FACE MASK DETECTION USING CONVOLUTIONAL NEURAL NETWORKS

A Mini Project Report Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Mr. Gajawada Venkat Sai Kumar(17071A05D7)
Mr. Goli Arun Kumar (17071A05E0)
Mr. Kodela Thomas Anvesh(17071A05E9)
Mr. Musham Sathesh (17075A05F8)
Mr. Thanniru Srikanth(18075A0536)

Under the Guidance of

Mrs. Channa Basamma

(Assistant Professor, Department of CSE, VNR VJIET)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING & TECHNOLOGY
An Autonomous Institute, NAAC Accredited With 'A++' Grade
NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B.Tech Courses
Approved by AICTE, New Delhi, Affiliated to JNTUH
Recognized as "College with Potential for Excellence" by UGC
VignanaJyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad 500 090, TS, India

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING and TECHNOLOGY

AN AUTONOMOUS INSTITUTE
(Approved by AICTE - New Delhi, Govt. of Telangana and Affiliated to JNTUH)
Accredited by NBA and NAAC with A++ Grade

VignanaJyothi Nagar, Bachupally, Nizampet (S.O.), Hyderabad-500 090.
Telangana, India.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report entitled "**FACE MASK DETECTION USING CONVOLUTIONAL NEURAL NETWORKS**" is a bonafide work done under our supervision and is being submitted by **Mr.Gajawada Venkat SaiKumar (17071A05D7), Mr.Goli Arun Kumar(17071A05E0),Mr.KodelaThomasAnvesh(17071A05E9),Mr.Musham Sathesh(17075A05F8)**, **Mr.Thanniru Srikanth(18075A0536)** in partial fulfilment for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering, of the VNRVJIET, Hyderabad during the academic year 2020-2021.

Certified further that to the best of our knowledge the work presented in this thesis has not been submitted to any other University or Institute for the award of any Degree or Diploma.

Mrs. Channa Basamma
Assistant Professor & Internal Guide
CSE Department, VNR VJIET

Dr. B.V. Kiranmayee
Associate Professor & HOD
CSE Department, VNRVJIET

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

AN AUTONOMOUS INSTITUTE
(Approved by AICTE - New Delhi, Govt. of Telangana and Affiliated to JNTUH)
Accredited by NBA and NAAC with A++ Grade

VignanaJyothi Nagar, Bachupally, Nizampet (S.O.), Hyderabad-500 090.
Telangana, India.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

Wedeclare that the thesis work entitled "**FACE MASK DETECTION USING CONVOLUTIONAL NEURAL NETWORKS**" submitted in the department of Computer Science and Engineering, Vallurupalli NageswaraRao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a bonafide record of our own work carried out under the supervision of **Mrs.Channa Basamma, Assistant Professor, Department of CSE, VNRVJIET.**Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree/diploma of any other institution or university previously.

Gajawada Venkat Sai Kumar(17071A05D7)

Goli Arun Kumar(17071A05E0)

Kodela Thomas Anvesh(17071A05E9)

Musham Sathesh(17071A05F8)

Thanniru Srikanth(18075A0536)

ACKNOWLEDGEMENT

Behind every achievement lies an unfathomable sea of gratitude to those who activated it, without it would ever never have come into existence. To them we lay the words of gratitude imprinting within us. We are indebted to our venerable principal **Dr. C. D. Naidu** for this unflinching devotion, which led us to complete this project. The support, encouragement given by him and his motivation lead us to complete this project. We express our thanks to internal guide **Mrs.Channa Basamma** and also Head of the department **Dr. B. V. Kiranmayee** for having provided us a lot of facilities to undertake the project work and guide us to complete the project. We express our sincere thanks to our faculty of the department of Computer Science and Engineering and the remaining members of our college **VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY** who extended their valuable support in helping us to complete the project in time.

Mr.Gajawada Venkat Sai Kumar(17071A05D7)
Mr.Goli Arun Kumar (17071A05E0)
Mr.Kodela ThomasAnvesh(17071A05E9)
Mr.Musham Sathesh(17071A05F8)
Mr.Thaniru Srikanth(18075A0536)

ABSTRACT

In today's scenario wearing a face mask is mandatory . Face Mask helps in preventing the spread of covid 19 . In this Project of building a Face Mask Detector we are going to use Convolutional Neural Network (CNN) , Python , Keras ,Tensorflow and OpenCV . For training this model we are going to bulid a dataset from the internet . After the dataset is ready ,data is Preprocessed and then the convolutional neural network model is trained using this data . Then this model will is optimized to detect the face mask accurately . With further improvements these types of models could be integrated with CCTV or other types cameras to detect and identify people without masks . With the prevailing worldwide situation due to covid19 pandemic , these types of systems would be very supportive for many kind of institutions around the world .

INDEX

LIST OF FIGURES

CONTENTS **Page No.**

CHAPTER 1

INTRODUCTION

1.1 Introduction to Machine Learning

Machine Learning is a tender which offers frameworks with the capability to perfunctorily take in and enhance from the information it secures without really being modified to do as such. This idea accentuates the advancement of projects that can consequently breakdown information and utilize it to absorb for themselves to improve decisions. This idea is critical to the area of Artificial Intelligence.

Learning initiates with data and impression of this data, for instance, coordinate finding, or direction, or representations, to search for designs, present regularly in material, and at last, distinguish between poor and enhanced judgments later with respect to cases that we give. The primary goal is to allow the PCs to learn ordinarily without human interpolation and modify exercises as indicated by the required.

ML enables programming applications to wind up more precisely in anticipating results without being expressly modified. The essential introduction of ML is to fabricate calculations that can get input information and utilize measurable going-over to get ahead and yield an incentive inside a satisfactory range.

1.2 Machine learning methods

ML algorithms are categorized as supervised or unsupervised.

i) Supervised Learning calculations can remove those that have been recognized previously to fresh information utilizing named cases to foresee future occasions. Beginning since the examination of a preparation dataset, the calculation delivers a gathered capacity to make forecasts about the yield estimates. The framework can give efforts to any first-hand contribution after suitable preparation. The learning calculation can likewise contrast its yield and the right, expected income, and realize blunders with a specific end objective to adjust the model as needed.

ii) In differentiation, unsupervised ML deviousness is developed when the data used to prepare is neither arranged nor marked. Unsupervised learning envisages how frameworks can surmise a capacity to portray a concealed structure from unlabelled information.

iii) Semi-regulated ML calculations fall someplace in the middle of managed and unsupervised learning, since they utilize both unlabelled information as well as marked one for preparing measures of some labeled information and a lot of unlabelled information. The frameworks that utilize this strategy can impressively enhance learning precision. Typically, semi-administered learning is picked when the obtained named information requires gifted and pertinent assets with a specific end goal to prepare it/gain from it.

1.3 Introduction to Neural Networks

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus a neural network is either a biological neural network, made up of real biological neurons or an artificial neural network, for solving artificial intelligence (AI) problems. The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are

modified by weight and summed. This activity is referred to as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be −1 and 1.
These artificial networks may be used for predictive modeling, adaptive control, and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.

How does it work in practice?
Once the network has been trained with enough learning examples, it reaches a point where you can present it with an entirely new set of inputs it's never seen before and see how it responds. For example, suppose you've been teaching a network by showing it lots of pictures of chairs and tables, represented in some appropriate way it can understand, and telling it whether each one is a chair or a table. After showing, let's say, 25 different chairs and 25 different tables, you feed it a picture of some new design it's not encountered before—let's say a chaise longue—and see what happens. Depending on how you've trained it, it'll attempt to categorize the new example as either a chair or a table, generalizing on the basis of its past experience—just like a human. Many of the things we all do every day involve recognizing patterns and using them to make decisions, so neural networks can help us out in zillions of different ways. They can help us forecast the stock market or the weather, operate radar scanning systems that automatically identify enemy aircraft or ships, and even help doctors to diagnose complex diseases on the basis of their symptoms. There might be neural networks ticking away inside your computer or your cell phone right this minute. If you use cell phone apps that recognize your handwriting on a touchscreen, they might be using a simple neural network to figure out which characters you're writing by looking out for distinct features in the marks you make with your fingers. Some kinds of voice recognition software also use neural networks. And so do some of the email programs that automatically differentiate between genuine emails and spam. Neural networks have even proved effective in translating text from one language to another. Google's automatic translation, for example, has made increasing use of this technology over the last few years to convert words in one language (the network's input) into the equivalent words in another language (the network's output). In 2016, Google announced it was using NMT to convert entire sentences, instantly, with a 55–85 percent reduction in error.

1.4 Existing Systems:

Face mask detection problem statement is new to the world. There might be some existing model.But there is no proper existing model on the internet .

There is model for face detection using haar cascade but it detects faces only without masks.
.
Disadvantages
It is hard to check whether every one is wearing a face mask or not using the existing system.

1.5 Proposed System

- The proposed system will help in detection of facemask easily anytime.
- This System uses MobileNetV2 model to detect face mask.
- MobileNetV2 is a popular CNN model used for Image Classification Purposes.

- This CNN model is made of several types of layer, like Convolutional Layer, Non-Linearity Layer, Rectification Layer, Rectified Linear Units (ReLU), Pooling Layer, Fully Connected Layer, Dropout Layer.

1.6 Proposed Model

This model firstly identifies the faces in the picture and then crops the faces.

This cropped faces are passed through the imagenet model to identify the faces without masks.

It may have accuracy between 90 - 95.

The proposed Methodology includes following steps:

1.Dataset Collection
2.Data Augmentation
3.Data Processing
4.Training the model
5.Performance Evaluation

Dataset Collection :

The Dataset have 2 classes of images real and fake. It has total of 85 images- 70 for training and 15 for testing. There are 60 real currency images and 25 fake currency images.

Data Augmentation:

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. Transformation of images are done by applying various operations like Image flip, Image rotate, image shift, Image brightness and Image Zoom.These transformations are supported via ImageDataGeneratorClass.

Data Processing:

After augmentation all the data have different heights and widths. Our model requires a fixed pixel for all data images. We resize all data in a fix resolution- 224x224 for our model.

Train the Model:

To train the model Transfer Learning method is used.Transfer Learning is process of taking a model trained on a large dataset and transfer its knowledge to a smaller dataset. For object recognition with a CNN, we freeze the early convolutional layers of the network and only train the last few layers which make a prediction.After Data augmentation and Processing, training of the model begins. The model script runs 100 training steps. Each step chooses 8 images at random from the training set, find their features and feeds them into the final layer to get predictions. Those predictions are then used to update final layer's weights.After adding new layers these layers of the model are also

trained using the same train dataset.

Performance Evaluation:

After all the training steps are completed, the scriptruns a final test accuracy evaluation on a set of images that are kept for testing purpose and accuracy of the model is evaluated using the formula:

$$\text{Accuracy} = \frac{\text{Samples correctly classified}}{\text{Total number of samples}}$$

After training the last layer of the pre trained MobileNetV2 model and testing it , the model has a testing accuracy of 0.99 which is a decent accuracy.

CHAPTER - 2

FEASIBILITY STUDY

A feasibility study involves taking a judgment call on whether a project is doable. The two criteria to judge feasibility are cost required and value to be delivered. A well-designed study should offer a historical background of the business or project, a description of the product or service, accounting statements, details of operations and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, such studies precede technical development and project implementation.

A feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions.

2.1 Technical Feasibility

Technical feasibility involves evaluation of the hardware and the software requirements of the proposed system.In this project, the technology involved is machine learning, deeplearning along with concepts of Python Programming , opencv,keras and tensorflow are used to implement the face mask detection .The tool that is used to execute the Python code is jupyter notebook.

2.2 Economic Feasibility

Economic Feasibility helps in assessing the viability, cost, and benefits associated with projects before financial resources are allocated. This assessment typically involves a cost/ benefits analysis of the project.The application is so designed that it requires minimal cost as there would be minimal need for manual work. The technologies used can easily be understood by the user without any investment. Since the model is trained, it reduces the cost that is required to deploy the man power and consequently eliminates the problem of time consumption.

2.3 Legal Feasibility

The proposed system doesn’t conflict with legal requirements like data protection acts or social media laws. It ensures legal data access and gives prominence to data security.

2.4 Operational Feasibility

The application involves design-dependent parameters such as reliability,maintainability, supportability, usability, disposability, sustainability, affordability, and others. It minimizes the drawbacks of the current system by building an application that automatically resolves the user queries and helps to analyses the user data.

2.5 Scheduling Feasibility

The project development took place in timely process by understanding time schedules of the project and maintains good time line for project development.

CHAPTER -3

LITERATURE SURVEY

A detailed study of the face mask detection is discussed in which properties of dataset, face mask detection study classifiers are exposed. Visual features of image is examined and some of the classifier techniques are discussed which are helpful in the further inspection of the methods of face recognition. The arise of neural networks has tremendously increased the scope to solve problems in data science.Face can be identified from facial images using Haar cascade classifier and Deep CNN which gives good accuracy rate with which we had an inference that deep learning can also be used for face mask detection. Face mask detection can be also performed with deep convolutional networks..

We discussed that transfer learning methods,mobilenetv2 models which basically loads the imagenet dataset and Tensorflow can be used as a backend. Tensorflow is a low level where as Keras is a model level. Tensorflow is developed by Google and it is most efficient to use it in a image recognition and face recognition with or without mask.

3.1 Machine learning

Machine learning is a buzzword these days, the reason for this is the huge amount of data production by applications and the increase of computation power. The term machine learning was first introduced by Arthur Samuel in 1959. We can define it in a summarized was as: "Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed."

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without beginning explicitly programmed.Machine learning focuses on the development of computer programs that can access data and use it learn from themselves.

3.2 Machine learning models

A Machine Learning system learns from historical data, builds the prediction models and whenever it receives new data, predicts the output for it. The accuracy of the predicted output depends up on the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predicts the output. Machine learning has changed our way of thinking about the problem.

3.3. Types of machine learning

Machine Learning is broadly classified in to three types they are:

1) Supervised Learning

2) Unsupervised Learning

3) Reinforcement Learning

Supervised Learning:

The system creates a model using labelled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not. The goal of supervised learning is to map input data with the output data. The supervised learning is based on supervision, and it is the same as when a student learns things in the supervision of the teacher. The example of supervised learning is spam filtering. Supervised learning can be grouped further in two categories of algorithms. They are:

1) Classification

2) Regression

Unsupervised Learning:

Unsupervised learning is a learning method in which a machine learns without any supervision.

The training is provided to the machine with the set of data that has not been labelled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns. In unsupervised learning, we don't have a predetermined result. The machine tries to find useful insights from the huge amount of data. It can be further classifieds into two categories of algorithms. They are:

1) Clustering

2) Association

Reinforcement Learning:

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance. The robotic dog, which automatically learns the movement of his arms, is an example of Reinforcement learning

3.4 Machine Learning Applications

Machine Learning is a diversified field and its applications are as follows:

Image Recognition: Image recognition is one of the most common applications of machine learning. It is used to identify objects, persons, places, digital images, etc. The popular use case of image recognition and face detection is, Automatic friend tagging suggestion. Facebook provides us a feature of auto friend tagging suggestion. Whenever we upload a photo with our Facebook friends, then we automatically get a tagging suggestion with name, and the technology behind this is machine learning's face detection and recognition algorithm.

Speech Recognition: While using Google, we get an option of "Search by voice," it comes under speech recognition, and it's a popular application of machine learning. Speech recognition is a process of converting voice instructions into text , and it is also known as "Speech to text", or "Computer speech recognition." At present, Machine Learning algorithms are widely used by various applications of speech recognition. Google assistant, Siri , Cortana, and Alexa are using speech recognition technology to follow the voice instructions.

Traffic Prediction: If we want to visit a new place, we take help of Google Maps, which shows us the correct path with the shortest route and predicts the traffic conditions. It predicts the traffic conditions such as whether traffic is cleared, slowmoving, or heavily congested .

Product Recommendation: Machine Learning is widely used various ecommerce and entertainment companies such as Amazon, Netflix, etc., for product recommendation to the user. Whenever we search for some product on Amazon, then we started getting an advertisement for the same product while internet surfing on the same browser and this is because of machine learning.

CHAPTER - 4

CONVOLUTIONAL NEURAL NETWORKS

4.1 Introduction to Convolutional Neural Network (CNN)

Convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing and financial time series.CNN image classifications take an input image, process it, and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers see an input image as an array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d(h = Height, w = Width, d = Dimension). Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC), and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

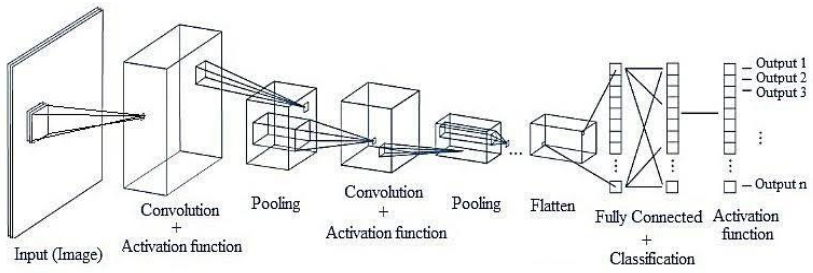


Fig 4.1 Convolutional Neural Network

4.2 Convolutional Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

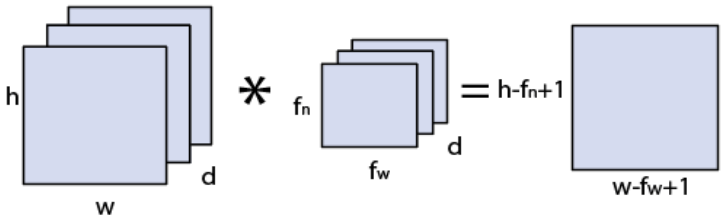
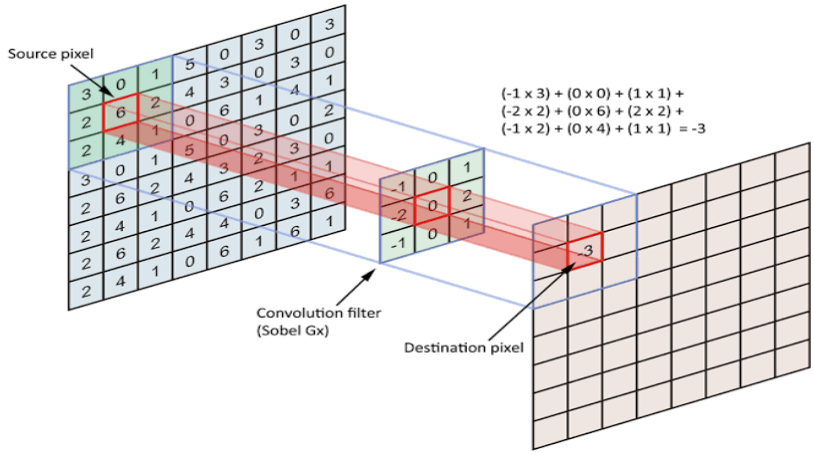


Image matrix multiplies kernl or filter matrix

Then the convolution of 8 x 8 image matrix multiplied with 3 x 3 filter matrix which is called **"Feature Map"** as output shown in below



4.3 Filters

Filters In convolutional (filtering and encoding by transformation) neural networks (CNN) every network layer acts as a detection filter for the presence of specific features or patterns present in the original data.

Linear Filter: The linear filter is a well-defined operation for any set of parameters (convolution kernel) or input data we can think of. We can now build a single layer, a single kernel, a convolutional neural network that approximates the linear filtering operation.

Median Filter: The median filter is a non-linear digital filtering technique, often used to remove noise from an image or signal. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median filtering is very

widely used in digital image processing because, under certain conditions, it preserves edges while removing noise (but see the discussion below), also having applications in signal processing.

Iterative Bilateral Filtering: Image noises are distortion in the image that arises due to fault in the camera or result of poor visibility due to changing weather conditions. Noises are also a random variation in the intensity levels of the pixels. Noise can be of various types like Gaussian noise, Salt, and pepper noise. In this proposed method, we use an iterative bilateral filter for noise removal.

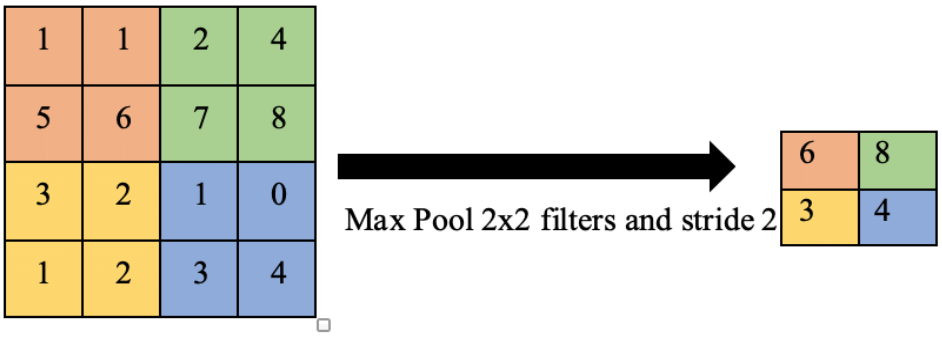
Gaussian filter: Gaussian filter is expressed as the normal distribution which is the limiting distribution of the normalized sum of iid random variables. It is characterized by the center and variance. Due to a large number of similar individuals, a variety of variables can be approximated by the Gaussian variable or Gaussian process, such as noise, heat, quantum, etc. The Gaussian filter has the highest value in the center and FWHM proportional to the standard deviation, decreasing rapidly in the area three times of standard deviation away from the center. With these properties, the Gaussian filter is used as a low pass filter for smoothing or denoising.

Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. The below example shows various convolution images after applying different types of filters (Kernels).

Operation	Filter
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

Strides

Stride is the number of pixels shifted over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.



Padding

Sometimes filters do not perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only the valid part of the image.

4.4 Activation Function

In the process of building a neural network, one of the choices you get to make is what activation function to use in the hidden layer as well as at the output layer of the network. The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron. We know, a neural network has neurons that work in correspondence of weight, bias, and their respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as back-propagation. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases. A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks. Popular types of activation functions and when to use them.

4.4.1. Binary Step

The first thing that comes to our mind when we have an activation function would be a threshold-based classifier i.e. whether or not the neuron should be activated based on the value from the linear transformation. In other words, if the input to the activation function is greater than a threshold, then the neuron is activated, else it is deactivated, i.e. its output is not considered for the next hidden layer. The binary step function can be used as an activation function while creating a binary classifier. As you can imagine, this function will not be useful when there are multiple classes in the target variable. That is one of the limitations of the binary step function.

4.4.2 Linear Function

The problem with the step function, the gradient of the function became zero. This is because there is no component of x in the binary step function. Instead of a binary function, we can use a linear function. Although the gradient here does not become zero, it is a constant which does not depend upon the input value x at all. This implies that the weights and biases will be updated during the backpropagation process but the updating factor would be the same. In this scenario, the neural network will not really improve the error since the gradient is the same for every iteration. The network will not be able to train well and capture the complex patterns from the data. Hence, a linear function might be ideal for simple tasks where interpretability is highly desired.

4.4.3 Sigmoid

The next activation function is the Sigmoid function. It is one of the most widely used non-linear activation functions. Sigmoid transforms the values between the range 0 and 1. When there are multiple neurons having a sigmoid function as their activation function, the output is non-linear.

4.4.4 ReLU

The ReLU function is another non-linear activation function.ReLU stands for the Rectified Linear Unit. 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. This means that the neurons will only be deactivated if the output of the linear transformation is less than 0. For the negative input values, the result is zero, which means the neuron does not get activated. Since only a certain number of neurons are activated, the ReLU function is far more computationally efficient when compared to the sigmoid.

4.4.5 Leaky ReLU

Leaky ReLU function is nothing but an improved version of the ReLU function. As we know that for the ReLU function, the gradient is 0 for $x<0$, which would deactivate the neurons in that region. Leaky ReLU is defined to address this problem. Instead of defining the Relu function as 0 for negative values of x, we define it as an extremely small linear component of x.

4.4.6 Parameterised ReLU

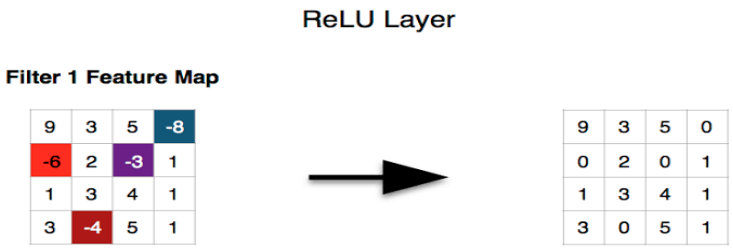
This is another variant of ReLU that aims to solve the problem of gradient's becoming zero for the left half of the axis. The parameterized ReLU, as the name suggests, introduces a new parameter as a slope of the negative part of the function.

Choosing the right Activation Function

- Sigmoid functions and their combinations generally work better in the case of classifiers
- Sigmoid and tanh functions are sometimes avoided due to the vanishing gradient problem
- ReLU function is a general activation function and is used in most cases these days.

Non Linearity (ReLU)

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is ***f(x) = max(0,x).*** Why ReLU is important: ReLU's purpose is to introduce non-linearity in our ConvNet. Since the real-world data would want our ConvNet to learn would be non-negative linear values.



There are other nonlinear functions such as tanh or sigmoid that can also be used instead of ReLU. Most of the data scientists use ReLU since performance-wise ReLU is better than the other two.

Pooling Layer

The pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling is also called subsampling or downsampling which reduces the

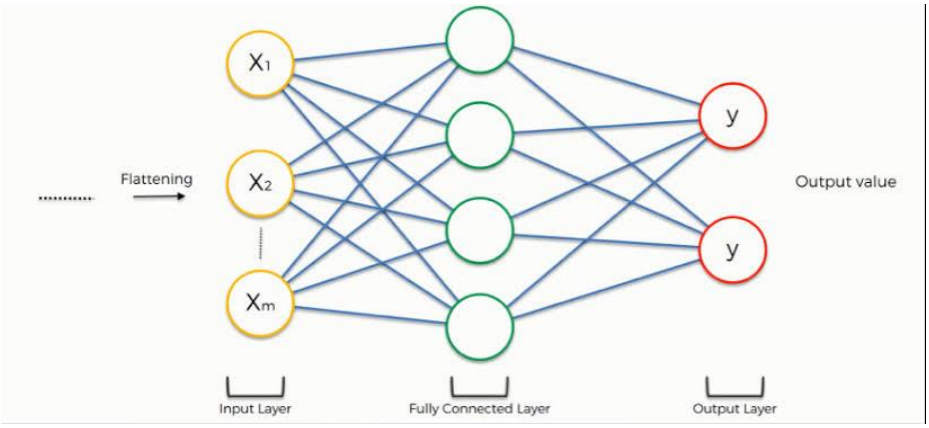
dimensionality of each map but retains important information. Spatial Pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

Fully Connected Layer

The layer we call the FC layer, we flatten our matrix into a vector and feed it into a fully connected layer like a neural network.



In the above diagram, the feature map matrix will be converted as a vector (x1, x2…..xn). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs as cat, dog, car, truck, etc.

4.5 CNN Advantages

1.The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. For example, given many pictures of cats and dogs, it learns distinctive features for each class by itself. 2.CNN is also computationally efficient.

3.Multi-layer neural networks are practical for many tasks.

4.Model complexity, learning rates, maximum-likelihood estimation, and prediction accuracy all scale linearly with model complexity, and training is usually quick and easy.

5.They have better performance on multiple features simultaneously in large images.

6.Convolutional Neural Networks are not simple in theory, but in practice, they are efficient, and extremely efficient too

7.Multi-layered networks are better for training because they can reuse much of the computation available in each layer.

4.6 Applications of CNN

1.Face recognition

2.Scene Labelling

3.Image Classification

4.Action Recognition

5.Human Pose Estimation

6.Document Analysis

4.7 Summary

1.Provide input image into convolution layer

2.Perform pooling to reduce dimensionality size

3. as many convolutional layers until satisfied

4.Flatten the output and feed into a fully connected layer (FC Layer)

5.Output the class using an activation function and classifies images.

CHAPTER-5

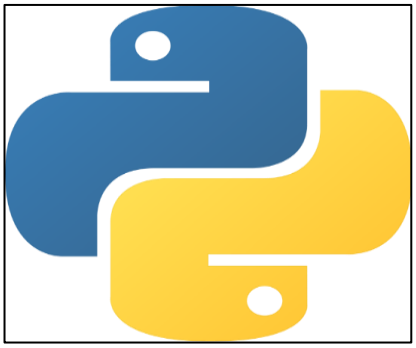
SYSTEM ANALYSIS

5.1 System Requirements

5.1.1 Software Requirements

5.1.1.1 Python

Python is a deciphered language, Guido van Rossum, Python has a diagram hypothesis that complements code decipherability, and a sentence structure that empowers programming architects to express thoughts in less lines of code noticeably using imperative whitespace. It gives builds up that engages a little immense. Incorporates a kind modified organization. Reinforces different perfect models, masterminded, essential, useful and, and a huge and exhaustive.



5.1.1.2 Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to make an offer reports that cover live code, conditions, observations, and account content. Utilizations include information cleaning and change, numerical reproduction, factual demonstrating, information representation, machine learning, and significantly more. The Scratchpad has to bolster for more than 40 programming dialects, including Python, R, Julia, and Scala. Note pads can be imparted to others using email, Dropbox, GitHub, and the Jupyter Notebook Watcher.

5.1.2 Hardware Requirements

RAM:	4GB and Higher
Processor:	Intel i3 and above
Hard Disk:	10GB: Minimum

5.2 System Architecture

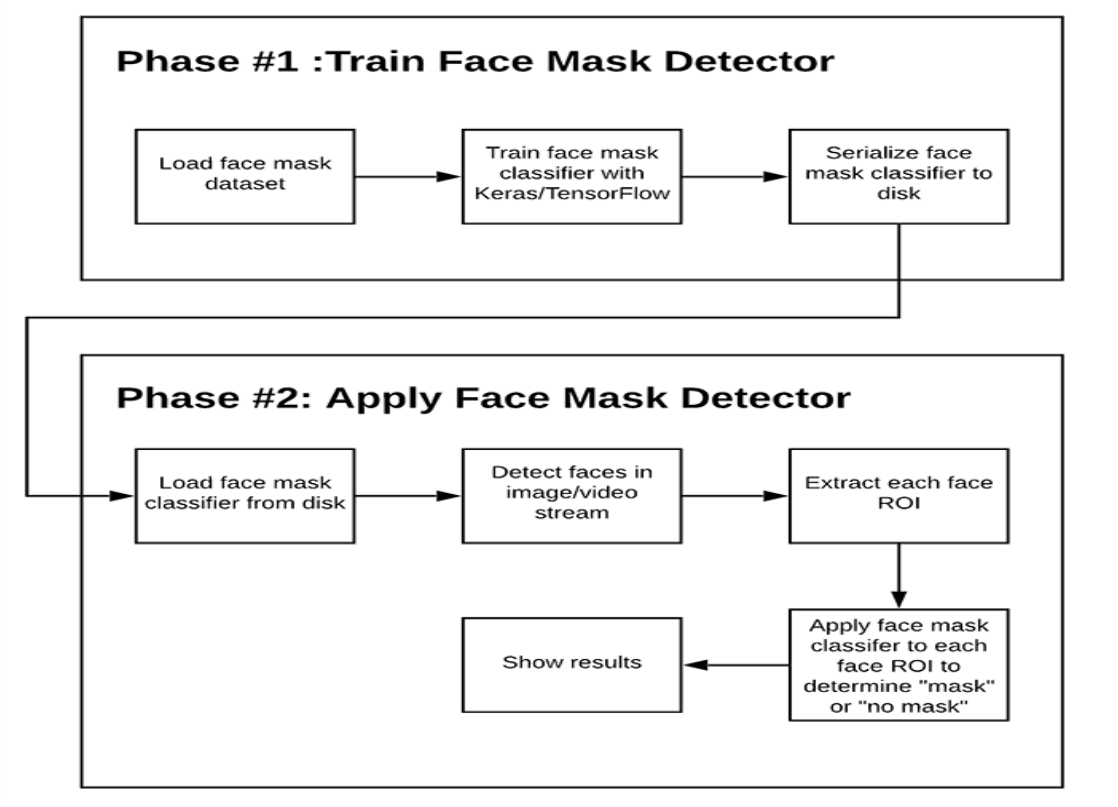


Figure 5.2: Phases and individual steps for building a COVID-19 face mask detector

Training: Here we'll focus on loading our face mask detection dataset from disk, training a model on this dataset, and then serializing the face mask detector to disk.

Deployment: 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_mask

a) Loading Dataset

In this, we need to Load the dataset (question) which contains the required information.

b) Data preprocessing

The dataset we are using consists of images with different colors, different sizes, and

different orientations. Therefore, we need to convert all the images into grayscale because we need to be sure that color should not be a critical point for detecting mask. After that, we need to have all the images in the same size (100x100) before applying it to the neural network**.**

c) Training

Here we'll focus on loading our face mask detection dataset from disk, training a model (using Keras/TensorFlow) on this dataset, and then serializing the face mask detector to disk.

d) Deployment

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_mask.

CHAPTER - 6

SOFTWARE DESIGN

6.1 UML Diagrams

Unified Modelling Language is a tool that helps a designer to present his ideas about the project to his client and his developer. Modeling plays a crucial role in designing software. A poorly designed model can lead to poorly developed software. A UML system has five different views that help in describing systems from different perspectives. Each view has a set of diagrams and components that represent the real-time objects.

a. User Model View:

i. It models user behavior in a system context.

ii. All the diagrams are drawn keeping in mind the user's response and reaction towards a system.

b. Structural Model View

i. This view consists of a class diagram and object diagram which is used to model the static structures.

ii. It uses objects, attributes, operations, and relationships.

c. Behavioural Model View

i. It mainly consists of the sequence diagram, collaboration diagram, state chart diagram and activity diagram. They mainly represent flow of actions between different objects involved in the system.

ii. They are used to visualize various dynamic aspects of the system architecture.

d. Implementation Model View

i. This view consists of component diagrams and deployment diagrams. This view models the static software modules for an organization.

ii. This usually contains the data files, documentation, the executables and source code.

iii. These are the physically replaceable components of the system. They are modelled using component diagrams.

6.1.1 Use Case Diagram

The basic representation for the interaction of the user with the system is represented using the use case diagram. It involves the relationship between the user and various use cases with the actors being involved. There are different kinds of relationships that are involved between the use cases and the actors. They include:

a. Association relationship

b. Generalization

c. Dependency

d. Realizations
e.Transitions

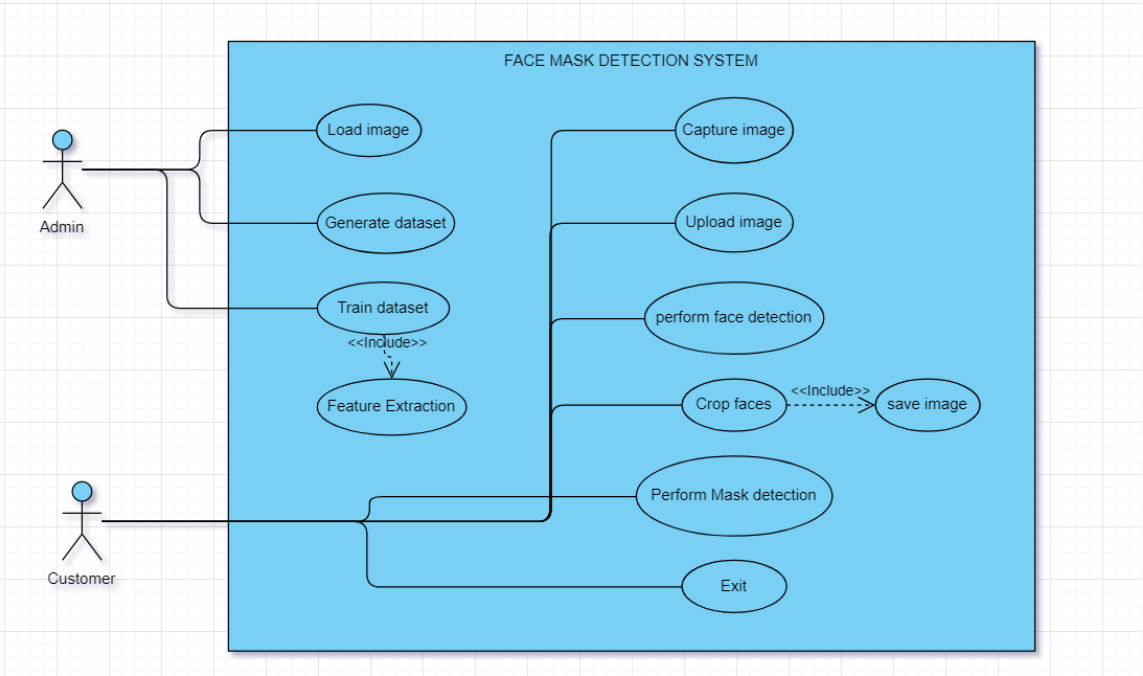


Fig 6.1.1 Use Case Diagram

6.1.2 Class Diagram

They are a static representation of an application. Only the class diagrams have the capability to be directly m OOP Languages because in OOPs everything is modeled in the form of classes and objects. Because of this reason the used widely at the time of construction. This is one of the most popularly used UML diagrams in the designer comm diagram plays an essential role in forward and reverse engineering.

a. It acts as a base for the component and deployment diagrams.
b. It mainly describes and defines the basic responsibilities of a system's application.

c. It implements the analysis and design view for a static application.

In a class diagram, each object is modeled as a class. Each class consists of a section or compartments.

1. Class name
2. Attributes of a class or operations
3. Methods or functions
4. Documentation (optional section)

The following points ought to be recollected while drawing a class diagram:

a. The name of the class diagram must be meaningful to portray the aspect of the framework.
b. Each component and its connections must be distinguished ahead of time.
c. Each class has a responsibility (attributes and methods) that must be identified clearly.
d. The number of properties for each class must be minimum. Since pointless properties will make t convoluted.
e. At whatever point required to depict some part of the diagram use notes Since toward the finish of the dia justifiable to the designer/coder.
f. Before finalizing the last version, the diagram must be drawn on plain paper and revise whatever numbe as would be prudent to make it redress.

1. Scopes: The UML diagrams have two different types of scopes for class members.

i. instance members scope
ii. classifier members scope

2. Classifier members are "static" members of a class in many programming languages. The sc itself.

i. Static attributes are common to all other objects that invoke the class.
ii. Static methods are not instantiated.

3. Instance members are nothing but the members that are local to an object.

i. The main purpose of instance members is to allow the objects to store their states.
ii. Declarations outside the methods are usually known as instance members.

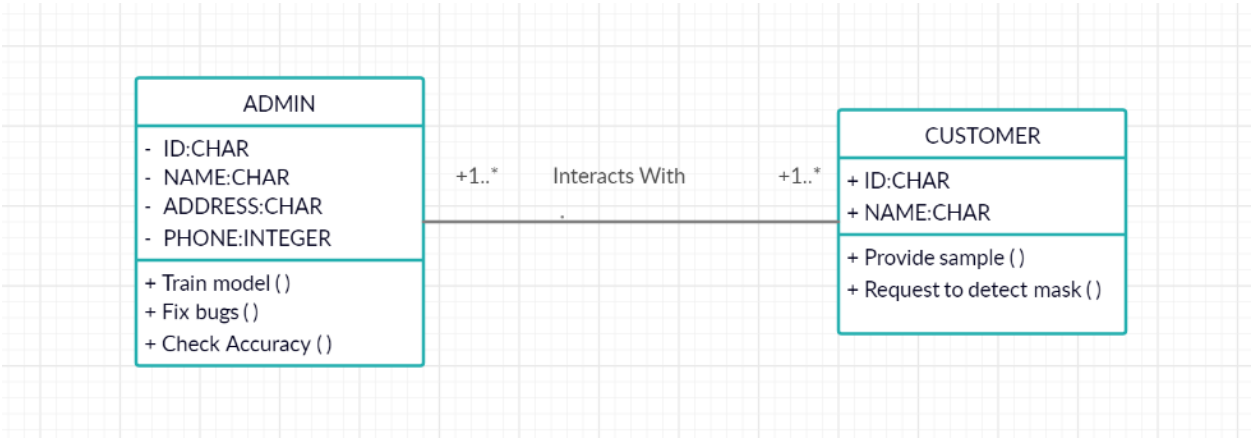


Fig 6.1.2 Class Diagram

6.1.3 Sequence Diagram

The Sequence Diagram depicts the time sequence among various objects in an application. It depicts t messages with which objects communicate with each other so that they carry out the required functionality. It lifelines which are usually parallel vertical lines. It consists of horizontal arrows that indicate the direction of th are exchanged in a proper order which makes the user easy to understand. The lifeline for a given object repres synchronous calls are represented with the help of a solid arrowhead whereas the asynchronous messages are r the help of open arrowheads. All objects are represented according to their time ordering. Timing of messages pla in sequence diagrams. An object is killed immediately after its use in sequence diagrams.

I). Common Properties:

An arrangement graph is much the same as a unique sort of diagram and offers some indistinguishable other diagrams. In any case, it varies from every single other diagram in its content.

II). Contents

Objects are normally named or unknown instances of a class, however may likewise speak to occurren things, for example components, collaboration, and nodes. Graphically, an object is represented as a rectangle b name.

III). Links

A link is a semantic association among objects i.e., an object of affiliation is called as a connection. It is line.

IV). Messages

A message is a determination of a correspondence between objects that passes on the data with the desire will follow.

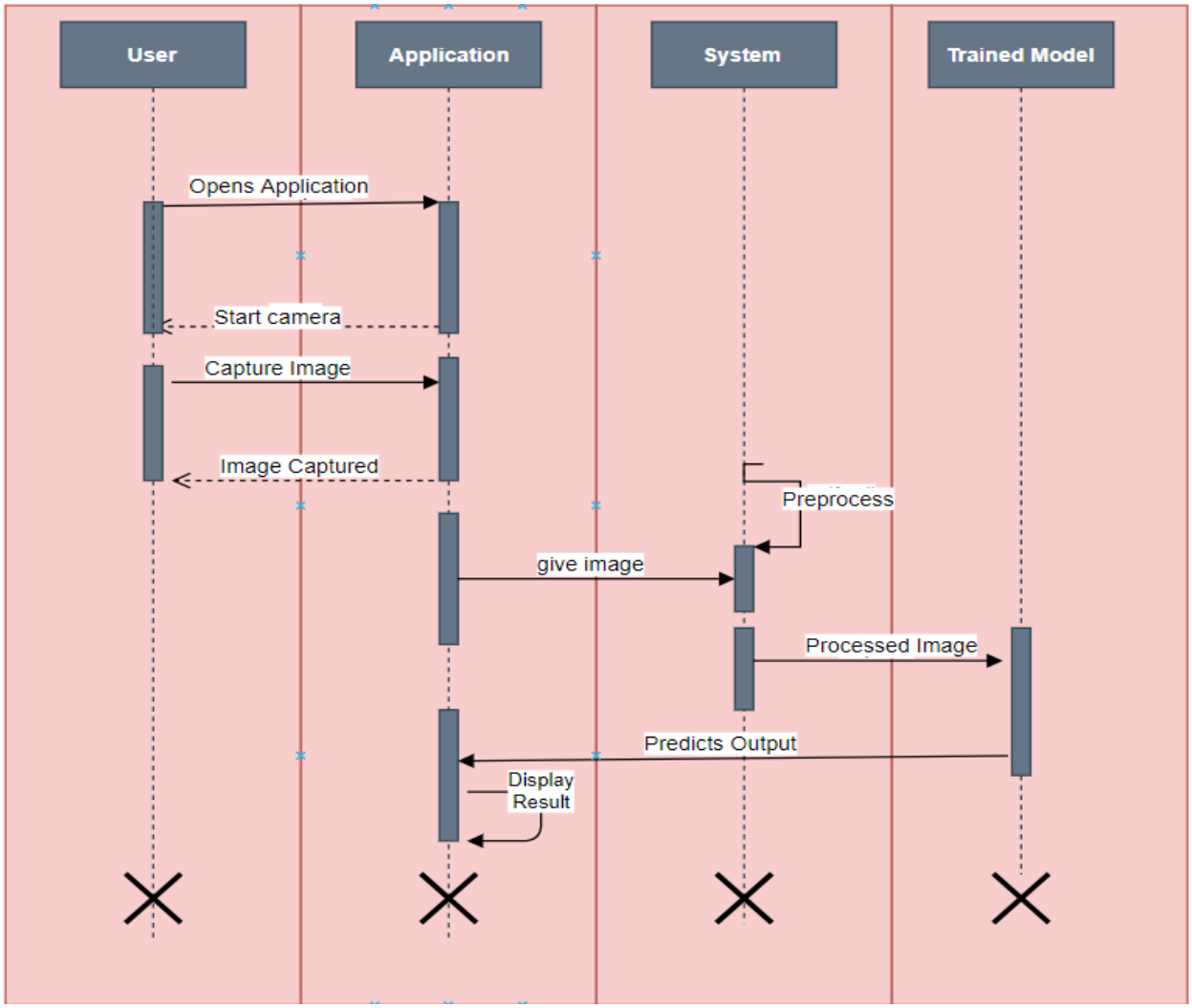


Fig 6.1.3 Sequence Diagram

6.1.4 Activity Diagram

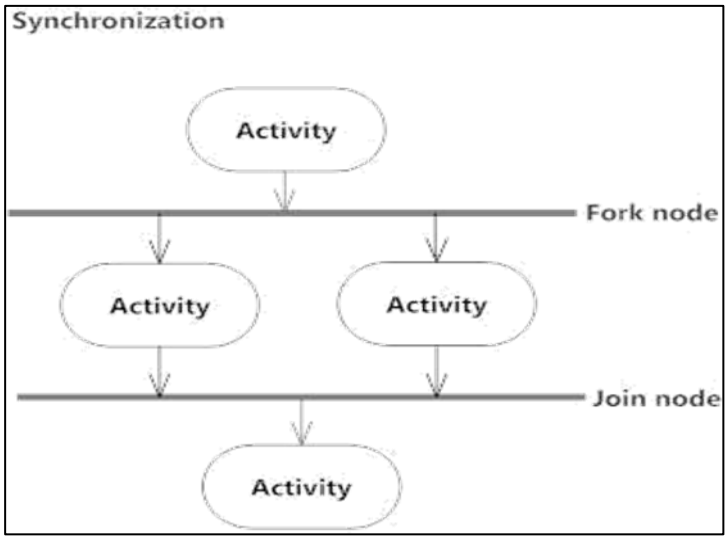
The flow from one activity to another activity can be represented in the form of a flow chart which is usually an ac
It forms a backbone for the UML diagrams. It depicts the dynamic aspects of all the objects within the system. T
from one object to another object is drawn which shows the basic operations that are to be performed.

Activity diagrams are constructed using the following:

1. Actions are represented using rounded rectangles;

2. Decisions are represented using diamonds;

3. Concurrent activity bars are represented using the start (split) or end (join) ;



4. Time event is represented as

5. Final state is represented using an encircled black circle.

The basic purpose of an activity diagram is same as that of other UML diagrams. The dynamic behaviour of the system is viewed by the activity diagram. They are used to construct the system using the backward and forward engineering mechanisms.

The purpose of an activity diagram is as follows:

1)For drawing the flow (i.e. activity) in a system.

2)For showing the flow of sequence from one activity to another activity.

3)For showing the concurrent and parallel flow of actions in the system.

The elements that are used in an activity diagram are as follows:
i)Association relationship
ii)Activities
iii) Conditions and Constraints

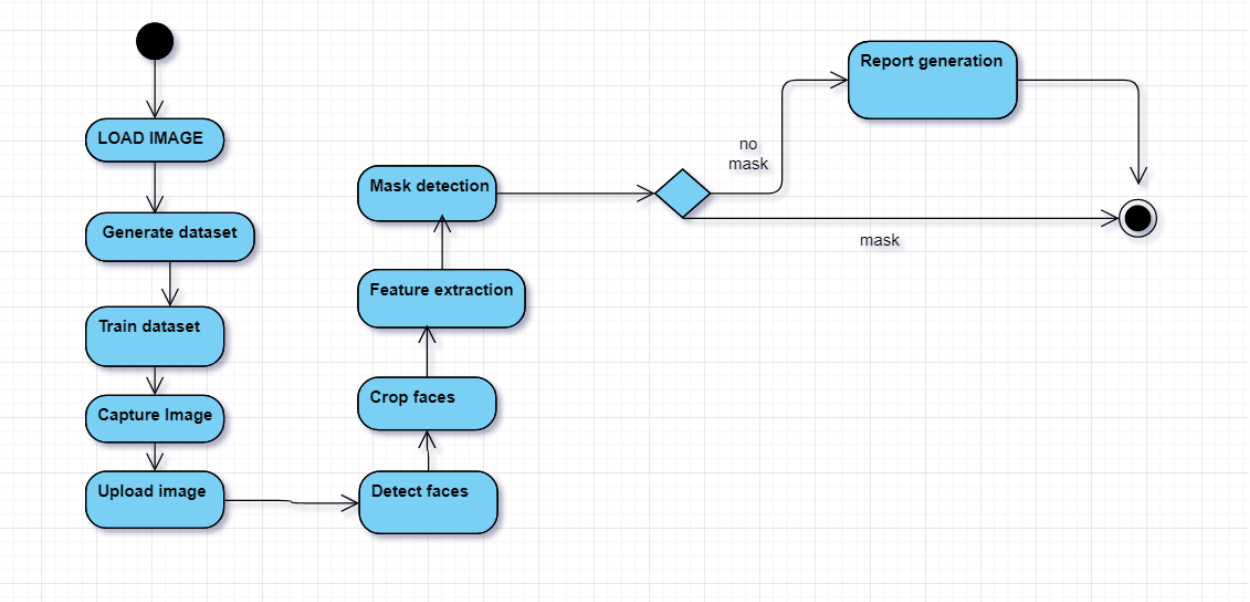


Fig 6.1.4 Activity Diagram

CHAPTER - 7

IMPLEMENTATION

7.1 Code for importing modules

Importing the required modules and libraries to apply different techniques and displaying the result.

```
[ ]  from keras.models import Sequential
     from keras.layers import Dense,Activation,Flatten,Dropout,AveragePooling2D
     from keras.layers import Conv2D,MaxPooling2D
     from keras.callbacks import ModelCheckpoint
     from keras.models import Model
     import tensorflow as tf
     import os
     import numpy as np
     import matplotlib.pyplot as plt
```

7.2 Code for Data Preprocessing

Here we freeze all the inner layers so that we can use transfer learning method and add the top layers to the model.

```
import cv2,os

data_path='/content/drive/My Drive/Face Mask Detection/dataset'
categories=os.listdir(data_path)
labels=[i for i in range(len(categories))]

label_dict=dict(zip(categories,labels))

print(label_dict)
print(categories)
print(labels)
```

```
{'with mask': 0, 'without mask': 1}
['with mask', 'without mask']
[0, 1]
```

7.3 Code for conversion of image to grey scale

Since images will be of different size,colour and orientation we must convert all images to grey scale since colour does not matter in mask detection and resize all the images

```
[ ] img_size=224
    data=[]
    target=[]


    for category in categories:
        folder_path=os.path.join(data_path,category)
        img_names=os.listdir(folder_path)

        for img_name in img_names:
            img_path=os.path.join(folder_path,img_name)
            img=cv2.imread(img_path)

            try:
                gray= cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                #Coverting the image into gray scale
                resized=cv2.resize(gray,(img_size,img_size))
                #resizing the gray scale into 100x100, since we need a fixed common size for all the images in the dataset
                data.append(resized)
                target.append(label_dict[category])
                #appending the image and the label(categorized) into the list (dataset)

            except Exception as e:
                print('Exception:',e)
                #if any exception rasied, the exception will be printed here. And pass to the next image
```

7.4 Code for loading model

Loading MobilenetV2 Model with weights from imagenet dataset excluding the top layer so that it can be trained on our dataset. The input shape for MobileNetV2 model is 224x224 with 3 channels (RGB).

```
from keras.applications import MobileNetV2
baseModel = MobileNetV2(weights='imagenet', include_top=False)
# Create the base model from the pre-trained model MobileNet V2
baseModel = MobileNetV2(weights='imagenet', include_top=False,input_shape=(224,224, 3))
```

7.5 Code for Training Model

We compile and train the model. We also save the checkpoints so that in case of any interruption we can resume training from checkpoint.

```
[ ]  from keras.models import Sequential
     from keras.layers import Dense,Activation,Flatten,Dropout,AveragePooling2D
     from keras.layers import Conv2D,MaxPooling2D
     from keras.callbacks import ModelCheckpoint
     from keras.models import Model
     import tensorflow as tf
     import os
     import numpy as np
     import matplotlib.pyplot as plt
     from keras.applications import MobileNetV2
     baseModel = MobileNetV2(weights='imagenet', include_top=False)
     # Create the base model from the pre-trained model MobileNet V2
     baseModel = MobileNetV2(weights='imagenet', include_top=False,input_shape=(224,224, 3))
     baseModel.trainable = False
     headModel = baseModel.output
     headModel = Dropout(0.75)(headModel)
     headModel = Flatten(name="flatten")(headModel)
     headModel = Dropout(0.75)(headModel)
     headModel = Dense(128, activation="relu")(headModel)
     headModel = Dropout(0.75)(headModel)
     headModel = Dense(2, activation="softmax")(headModel)
     model = Model(inputs=baseModel.input, outputs=headModel)

     #The Final layer with two outputs for two categories

     model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
[ ]  from sklearn.model_selection import train_test_split

     train_data,test_data,train_target,test_target=train_test_split(data,target,test_size=0.1)

[ ]  checkpoint = ModelCheckpoint('model-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
     history=model.fit(train_data,train_target,epochs=5,callbacks=[checkpoint],validation_split=0.2)

     Train on 990 samples, validate on 248 samples
     Epoch 1/5
     990/990 [==============================] - 39s 40ms/step - loss: 2.8808 - accuracy: 0.7838 - val_loss: 0.0791 - val_accuracy: 0.9839
     Epoch 2/5
     990/990 [==============================] - 22s 23ms/step - loss: 0.1833 - accuracy: 0.9808 - val_loss: 0.0978 - val_accuracy: 0.9879
     Epoch 3/5
     990/990 [==============================] - 22s 23ms/step - loss: 0.1240 - accuracy: 0.9848 - val_loss: 0.0738 - val_accuracy: 0.9960
     Epoch 4/5
     990/990 [==============================] - 22s 23ms/step - loss: 0.0247 - accuracy: 0.9949 - val_loss: 2.3841e-07 - val_accuracy: 1.0000
     Epoch 5/5
     990/990 [==============================] - 23s 23ms/step - loss: 0.0500 - accuracy: 0.9970 - val_loss: 0.0207 - val_accuracy: 0.9960
```

7.6 Code for Applying the trained model using webcam

Inorder to collect the data from the users we must link the webcam so that the model can take the data from webcam and displays the result.

```
[ ]  from keras.models import load_model
     import cv2
     import numpy as np
```

```
Using TensorFlow backend.
```

```
[ ]  model = load_model('model-004.model')

     face_clsfr=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

     source=cv2.VideoCapture(0)

     labels_dict={0:'MASK',1:'NO MASK'}
     color_dict={0:(0,255,0),1:(0,0,255)}
```

```
while(True):

    ret,img=source.read()
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    faces=face_clsfr.detectMultiScale(gray,1.3,5)
    for x,y,w,h in faces:
        face_img=gray[y:y+w,x:x+w]
        resized=cv2.resize(face_img,(224,224))
        normalized=resized/255.0
        reshaped=np.reshape(normalized,(1,224,224,1))
        result=model.predict(reshaped)

        label=np.argmax(result,axis=1)[0]

        cv2.rectangle(img,(x,y),(x+w,y+h),color_dict[label],2)
        cv2.rectangle(img,(x,y-40),(x+w,y),color_dict[label],-1)
        cv2.putText(img, labels_dict[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)

    cv2.imshow('LIVE',img)
    key=cv2.waitKey(1)

    if(key==27):
        break

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

CHAPTER 8

TESTING

8.1 Performance Evaluation:

To evaluate the performance of the model , the model should be tested on data that it has never seen before. For that purpose we generate the testing dataset and test the model on testing dataset.

Code for Model Evaluation and its corresponding output:

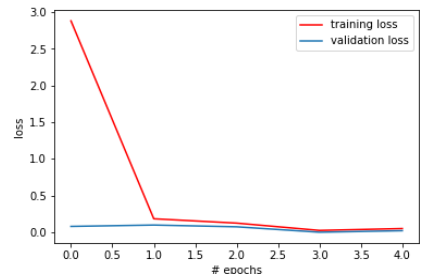
```
[ ] from sklearn.model_selection import train_test_split

    train_data,test_data,train_target,test_target=train_test_split(data,target,test_size=0.1)
```

```
[ ] checkpoint = ModelCheckpoint('model-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
    history=model.fit(train_data,train_target,epochs=5,callbacks=[checkpoint],validation_split=0.2)
```

```
Train on 990 samples, validate on 248 samples
Epoch 1/5
990/990 [==============================] - 39s 40ms/step - loss: 2.8808 - accuracy: 0.7838 - val_loss: 0.0791 - val_accuracy: 0.9839
Epoch 2/5
990/990 [==============================] - 22s 23ms/step - loss: 0.1833 - accuracy: 0.9808 - val_loss: 0.0978 - val_accuracy: 0.9879
Epoch 3/5
990/990 [==============================] - 22s 23ms/step - loss: 0.1240 - accuracy: 0.9848 - val_loss: 0.0738 - val_accuracy: 0.9960
Epoch 4/5
990/990 [==============================] - 22s 23ms/step - loss: 0.0247 - accuracy: 0.9949 - val_loss: 2.3841e-07 - val_accuracy: 1.0000
Epoch 5/5
990/990 [==============================] - 23s 23ms/step - loss: 0.0500 - accuracy: 0.9970 - val_loss: 0.0207 - val_accuracy: 0.9960
```

After the model is trained for 5 epoch it has got a validation accuracy of 0.996 and a minimum validation loss of 0.207.

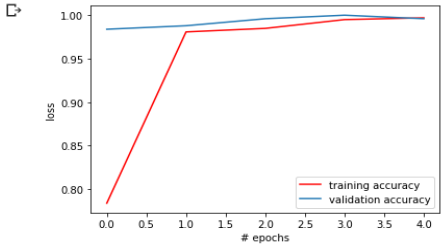
```
[ ] from matplotlib import pyplot as plt

    plt.plot(history.history['loss'],'r',label='training loss')
    plt.plot(history.history['val_loss'],label='validation loss')
    plt.xlabel('# epochs')
    plt.ylabel('loss')
    plt.legend()
    plt.show()
```



Graph for validation and training loss.

Code for checking the training and validation accuracy of the model.

```
[ ]  plt.plot(history.history['accuracy'],'r',label='training accuracy')
     plt.plot(history.history['val_accuracy'],label='validation accuracy')
     plt.xlabel('# epochs')
     plt.ylabel('loss')
     plt.legend()
     plt.show()
```



8.2 Unit testing

Every individual module has been tried against the necessity with some test information. Any image has to undergo two steps for getting final result. One is Preprocessing where the size of the image is changed so that it is suitable to the model and the second one is prediction where the model takes the image and predicts output.

No Mask

CHAPTER - 9

CONCLUSION

9.1 Advantages:

In present situation it is very much important for every one to wear a face mask. So there is a need of system which observes every individual and alerts them to wear a face mask. The Face Mask Detection System can be used at airports to detect travelers without masks. Face data of travelers can be captured in the system at the entrance. If a traveler is found to be without a face mask, their picture is sent to the airport authorities so that they could take quick action. If the person's face is already stored, like the face of an Airport worker, it can send the alert to the worker's phone directly.

Using Face Mask Detection System, Hospitals can monitor if their staff is wearing masks during their shift or not. If any health worker is found without a mask, they will receive a notification with a reminder to wear a mask. Also, if quarantine people who are required to wear a mask, the system can keep an eye and detect if the mask is present or not and send notification automatically or report to the authorities.

9.2 Limitations:

1.It will not detect faces without mask if face is side turned.
2.It cannot detect all faces if there are more people.
3.If resolution of picture is very low it will not detect properly.

9.3 Future Scope

The main limitation of this system is the small datasets available. Provided with large datasets the model can be trained efficiently to give more accurate results. The model can also be trained for different orientations of face and different face masks.To increase the accuracy of model , it can be trained from scratch which requires high computational power.

9.4 Conclusion

The use of face mask is very much important in present situation to stop the spread of deadly virus. This model is optimized to detect the face mask accurately . With further improvements these types of models could be integrated with CCTV or other types cameras to detect and identify people without masks . With the prevailing worldwide situation due to covid19 pandemic , these types of systems would be very supportive for many kind of institutions around the world

REFERENCES

1. P. Viola and M. J. Jones, "Robust real-time face detection", *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137-154, May 2004.
2. A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in Neural Information Processing Systems 25*, pp. 1097-1105, 2012.
3. 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.
4. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *CoRR*, 2014.
5. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, et al., "Going deeper with convolutions", 2015.
6. .K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778, 2016.
7. X. Fu and H. Qu, "Research on semantic segmentation of high-resolution remote sensing image based on full convolutional neural network", 2018 12th International Symposium on Antennas

Websites

1. https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/
2. https://towardsdatascience.com/covid-19-face-mask-detection-using-tensorflow-and-opencv-702dd833515b
3. https://medium.com/@harshilp/real-time-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning-d0744fe048c6
4. https://www.leewayhertz.com/face-mask-detection-system/

5. https://mobisoftinfotech.com/resources/blog/face-mask-detection-system/

6. https://www.ideas2it.com/blogs/face-mask-detector-using-deep-learning-pytorch-and-computer-vision-opencv/

BIBLIOGRAPHY

1.Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Book by Aurélien Géron

2.TensorFlow 1.x Deep Learning Cookbook Book by Antonio Gulli and Amitha kapoor

3.Deep Learning with Python Book by Francois Chollet

4.Object Detection in Low-spatial-resolution Aerial Imagery Using Convolutional Neural Networks Book by Brannon W. Chapman

5.Hands-On Computer Vision with TensorFlow 2 Book by Benjamin Planche and Eliot Andres