FACE MASK DETECTION USING YOLO v5

Mrs. P.Vijaya Lakshmi , Assistant Professor

Veneela Choppari - 19VE1A05B8

Guda Sowmya- 19VE1A0578

K.Shreshta - 19VE1A0585

G. Pavan Chand - 20VE5A0508

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

ABSTRACT

The novel Coronavirus had brought a new normal life in which the social distance and wearing of face masks plays a vital role in controlling the spread of virus. But most of the people are not wearing face masks in public places which increases the spread of viruses. This may result in a serious problem of increased spreading. Hence to avoid such situations we have to scrutinize and make people aware of wearing face masks. Humans cannot be involved for this process, due to the chance of getting affected by corona. Hence here comes the need for Artificial Intelligence (AI), which is the main theme of our project. Our project involves the identification of persons wearing face masks and not wearing face by creating a customized YOLOV5 model. YOLO is an Object Detection Algorithm that divides the images into grid system. We also use an online annotation tool or website called as makesense.ai.

KEYWORDS: *Artificial Intelligence, YOLOV5, Scrutinize.*

1 INTRODUCTION

1.1 MOTIVATION

The globe has not still sufficiently Restore from this pandemic and the cure that can efficiently treat Covid-19 is still expected to be found. Nevertheless, to reduce the impact of the universal on the country's frugality, various governments have admitted a restricted number of financial actions to be regained already the number of new cases of Covid19 has discontinued beneath the level. As these nations tentatively restarting their financial endeavors, concerns have arose concerning business security in the newpost-Covid-19 environment.

To humiliate the likelihood of contamination, it is considered that public endure wear masks and assert a distance of at least 1 bit each additional. Deep knowledge has acquire more consideration in object discovery and was used for human discovery purposes and evolve a face mask discovery form that can discover either the individual is wearing mask a suggestion of correction. This may be accomplished by judgment of the categorization results by resolving certain-time pouring from the Cam. In deep knowledge projects, we need a preparation basic document file. It is the real dataset used to train the model for operating various conduct.

1.2 PROBLEM STATEMENT

In the middle of Covid-19 crisis, wearing masks is a fundamental need nowadays. In public places because of the large volume of people it becomes tough for security officials to check every person who is not wearing a mask. This model detects a single person wearing a mask or not.

The main objective of the face detection model is to detect the face of individuals and conclude whether they are wearing masks or not at that particular moment when they are captured in the image.

2.LITERATURE SURVEY

Recent studies show that models developed based on deep learning have made extraordinary progress in object detection and computer vision compared to traditional models. Deep learning-based object detection models can generally be divided into two categories. The first of these are single-stage models such as YOLO [3] and SSD [4], the second is two-stage models such as R-CNN [5], Fast R-CNN [6] and Faster R-CNN [7].

Various studies have been conducted on face mask detection and classification using different deep learning techniques. The first study on face mask classification is a hybrid model developed by Loey et al. They used the ResNet50 model, one of the deep learning architectures, for feature extraction, and the decision trees (DT) and support vector machines (SVM) from traditional machine learning models for classification. They tested the developed model on three datasets and reported that the SVM classifier achieved between 99% and 100% accuracy rate [8].

The other classification study was suggested by Qin and Li. They have developed a facial mask recognition system using a classification network (SRCNet) that incorporates

image super resolution. They tested their proposed system on Medical Masks Dataset, which contains 3835 images. The proposed SRCNet model achieved 98.70% accuracy [9].

The studies on facemask detection were summarized below. Inamdar and Mehendale proposed a model called Facemasknet using deep learning techniques for mask detection. Using the proposed model, they made a triple classification: no mask, the mask has worn incorrectly and the mask has worn properly. They reported that the model, which can be used in both images and video streaming, achieved 98.6% accuracy rate [10].

Wang et al. created three public data sets: Masked Face Detection Dataset (MFDD), Real-world Masked Face Recognition Dataset (RMFRD) and Simulated Masked Face Recognition Dataset (SMFRD). They also proposed a multiparticle masked face recognition model to recognize masked faces. The obtained findings showed that the proposed model reached an accuracy of 95% [11]. In another study, Yadav has proposed a model that focuses on automatic real-time tracking of people to detect safe social distance and face masks in public places. The proposed model consists of combining the MobileNetV2 model with the Single Shot object Detection (SSD) framework. The obtained findings showed that the system could detect social distance and facial masks with a sensitivity score of 91.7% [12]. Jiang et al., have developed a facemask detector called RetinaFaceMask. The developed model is a single-stage detector and includes a feature pyramid network to combine high-level semantic information with multiple feature maps, and an attention module to detect facemasks. They reported that RetinaFaceMask achieved high accuracy facemask[13].

2.1 Existing System

The existing system to detect face masks is developed using YOLO V3.
In the existing system, they introduced a backbone network which can allocate more resources and to accelerate the training process and improve performance.
It produces an accuracy of 76%. The accuracy can be improved while using the application with different algorithms.

2.2 Limitations of Existing System

1.Our model struggles with small objects that appear in groups, such as flocks of birds.

2. Our main source of error is incorrect localizations.

3.REQUIREMENT SPECIFICATION

This chapter gives an overview of the software and hardware components required for our project.

3.1 SOFTWARE REQUIREMENTS

- Programming Language : Python
- Operating System : Windows 10
- IDE : Visual Studio
- UML Design : Start UML
- Tools : PIP

3.2 HARDWARE REQUIREMENTS

- Processor : Intel i3 and above
- RAM : 4GB and Higher
- Hard Disk : 500GB: Minimum

3.3 FUNCTIONAL REQUIREMENTS

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

- Data Collection
- Handling Missing Values
- Remove Duplicate rows
- Data Cleaning
- Data Labeling
- Data Scaling
- Finding Feature Importance
- Data Visualization
- Model Creating
- Model Training
- Model Evaluation
- Model Deployment

3.4 NON-FUNCTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. Non-functional requirements add tremendous value to business analysis. It is commonly misunderstood by a lot of people. It is important for business stakeholders, and Clients to clearly explain the requirements and their expectations in measurable terms. If the non-functional requirements are not measurable then they should be revised or rewritten to gain better clarity. For example, User stories help in mitigating the gap between developers and the user community in Agile Methodology.

- **Usability:**
  Prioritize the important functions of the system based on usage patterns. Frequently used functions should be tested for usability, as should complex and critical functions. Be sure to create a requirement for this.

- **Reliability:**
  Reliability defines the trust in the system that is developed after using it for a period of time. It defines the likeability of the software to work without failure for a given time period.
  The number of bugs in the code, hardware failures, and problems can reduce the reliability of the software.
  Your goal should be a long MTBF (mean time between failures). It is defined as the average period of time the system runs before failing.
  Create a requirement that data created in the system will be retained for a number of years without the data being changed by the system.
  It's a good idea to also include requirements that make it easier to monitor system performance.

- **Performance:**
  What should system response times be, as measured from any point, under what circumstances?
  Are there specific peak times when the load on the system will be unusually high?
  Think of stress periods, for example, at the end of the month or in conjunction with payroll disbursement.

- **Supportability:**
  The system needs to be cost-effective to maintain.
  Maintainability requirements may cover diverse levels of documentation, such as system documentation, as well as test documentation, e.g. which test cases and test plans will accompany the system.

4 SYSTEM DESIGN

4.1 System Design

In this phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

There are two kinds of design documents developed in this phase: High-Level Design (HLD)

- Brief description and name of each module
- An outline about the functionality of every module
- Interface relationship and dependencies between modules
- Database tables identified along with their key elements
    - Complete architecture diagrams along with technology details Low-Level Design(LLD)
- Functional logic of the modules
- Database tables, which include type and size
- Complete detail of the interface
- Addresses all types of dependency issues
- Listing of error messages
- Complete input and outputs for every module

4.2 UML Design:

Unified Modeling Language (UML) is a general-purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.
UML is not a programming language; it is rather a visual language. We use UML diagrams to portray the behavior and structure of a system, UML helps software engineers, businessmen and system architects with modeling, design and analysis. The Object Management Group (OMG) adopted Unified Modeling Language as a standard in 1997. It's been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

Do we really need UML?

- Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
- Businessmen do not understand code. So UML becomes essential to communicate with non programmer's essential requirements, functionalities and processes of the system.
- A lot of time is saved down the line when teams are able to visualize processes, user interactions and static structure of the system.
- UML is linked with object oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

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

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

Types of UML Diagrams:

Structural Diagrams:

Capture static aspects or structure of a system. Structural Diagrams include: Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.

Behavior Diagrams:

Capture dynamic aspects or behavior of the system. Behavior diagrams include: Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

The image below shows the hierarchy of diagrams according to UML

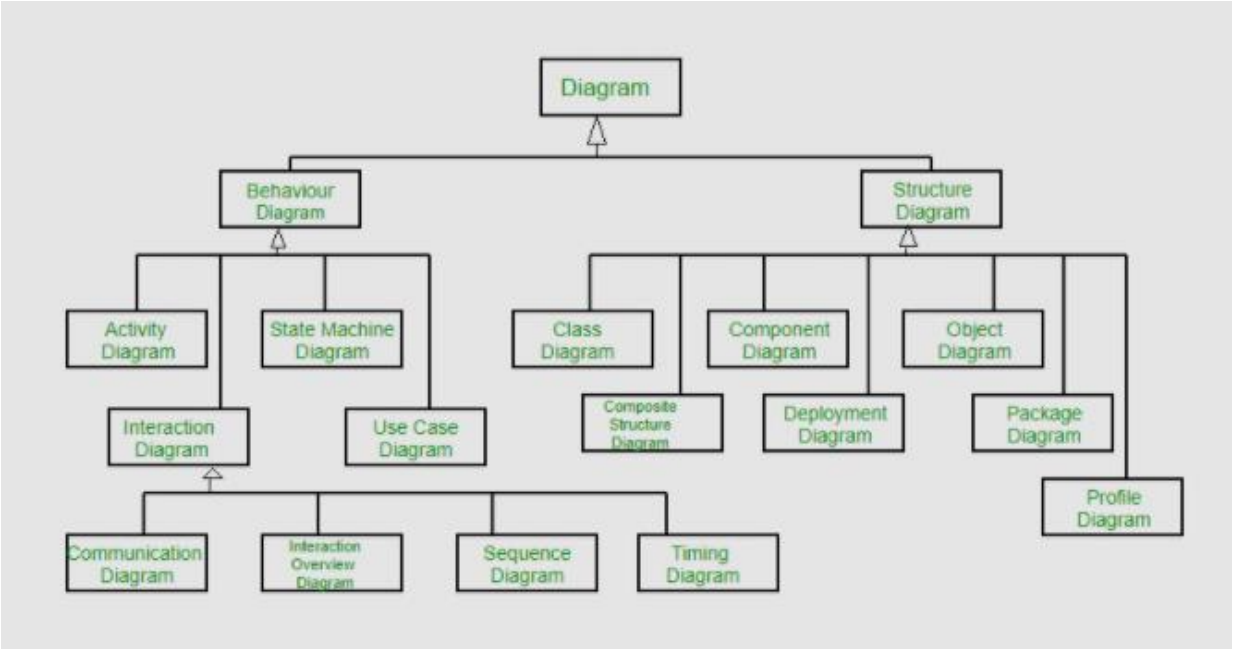


Figure-4.2 UML Hierarchy diagrams

4.2.1 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.

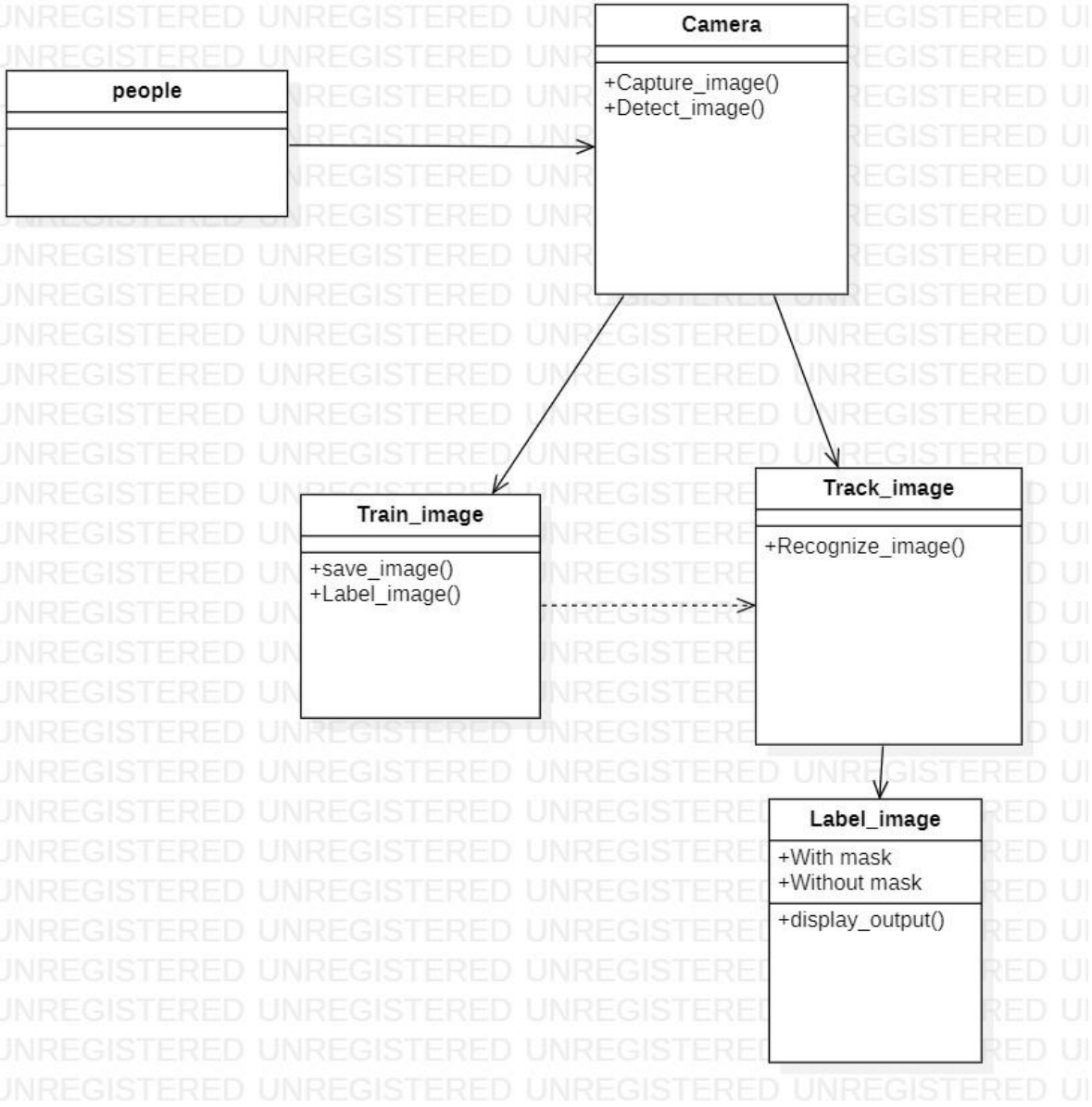


Figure-4.2.1 Class Diagram

4.2.2 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.

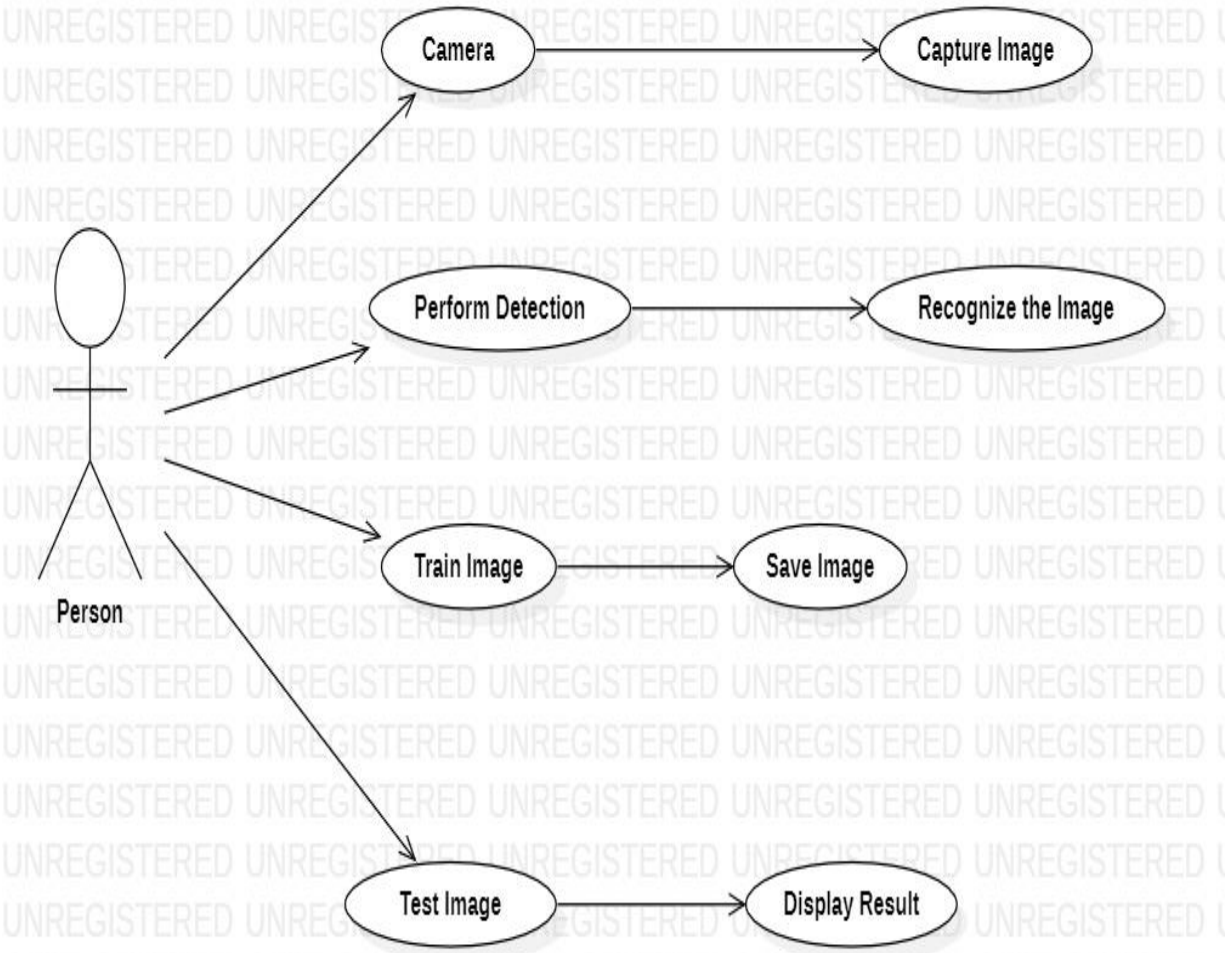


Figure-4.2.2 Use Case Diagram

4.2.3 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling 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.

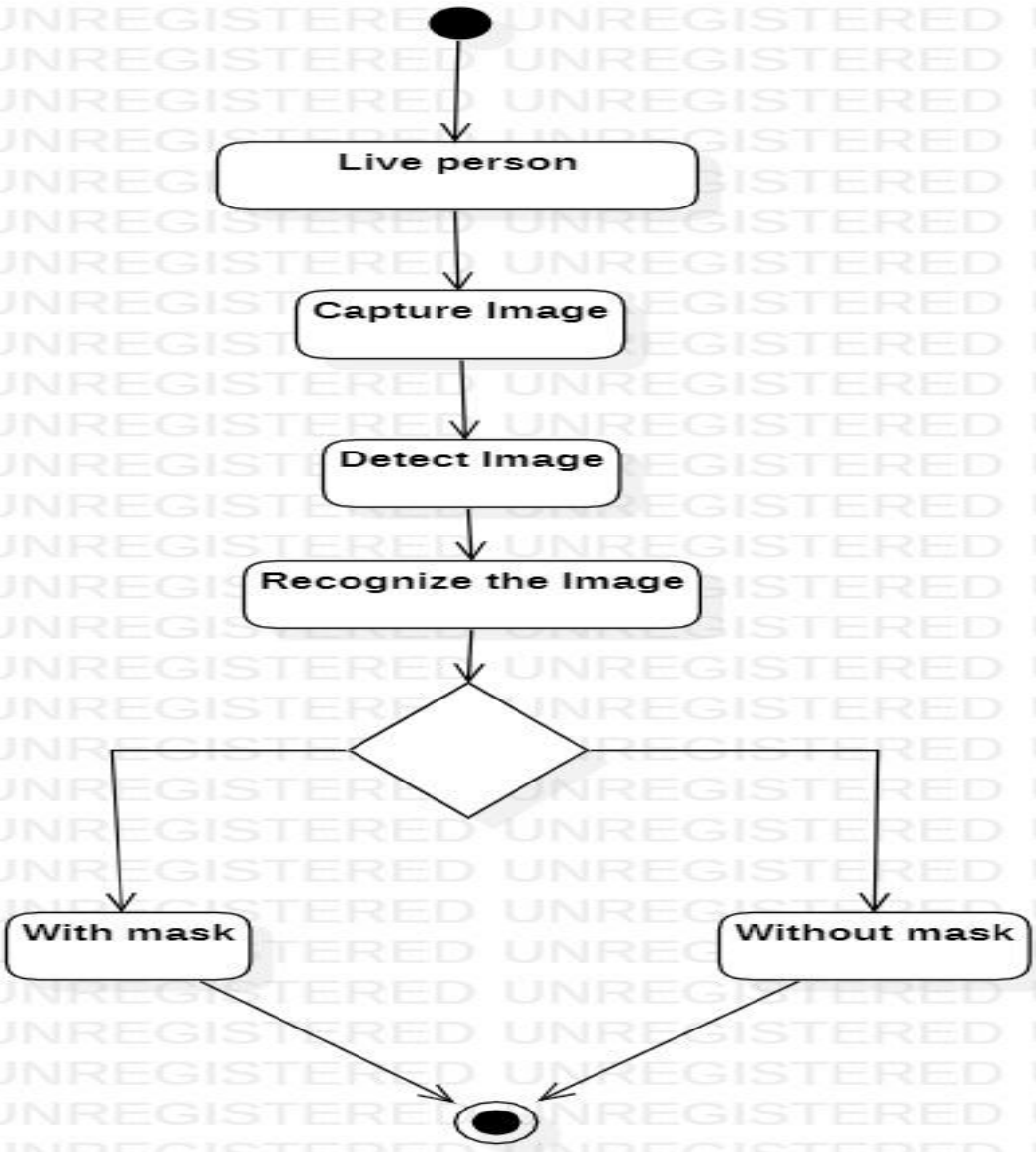


Figure-4.2.3 Activity Diagram

4.2.4 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling 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.

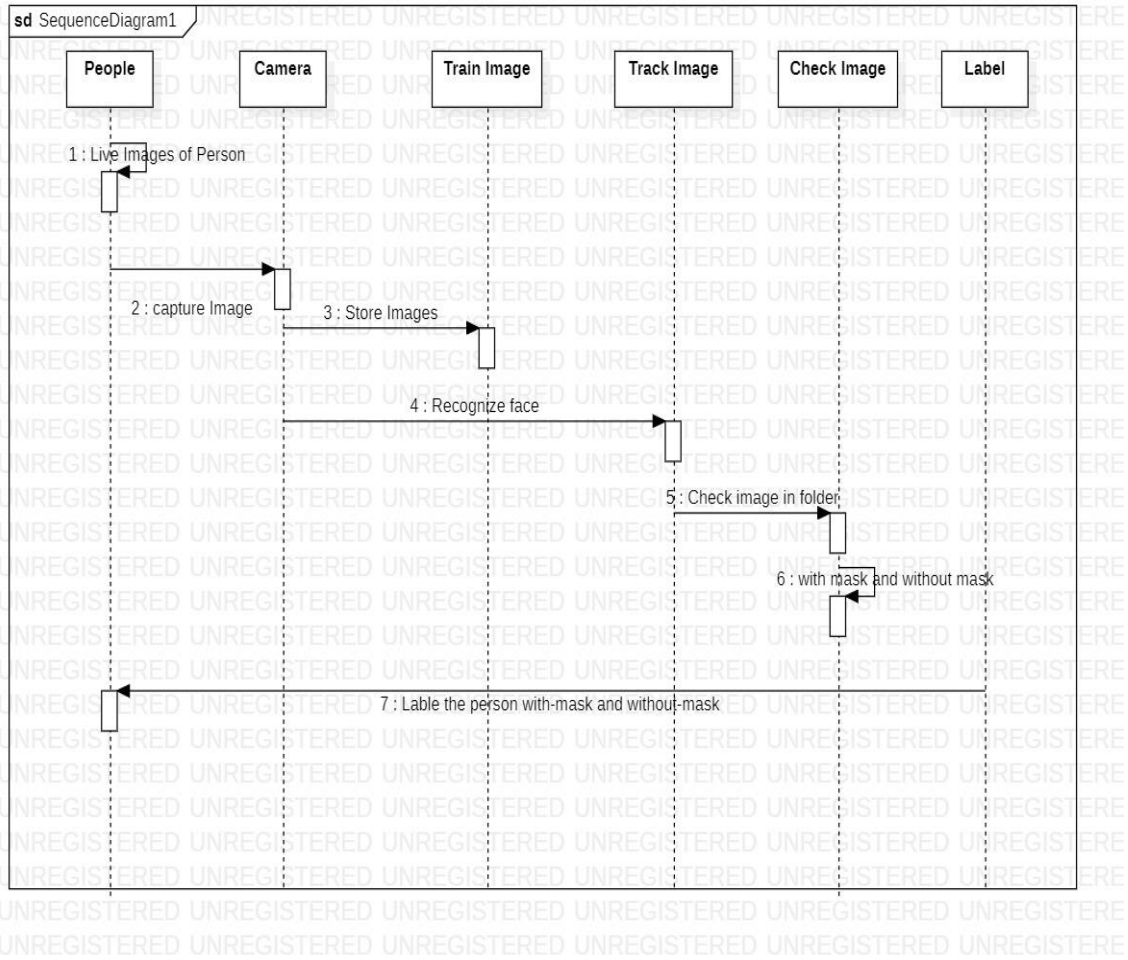


Figure-4.2.4 Sequence Diagram

4.3 SYSTEM ARCHITECTURE:

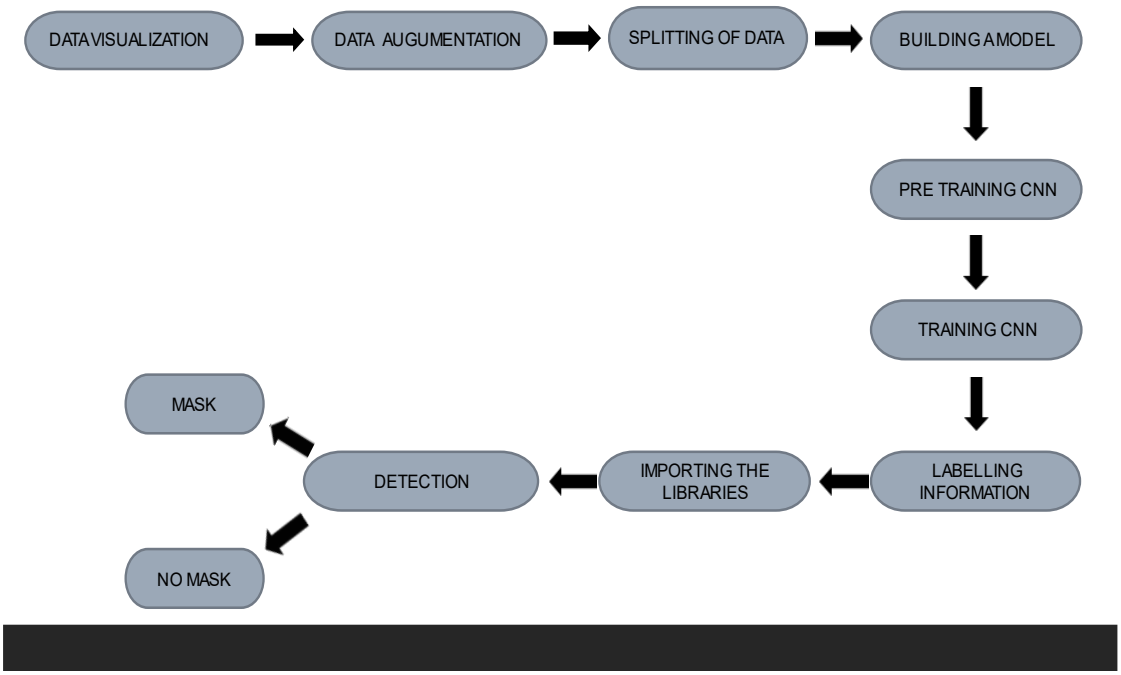


Figure-4.3 System Architecture

4.4 DATA FLOW DIAGRAM

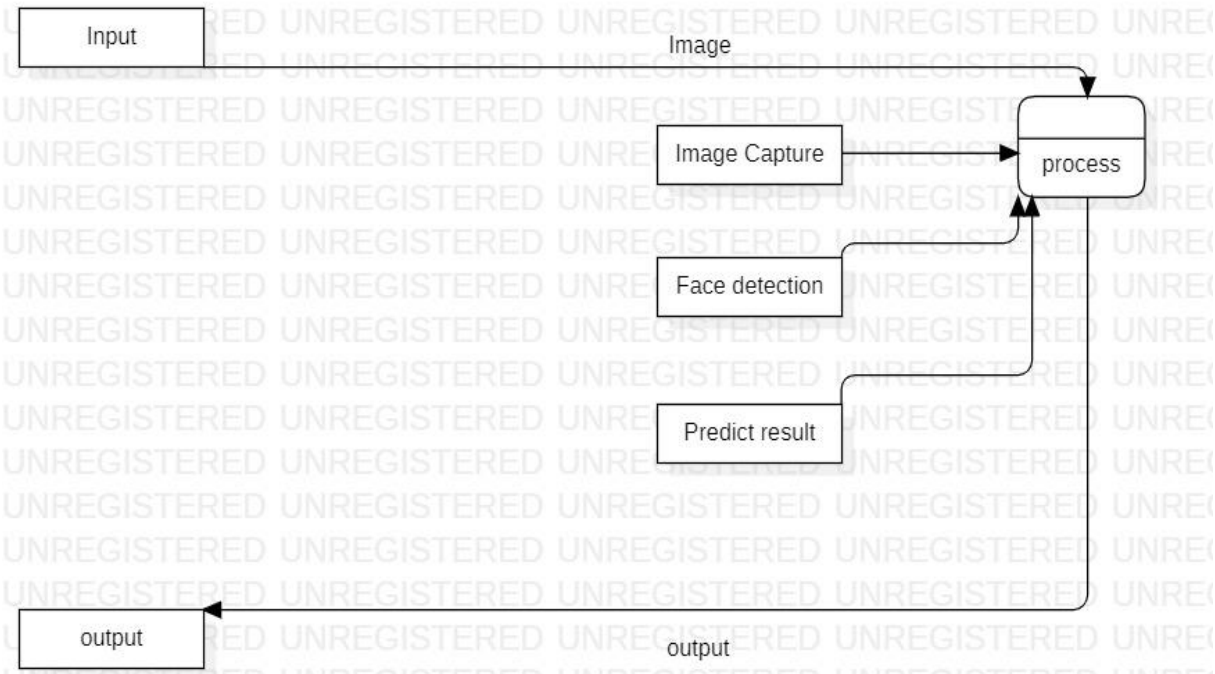


Figure 4.4 Data Flow Diagram

4.5 TECHNOLOGY DESCRIPTION:

4.5.1 PYTHON

Python is a **high-level, interpreted**, **interactive** and **object-oriented scripting language**. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

  Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

➢ HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

➢ PYTHON FEATURES

Python's features include:

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

- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

➢ **ADVANTAGES/BENEFITS OF PYTHON**: The diverse application of the Python language is a result of the combination of features which give this language an edge over others. Some of the benefits of programming in Python include:

- Presence of Third-Party Modules
- Open Source
- Learning Ease
- User-friendly
- Productivity and Speed

4.5.2 IDE:

An integrated development environment (IDE) is a feature-rich program that supports many aspects of software development. The Visual Studio IDE is a creative launching pad that you can use to edit, debug, and build code, and then publish an app. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to enhance the software development process.

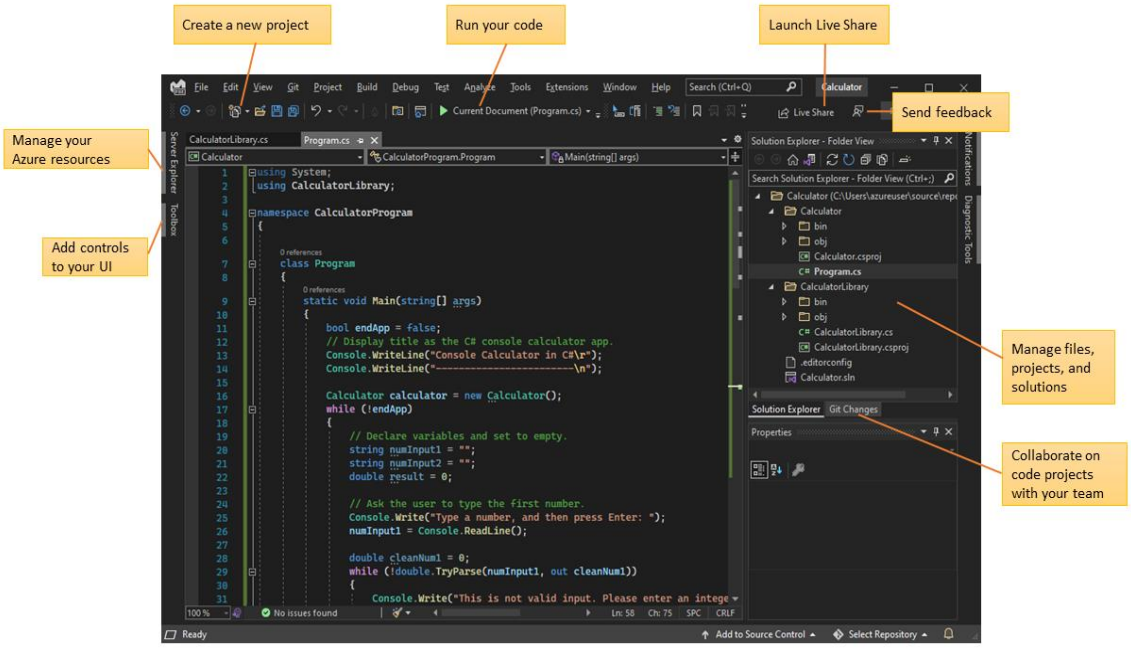


Fig 4.5.2 Visual studio

4.5.3 DEEP LEARNING:

Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions.

Deep learning AI is able to learn without human supervision, drawing from data that is both unstructured and unlabeled.

In this, face mask detection is built using Deep Learning technique called as Convolution Neural Networks (CNN). Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower-level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features. Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features.

4.5.4 ARCHITECTURE OF NEURAL NETWORKS:

- **FEED-FORWARD NETWORKS**:

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

- **FEEDBACK NETWORKS:**

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

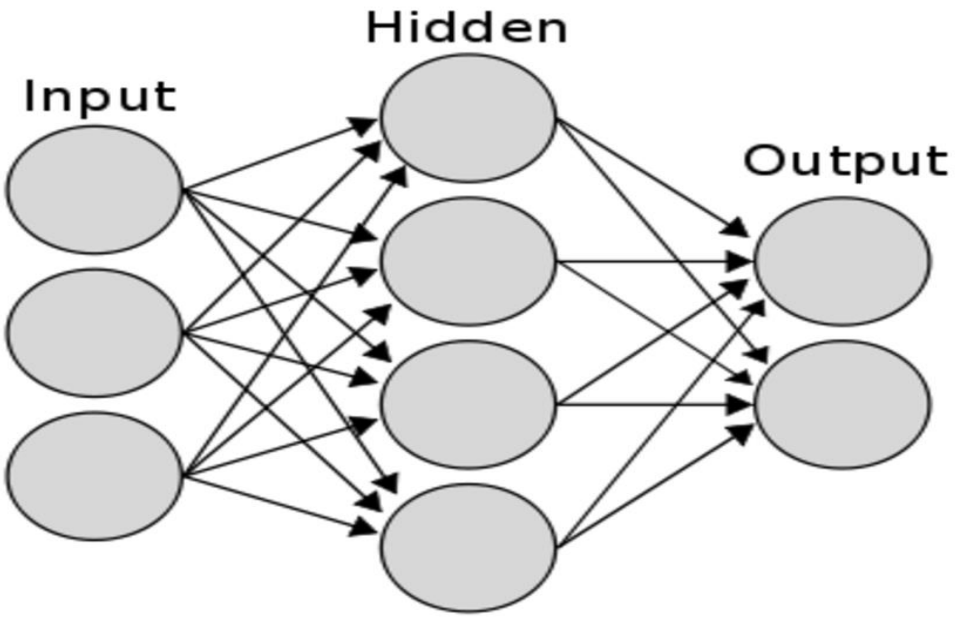


Fig 4.5.4 layers in NN

4.5.5 Convolution Neural Network

A convolution neural network is a special architecture of artificial neural network proposed by yann lecun in 1988. One of the most popular uses of the architecture is image classification. CNNs have wide applications in image and video recognition, recommender systems and natural language processing. In this article, the example that this project will take is related to Computer Vision. However, the basic concept remains the same and can be applied to any other use-case!

CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on CNNs. In more detail the image is passed through a series of convolution, nonlinear, pooling layers and fully output.

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the visual cortex. CNNs use relatively little pre-processing compared to other image classification algorithms. CNN is a special kind of multi- layer NNs applied to 2-d arrays (usually images), based on spatially localized neural input. CNN Generate ‘patterns of patterns’ for pattern recognition.

Each layer combines patches from previous layers. Convolutional Networks are trainable multistage architectures composed of multiple stages Input and output of each stage are sets of arrays called feature maps. At output, each feature map represents a particular feature extracted at all locations on input. Each stage is composed of: a filter bank layer, a non-linearity layer, and a feature pooling layer. A ConvNet is composed of 1, 2 or 3 such 3-layer.

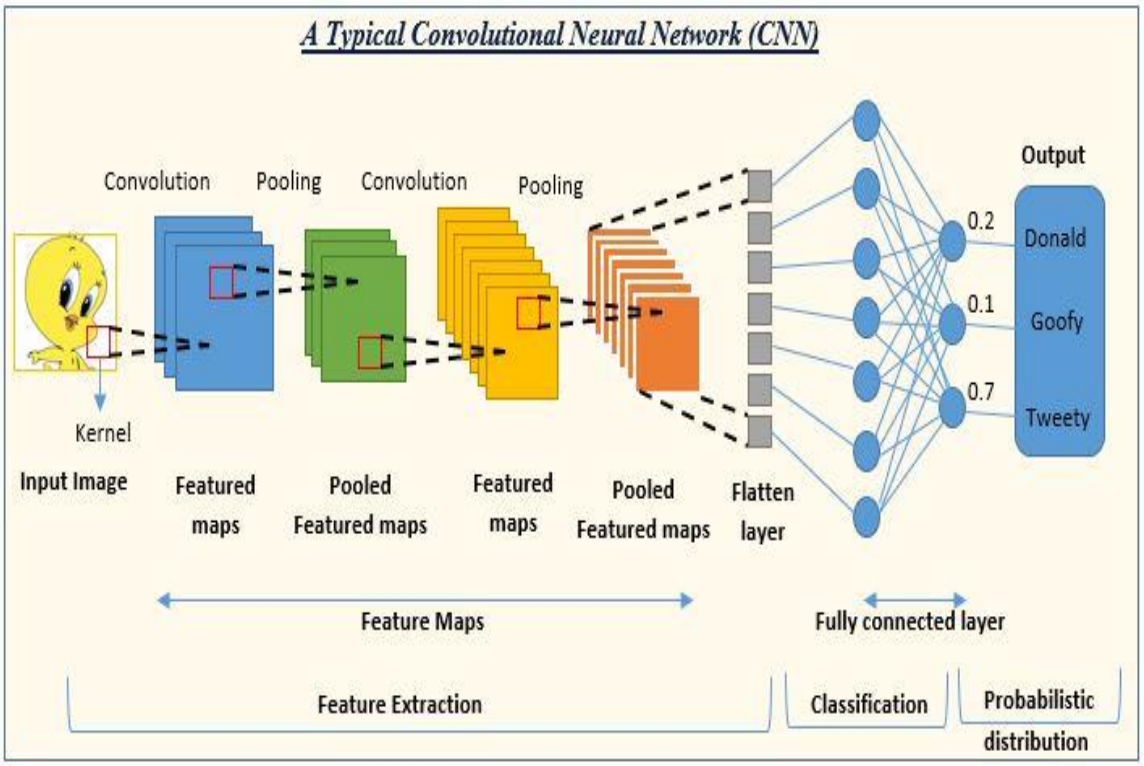


Fig 4.5.5 Convolutional Neural Network

5. IMPLEMENTATION

5.1 Libraries

Numpy:

NumPy is a Python package which stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object, provide tools for integrating C, C++ etc. It is also useful in linear algebra, random number capability etc. NumPy array can also be used as an efficient multi-dimensional container for generic data. Now, let me tell you what exactly is a python numpy array.

To install Python NumPy, go to your command prompt and type "pip install numpy". Once the installation is completed, go to your IDE (For example: PyCharm) and simply import it by typing: "import numpy as np".

Pandas:

Pandas are an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.
Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Standard Python distribution doesn't come bundled with Pandas module. A lightweight alternative is to install NumPy using popular Python package installer, pip**.** pip install pandas

Open-CV:

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.
This OpenCV tutorial will help you learn the Image-processing from Basics to Advance, like operations on Images, Videos using a huge set of Opencv-programs and projects.

TensorFlow:

TensorFlow is an open source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow.
Let us now consider the following important features of TensorFlow −

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data use.

Keras

Deep learning is one of the major subfields of machine learning framework. Machine learning is the study of design of algorithms, inspired from the model of human brain. Deep learning is becoming more popular in data science fields like robotics, artificial intelligence (AI), audio &

video recognition and image recognition. Artificial neural network is the core of deep learning methodologies. Deep learning is supported by various libraries such as Theano, TensorFlow, Caffe, Mxnet etc., Keras is one of the most powerful and easy to use python library, which is built on top of popular deep learning libraries like TensorFlow, Theano, etc., for creating deep learning models.

imutils

A series of convenience functions to make basic image processing functions such as translation as well as rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much easier with OpenCV and both Python 2.7 and Python 3.

5.2 EXECUTABLE CODE

```
# import the necessary packages

from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os

def detect_and_predict_mask(frame, faceNet, maskNet):

        # grab the dimensions of the frame and then construct a blob
        # from it

        (h, w) = frame.shape[:2]
        blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
                (104.0, 177.0, 123.0))

        # pass the blob through the network and obtain the face detections

        faceNet.setInput(blob)
        detections = faceNet.forward()
        print(detections.shape)

        # initialize our list of faces, their corresponding locations,
        # and the list of predictions from our face mask network
```

```
faces = []
locs = []
preds = []

# loop over the detections

for i in range(0, detections.shape[2]):

        # extract the confidence (i.e., probability) associated with
        # the detection

        confidence = detections[0, 0, i, 2]

        # filter out weak detections by ensuring the confidence is
        # greater than the minimum confidence

        if confidence > 0.5:

                # compute the (x, y)-coordinates of the bounding box for
                # the object

                box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                (startX, startY, endX, endY) = box.astype("int")

                # ensure the bounding boxes fall within the dimensions of
                # the frame

                (startX, startY) = (max(0, startX), max(0, startY))
                (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

                # extract the face ROI, convert it from BGR to RGB channel
                # ordering, resize it to 224x224, and preprocess it

                face = frame[startY:endY, startX:endX]
                face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
                face = cv2.resize(face, (224, 224))
                face = img_to_array(face)
                face = preprocess_input(face)

                # add the face and bounding boxes to their respective
                # lists
```

```
                faces.append(face)
                locs.append((startX, startY, endX, endY))

        # only make a predictions if at least one face was detected

        if len(faces) > 0:

            # for faster inference we'll make batch predictions on *all*
            # faces at the same time rather than one-by-one predictions
            # in the above `for` loop

            faces = np.array(faces, dtype="float32")
            preds = maskNet.predict(faces, batch_size=32)

        # return a 2-tuple of the face locations and their corresponding
        # locations

        return (locs, preds)

# load our serialized face detector model from disk

prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk

maskNet = load_model("mask_detector.model")

# initialize the video stream

print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()

# loop over the frames from the video stream

while True:

        # grab the frame from the threaded video stream and resize it
        # to have a maximum width of 400 pixels

        frame = vs.read()
        frame = imutils.resize(frame, width=400)
```

```
        # detect faces in the frame and determine if they are wearing a
        # face mask or not

        (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

        # loop over the detected face locations and their corresponding
        # locations

        for (box, pred) in zip(locs, preds):

                # unpack the bounding box and predictions
                (startX, startY, endX, endY) = box
                (mask, withoutMask) = pred

                # determine the class label and color we'll use to draw
                # the bounding box and text

                label = "Mask" if mask > withoutMask else "No Mask"
                color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

                # include the probability in the label

                label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

                # display the label and bounding box rectangle on the output
                # frame

                cv2.putText(frame, label, (startX, startY - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
                cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

        # show the output frame

        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop

        if key == ord("q"):
                break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

6. TESTING

6.1 TESTING DEFINITION:

- Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do.

- The benefits of testing include preventing bugs, reducing development costs and improving performance.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING:

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.
The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defect

6.3 Test cases

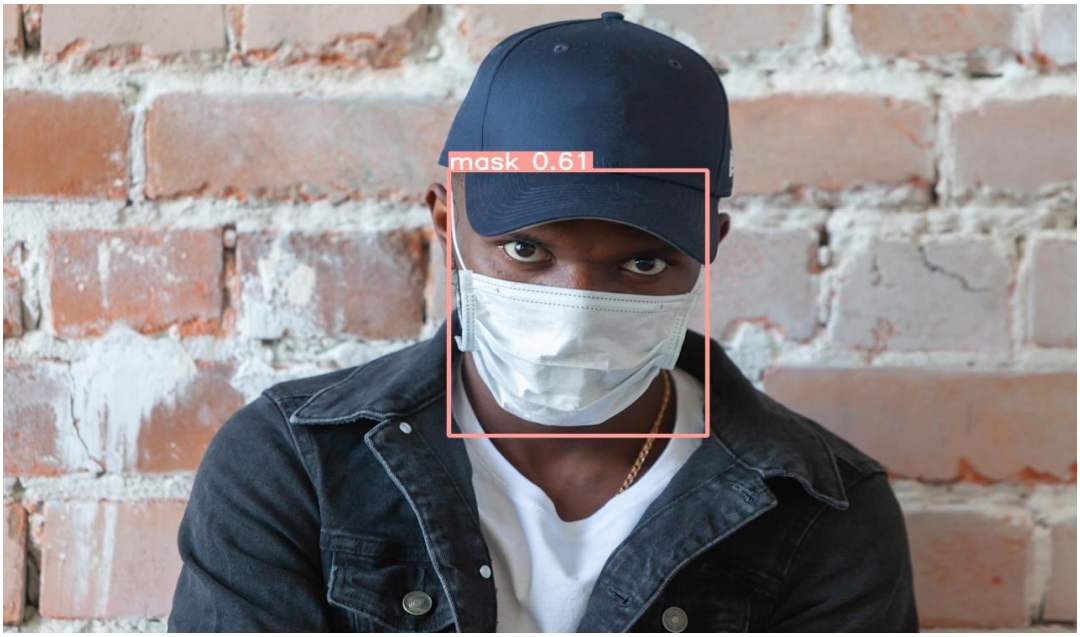
Tested	Test name	Inputs	Expected output	Actual Output	status
1	Load Dataset	Csv file	Read dataset	Load dataset	success
2	Split dataset	Train80% and test20%	Divide the training set and Testing set	Split train and Test	success
3	Train Model	Train dataset, random value, predicted class	Train with best accuracy	Train with best accuracy	success
4	Validate Model	No .of Epochs	Validate the Model with best fit	Model Generated	success
8	Testing Camera	Camera with low clarity	Predict Mask or No Mask	Not accurate	Failed

7 RESULTS

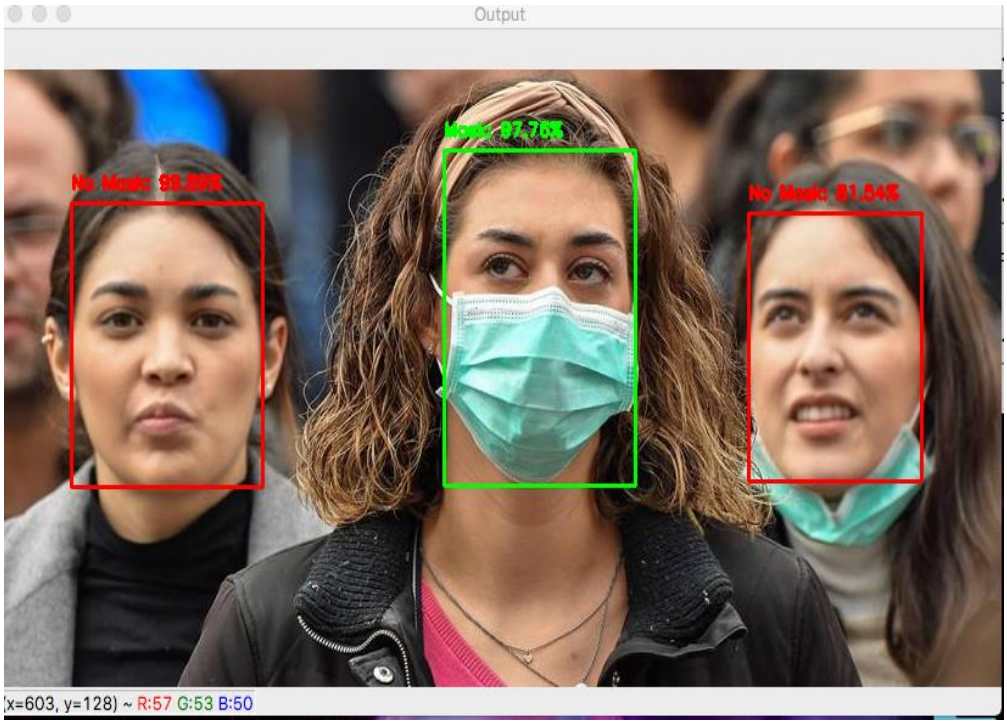
The model works to detect people wearing mask and not wearing masks. The model uses pre-trained YOLOv5 model which has 80% accuracy

7.1 Detection of fask mask through Images

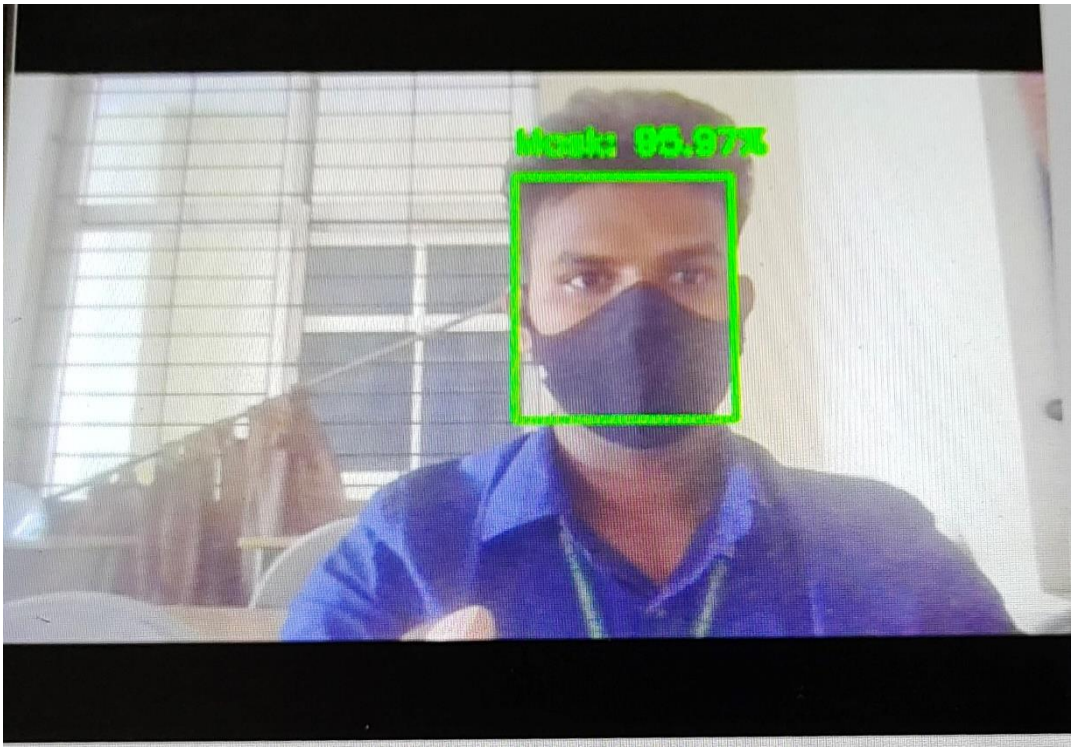
Screenshot-7.1.1 Input image with mask



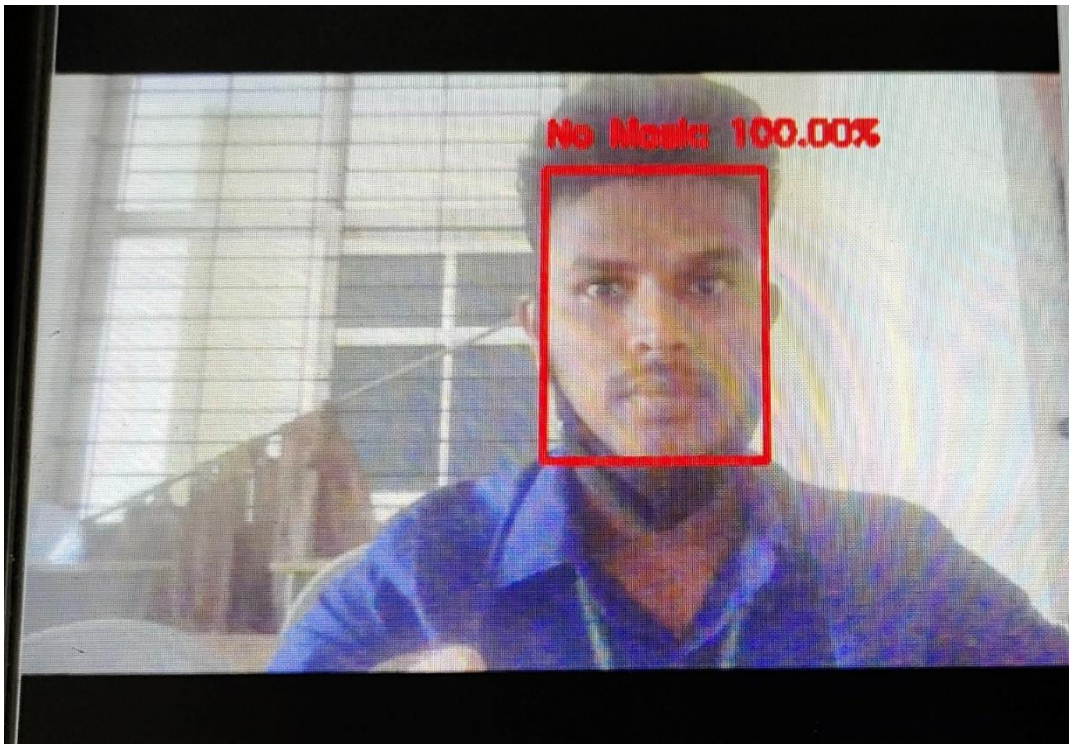
Screenshot-7.1.2 Result Detected By Model (with mask)



7.2 Detection of face mask through Live



Screenshot-7.2.1 Detection of mask through Live



Screenshot-7.2.2 Detection of No Mask through Live

8. CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

In this project the target of the developed model is to perform well for the detection of face mask and to detect the presence of the masks in real time image and video. In the set target, the developed model reached up-to an accuracy of 80%. In this, a minor drawback of the current developed model is that, it captures the image of all persons coming in the frame but sometimes results are not accurate, this is affecting the accuracy of the designed method because it should be applicable to everyone. This implies that if a person is wearing a mask but displaying it as not on mask sometimes is the defect in the model, A possible solution for this error is to use a detection algorithm in the second layer as well, which may compromise the speed and may require more sophisticated hardware.

8.2 Future Scope

This work developed the face mask detection by using YOLO V5 algorithm. The YOLO V5 algorithm consists of deep learning method which is able to detect the object properly.
From the experiment results, the algorithm is able to detect and distinguish a not-wearing and a wearing- mask precisely with any condition of surrounding environment.

In future, we focus on generated more dataset from the complex scenario of population in India and also performed experimentation on future deep learning models for improving the detection of mask wear or not by people.

9. REFERENCES

[1] M. S. Ejaz and M. R. Islam, "Masked Face Recognition Using Convolutional Neural Network," 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), 2019, pp. 1-6, doi: 10.1109/STI47673.2019.9068044.

[2] M. R. Bhuiyan, S. A. Khushbu and M. S. Islam, "A Deep Learning Based Assistive System to Classify COVID-19 Face Mask for Human Safety with YOLOv3," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)

[3] M. M. Rahman, M. M. H. Manik, M. M. Islam, S. Mahmud and J. -H. Kim, "An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network," 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), 2020

[4] Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” in Advances in neural information processing systems, 2014, pp. 198hy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” 2014.

[5] F. S. Samaria and A. C. Harter, “Parameterisation of a stochastic model for human face identification,” in Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on, pp. 138–142, IEEE, 1994.

[6] D. Yi, Z. Lei, S. Liao, and S. Z. Li, “Learning face representation from scratch,” CoRR

abs/1411.7923, 2014.
[7] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, "A practical transfer learning algorithm for face verification," in Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 3208–3215, IE8–1996.
[8] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," ACM computing surveys (CSUR), vol. 35, no. 4, pp. 399–458, 2003.
[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. KarpatEE, 2013.
[10] P. N. Belhumeur, J. P. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," Pattern Analysis and Machine Intelligence, IEEE Transactions on 19(7), pp. 711– 720, 1997.
[11] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, "A practical transfer learning algorithm for face verification," in Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 3208–3215, IEEE, 2013.
[12] Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. CoRR, abs/1406.4773, 2014. 1, 2, 3
[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. Nature, 1986. 2, 4