FACEMASK MONITORING SYSTEM

A Major Project

Submitted to

Jawaharlal Nehru Technological University, Hyderabad

In partial fulfillment of the requirements for the

award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

BHANDEKAR DINESH (17VE1A0511)

BIKKUMALLA RISHI PRANAY RAJ (17VE1A0512)

KAMAL PATEL (17VE1A0531)

THALLA ASHISH SAI (17UJ1A0522)

Under the guidance of

MRS. PULI SRILATHA

Assistant Professor

SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Affiliated to JNTUH, Approved by A.I.C.T.E and Accredited by NAAC, New Delhi)
Bandlaguda, Beside Indu Aranya, Nagole,
Hyderabad-500068, Ranga Reddy

SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Major Project report on "FACEMASK MONITORING SYSTEM" submitted by DINESH BHANDEKAR,BIKKUMALLA RISHI PRANAY RAJ,KAMAL PATEL,THALLA ASHISH SAI bearing Hall tickets Nos. 17VE1A0511, 17VE1A0512, 17VE1A0531, 17UJ1A0522 in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** from Jawaharlal Nehru Technological University, Kukatpally, Hyderabad for the academic year 2020-2021 is a record of Bonafide work carried out by them under our guidance and Supervision.

Internal Guide	**Head of the Department**
MRS. PULI SRILATHA **Assistant Professor**	**Dr. SHAIK ABDUL NABI** **Professor**
Project Co-Ordinator	**External Examiner**
Dr. M. JAYA RAM **Professor**	

SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **BHANDEKAR DINESH, BIKKUMALLA RISHI PRANAY RAJ, KAMAL PATEL, THALLA ASHISH SAI** bearing Hall ticket Nos. **17VE1A0511, 17VE1A0512, 17VE1A0531, 17UJ1A0522** hereby declare that the Major Project titled "FACEMASK MONITORING SYSTEM" done by us under the guidance of **Mrs. PULI. SRILATHA**, which is submitted in the partial fulfillment of the requirement for the award of the B-Tech in **Computer Science and Engineering** Branch at **Sreyas Institute of Engineering & Technology** for Jawaharlal Nehru Technological University, Hyderabad is my original work.

BHANDEKAR DINESH (17VE1A0511)
BIKKUMALLA RISHI PRANAY RAJ (17VE1A0512)
KAMAL PATEL (17VE1A0531)
THALLA ASHISH SAI (17UJ1A0522)

ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without mention of the people who made it possible through their guidance and encouragement crowns all the efforts with success.

We take this opportunity to acknowledge with thanks and deep sense of gratitude to **Mrs. PULI. SRILATHA, Assistant Professor, Department of Computer Science and Engineering** for her constant encouragement and valuable guidance during the Project work

A Special note of Thanks to **Dr. SHAIK ABDUL NABI, Head of the Department** and **Dr. M. JAYA RAM, Project Co-Ordinator** who has been a source of Continuous motivation and support. They had taken time and effort to guide and correct me all through the span of this work.

We owe very much to the **Department Faculty, Principal** and the **Management** who made my term at Sreyas a Stepping stone for my career. We treasure every moment we had spent in the college.

Last but not least, my heartiest gratitude to my parents and friends for their continuous encouragement and blessings. Without their support this work would not have been possible.

BIKKUMALLA RISHI PRANAY RAJ (17VE1A0512)

BHANDEKAR DINESH (17VE1A0511)

KAMAL PATEL (17VE1A0531)

THALLA ASHISH SAI (17UJ1A0522)

ABSTRACT

The project is about detecting whether a person is wearing a mask or not which is run on the host network using the camera interface using OpenCV python library and for the API purpose we are using Swagger UI. Face Mask Detection system built with OpenCV, Keras/TensorFlow using Computer Vision concepts in order to detect face masks in static images as well as in real-time video streams. The model is accurate, and since we used the MobileNetV2 architecture.

We are using Swagger UI because it allows anyone either their own development team or consumers to visualize and interact with the API's resources without having any of the implementation logic in place. When this application is run on the host network it automatically captures images every one minute and sends the data in the form of JSON data which interprets how many members are wearing masks or not. We can also send images as an input to this application which sends output as labels with either mask or no mask. It can be run on any device since it's an API.

In the present scenario due to Covid-19, there are no efficient face mask detection applications which are now in high demand for transportation means, densely populated areas, residential districts, large-scale manufacturers and other enterprises to ensure safety. It is very useful to detect masks on the faces on real time scenarios.

KEYWORDS: API, OpenCV, AI, Deep learning, MobileNetV2, COVID-19.

TABLE OF CONTENTS

LIST OF FIGURES

<u>LIST OF TABLES</u>

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Face Mask Detection system built with OpenCV, Keras/TensorFlow using Deep Learning and Computer Vision concepts in order to detect face masks in static images as well as in real-time video streams. In the present scenario due to Covid-19, there is no efficient face mask detection applications which are now in high demand for transportation means, densely populated areas, residential districts, large-scale manufacturers and other enterprises to ensure safety. Also, the absence of large datasets of ‘with mask’ images have made this task more cumbersome and challenging.

An API is a set of definitions and protocols for building and integrating application software. It’s sometimes referred to as a contract between an information provider and an information user—establishing the content required from the consumer (the call) and the content required by the producer (the response).In other words, if you want to interact with a computer or system to retrieve information or perform a function, an API helps you communicate what you want to that system so it can understand and fulfil the request. You can think of an API as a mediator between the users or clients and the resources or web services they want to get. It’s also a way for an organization to share resources and information while maintaining security, control, and authentication—determining who gets access to what.

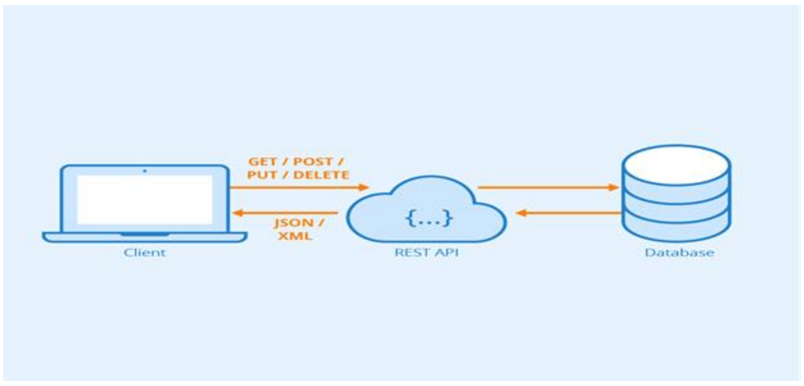


Fig 1: REST API

1.2 SCOPE

The main scope of this project is to detect whether the person is wearing mask or not through an API. We can integrate this API with the CCTVs to detect whether a person is wearing mask or not at hospitals, airports, offices., etc.

1.3 EXISTING SYSTEM

In the past few years, face recognition owned significant consideration and appreciated as one of the most promising applications in the field of image analysis. Face detection can consider a substantial part of face recognition operations. According to its strength to focus computational resources on the section of an image holding a face. The method of face detection in pictures is complicated because of variability present across human faces such as pose, expression, position and orientation, skin colour, the presence of glasses or facial hair, differences in camera gain, lighting conditions, and image resolution.

There were only limited libraries to develop an API which is user friendly and takes less time to create an API. And to monitor whether the person is wearing a mask or not one has to monitor the humans continuously in the crowded areas like shopping malls, theatres, large gatherings etc., So both human effort and time are needed.

1.4 PROPOSED SYSTEM

With the level of advancements in the technology we can see that may libraries have been developed. So, with those libraries we can create many applications and API's which are user friendly. In this we'll be developing an interface for an enterprise which is integrated with CCTV's so that it detects humans with masks and without masks.

The algorithm is trained to capture facial features in real-time video streams and images and recognize whether everyone's wearing a protective mask with a 99.98% accuracy rate. Equally effective for both individual and group detection, our face mask detection system can supplement or reduce the number of enforcement agents on the ground.

In this the admin or the host has to run the application so that the client can monitor using an API. Client can either give images or videos as input for the API. Videos can be both static and dynamic. After performing the initial analysis, the system classifies every person as "wearing a mask" or flags as "not wearing a mask" and sends an instant alert, so you can take further action — dispatch a public audio announcement, send a custom message to a digital screen, or a personalized message to the person’s phone. Proactively manage and correct visitors' behavior while remaining compliant with privacy regulations.

CHAPTER 2

LITERATURE SURVEY

Title: “An Automatic System to Monitor the Physical Distance and Face Mask Wearing of Construction Workers in COVID-19 Pandemic.”

Author: Moein Razavi, Hamed Alikhani, Vahid Janfaza, Benyamin Sadeghi, Ehsan Alikhani

Publication Date: 5/01/2021

Description:

The COVID-19 pandemic has caused many shutdowns in different industries around the world. Sectors such as infrastructure construction and maintenance projects have not been suspended due to their significant effect on people's routine life. In such projects, workers work close together that makes a high risk of infection. The World Health Organization recommends wearing a face mask and practicing physical distancing to mitigate the virus's spread. This paper developed a computer vision system to automatically detect the violation of face mask wearing and physical distancing among construction workers to assure their safety on infrastructure projects during the pandemic. For the face mask detection, the paper collected and annotated 1,000 images, including different types of face mask wearing, and added them to a pre-existing face mask dataset to develop a dataset of 1,853 images. Then trained and tested multiple TensorFlow state-of-the-art object detection models on the face mask dataset and chose the Faster R-CNN Inception ResNet V2 network that yielded the accuracy of 99.8%. For physical distance detection, the paper employed the Faster R-CNN Inception V2 to detect people. A transformation matrix was used to eliminate the camera angle's effect on the object distances on the image. The Euclidian distance used the pixels of the transformed image to compute the actual distance between people. A threshold of six feet was considered to capture physical distance violation. The paper also used transfer learning for training the model. The final model was applied on four videos of road maintenance projects in Houston, TX, that effectively detected the face mask and physical distance. We recommend that construction owners use the proposed system to enhance construction workers' safety in the pandemic situation.

CHAPTER 3

SYSTEM ANALYSIS

3.1 MODULES AND ITS DESCRIPTION

REST API

A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding.

An API is a set of definitions and protocols for building and integrating application software. It’s sometimes referred to as a contract between an information provider and an information user—establishing the content required from the consumer (the call) and the content required by the producer (the response).

In other words, if you want to interact with a computer or system to retrieve information or perform a function, an API helps you communicate what you want to that system so it can understand and fulfil the request.

You can think of an API as a mediator between the users or clients and the resources or web services they want to get. It’s also a way for an organization to share resources and information while maintaining security, control, and authentication—determining who gets access to what.

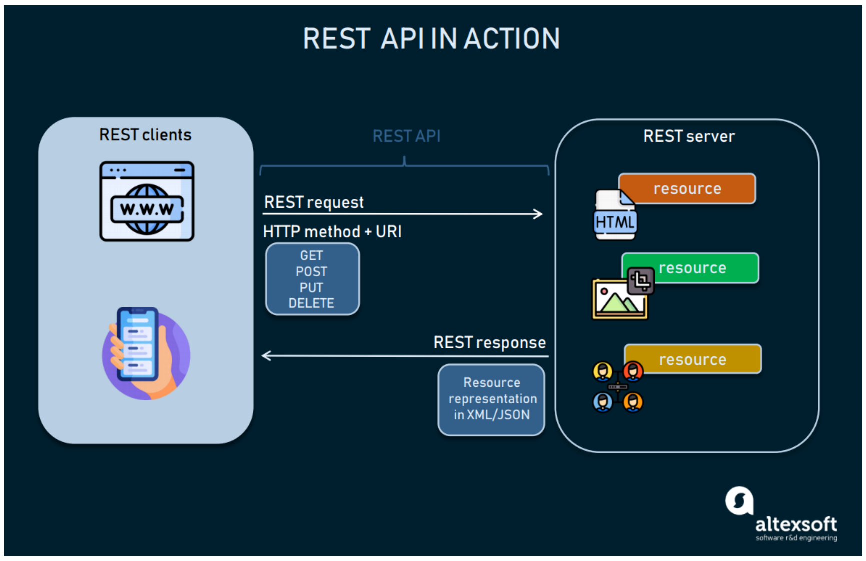


Fig 2: REST API in action

Another advantage of an API is that you don’t have to know the specifics of caching—how your resource is retrieved or where it comes from.

In order for an API to be considered RESTful, it has to conform to these criteria:

- A client-server architecture made up of clients, servers, and resources, with requests managed through HTTP.
- Stateless client-server communication, meaning no client information is stored between get requests and each request is separate and unconnected.
- Cacheable data that streamlines client-server interactions.
- A uniform interface between components so that information is transferred in a standard form. This requires that:
    - resources requested are identifiable and separate from the representations sent to the client.
    - resources can be manipulated by the client via the representation they receive because the representation contains enough information to do so.
    - self-descriptive messages returned to the client have enough information to describe how the client should process it.
    - hypertext/hypermedia is available, meaning that after accessing a resource the client should be able to use hyperlinks to find all other currently available actions they can take.
- A layered system that organizes each type of server (those responsible for security, load-balancing, etc.) involved the retrieval of requested information into hierarchies, invisible to the client.
- Code-on-demand (optional): the ability to send executable code from the server to the client when requested, extending client functionality.

FAST API

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints.

The key features are:

- Fast
- Fast to code
- Fewer bugs

- Intuitive
- Easy and short
- Robust
- Standards-based

INSTALLATION

```
$ pip install fastapi
```

You will also need an ASGI server, for production such as uvicorn.

```
$ pip install uvicorn
```

EXAMPLE

Create a file main.py with:

```
from typing import Optional
from fastapi import FastAPI
app = FastAPI()
@app.get("/")
def read_root():
    return {"Hello": "World"}
@app.get("/items/{item_id}")
def read_item(item_id: int, q: Optional[str] = None):
    return {"item_id": item_id, "q": q}
```

You will see the JSON response as:

```
{"item_id": 5, "q": "somequery"}
```

You already created an API that:

Receives HTTP requests in the paths / and /items/{item_id}.

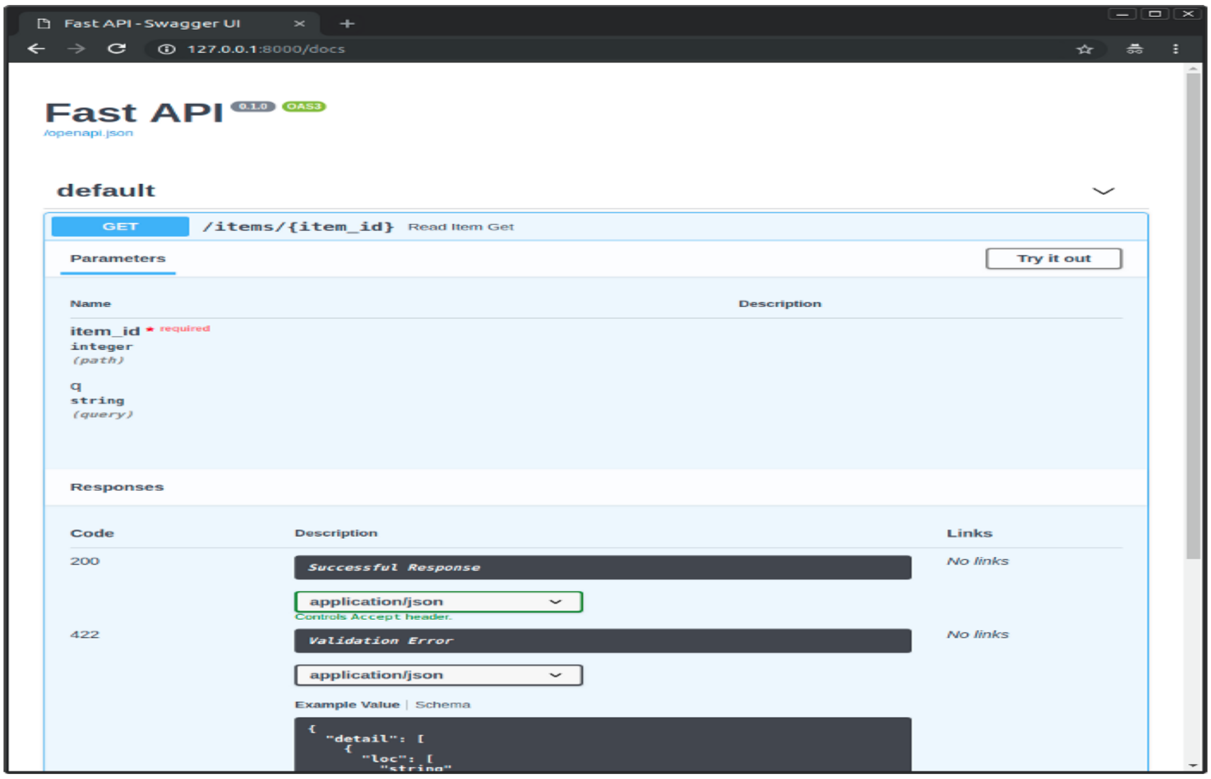
Both paths take GET operations (also known as HTTP methods).

The path /items/{item_id} has a path parameter item_id that should be an int.

The path /items/{item_id} has an optional str query parameter q.

INTERACTIVE API docs

You will see the automatic interactive API documentation (provided by Swagger UI):


Fast API - Swagger UI

127.0.0.1:8000/docs

Fast API 0.1.0 OAS3

/openapi.json

default

GET /items/{item_id} Read Item Get

Parameters Try it out

Name	Description
item_id * required integer *(path)*	
q string *(query)*	

Responses

Code	Description	Links
200	*Successful Response* application/json Controls Accept header.	*No links*
422	*Validation Error* application/json Example Value | Schema	*No links*

```
{
  "detail": [
    {
      "loc": [
        "string"
```

Fig 3: Fast API

STARLETTE LIBRARY

Starlette is a lightweight ASGI framework/toolkit, which is ideal for building high performance asyncio services.

It is production-ready, and gives you the following:

- Seriously impressive performance.
- WebSocket support.
- GraphQL support.
- In-process background tasks.
- Startup and shutdown events.
- Test client built on requests.
- CORS, GZip, Static Files, Streaming responses.
- Session and Cookie support.
- 100% test coverage.
- 100% type annotated codebase.
- Zero hard dependencies.

REQUIREMENTS

Python 3.6+

INSTALLATION:

```
pip3 install starlette
```

Example:

```
from starlette import Response
class App:
    def __init__ (self, scope):
        self. scope = scope
    async def __call__(self, receive, send):
        response = Response ('Hello, world!', media_type='text/plain')
        await response (receive, send)
```

You can run the application with any ASGI server, including uvicorn, daphne, or hypercorn

SWAGGER UI

Swagger UI is a collection of HTML, JavaScript, and CSS assets that dynamically generate beautiful documentation from a Swagger-compliant API.

Swagger UI allows anyone — be it your development team or your end consumers — to visualize and interact with the API's resources without having any of the implementation logic in place. It's automatically generated from your OpenAPI (formerly known as Swagger) Specification, with the visual documentation making it easy for back end implementation and client-side consumption.

KEY FEATURES

- Dependency Free: The UI works in any development environment, be it locally or in the web.
- Human Friendly: Allow end developers to effortlessly interact and try out every single operation your API exposes for easy consumption
- Easy to Navigate: Quickly find and work with resources and endpoints with neatly categorized documentation

- All Browser Support: Cater to every possible scenario with Swagger UI working in all major browsers
- Fully Customizable: Style and tweak your Swagger UI the way you want with full source code access
- Complete OAS Support: Visualize APIs defined in Swagger 2.0 or OAS 3.0

Swagger UI is just one open source project in the thousands that exist in the Swagger ecosystem. The source code is publicly hosted on GitHub, and you can start contributing to the open source Swagger UI project.

FACEMASK DETECTION

Face Mask Detection system built with OpenCV, Keras/TensorFlow using Deep Learning and Computer Vision concepts in order to detect face masks in static images as well as in real-time video streams. In the present scenario due to Covid-19, there is no efficient face mask detection applications which are now in high demand for transportation means, densely populated areas, residential districts, large-scale manufacturers and other enterprises to ensure safety. Also, the absence of large datasets of ‘with mask’ images has made this task more cumbersome and challenging.

TWO PHASE FACEMASK DETECTOR

In order to train a custom face mask detector, we need to break our project into two distinct phases, each with its own respective sub-steps (as shown by Figure 2 below):

1.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

2.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.

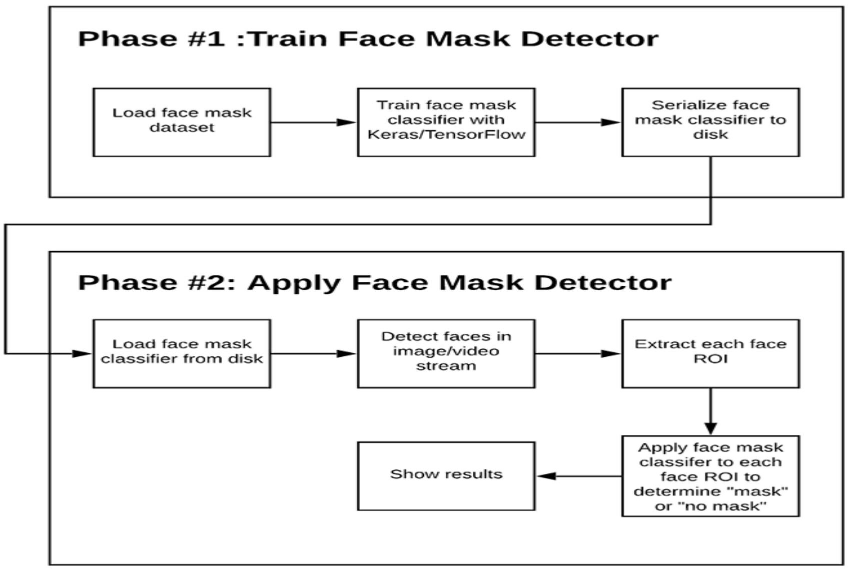

Phase #1 :Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifer to each face ROI to determine "mask" or "no mask"
Show results

Fig 4: Two face mask detector

CAFFE BASED FACE DETECTOR

Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR) and by community contributors. Yangqing Jia created the project during his PhD at UC Berkeley. Caffe is released under the BSD 2-Clause license.

It is a Caffe model which is based on the Single Shot-Multibox Detector (SSD) and uses ResNet-10 architecture as its backbone. It was introduced post OpenCV 3.3 in its deep neural network module. There is also a quantized TensorFlow version that can be used but we will use the Caffe Model.

Deep networks are compositional models that are naturally represented as a collection of inter-connected layers that work on chunks of data. Caffe defines a net layer-by-layer in its own model schema. The network defines the entire model bottom-to-top from input data to loss. As data and derivatives flow through the network in the forward and backward passes Caffe stores, communicates, and manipulates the information as blobs: the blob is the standard array and unified memory interface for the framework. The layer comes next as the foundation of both model and computation. The net follows as the collection and connection of layers. The details of blob describe how information is stored and communicated in and across layers and nets.

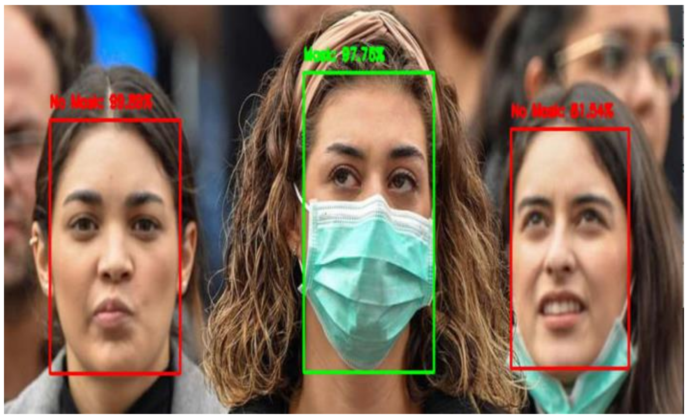


Fig 5: Real time face mask detector

Applications. Caffe is being used in academic research projects, startup prototypes, and even large-scale industrial applications in vision, speech, and multimedia. Yahoo! has also integrated caffe with Apache Spark to create CaffeOnSpark, a distributed deep learning framework.

CONVOLUTIONAL NEURAL NETWORKS

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image) and height (as image generally have red, green, and blue channels).

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlaps to cover the entire visual area.

LAYERS OF CNN

Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. In a regular Neural Network there are three types of layers:

Input Layers: It's the layer in which we give input to our model. The number of neurons in this layer is equal to total number of features in our data (number of pixels in case of an image).

Hidden Layer: The input from Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by addition of learnable biases followed by activation function which makes the network nonlinear.

Output Layer: The output from the hidden layer is then fed into a logistic function like sigmoid or SoftMax which converts the output of each class into probability score of each class.

The data is then fed into the model and output from each layer is obtained this step is called feedforward, we then calculate the error using an error function, some common error functions are cross entropy, square loss error etc. After that, we back propagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss.

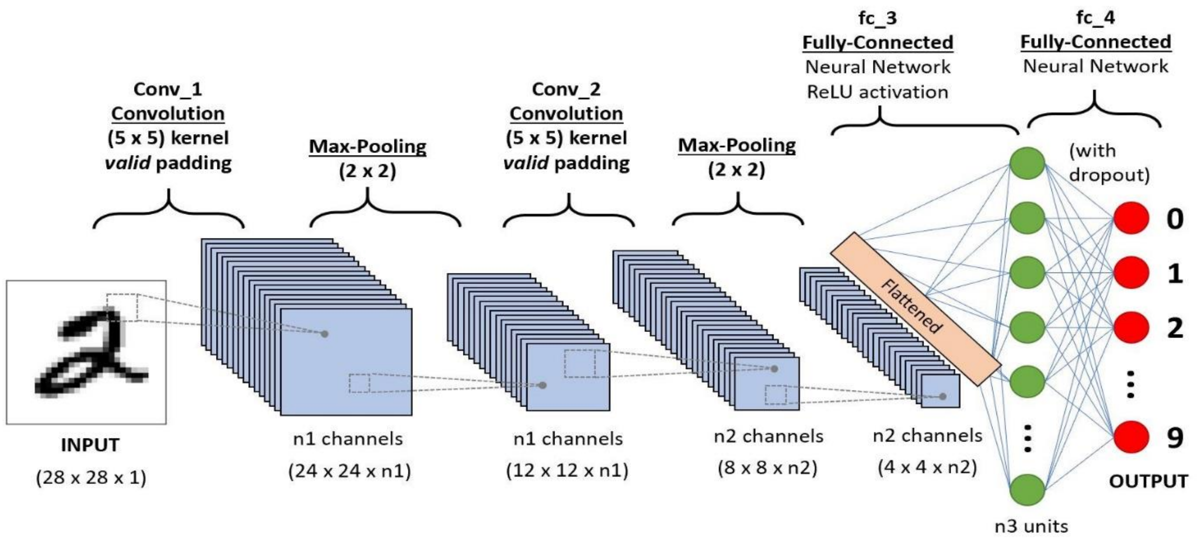


Fig 6: A CNN sequence to classify handwritten digits

ARCHITECTURE

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically, this includes a layer that does multiplication or other dot product, and its activation function is commonly ReLU. This is followed by other convolution layers such as pooling layers, fully connected layers and normalization layers.

Convolutional layers

In a CNN, the input is a tensor with shape (number of images) x (image height) x (image width) x (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map height) x (feature map width) x (feature map channels). A convolutional layer within a neural network should have the following attributes:

- Convolutional filters/kernels defined by a width and height (hyper-parameters).
- The number of input channels and output channels (hyper-parameter).
- The depth of the convolution kernel/filter (the input channels) must equal the number channels (depth) of the input feature map.
- Convolution operation specific hyperparameters like padding size and stride.

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features and classify data, this architecture is impractical for images. It would require a very high number of neurons, even in a shallow architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable.

For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for each neuron in the second layer. Instead, convolution reduces the number of free parameters, allowing the network to be deeper. For example, regardless of image size, tiling 5 x 5 region, each with the same shared weights, requires only 25 learnable parameters. Using regularized weights over fewer parameters avoids the vanishing gradient and exploding gradient problems seen during backpropagation in traditional neural networks.

Pooling layers

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. There are two common types of pooling: max and average. Max pooling uses the maximum value of each cluster of neurons at the prior layer, while average pooling instead uses the average value.

Fully connected layers

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

Receptive field

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every neuron of the previous layer. In a convolutional layer, each neuron receives input from only a restricted area of the previous layer called the neuron's receptive field. Typically, the area is a square (e.g., 5 by 5 neurons). (So, in a fully connected layer, the receptive field is the entire previous layer.) Thus, in each convolutional layer, each neuron takes input from a larger area of pixels in the input image than previous layers. This is due to applying the convolution over and over, which considers the value of a pixel and its surrounding pixels.

Weights

Each neuron in a neural network computes an output value by applying a specific function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning consists of iteratively adjusting these biases and weights.

The vector of weights and the bias are called filters and represent particular features of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces memory footprint because a single bias and a single vector of weights are used across all receptive fields sharing that filter, as opposed to each receptive field having its own bias and vector weighting.

FEATURE EXTRACTION

In order to carry out image recognition/classification, the neural network must carry out feature extraction. Features are the elements of the data that you care about which will be fed through the network. In the specific case of image recognition, the features are the groups of pixels, like edges and points, of an object that the network will analyse for patterns.

Feature recognition (or feature extraction) is the process of pulling the relevant features out from an input image so that these features can be analysed. Many images contain annotations or metadata about the image that helps the network find the relevant features.

CLASSIFICATION

Image Recognition refers to the task of inputting an image into a neural network and having it output some kind of label for that image. The label that the network outputs will correspond to a pre-defined class. There can be multiple classes that the image can be labelled as, or just one. If there is a single class, the term "recognition" is often applied, whereas a multi-class recognition task is often called "classification".

A subset of image classification is object detection, where specific instances of objects are identified as belonging to a certain class like animals, cars, or people.

HOW NEURAL NETWORKS LEARN TO RECOGNIZE IMAGES

Getting an intuition of how a neural network recognizes images will help you when you are implementing a neural network model, so let's briefly explore the image recognition process in the next few sections.

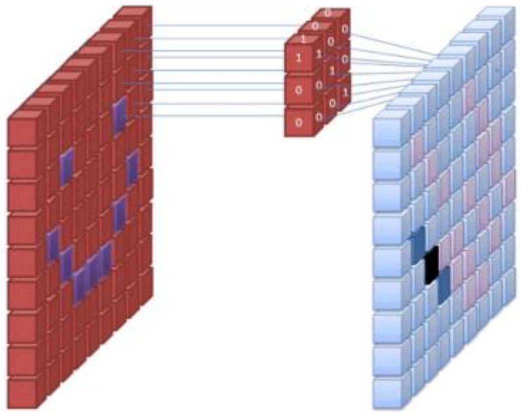


Fig 7: Feature Extraction with Filters

The first layer of a neural network takes in all the pixels within an image. After all the data has been fed into the network, different filters are applied to the image, which forms representations of different parts of the image. This is feature extraction and it creates "feature maps".

This process of extracting features from an image is accomplished with a "convolutional layer", and convolution is simply forming a representation of part of an image. It is from this convolution concept that we get the term Convolutional Neural Network (CNN), the type of neural network most commonly used in image classification/recognition.

If you want to visualize how creating feature maps works, think about shining a flashlight over a picture in a dark room. As you slide the beam over the picture you are learning about features of the image. A filter is what the network uses to form a representation of the image, and in this metaphor, the light from the flashlight is the filter.

The width of your flashlight's beam controls how much of the image you examine at one time, and neural networks have a similar parameter, the filter size. Filter size affects how much of the image, how many pixels, are being examined at one time. A common filter size used in CNNs is 3, and this covers both height and width, so the filter examines a 3 x 3 area of pixels.

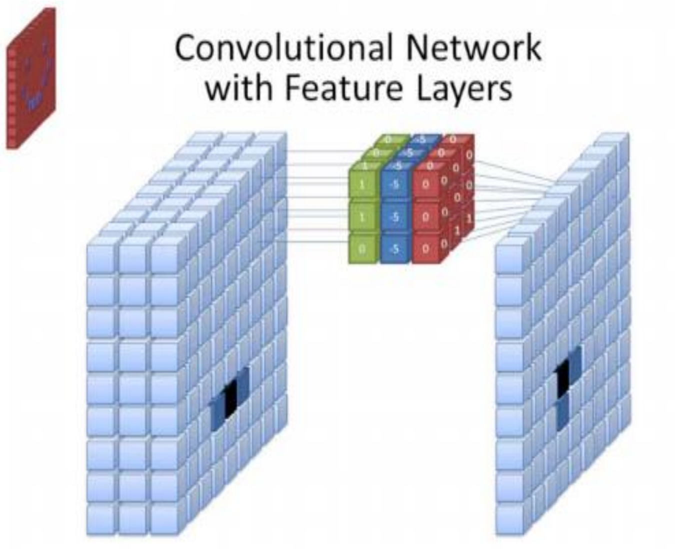


Fig 8: Convolutional Network with feature layers

While the filter size covers the height and width of the filter, the filter's depth must also be specified.

HOW DOES A 2D IMAGE HAVE DEPTH?

Digital images are rendered as height, width, and some RGB value that defines the pixel's colours, so the "depth" that is being tracked is the number of color channels the image has. Grayscale (non-color) images only have 1 color channel while color images have 3 depth channels.

All of this means that for a filter of size 3 applied to a full-color image, the dimensions of that filter will be 3 x 3 x 3. For every pixel covered by that filter, the network multiplies the filter values with the values in the pixels themselves to get a numerical representation of that pixel. This process is then done for the entire image to achieve a complete representation. The filter is moved across the rest of the image according to a parameter called "stride", which defines how many pixels the filter is to be moved by after it calculates the value in its current position. A conventional stride size for a CNN is 2.

The end result of all this calculation is a feature map. This process is typically done with more than one filter, which helps preserve the complexity of the image.

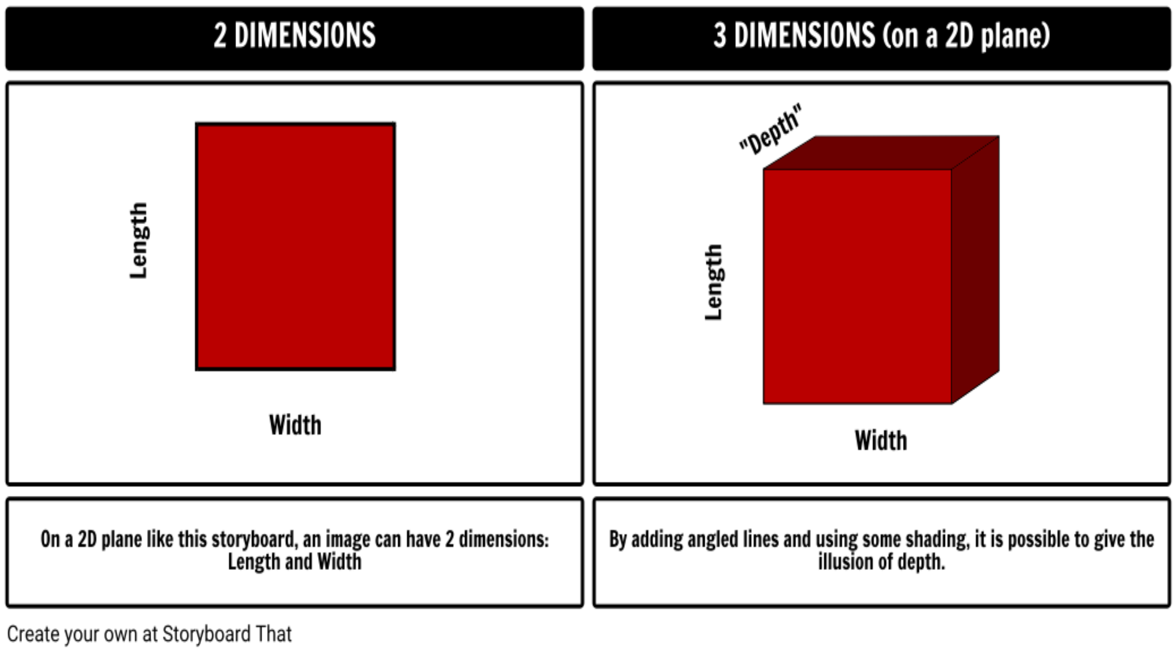

2 DIMENSIONS
3 DIMENSIONS (on a 2D plane)
Length
Width
"Depth"
Length
Width
On a 2D plane like this storyboard, an image can have 2 dimensions: Length and Width
By adding angled lines and using some shading, it is possible to give the illusion of depth.
Create your own at Storyboard That

Fig 9: 2D Image depth and 3D Image depth

SYSTEM DESIGN

Input in this case is image and output are the names of object which is nothing but label.

If the image is labelled data then CNN comes into picture. It identifies the key features which is nothing but feature extraction and based on that classification takes place by comparing with the pretrained dataset which finally produces the output as class label.

If the image is unlabelled data then it compares with pretrained data and training is done until it achieves good accuracy.

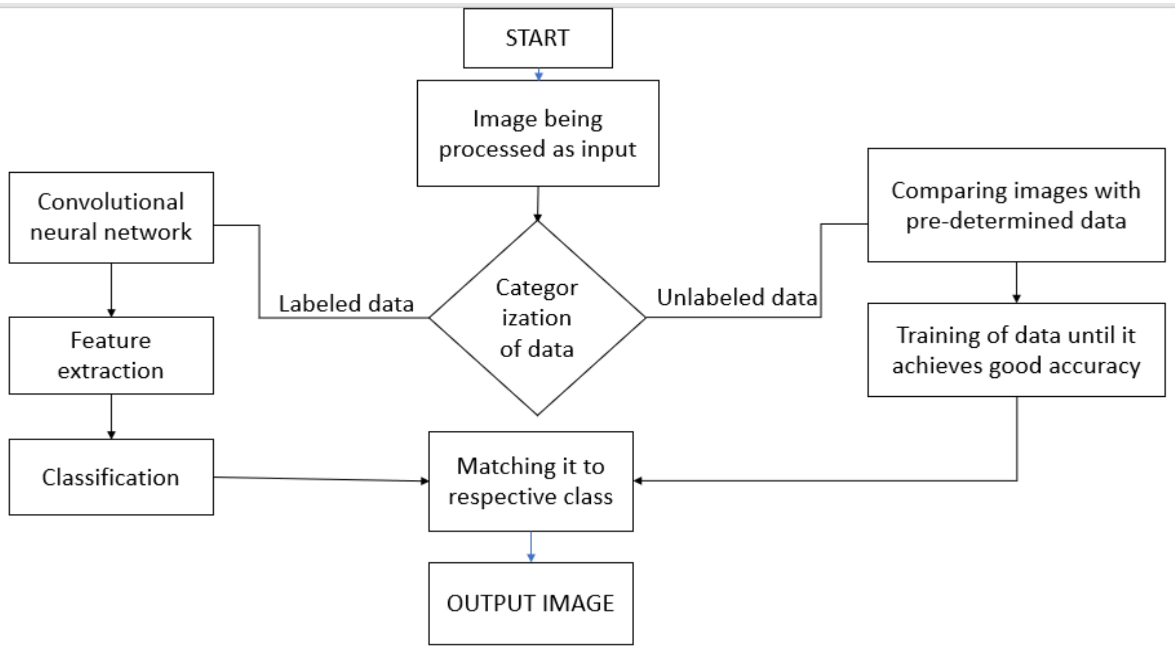

START
Image being processed as input
Convolutional neural network
Labeled data
Categorization of data
Unlabeled data
Comparing images with pre-determined data
Feature extraction
Training of data until it achieves good accuracy
Classification
Matching it to respective class
OUTPUT IMAGE

Fig 10: System Design

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.

KEY FEATURES

- Iterate at the speed of thought
- Exascale machine learning
- Deploy anywhere

- A vast ecosystem
- An accessible superpower

INSTALLATION

pip install keras

STEPS

1. Load Data.
2. Define Keras Model.
3. Compile Keras Model.
4. Fit Keras Model.
5. Evaluate Keras Model.
6. Tie It All Together.
7. Make Predictions

USE

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

H5 is a file format to store structured data, it's not a model by itself. Keras saves models in this format as it can easily store the weights and model configuration in a single file.

This short introduction uses Keras to:

1. Build a neural network that classifies images.

2. Train this neural network.

3. And, finally, evaluate the accuracy of the model.

Implementation of the Keras API meant to be a high-level API for TensorFlow.

MOBILENET

MobileNet model is designed to be used in mobile applications, and it is TensorFlow's first mobile computer vision model.

MobileNet uses depthwise separable convolutions. It significantly reduces the number of parameters when compared to the network with regular convolutions with the same depth in the nets. This results in lightweight deep neural networks.

A depthwise separable convolution is made from two operations.

1. Depthwise convolution.
2. Pointwise convolution.

MobileNet is a class of CNN that was open-sourced by Google, and therefore, this gives us an excellent starting point for training our classifiers that are insanely small and insanely fast.

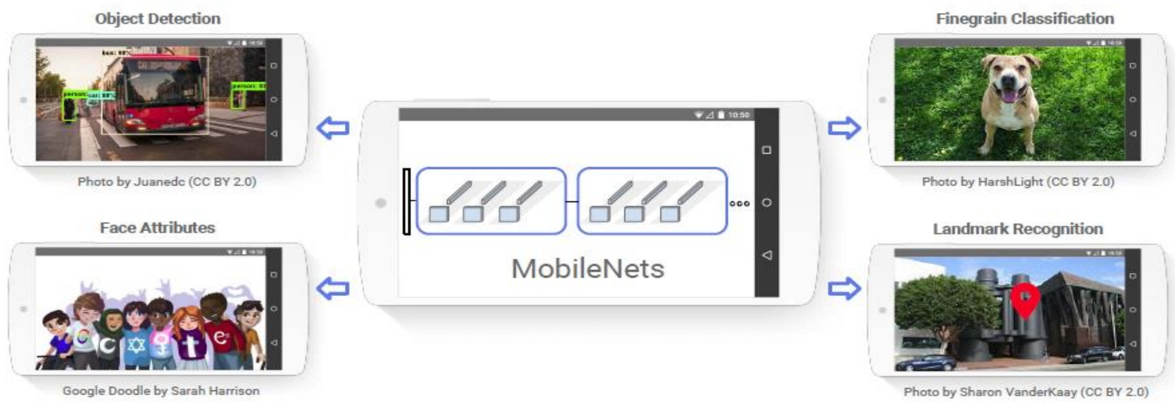

Object Detection
Photo by Juanedc (CC BY 2.0)
Face Attributes
Google Doodle by Sarah Harrison
MobileNets
Finegrain Classification
Photo by HarshLight (CC BY 2.0)
Landmark Recognition
Photo by Sharon VanderKaay (CC BY 2.0)

Fig 11: When MobileNet is Applied to Real Life

The speed and power consumption of the network is proportional to the number of MACs (Multiply-Accumulates) which is a measure of the number of fused Multiplication and Addition operations.

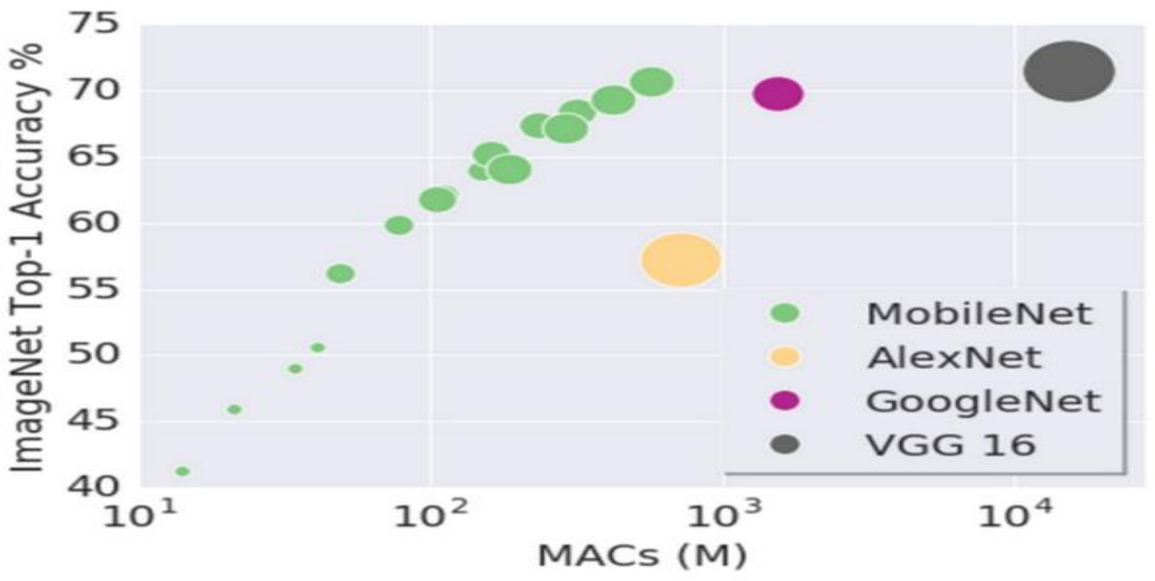


Fig 12: Architecture of MobileNet

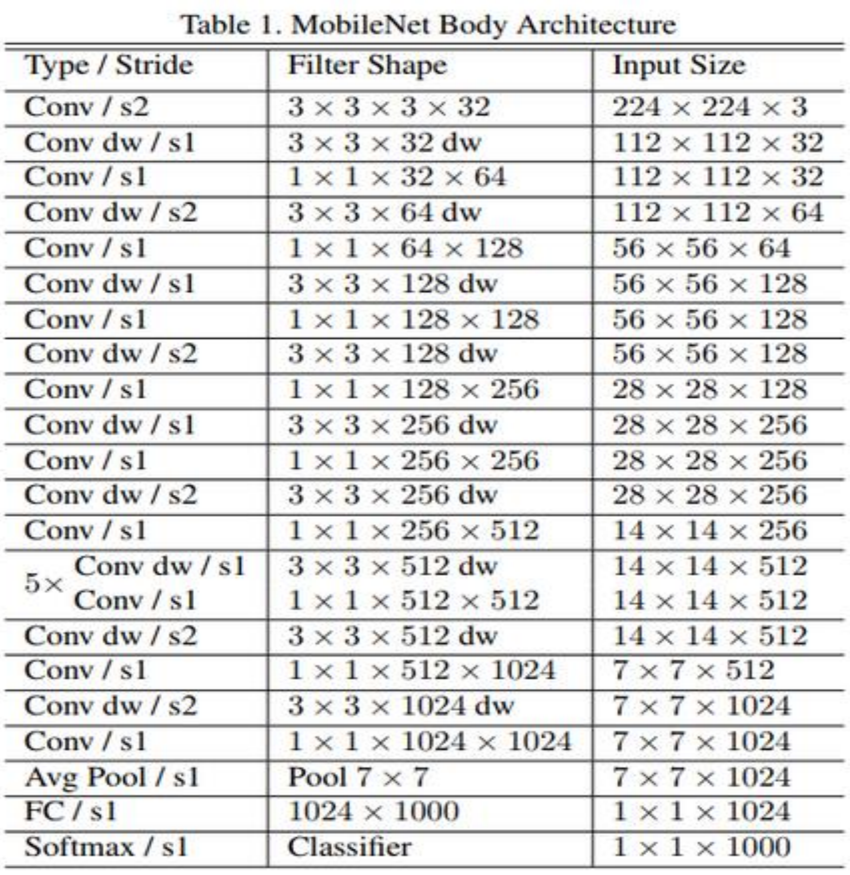


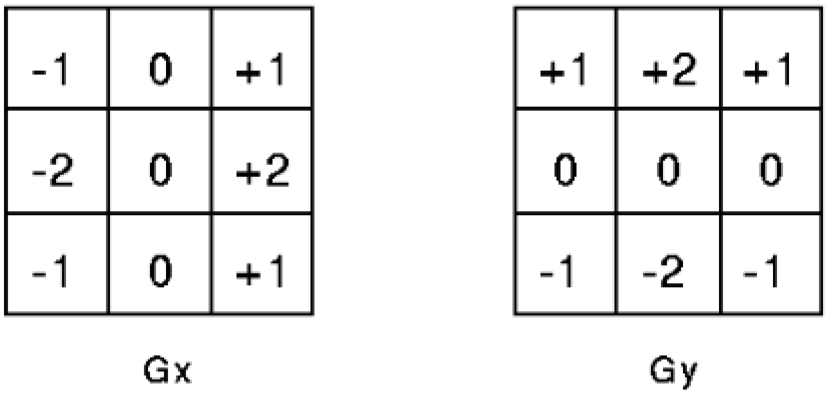
Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Fig 13: MobileNet Layers

1)Depthwise Separable Convolution

This convolution originated from the idea that a filter's depth and spatial dimension can be separated- thus, the name separable. Let us take the example of Sobel filter, used in image processing to detect edges.



-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Fig 14: Sobel Filter Gx for the vertical edge, Gy for horizontal edge detection

You can separate the height and width dimensions of these filters. Gx filter can be viewed as a matrix product of [1 2 1] transpose with [-1 0 1].

We notice that the filter had disguised itself. It shows it had nine parameters, but it has 6. This has been possible because of the separation of its height and width dimensions.

The same idea applied to separate depth dimension from horizontal (width*height) gives us depth-wise separable convolution where we perform depth-wise convolution. After that, we use a 1*1 filter to cover the depth dimension.

One thing to notice is how much parameters are reduced by this convolution to output the same no. of channels. To produce one channel, we need 3*3*3 parameters to perform depth-wise convolution and 1*3 parameters to perform further convolution in-depth dimension.

But If we need three output channels, we only need 31*3 depth filter, giving us a total of 36 (= 27 +9) parameters while for the same no. of output channels in regular convolution, we need 33*3*3 filters giving us a total of 81 parameters.

Depthwise separable convolution is a depthwise convolution followed by a pointwise convolution as follows:

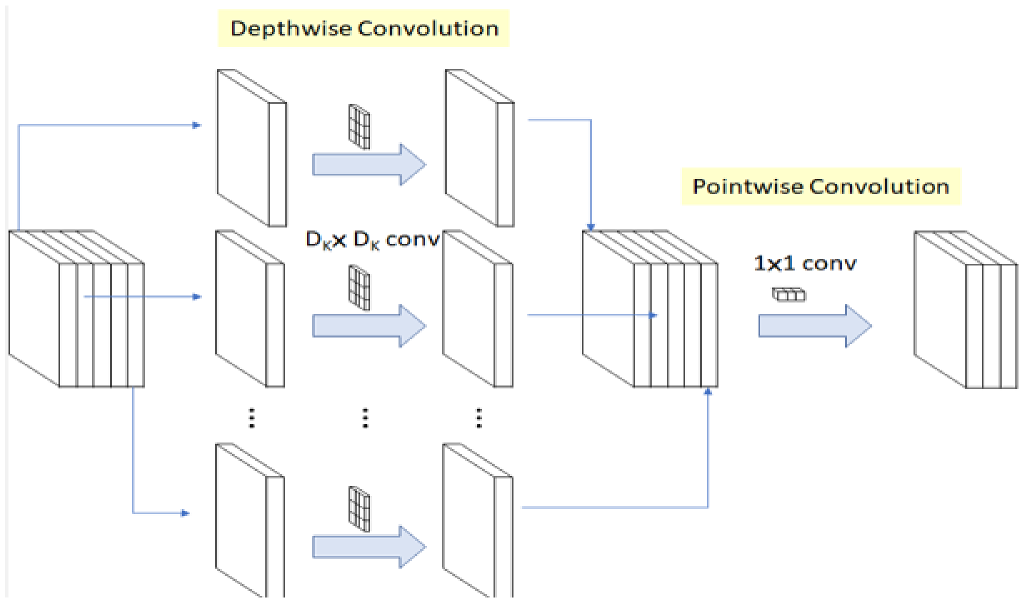


Fig 15: Depthwise Separable Convolution

1. **Depthwise convolution** is the **channel-wise DK×DK spatial convolution**. Suppose in the figure above, and we have five channels; then, we will have 5 DK×DK spatial convolutions.

2. **Pointwise convolution** is the **1×1 convolution** to change the dimension.

3. **Depthwise convolution** It is a map of a single convolution on each input channel separately. Therefore, its number of output channels is the same as the number of the input channels. Its computational cost is $Df^2 * M * Dk^2$.

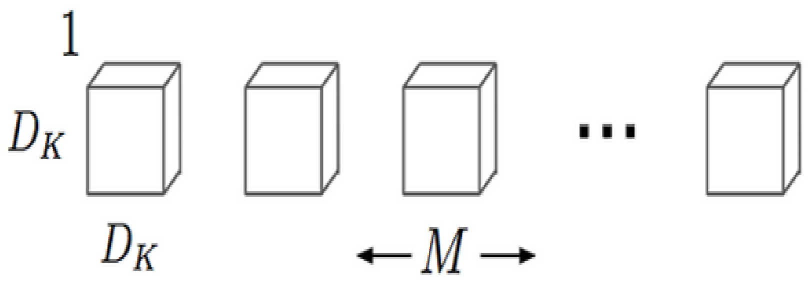


Fig 16: Depthwise convolution.

2)POINTWISE CONVOLUTION

Convolution with a kernel size of 1x1 that simply combines the features created by the depthwise convolution. Its computational cost is **$M * N * Df^2$**.

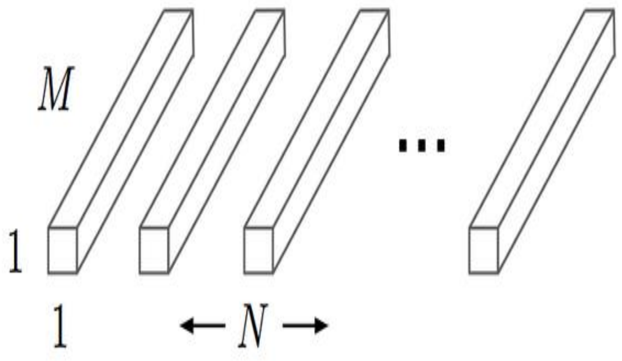


Fig 17: Pointwise Convolution

Difference between Standard Convolution and Depthwise separable convolution

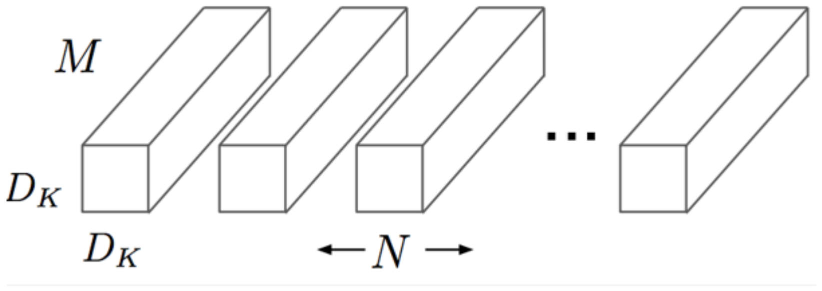


Fig 18: Standard Convolution.

The main difference between MobileNet architecture and a traditional CNN instead of a single 3x3 convolution layer followed by the batch norm and ReLU. Mobile Nets split the convolution into a 3x3 depth-wise conv and a 1x1 pointwise conv, as shown in the figure.

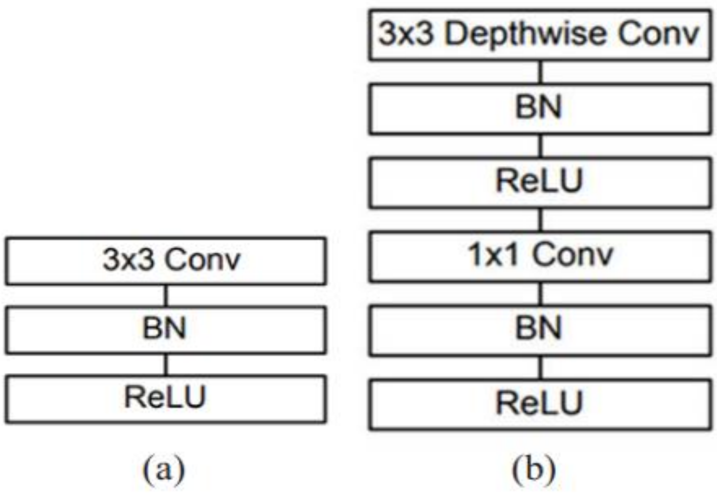


Fig 19: (a) Standard convolutional layer with batch normalization and ReLU. (b) Depth-wise separable convolution with depth-wise and pointwise layers followed by batch normalization and ReLU.

MOBILENETV2

MobileNetV2 is a significant improvement over MobileNetV1 and pushes the state of the art for mobile visual recognition including classification, object detection and semantic segmentation. MobileNetV2 is released as part of TensorFlow-Slim Image Classification Library, or you can start exploring MobileNetV2 right away in Colaboratory. Alternately, you can download the notebook and explore it locally using Jupyter. MobileNetV2 is also available as modules on TF-Hub, and pretrained checkpoints can be found on GitHub.

MobileNetV2 builds upon the ideas from MobileNetV1 [1], using depthwise separable convolution as efficient building blocks. However, V2 introduces two new features to the architecture: 1) linear bottlenecks between the layers, and 2) shortcut connections between the bottlenecks1. The basic structure is shown below.

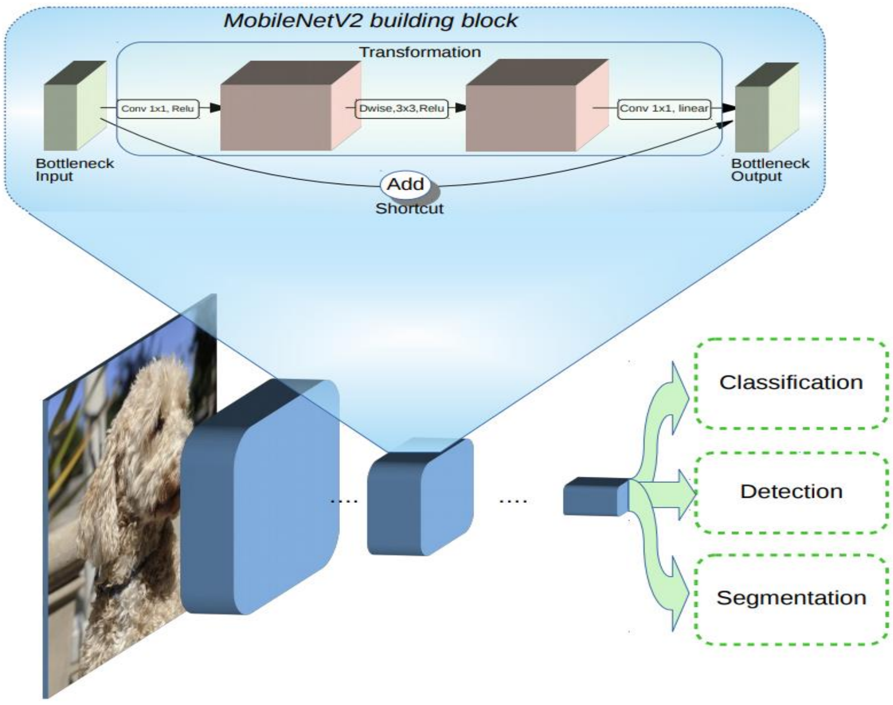


Fig 20: Overview of MobileNetV2 Architecture. Blue blocks represent composite convolutional building blocks as shown above.

The intuition is that the bottlenecks encode the model's intermediate inputs and outputs while the inner layer encapsulates the model's ability to transform from lower-level concepts such as pixels to higher level descriptors such as image categories. Finally, as with traditional residual connections, shortcuts enable faster training and better accuracy. You can learn more about the technical details in our paper, "MobileNet V2: Inverted Residuals and Linear Bottlenecks".

How does it compare to the first generation of MobileNets?

Overall, the MobileNetV2 models are faster for the same accuracy across the entire latency spectrum. In particular, the new models use 2x fewer operations, need 30% fewer parameters and are about 30-40% faster on a Google Pixel phone than MobileNetV1 models, all while achieving higher accuracy.

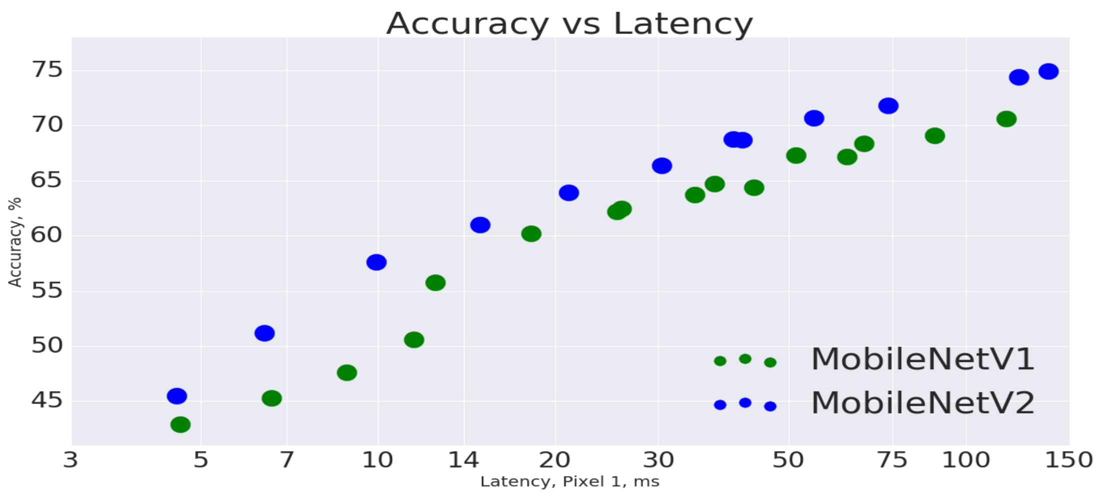


Fig 21: Accuracy

MobileNetV2 improves speed (reduced latency) and increased ImageNet Top 1 accuracy

MobileNetV2 is a very effective feature extractor for object detection and segmentation. For example, for detection when paired with the newly introduced SSDLite [2] the new model is about 35% faster with the same accuracy than MobileNetV1.

Model	Params	Multiply-Adds	mAP	Mobile CPU
MobileNetV1 + SSDLite	5.1M	1.3B	22.2%	270ms
MobileNetV2 + SSDLite	4.3M	0.8B	22.1%	200ms

To enable on-device semantic segmentation, we employ MobileNetV2 as a feature extractor in a reduced form of DeepLabv3 [3], that was announced recently. On the semantic segmentation benchmark, PASCAL VOC 2012, our resulting model attains a similar performance as employing MobileNetV1 as feature extractor, but requires 5.3 times fewer parameters and 5.2 times fewer operations in terms of Multiply-Adds.

Model	Params	Multiply-Adds	mIOU
MobileNetV1 + DeepLabV3	11.15M	14.25B	75.29%
MobileNetV2 + DeepLabV3	2.11M	2.75B	75.32%

As we have seen MobileNetV2 provides a very efficient mobile-oriented model that can be used as a base for many visual recognition tasks. We hope by sharing it with the broader academic and open-source community we can help to advance research and application development.

RESNET

A residual neural network (ResNet) is an artificial neural network (ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers.

It is used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully.

FUNCTIONS

ResNet101(...): Instantiates the ResNet101 architecture.

ResNet152(...): Instantiates the ResNet152 architecture.

ResNet50(...): Instantiates the ResNet50 architecture.

decode_predictions(...): Decodes the prediction of an ImageNet model.

preprocess_input(...): Preprocesses a tensor or NumPy array encoding a batch of images.

RESNET50

ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database [1]. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

The ResNet-50 model consists of 5 stages each with a convolution and Identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. The ResNet-50 has over 23 million trainable parameters.

ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer.

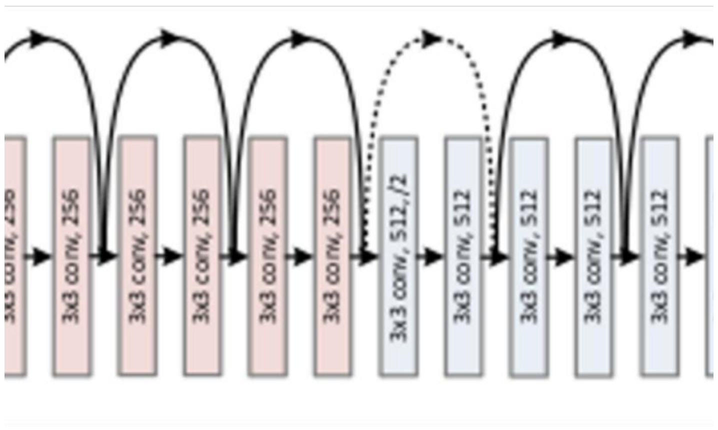


Fig 22: Implementation

ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database [1]. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

The ResNet-50 has accuracy 81% in 30 epochs and the MobileNet has accuracy 65% in 100 epochs. ResNet-50 has a greater number of parameters to be used so it is obvious that it will show better performance as compared to the MobileNet.

FUNCTIONS

ResNet50(...): Instantiates the ResNet50 architecture.

decode_predictions(...): Decodes the prediction of an ImageNet model.

preprocess_input(...): Preprocesses a tensor or NumPy array encoding a batch of images.

RESNETV2

After the release of the second paper on ResNet [4], the original model presented in the previous section has been known as ResNet v1. The improved ResNet is commonly called ResNet v2. The improvement is mainly found in the arrangement of layers in the residual block as shown in following figure.

The prominent changes in ResNet v2 are:

- The use of a stack of 1 × 1 - 3 × 3 - 1 × 1 BN-ReLU-Conv2D

- Batch normalization and ReLU activation come before 2D convolution

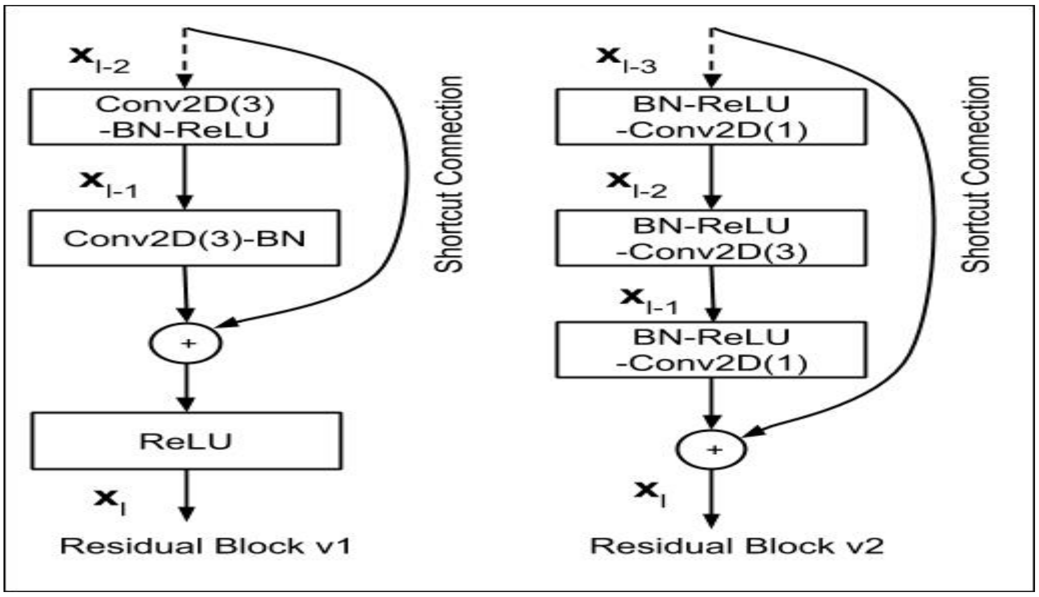


Fig 23: A comparison of residual blocks between ResNet v1 and ResNet v2

FUNCTIONS

ResNet101V2(...): Instantiates the ResNet101V2 architecture.

ResNet152V2(...): Instantiates the ResNet152V2 architecture.

ResNet50V2(...): Instantiates the ResNet50V2 architecture.

decode_predictions(...): Decodes the prediction of an ImageNet model.

preprocess_input(...): Preprocesses a tensor or NumPy array encoding a batch of images.

blobFromImage

Now the next step is to load images in a batch and run them through the network. For this, we use the cv2.dnn.blobFromImage method. This method creates 4-dimensional blob from input images. Let's look at the signature of this method:

blob = cv.dnn.blobFromImage (image, scalefactor, size, mean, swapRB, crop)

Where:

image: is the input image that we want to send to the neural network for inference.

scalefactor: If we want to scale our images by multiplying them by a constant number. A lot of times we divide all of our uint8 images by 255, this way all the pixels are between 0 and 1(0/255-255/255). The default value is 1.0 which means no scaling.

size: The spatial size of the output image. It will be equal to the input size required for the follow-on neural networks as the output of blobFromImage.

swapRB: Boolean to indicate if we want to swap the first and last channel in 3 channel images. OpenCV assumes that images are in BGR format by default but if we want to swap this order to RGB, we can set this flag to True which is also the default.

mean: In order to handle intensity variations and normalization, sometimes we calculate the average pixel value on the training dataset and subtract it from each image during training. If we are doing mean subtraction during training, then we must apply it during inference. This mean will be a tuple corresponding to R, G, B channels. For example, mean values on the imagenet dataset is R=103.93, G=116.77, and B=123.68. If we use swapRB=False, then this order will be (B, G, R).

crop: Boolean flag to indicate if we want to center crop our images. If it's set to True, the input image is cropped from the center in such a way that smaller dimension is equal to the corresponding dimension in size and other dimension is equal or larger. However, if we set it to False, it would preserve the aspect ratio and just resize to dimensions in size.

IMAGE SCALING

Image scaling is the process of resizing a digital image. Scaling down an image makes it smaller while scaling up an image makes it larger. Both raster graphics and vector graphics can be scaled, but they produce different results.

An image can be scaled explicitly with an image viewer or editing software, or it can be done automatically by a program to fit an image into a differently sized area. Reducing an image, as is done to create thumbnail pictures, can use several methods but largely employs a type of sampling called under sampling to reduce the image and maintain the original quality. Increasing the size of an image can be more complex, because the number of pixels required to fill the larger area is greater than the number of pixels in the original image. When image scaling is used to increase the size of an image, one of several algorithms is used to approximate the color of the additional pixels in the larger image.

Mobilenetv2.preprocessing_input

Preprocesses a tensor or NumPy array encoding a batch of images.

```
tf.keras.applications.mobilenet_v2.preprocess_input (
    x, data_format=None
)
```
Example:
applications.MobileNet:

```
i = tf.keras.layers.Input([None, None, 3], dtype = tf.uint8)
x = tf.cast(i, tf.float32)
x = tf.keras.applications.mobilenet.preprocess_input(x)
core = tf.keras.applications.MobileNet()
x = core(x)
model = tf.keras.Model(inputs=[i], outputs=[x])

image = tf.image.decode_png(tf.io.read_file('file.png'))
result = model(image)
```

Args

X: A floating point numpy.array or a tf.Tensor, 3D or 4D with 3 color channels, with values in the range [0, 255]. The preprocessed data are written over the input data if the data types are compatible. To avoid this behaviour, numpy.copy(x) can be used.

data_format: Optional data format of the image tensor/array. Defaults to None, in which case the global setting tf.keras.backend.image_data_format() is used (unless you changed it, it defaults to "channels_last").

Returns: Preprocessed numpy.array or a tf.Tensor with type float32.

The inputs pixel values are scaled between -1 and 1, sample-wise.

Raises ValueError: In case of unknown data_format argument.

TF KERAS PREPROCESSING

Provides keras data pre-processing utils to pre-process tf.data. Datasets before they are fed to the model.

MODULES

image module: Set of tools for real-time data augmentation on image data.

sequence module: Utilities for preprocessing sequence data.

text module: Utilities for text input preprocessing.

FUNCTIONS

image_dataset_from_directory(...): Generates a tf.data.Dataset from image files in a directory.

text_dataset_from_directory(...): Generates a tf.data.Dataset from text files in a directory.

timeseries_dataset_from_array(...): Creates a dataset of sliding windows over a timeseries provided as array.

H5 MODEL

H5 is a file format to store structured data, it's not a model by itself. Keras saves models in this format as it can easily store the weights and model configuration in a single file.

A Keras model consists of multiple components:

- The architecture, or configuration, which specifies what layers the model contain, and how they're connected.
- A set of weights values (the "state of the model").
- An optimizer (defined by compiling the model).
- A set of losses and metrics (defined by compiling the model or calling add_loss() or add_metric()).

The Keras API makes it possible to save all of these pieces to disk at once, or to only selectively save some of them:

- Saving everything into a single archive in the TensorFlow SavedModel format (or in the older Keras H5 format). This is the standard practice.
- Saving the architecture / configuration only, typically as a JSON file.
- Saving the weights values only. This is generally used when training the model.

It is the default when you use model.save() . You can switch to the H5 format by: Passing save_format='h5' to save(). Passing a filename that ends in .h5 or .keras to save() .

Saving a Keras model:

```
model = ...  # Get model (Sequential, Functional Model, or Model subclass)
model.save('path/to/location')
```

Loading the model back:

```
from tensorflow import keras
model = keras.models.load_model('path/to/location')
```

WEIGHT OF DNN

Within each node is a set of inputs, weight, and a bias value. As an input enters the node, it gets multiplied by a weight value and the resulting output is either observed, or passed to the next layer in the neural network. Often the weights of a neural network are contained within the hidden layers of the network.

MASK DETECTOR ARCHITECTURE

MobileNetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.

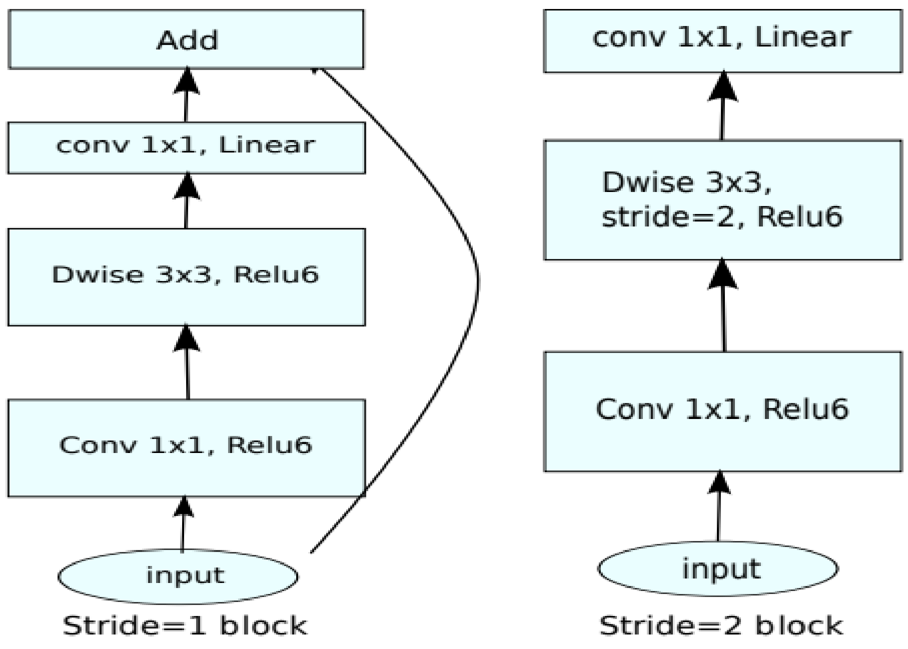


Fig 24: Residual vs Inverted Block

PROTOTXT

A PROTOTXT file is a prototype machine learning model created for use with Caffe. It contains an image classification or image segmentation model that is intended to be trained in Caffe. PROTOTXT files are used to create .CAFFEMODEL files.

Caffe (Convolutional Architecture for Fast Feature Embedding) is a deep learning framework that allows users to create image classification and image segmentation models. Initially, users create and save their models as plain text. prototxt files. After a user trains and refines their model using Caffe, the program saves the user's trained model as a CAFFEMODEL file.

PROTOTXT files are serialized using Google's Protocol Buffers serialization library, and they contain:

A list of the neural network layers a model contains

Each layer's parameters, including its name, type, input dimensions, and output dimensions

Specifications for connections between layers

How do I open a PROTOTXT file?

PROTOTXT files are intended to be used as prototype Caffe (cross-platform) machine learning models. However, because PROTOTXT files are plain text files, you can open and edit them in any text editor if needed.

PRE-TRAINED MODEL

A pre-trained model is a model created by someone else to solve a similar problem. Instead of building a model from scratch to solve a similar problem, you use the model trained on other problem as a starting point.

For example, if you want to build a self-learning car. You can spend years to build a decent image recognition algorithm from scratch or you can take inception model (a pre-trained model) from Google which was built on ImageNet data to identify images in those pictures.

A pre-trained model may not be 100% accurate in your application, but it saves huge efforts required to re-invent the wheel.

Ways to Fine tune the model

Feature extraction – We can use a pre-trained model as a feature extraction mechanism. What we can do is that we can remove the output layer (the one which gives the probabilities for being in each of the 1000 classes) and then use the entire network as a fixed feature extractor for the new data set.

Use the Architecture of the pre-trained model – What we can do is that we use architecture of the model while we initialize all the weights randomly and train the model according to our dataset again.

Train some layers while freeze others – Another way to use a pre-trained model is to train is partially. What we can do is we keep the weights of initial layers of the model frozen while we retrain only the higher layers. We can try and test as to how many layers to be frozen and how many to be trained.

CHAPTER 4

SYSTEM REQUIRMENTS

4.1 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

Technologies Used – Python

Tool used - Visual Studio Code

Browser Required – Chrome, Firefox, Edge

4.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

Operating System - Windows

Processor – 1.7 GHZ or Higher

Ram - 4 GB

Hard Disk - 20 GB

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

Architecture diagram consists of both client and server. Client has its own UI where as server has REST API. Client need to run the application then it will generate the IP Address which acts an interface. Client need to send the input as either image/video or dynamic video. If the client requests for the dynamic video then camera will be accessed. The interface which we've used is REST API. The process of facemask detection runs on the server only which involves feature extraction So first it detects the faces using face detection model then upon detecting faces it detects for the masks using mask detection model which has its own Caffe models and prototypes.

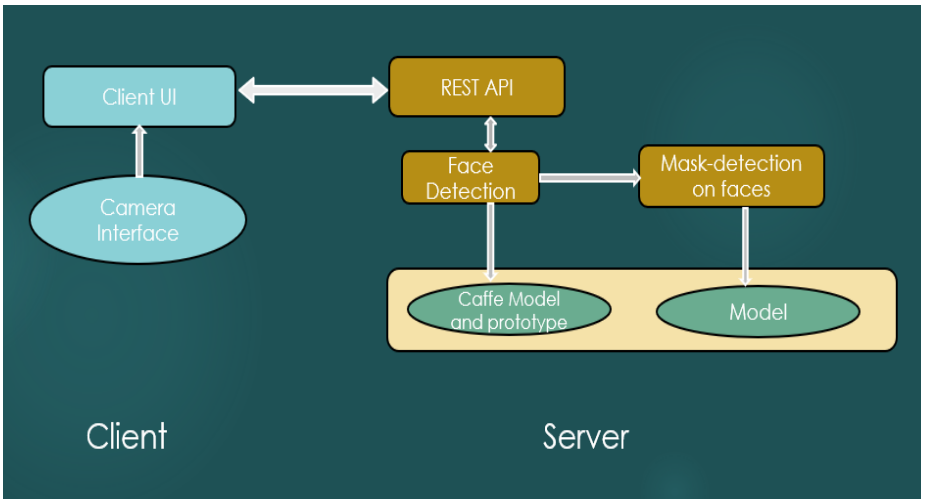


Fig 25: Architecture Design

5.2 DATA F LOW DIAGRAM

Data flow diagram is a traditional way to visualize the information flows within a system.

When a human face is placed or shown to the camera then pre-processing is done which takes input as raw data in the form of individual frames, then the feature extraction is done in the face detection where a human face is detected and then binary classification is done to check whether that human face has a face mask on it or not.

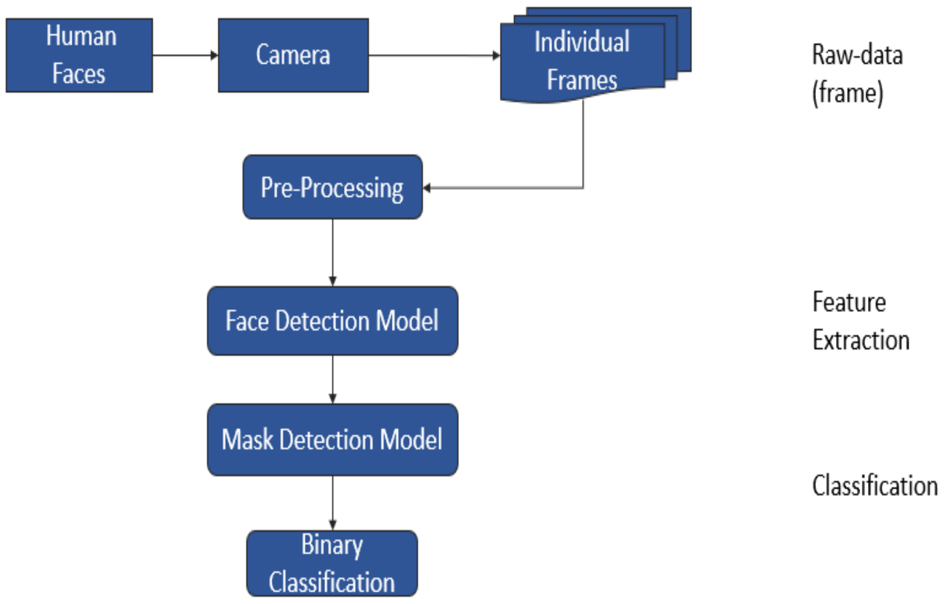


Fig 26: Data Flow Diagram

5.3 DESIGN AND ANALYSIS

It is nothing what is the procedure used to implement the task basically known as algorithm. Algorithm starts with start and ends with stop. Upon running the application client will be able to see an interface which is technically known as Swagger UI. Then client need to select an option. If he chooses image/video then he needs to give input statically to the interface then those input will be forwarded to the server which runs the facemask detection. Upon successful completion of processing client can be able to download the results. If the client chooses for dynamic video then he needs to grant the camera which dynamically performs facemask detection. It also captures the images for every 5 seconds which can be viewed after closing the camera.

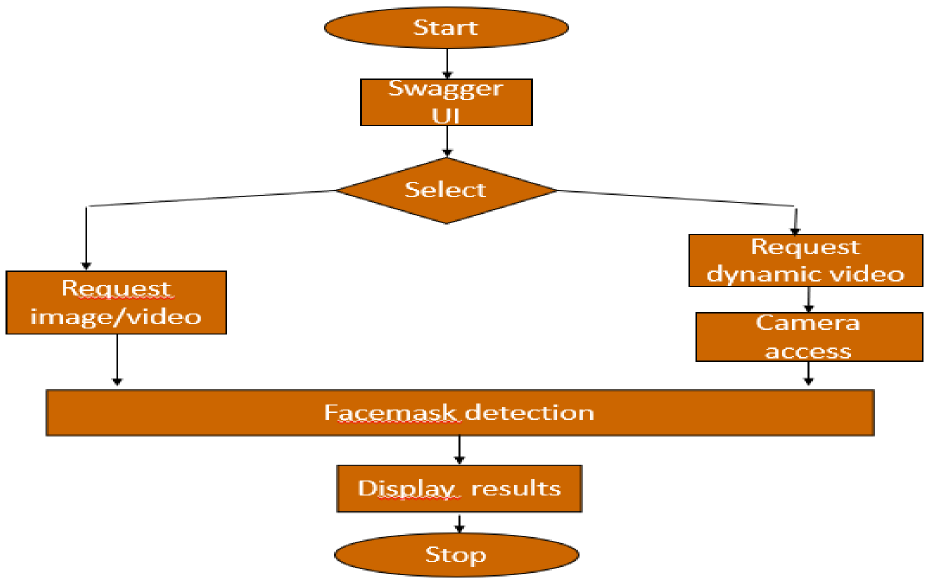


Fig 27: Design and Analysis

5.4 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: A Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

In this case the user has perform the following actions.

First the user has to load an application which generates IP address upon going through that IP address an interface will appears which is technically called as Swagger UI.

Then the user needs to select the file. It might be either image or video. There is also an option of dynamic video which is nothing but Live video. Upon selecting the image/video option user need to provide input as image/video. Then the request is forwarded to the server. Server does the work of facemask detection which involves feature extraction. After successful completion output can downloaded and displayed by the user.

If the user selects for dynamic video then it requests for the camera access and then facemask detection takes place. Here it captures the images for every 5 seconds till the user closes the prompt. In this case the captured images are outputs.

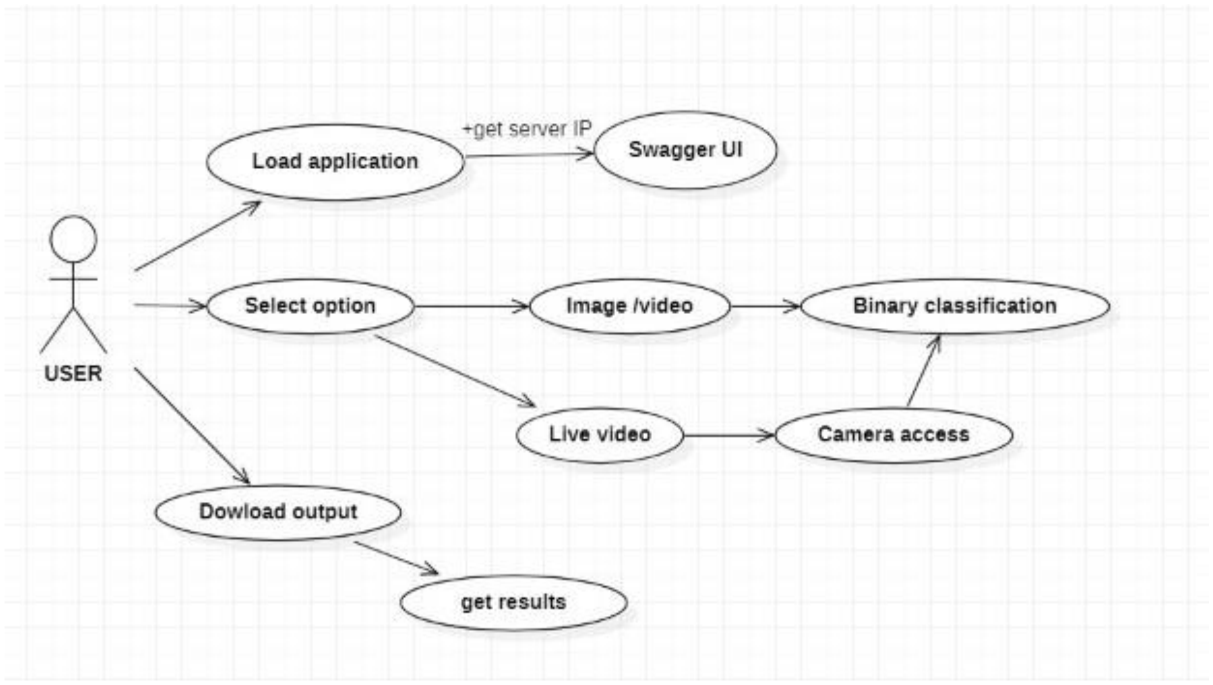


Fig 28: Usecase Diagram

CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

We can see that there are three class diagrams which are User, Server and Facemask detection.

User: The attributes of user are Swagger UI whereas the operations are image, video and live camera. Upon running the application user needs provide input either in the form of images or videos. That request is sent to the server and upon processing server sends it back to the user. User can download and display the results.

Server: The attributes of server are Server IP and REST API whereas the operations are validateInput, ForwardOutput and SendResponse.

Server takes request from the user and validates the input then it forwards the data to facemask detection. Facemask detection does the work of feature extraction and upon successful of feature extraction it sends the response to the sever which then will be sent to the user.

Facemask Detection: The attributes of facemask detection are validateInput whereas the operations are FaceDetection, Maskdetection and Confidence.

When the data is forwarded from server to facemask detection it validates the data then it does feature extraction. First it detects face in the images/videos then upon detecting faces it detects for mask upon those faces. We can get the confidence level of the detected masks ranging in between 0-100. After detecting the output is sent to the server which is then forwarded to the client.

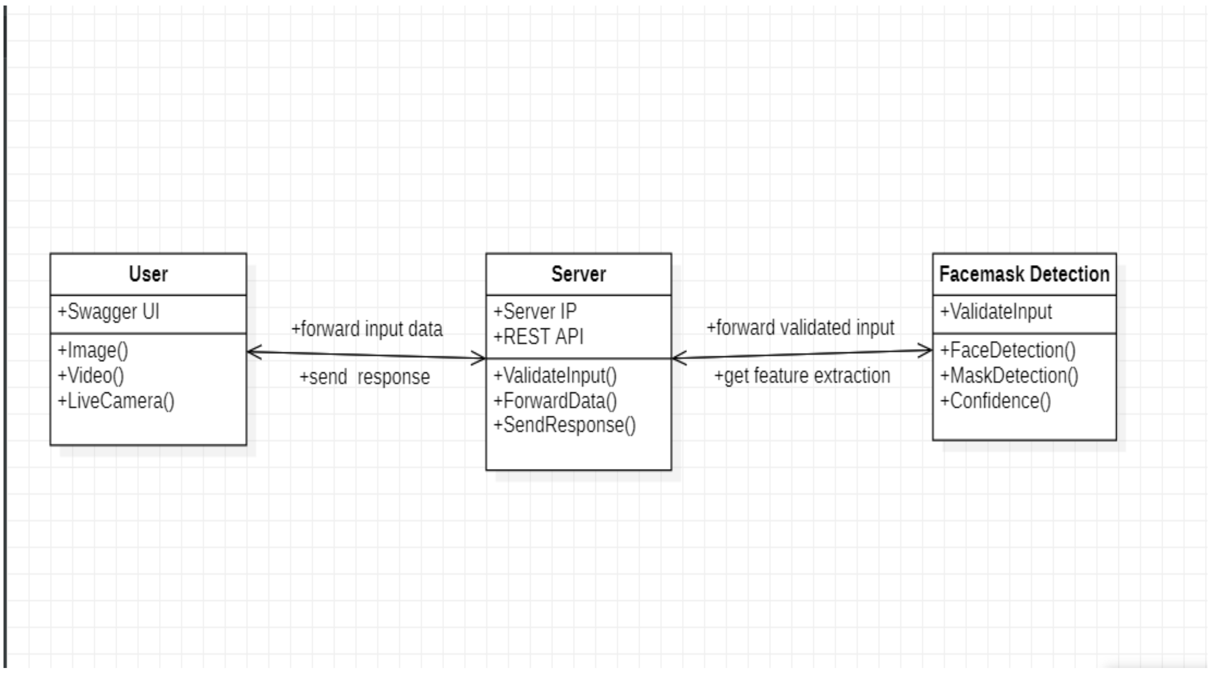


Fig 29: Class Diagram

SEQUENCE DIAGRAM:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

There are 5 lifelines present in this application which are User, Swagger UI, Server Script, Facemask Detection and Camera. Following are the sequence of actions performed by the user.

First the user has to load the application then he/she has to select the option which might be either image/video or dynamic video. If the user selects the 1st option then the request is made to the user to select the image/video then that request is forwarded through the server to facemask detection. It does the feature extraction which involves both face detection and mask detection. After successful completion of feature extraction output will be sent to user which can be downloaded and displayed.

If the user requests for the dynamic video then it asks for the camera access. It dynamically runs the facemask detection. It captures the images for every 5 seconds which can be viewed later after closing the camera.

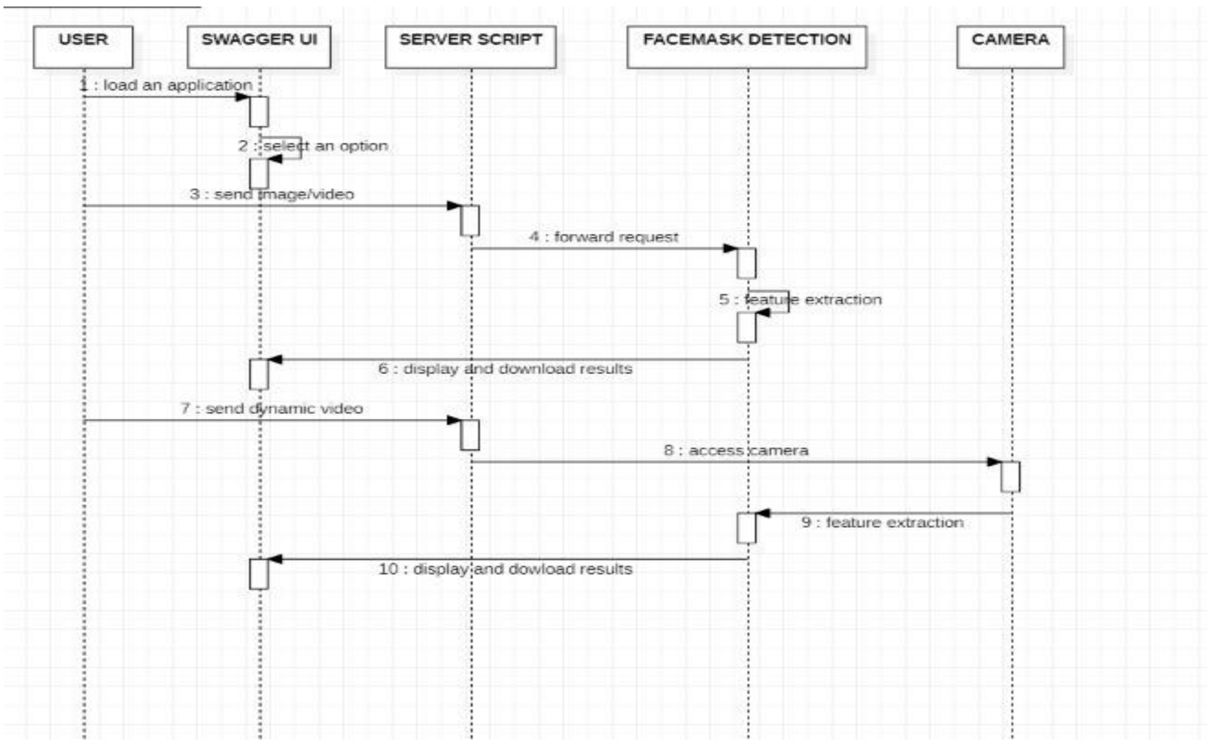


Fig 30: Sequence Diagram

ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

When the host runs a sever script it generates a server IP address which is then sent to the clients, which on clicking, a Swagger UI is loaded where the client can give the input as a video or an image at a time. This video/image will be then processed [binary classification] in the server and an output is generated that we can download it and see the output whether a person is wearing a mask or not. In case if the client wants to gives the input as a dynamic/live video, then a different script has to be run so that we'll be seeing the output dynamically, and also the camera can take snaps for a particular time that the user has set and sends to the client which shows whether the people in the frame are wearing a face mask or not.

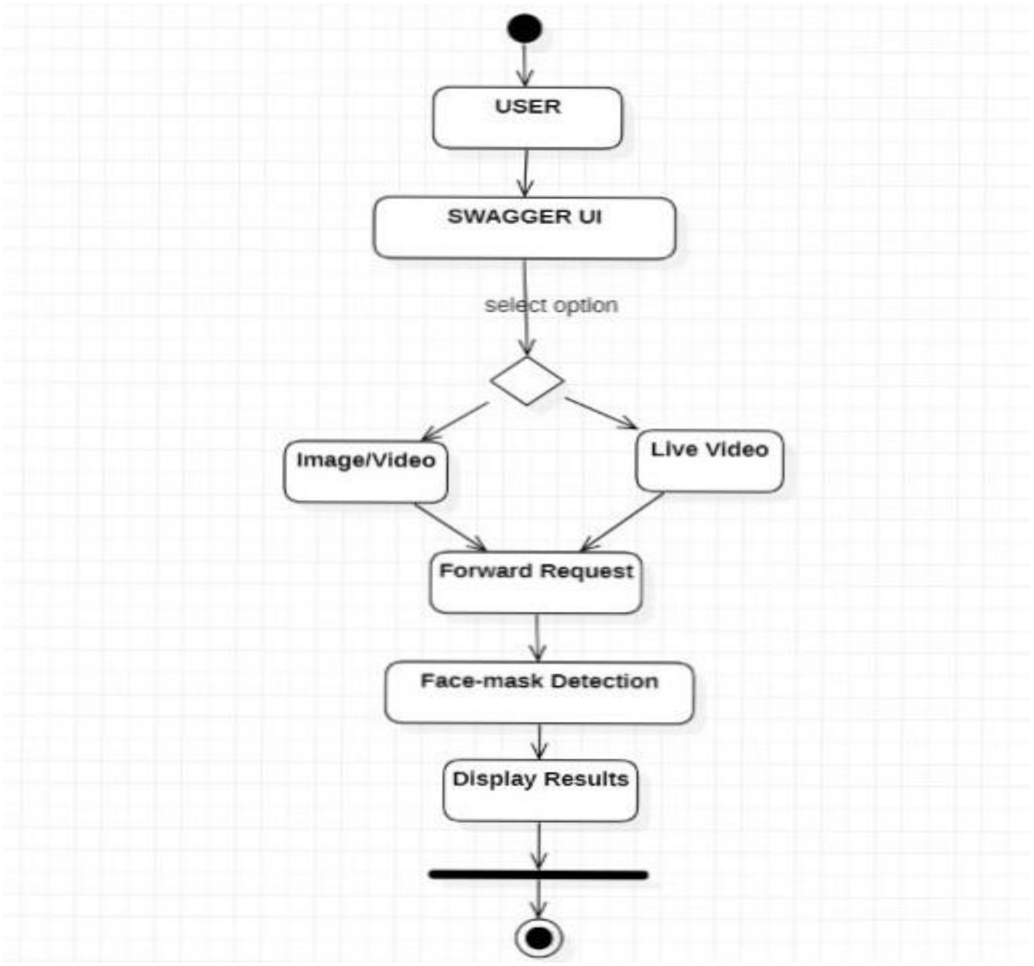


Fig 31: Activity Diagram

CHAPTER 6

CODING

fastApi.py:

```python
import shutil
import cv2

from fastapi import FastAPI
from fastapi.responses import FileResponse,HTMLResponse
import uvicorn
from fastapi import FastAPI, File, UploadFile

import detect_mask_image
import detect_mask_mp4

def execute_video(input_file, output_file):
    """
    input : video path
    output : saves scanned video
    """
    result = detect_mask_mp4.mask_video(input_file, output_file)

def execute_image (input_file, output_file):
    """
    input : image
    output : saves scanned image
    """
    result, label = detect_mask_image.mask_image(input_file)
    cv2.imwrite(output_file , result)

    return result, label

work_place= "work place/"
app=FastAPI()

@app.post('/facemask_detect')
def get_output(input_file: UploadFile = File(...)):
    """
    input types supported : photos - jpeg, png, jpg
                            video  - mp4
    """
    label = "none"
    input_wp=work_place+"inputs/"
    output_wp=work_place+"outputs/"
    print("type of file :", type(input_file))
    print("name of file :",input_file.filename)

    #uploading the file to host
    with open(input_wp+input_file.filename, "wb+") as b:
        shutil.copyfileobj(input_file.file, b)

    inp_file=input_wp+input_file.filename
    out_file=output_wp+input_file.filename+".jpg"

    if input_file.filename[-4:] == ".mp4":
        out_file = output_wp+input_file.filename+".avi"
        execute_video(inp_file , out_file)

    else:
        result, label = execute_image(inp_file,out_file)


    response = FileResponse(out_file , media_type='application/octet-
stream',filename="output.jpg" if input_file.filename[-
4:] != ".mp4" else "output.avi")
    response.headers["mask-status"] = label
    return response

@app.get("/")
async def main():
    content = """
        <body>
        <form action="/facemask_detect" enctype="multipart/form-
data" method="post">
        <input name="input_file" type="file" multiple>
        <input type="submit">
        </form>


        </body>
    """
    return HTMLResponse(content=content)


from fastapi import FastAPI, Request

if __name__ == '__main__':
    import socket
    hostname = socket.gethostname()
    local_ip = socket.gethostbyname(hostname)
    uvicorn.run(app, port=8000)
```

CHAPTER 7

TESTING

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification.

7.1 SOFTWARE VALIDATION

Validation is process of examining whether or not the software satisfies the user requirements. It is carried out at the end of the SDLC. If the software matches requirements for which it was made, it is validated.

• Validation ensures the product under development is as per the user requirements.

• Validation answers the question – "Are we developing the product which attempts all that user needs from this software?".

• Validation emphasizes on user requirements.

7.2 SOFTWARE VERIFICATION

Verification is the process of confirming if the software is meeting the business requirements, and is developed adhering to the proper specifications and methodologies.

• Verification ensures the product being developed is according to design specifications.

• Verification answers the question– "Are we developing this product by firmly following all design specifications?"

• Verifications concentrate on the design and system specifications.

7.3 TARGET OF THE TEST AREA

• **Errors** -These are actual coding mistakes made by developers. In addition, there is a difference in output of software and desired output, considered as an error.

• **Fault** - When error exists fault occurs. A fault, also known as a bug, is a result of an error which can cause system to fail.

• **Failure** - failure is said to be the inability of the system to perform the desired task. Failure occurs when fault exists in the system.

7.4 BLACK-BOX TESTING

It is carried out to test functionality of the program. It is also called ‘Behavioral’ testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested ‘ok’, and problematic otherwise.

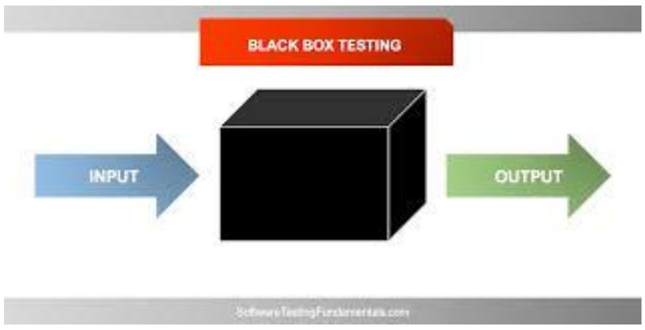


Fig 32: Black box testing

In this testing method, the design and structure of the code are not known to the tester, and testing engineers and end users conduct this test on the software.

7.4.1 BLACK-BOX TESTING TECHNIQUES

• **Equivalence class** - The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.

• **Boundary values** - The input is divided into higher and lower end values. If these values pass the test, it is assumed that all values in between may pass too.

• **Cause-effect graphing** - In both previous methods, only one input value at a time is tested. Cause (input) – Effect (output) is a testing technique where combinations of input values are tested in a systematic way.

• **Pair-wise Testing** - The behavior of software depends on multiple parameters. In pair wise testing, the multiple parameters are tested pair-wise for their different values.

• **State-based testing** - The system changes state on provision of input. These systems are tested based on their states and input.

7.5 TESTING LEVELS

Testing itself may be defined at various levels of SDLC. The testing process runs parallel to software development. Before jumping on the next stage, a stage is tested, validated and verified. Testing separately is done just to make sure that there are no hidden bugs or issues left in the software. Software is tested on various levels –

7.5.1 UNIT TESTING

While coding, the programmer performs some tests on that unit of program to know if it is error free. Testing is performed under white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

7.5.2 INTEGRATION TESTING

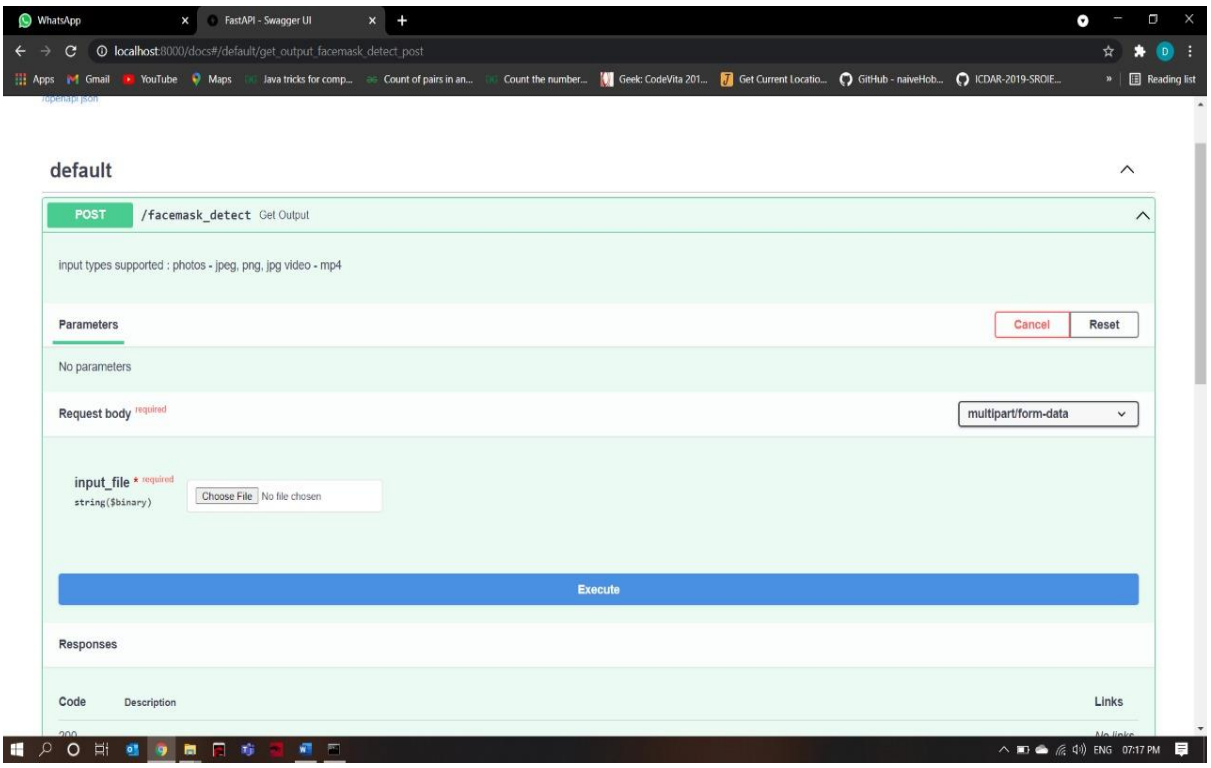
Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passing and data updating etc.

7.6 TESTCASES AND RESULTS

Req_id	**Tkt_id**	**Req_descrption**	**Expected Output**	**Actual Output**	**Req_tckt_status**
1	ID1	Testing API	It should be diverted to the API.	It displayed the API.	Pass
2	ID2	Choose File	It should the select file.	It selected both images and videos.	Pass
3	ID3	Test Input Validation	API should accept valid input format (.png, .jpg, .jpeg,.mp4) if format is different send "bad request status 400 error".	It has allowed only the required input format.	Pass
4	ID4	Detecting Facemask	It should detect weather all human faces are wearing mask or not	It displayed green box if the person is wearing mask else displayed in red box	Pass
5	ID5	With mask	It should detect if mask is not wearing by individual humans in an image in different lighting conditions	It has some trouble while detecting masks in low light condition	Fail
6	ID6	Without mask	It should detect if mask is wearing by individual humans in an image in different lighting conditions	It detects no mask successfully even in low light.	Pass

CHAPTER 8

SCREENSHOTS



default

/facemask_detect Get Output

input types supported : photos - jpeg, png, jpg video - mp4

Parameters

No parameters

Request body required

multipart/form-data

input_file * required
string($binary)

No file chosen

Responses

Code Description Links

Fig 33: API

IN CASE OF IMAGES

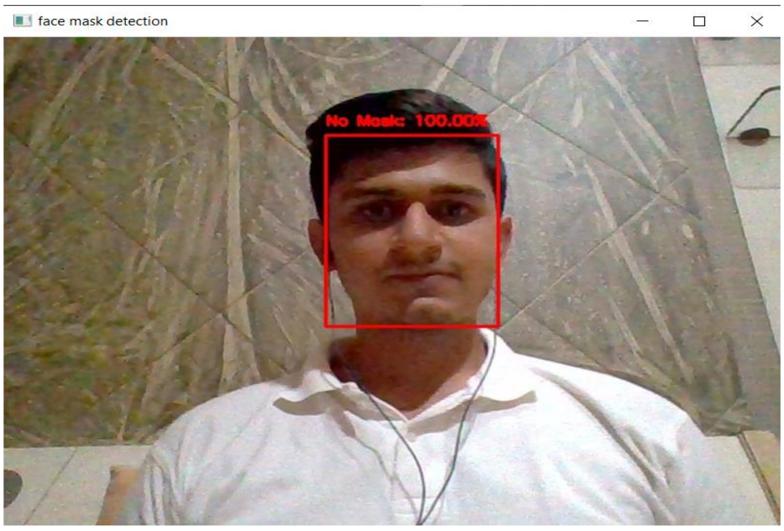


Fig 34: without_image

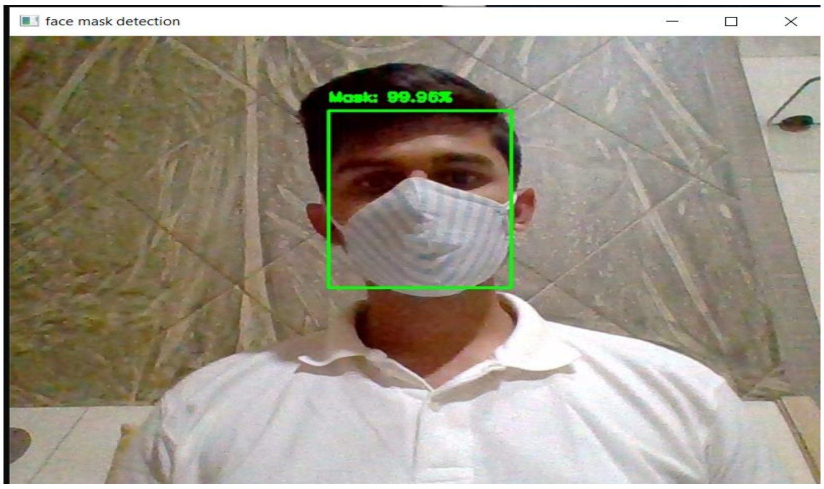
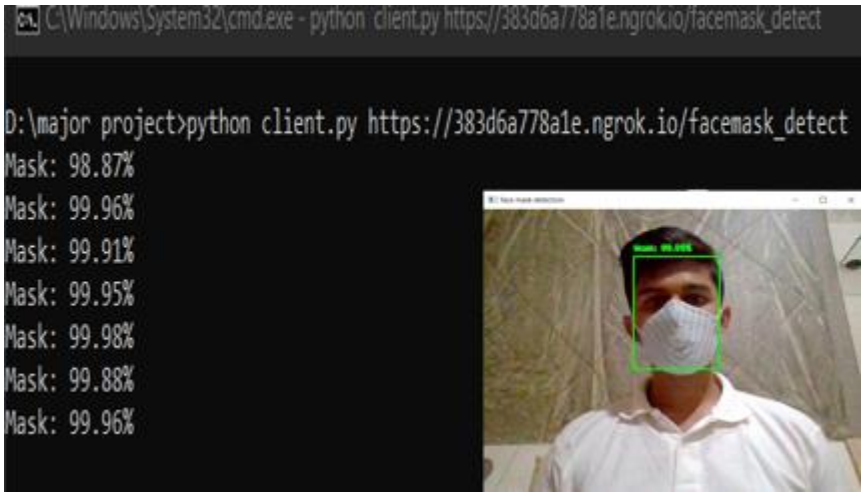


Fig 35: withmask_image

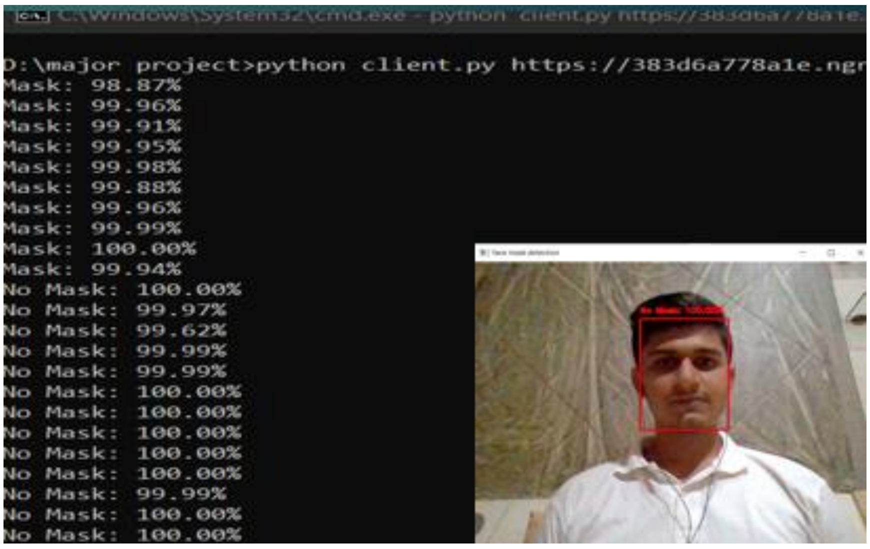
IN CASE OF DYNAMIC VIDEO



```
C:\Windows\System32\cmd.exe - python client.py https://383d6a778a1e.ngrok.io/facemask_detect

D:\major project>python client.py https://383d6a778a1e.ngrok.io/facemask_detect
Mask: 98.87%
Mask: 99.96%
Mask: 99.91%
Mask: 99.95%
Mask: 99.98%
Mask: 99.88%
Mask: 99.96%
```

Fig 36: withmask_video



```
C:\Windows\System32\cmd.exe - python client.py https://383d6a778a1e.
D:\major project>python client.py https://383d6a778a1e.ngr
Mask: 98.87%
Mask: 99.96%
Mask: 99.91%
Mask: 99.95%
Mask: 99.98%
Mask: 99.88%
Mask: 99.96%
Mask: 99.99%
Mask: 100.00%
Mask: 99.94%
No Mask: 100.00%
No Mask: 99.97%
No Mask: 99.62%
No Mask: 99.99%
No Mask: 99.99%
No Mask: 100.00%
No Mask: 100.00%
No Mask: 100.00%
No Mask: 100.00%
No Mask: 100.00%
No Mask: 99.99%
No Mask: 100.00%
No Mask: 100.00%
```

Fig 37: withoutmask_video

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION

By running this application, we can say that it has successfully detected face mask which can be used to monitor the facemask in the large gatherings like shopping malls, theatres or any organizations, etc. Wearing a mask may be obligatory in the near future, considering the Covid-19 crisis. The API which we developed is user-friendly anyone can be able to run the API. Just the user needs to provide input in the form of either images or videos. After giving input he needs to click the execute button it does the processing work then upon successful processing the user can download and display the results. Many government agencies will require customers to wear masks correctly in order to use 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. In this critical time the only way possible to control the spread of virus is to make sure that the people are wearing the mask properly and maintaining proper hygiene, in which this application can play a vital role.

9.2 FUTURE ENHANCEMENT

If this application is integrated with the employees database in organizations then if the employees are not wearing masks properly then we will be able to send a warning message to that employee using his/her id .In the similar way we can also ensure that all the students are wearing a mask properly in the campus if not then a reminder is sent and incase the same student violates this protocol the a fine is to be charged, which we can notify that student using student's contact details in college database.

If few more improvements and modifications are done to this application then we can also use this in public place and in case someone violates the protocol their faces are detected and using the database we can find them. We can also implement the concept of social distancing that would make it a complete system which can bring a dramatic impact on the spread of virus.

9.3 REFERNCES

https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/

https://www.leewayhertz.com/face-mask-detection-system/

https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html

https://www.redhat.com/en/topics/api/what-is-a-rest-api

https://tiangolo.medium.com/introducing-fastapi-fdc1206d453f

https://www.starlette.io/

https://pypi.org/project/starlette/0.1.5/

https://swagger.io/tools/swagger-ui/#

https://caffe.berkeleyvision.org/

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

https://caffe.berkeleyvision.org/tutorial/net_layer_blob.html

https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/

https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html

https://www.leewayhertz.com/face-mask-detection-system/