A MINI PROJECT REPORT

ON

FACE MASK DETECTION SYSTEM

For the partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

Vinay kumar 2101921520194

Under the Supervision of

Ms. Nidhi Gupta

G.L. BAJAJ INSTITUTE OF TECHNOLOGY & MANAGEMENT, GREATER NOIDA

Affiliated to

DR. APJ ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

2022-23

Declaration

I hereby declare that the project work presented in this report entitled **"FACE MASK DETECTION SYSTEM",** in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Science & Engineering, submitted to A.P.J. Abdul Kalam Technical University, Lucknow, is based on my own work carried out at Department of Computer Science & Engineering, G.L. Bajaj Institute of Technology & Management, Greater Noida. The work contained in the report is original and project work reported in this report has not been submitted by me/us for award of any other degree or diploma.

Signature:
Name: Vinay kumar
Roll No:210192152194

Date:
Place: Greater Noida

Certificate

This is to certify that the Project report entitled **"Face Mask Detection System" done by Vinay kumar (2101921520194)** of Branch CsAI is an original work carried out by them in Department of Computer Science & Engineering, G.L Bajaj Institute of Technology & Management, Greater Noida under my guidance. The matter embodied in this project work has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

Date:

Ms. Nidhi Gupta
Signature of the Supervisor

Dr. Sansar Singh Chauhan
Head of the Department

Acknowledgement

The merciful guidance bestowed to us by the almighty made us stick out this project to a successful end. We humbly pray with sincere heart for his guidance to continue forever.

I pay thanks to our project guide **Ms. Nidhi Gupta** who has given guidance and light to us during this project. Her versatile knowledge has cased us in the critical times during the span of this project.

I pay special thanks to our Head of Department **Dr. Sansar Singh Chauhan** who has been always present as a support and help us in all possible way during this project

I also take this opportunity to express our gratitude to all those people who have been directly and indirectly with us during the completion of the project.

I want to thanks our friends who have always encouraged us during this project.

At the last but not least thanks to all the faculty of CSE department who provided valuable suggestions during the period of project.

Abstract

Global pandemic COVID-19 circumstances emerged in an epidemic of dangerous disease in all over the world. Wearing a face mask will help prevent the spread of infection and prevent the individual from contracting any airborne infectious germs. Using Face Mask Detection System, one can monitor if the people are wearing masks or not. Here HAAR-CASACADE algorithm is used for image detection. Collating with other existing algorithms, this classifier produces a high recognition rate even with varying expressions, efficient feature selection and low assortment of false positive features. HAAR feature-based cascade classifier system utilizes only 200 features out of 6000 features to yield a recognition rate of 85-95%. According to this motivation we demand mask detection as a unique and public health service system during the global pandemic COVID-19 epidemic. The model is trained by face mask image and non-face mask image. Keywords: COVID-19 epidemic, HAAR-CASACADE algorithm, mask detection, face mask image, non-face mask image.

TABLE OF CONTENT

LIST OF FIGURES

Chapter 1

Introduction

1.1 Overview

The following is a quick summary of the suggested approach:

1.1.1 Dataset Collection: Images from various sources are used to build a dataset. The size of datasets can be expanded by the application of data enhancement techniques. The photographs are stored in two files, “training dataset” and “test dataset,” each of which comprises 80 and 20% of the images, respectively. Bounding boxes, sometimes known as “data annotations,” are created around an area of interest using a variety of methods. Labelling pictures as “mask” or “NO mask” will be done using the LabelImg tool in the proposed system.

1.1.2 Image Enhancement: To draw attention to the foreground elements, the image is improved through pre-processing methods and segmentation techniques.

1.1.3 Model Implementation: We ran the tests on an Intel Core i7 processor with a Nvidia GTX 1,080 graphics card and Windows 10. Python 3.5 was used as the programming language in this project.

1.1.4 Training the model: To distinguish between those wearing “masks” and those who aren't, the model is trained in an online GPU environment called Google Colab. A folder referred to as “the trained folder” is used for training purposes.

1.1.5 Prediction: Using the test folder, the model is tested for its ability to identify and classify masks and no-masks that were found in the original photos.

1.2 Background

1.2.1 An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network [1]: COVID-19 pandemic caused by novel coronavirus is continuously spreading until now all over the world. The impact of COVID-19 has been fallen on almost all sectors of development. The healthcare system is going through a crisis. Many precautionary measures have been taken to reduce the spread of this disease where wearing a mask is one of them. In this paper, we propose a system that restrict the growth of COVID-19 by finding out people who are not wearing any facial mask in a smart city network where all the public places are monitored with Closed-Circuit Television (CCTV) cameras. While a person without a mask is detected, the corresponding authority is informed through the city network. A deep learning architecture is trained on a dataset that consists of images of people with and without masks collected from various sources. The trained architecture achieved 98.7% accuracy on distinguishing people with and without a facial mask for previously unseen test data. It is hoped that our study would be a useful tool to reduce the spread of this communicable disease for many countries in the world.

1.2.2 Masked Face Recognition Using Convolutional Neural Network [2]: Recognition from faces is a popular and significant technology in recent years. Face alterations and the presence of different masks make it too much challenging. In the real-world, when a person is uncooperative with the systems such as in video surveillance then masking is further common scenarios. For these masks, current face recognition performance degrades. An abundant number of researches work has been performed for recognizing faces under different conditions like changing pose or illumination, degraded images, etc. Still, difficulties created by masks are usually disregarded. The primary concern to this work is about facial masks, and especially to enhance the recognition accuracy of different masked faces. A feasible approach has been proposed that consists of first detecting the facial regions. The occluded face detection problem has been approached using Multi-Task Cascaded Convolutional Neural Network (MTCNN). Then facial features extraction is performed using the Google Face Net embedding model.

It is not feasible to manually track the implementation of this policy. Technology holds the key here. We introduce a Deep Learning based system that can detect instances where face masks are not used properly. Our system consists of a dual stage Convolutional Neural Network (CNN) architecture capable of detecting masked and unmasked faces and can be integrated with pre-installed CCTV cameras. This will help track safety violations, promote the use of

face masks, and ensure a safe working environment. The study is carried out by analysing the necessary technologies involved in the previous research and formulating the efficient model that help the people in real-time. Application development using keras and Tensorflow is most widely used in current trends, henceforth we are suggesting the Python based image processing and machine learning technique to achieve the robust structure the feasibility of considering a diversity of cases using the proposed method with a high level of confidence and accuracy in social distancing monitoring and risk assessment with the help of Deep Learning and Computer Vision.

1.3 Motivation

The world has not yet fully Recover from this pandemic and the vaccine that can effectively treat Covid-19 is yet to be discovered. However, to reduce the impact of the pandemic on the country's economy, several governments have allowed a limited number of economic activities to be resumed once the number of new cases of Covid-19 has dropped below a certain level. As these countries cautiously restarting their economic activities, concerns have emerged regarding workplace safety in the new post-Covid-19 environment.

To reduce the possibility of infection, it is advised that people should wear masks and maintain a distance of at least 1 meter from each other. Deep learning has gained more attention in object detection and was used for human detection purposes and develop a face mask detection tool that can detect whether the individual is wearing mask or not. This can be done by evaluation of the classification results by analysing real-time streaming from the Camera. In deep learning projects, we need a training data set. It is the actual dataset used to train the model for performing various actions. There has been a tremendous amount of interest in deep learning in the past few years, notably in fields like machine vision, text analytics, object recognition, and other information processing aspects. The majority of the past research in object detection has been conducted using convolutional neural network models. Using deep learning architectures, convolutional neural networks (CNNs) have become more popular in recent years for a variety of tasks, such as picture identification, speech synthesis, object tracking, and image thresholding. When it comes to the abovementioned domains, CNN exhibits an excellent capacity to retrieve features from images. A growing number of research methods are replacing traditional classification methods with CNNs in order to more effectively capture image information and achieve improved classification performance. Due to energy limitations, numerous deep neural networks are unsuited for mobile-based facial image classification since

their evaluation phase is time consuming and expensive. We describe a Mobile Net-based facial image classification model that uses a Depth wise separable convolution technique to handle this problem. DSC (Depth wise separable convolution) was first presented in and is commonly used in image processing for classification tasks. The Depth wise separable convolution is a quantized version of the ordinary convolution. Convolutions are often separated into Depth wise and 1×1 pointwise convolution. Rather than applying each filter to all input channels as in traditional convolution, the Depth wise convolution layer applies one filtering to one pulse and then uses a 1×1 pointwise convolution to combine the Depth wise convolution results. Depth wise separable convolution reduces the number of learnable parameters and the expense of test and train computations.

1.4 Methodology

1.4.1 PROPOSED SYSTEM:

1.This system is capable to train the dataset of both persons wearing masks and without wearing masks.

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

3.It also can access the webcam and predict the result.

1.4.2 TENSORFLOW FRAMEWORK:

Tensor flow is an open-source software library. Tensor flow was originally developed by researchers and engineers. It is working on the Google Brain Team within Google's Machine Intelligence research organization the purposes of conducting machine learning and deep neural networks research.

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

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

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

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

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

Version 1.0.0 was released on February 11, While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units).

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

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

The name Tensor Flow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

1.4.3 OPENCV:

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

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

3. Currently Open CV supports a wide variety of programming languages like C++, Python,

Java etc. and is available on different platforms including Windows, Linux, OS X, Android, iOS etc.

4. Also, interfaces based on CUDA and OpenCL are also under active development for

high-speed GPU operations. Open CV-Python is the Python API of Open CV.

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

6. OpenCV (Open-Source Computer Vision Library) is an opensource computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications

and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

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

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

1.4.4 NUMPY:

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

The Python programming language was not initially designed for numerical computing, but attracted

the attention of the scientific and engineering community early on, so that a special interest group called matrix-sig was founded in 1995 with the aim of defining an array computing package. Among its members was Python designer and maintainer Guido van Rossum, who implemented extensions to Python's syntax (in particular the indexing syntax) to make array computing easier.

An implementation of a matrix package was completed by Jim Fulton, then generalized by Jim Hugunin to become Numeric also variously called Numerical Python extensions or NumPy

Hugunin, a graduate student at Massachusetts Institute of Technology (MIT) joined the Corporation for National Research Initiatives (CNRI) to work on J Python in 1997 leaving Paul Dubois of Lawrence Livermore National Laboratory (LLNL) to take over as maintainer.

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

1.4.5 MATPLOT:

Mat plot is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, WX Python, Qt, or GTK+. There is also a procedural "Pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged SciPy makes use of Matplotlib.

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

1.4.6 PYTHON CONCEPTS:

Python was developed into an easy-to-use programming language. It uses English words instead of punctuation, and has fewer syntax than other languages. Python is a highly developed, translated, interactive, and object-oriented language.

Python translated - Interpreter processing Python during launch. Before using your software, you do not need to install it. This is similar to PERL and PHP editing languages.

Python interactive - To write your own applications, you can sit in Python Prompt and communicate directly with the interpreter.

Python Object-Oriented - Python supports the Object-Oriented program style or method, encoding the code within objects.

Python is a language for beginners - Python is an excellent language for beginners, as it allows for the creation of a variety of programs, from simple text applications to web browsers and games.

1.4.6.1 Python Features

Python features include -

1.Easy-to-learn - Python includes a small number of keywords, precise structure, and well-defined syntax. T This allows the student to learn the language faster

2.Easy to read - Python code is clearly defined and visible to the naked eye.

3.Easy-to-maintain - Python source code is easy to maintain.

4.Standard General Library - Python's bulk library is very portable and shortcut compatible with UNIX, Windows, and Macintosh.

5.Interaction mode - Python supports interaction mode that allows interaction testing and correction of captions errors.

6.Portable - Python works on a variety of computer systems and has the same user interface for all.

7.Extensible - Low-level modules can be added to Python interpreter. These modules allow system developers to improve the efficiency of their tools either by installing or customizing them.

8.Details - All major commercial information is provided by Python ways of meeting.

9.GUI Programming - Python assists with the creation and installation of a user interface for images of various program phones, libraries, and applications, including Windows MFC, Macintosh, and Unix's X Window.

10.Scalable - Major projects benefit from Python building and support, while Shell writing is not.

1.4.6.2 Python Numbers

Numeric values are stored in number data types. When you give a number a value, it becomes a number object.

1.4.6.3 Python Strings

In this python uses a string is defined as a collection set of characters enclosed in quotation marks. Python allows you to use any number of quotes in pairs. The slice operator ([] and [:]) can be used to extract subsets of strings, with indexes starting at 0 at the start of the string and working their way to -1 at the end.

1.4.6.4 Python Lists

The most diverse types of Python data are lists. Items are separated by commas and placed in square brackets in the list ([]). Lists are similar to C-order in some ways. Listings can be for many types of data, which is one difference between them.

The slide operator ([] and [:]) can be used to retrieve values stored in the list, the indicators start with 0 at the beginning of the list and work their way to the end of the list. The concatenation operator of the list is a plus sign (+), while the repeater is an asterisk (*).

1.4.6.5 Python Tuples

A cone is a type of data type similar to a sequence of items. Cone is a set of values

separated by commas. The pods, unlike the list, are surrounded by parentheses.

Lists are placed in parentheses ([]), and the elements and sizes can be changed, but the lumps are wrapped in brackets (()) and cannot be sorted. Powders are the same as reading lists only.

1.4.6.6 Python Dictionary

Python dictionaries in Python are a way of a hash table. They are similar to Perl's combination schemes or hashes, and are made up of two key numbers.

The dictionary key can be any type of Python, but numbers and strings are very common. Prices, on the other hand, can be anything you choose Python.

Curly braces () surround dictionaries, and square braces ([) are used to assign and access values.

Different modes in python

Python Normal and interactive are the two basic Python modes.

The scripted and completed.py files are executed in the Python interpreter in the regular manner.

Interactive mode is a command line shell that provides instant response for each statement while simultaneously running previously provided statements in active memory.

The feed programme is assessed in part and whole as fresh lines are fed into the interpreter.

1.4.6 PANDAS

Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analysing data much easier. Pandas is fast and it has high-performance & productivity for users.

Pandas is a python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labelled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible opensource data analysis/manipulation tool available in any language. It is already well on its way toward this goal.

Explore data analysis with Python. Pandas Data Frames make manipulating your data easy, from selecting or replacing columns and indices to reshaping your data. Pandas is well suited for many different kinds of data:

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

Ordered and unordered (not necessarily fixed-frequency) time series data.

Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels

Any other form of observational / statistical data sets. The data need not be labelled at all to be placed into a Pandas data structure.

1.4.7 KERAS

KERAS is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages.

It also has extensive documentation and developer guides. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. Keras is a minimalist Python library for deep learning that can run on top of Theano or Tensor Flow.

It was developed to make implementing deep learning models as fast and easy as possible for research and development.

FOUR PRINCIPLES:

- Modularity: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.
- Minimalism: The library provides just enough to achieve an outcome, no frills and maximizing readability.
- Extensibility: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.
- Python: No separate model files with custom file formats. Everything is native Python. Keras is designed for minimalism and modularity allowing you to very quickly define deep learning models and run them on top of a Theano or TensorFlow backend.

1.4.8 Machine Learning approaches:

1.4.8.1. Viola–Jones object detection framework based on HAAR Features

1.4.8.2. Scale-invariant feature transform (SIFT)

1.4.8.3. Histogram of oriented gradients (HOG) features

Machine learning (ML)is the study of computer algorithms that improve automatically through experience. Itis seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Approaches for Machine Learning:

1.4.8.1 Viola-Jones Object detection framework based in HAAR features:

The Viola-Jones algorithm is one of the most popular algorithms for objects recognition in an image. This research paper deals with the possibilities of parametric optimization of the Viola-Jones algorithm to achieve maximum efficiency of the algorithm in specific environmental conditions. It is shown that with the use of additional modifications it is possible to increase the speed of the algorithm in a

particular image by 2-5 times with the loss of accuracy and completeness of the work by not more than the 3-5%.

1.4.8.2 Scale-invariant feature transform (SIFT):

The scale-invariant feature transform (SIFT) is a feature detection algorithm in computer vision to detect and describe local features in images. SIFT key points of objects are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature

vectors. From the full set of matches, subsets of key points that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches. The determination of consistent clusters is performed rapidly by using an efficient hash table implementation of the generalized Hough transform. Each cluster of 3 or more features that agree on an object and its pose is then subject to further detailed model verification and subsequently outliers are discarded, Finally the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable false matches. Object matches that pass all these tests can be identified as correct with high confidence.

1.4.8.3 Histogram of oriented gradients (HOG):

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object-detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts.

1.4.9 DEEP LEARNING

1.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.

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

3.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.

1.4.10 NEURAL NETWORKS VERSUS CONVENTIONAL COMPUTERS:

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

Neural networks learn by example. They cannot be programmed to is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable. On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high-level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

1.4.11 ARCHITECTURE OF NEURAL NETWORKS:

1.4.11.1 FEED-FORWARD NETWORKS:

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with

outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down. to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

1.4.11.2 FEEDBACK NETWORKS:

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

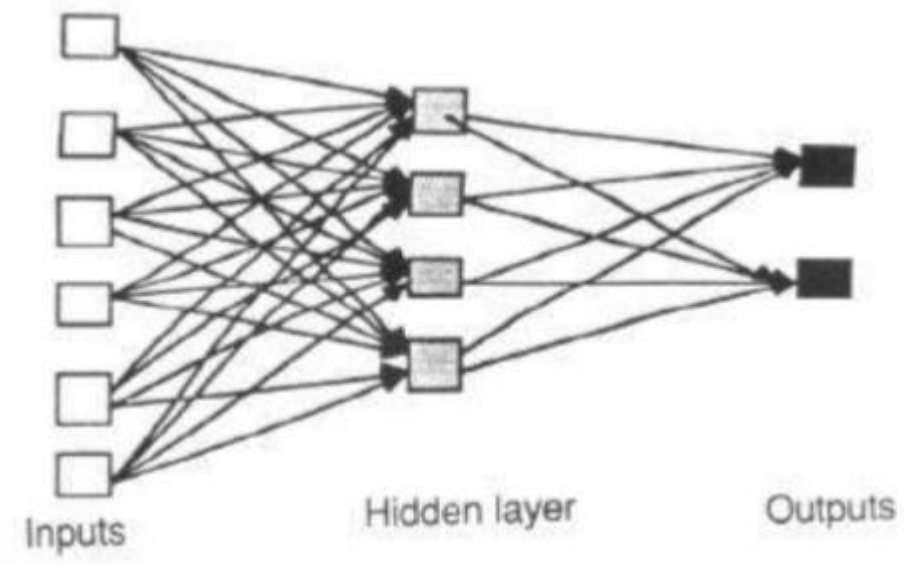

Figure 1.1 Layers in NN

1.4.11.3 NETWORK LAYERS:

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of input units is connected to a layer of hidden units, which is connected to a layer of output units.

The activity of the input units represents the raw information that is fed into the network.

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

1.4.12 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 connected layers, then generates the 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 analysing 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 stages, followed by a classification module.

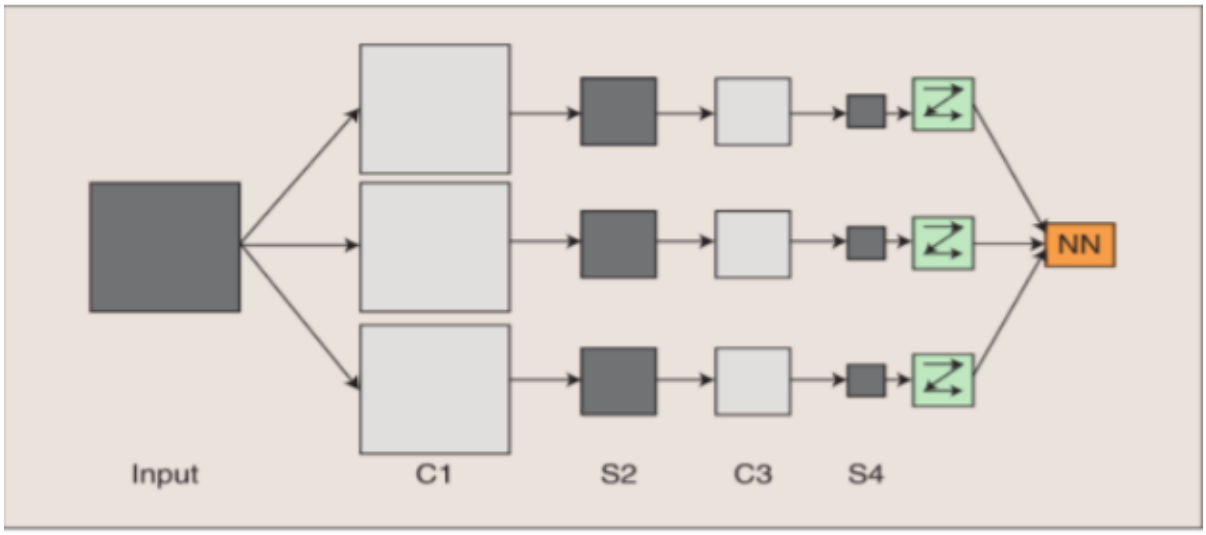

Figure 1.2 Basic structure of CNN

, where C1, C3 are convolution layers and S2, S4 are pooled/sampled layers.

Filter: A trainable filter (kernel) in filter bank connects input feature map to output feature map Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli.

1.4.13 CONVOLUTIONAL LAYER

It is always first. The image (matrix with pixel values) is entered into it. Image that the reacting of the input matrix begins at the top left of image. Next the software selects the smaller matrix

there, which is called a filter. Then the filter produces convolution that is moves along the input image. The filter task is to multiple its value by the original pixel values. All these multiplications are summed up and one number is obtained at the end. Since the filter has read the image only in the upper left corner it moves further by one unit right performing a similar operation. After passing the filter across all positions, a matrix is obtained, but smaller than an input matrix.

This operation, from a human perspective is analogous to identifying boundaries and simple Colors on the image. But in order to recognize the fish whole network is needed. The network will be -consists of several convolution layers mixed with nonlinear and pooling layers. Convolution is the first layer to extract features from an input image. Convolution features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

● An image matrix of dimension (h x w x d)

● A filter (fh x fw x d)

● Outputs a volume dimension (h-fh+1) x (w-fw+1) x1.

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below

Figure 1.3 convolution with a filter example

Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called “Feature Map” as output shown in next page

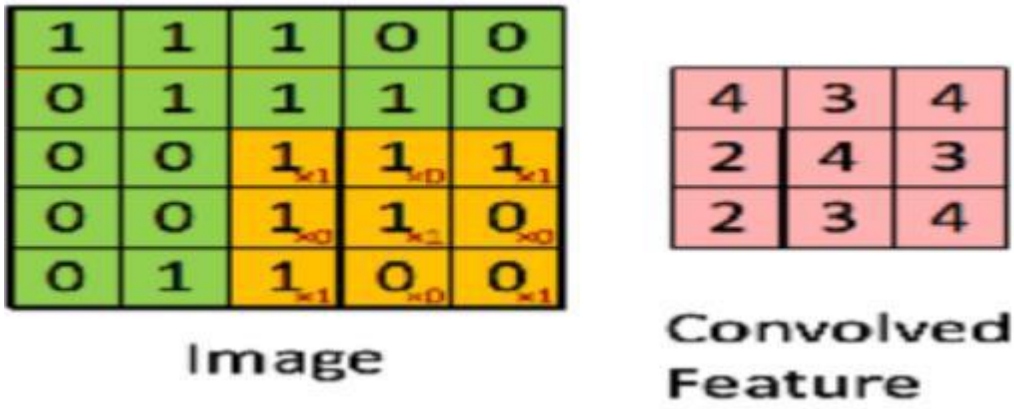

Figure 1.4 Output of Convolution layer

1.4.14 THE NON-LINEAR LAYER:

It is adder after each convolution operation. It has the activation function, which brings nonlinear property, without this property a network would not be, sufficiently intense and will not be able to model the response variable**.**

1.4.15 THE POOLING LAYER:

It follows the nonlinear layer. It works with width and height of the image and performs a down sampling operation on them. As a result, image volume is reduced. This means that if some features already been identified in the previous convolution operation, then a detailed image is no longer needed for further processing and is compressed to less detailed pictures.

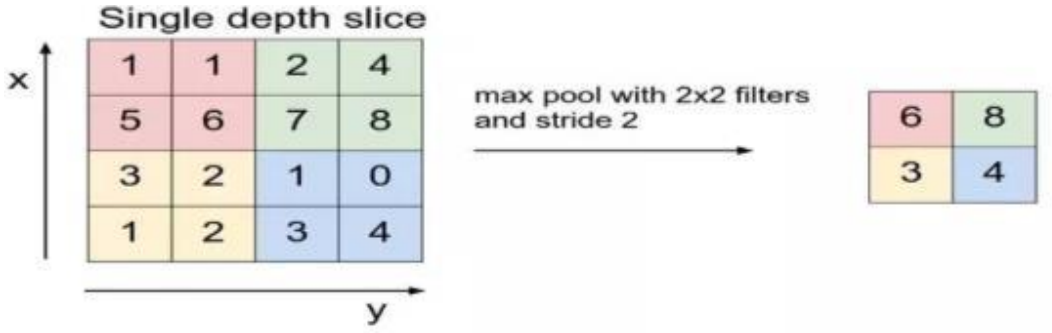

Figure 1.5 Max Pooling Layer

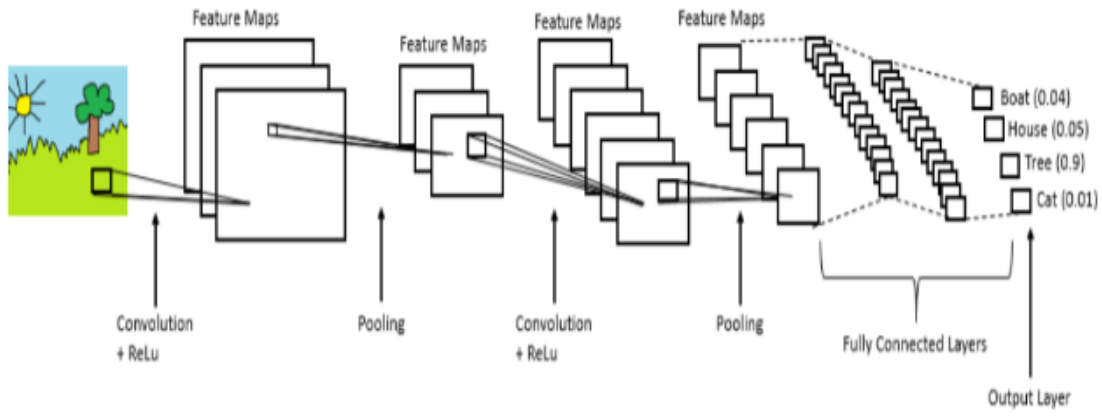

Figure 1.6 Overall structure of CNN

1.4.16 FULLY CONNECTED LAYER:

After completion of series of convolution, non-linear and pooling layer, it's necessary to attach a fully connected layer. This layer takes the output information from convolution network. Attaching a fully connected layer to the end of the network results in N dimensional vector, where N is the number of classes from which the model selects the desired class

1.4.17 CNN MODEL

1.This CNN model is built using the Tensorflow framework and the OpenCv library which is highly used for real-time applications.

2.This model can also be used to develop a full-fledged software to scan every person before they can enter the public gathering

1.4.17.1 LAYERS IN CNN MODEL

Conv2D

MaxPooling2D

Flatten ()

Dropout

Dense

Conv2D Layer:

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

2.MaxPooling2D:

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

3.Flatten () Layer:

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

4.Dropout Layer:

It is used to prevent the model from overfitting.

5.Dense Layer:

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

1.4.18 SYSTEM ARCHITECTURE:

Data Visualization.

Data Augmentation.

Splitting the data.

Labelling the Information.

Importing the Face detection.

Detecting the Faces with and without Masks.

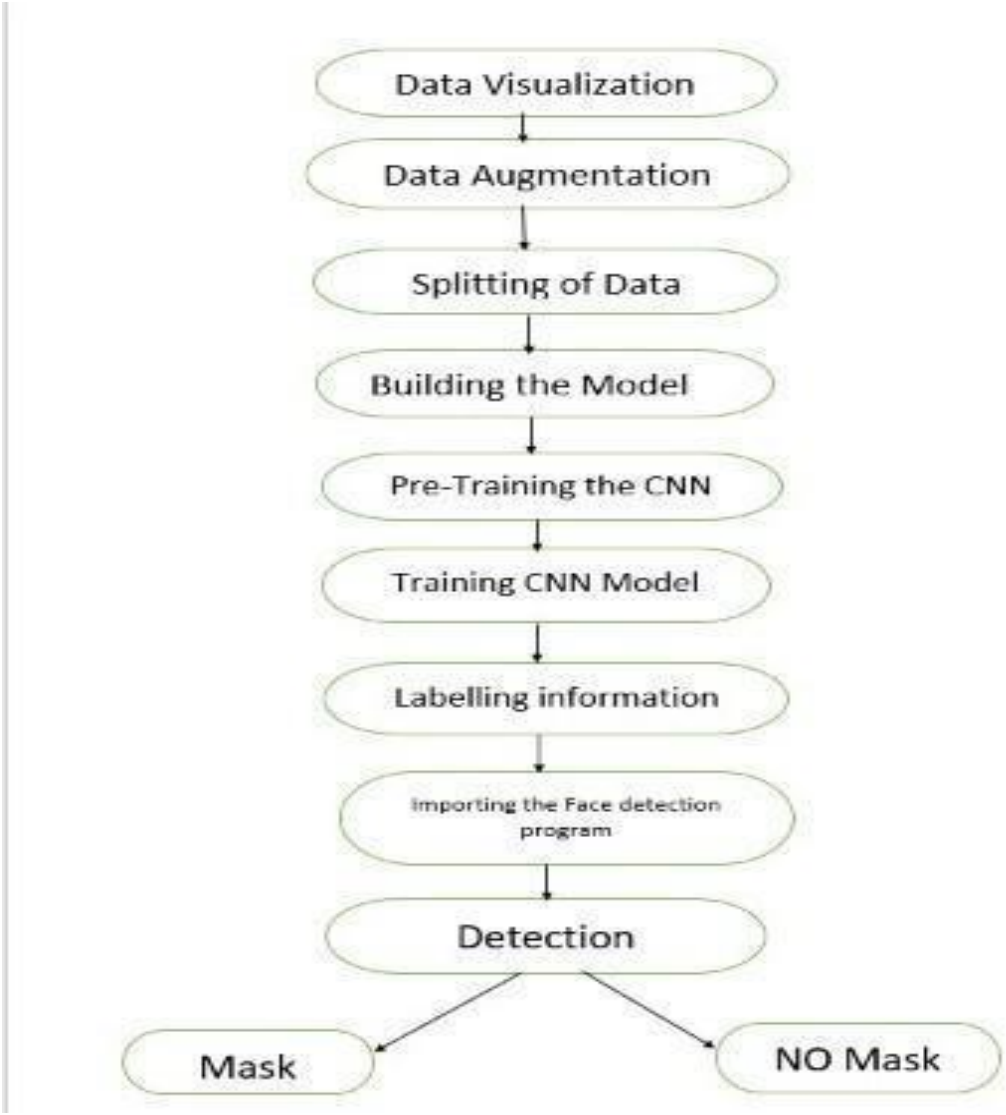

Figure 1.7 System architecture

Data Visualization

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

Data Augmentation

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

Splitting the data

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

Building the Model

In the next step, we build our Sequential CNN model with various layers such as Conv2D, MaxPooling2D, Flatten, Dropout and Dense.

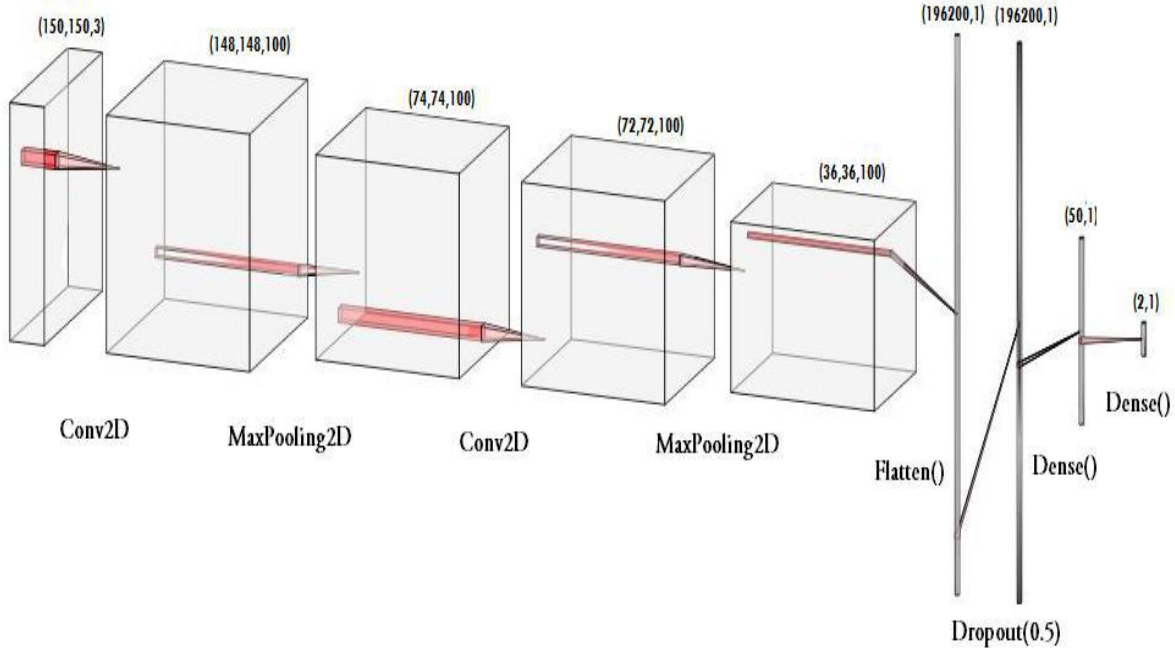

Figure 1.8 Building the model

Pre-Training the CNN model

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

Training the CNN model

This step is the main step where we fit our images in the training set and the test set to our Sequential model, we built using keras library. I have trained the model for 30 epochs

(iterations). However, we can train for more number of epochs to attain higher accuracy lest there occurs over-fitting.

Labelling the Information

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

Importing the Face detection Program

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

Detecting the Faces with and without Masks

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

Chapter 2

EXISTING SYSTEM

2.1 INTRODUCTION

An existing software system is any software application that is currently in use. It includes everything from newly released software to those that have existed for years. Many systems reach a point sometime during their operational life when they need to be thoroughly examined to determine whether they are supportable and can continue to be supported.

Some indicators of potential maintenance problems include:

• It has existed longer than originally planned.

• Its changes have been poorly captured.

• How it works is difficult to understand.

• It resists modification; i.e., the incorporation of changes is difficult, unexpected

defects show up when upgrades or enhancements occur.

• Its change management process is inadequate or non-existent.

• Its documentation is minimal or non-existent; thus, maintenance is performed using

source code.

• Its hardware or primary programming language is no longer supported by industry.

2.2 EXISTING SYSTEM

face detection problem has been approached using Multi-Task Cascaded Convolutional Neural Network (MTCNN). Then facial features extraction is performed using the Google Face Net embedding model.

1.This system is capable to train the dataset of both persons wearing masks and without wearing masks.

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

Chapter 3

PROBLEM FORMULATION

3.1 PROBLEM STATEMENT

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.

3.2 CAUSES

The rapid spreading of Coronavirus disease (COVID-19) is a major health risk that the whole world is facing for the last two years. One of the main causes of the fast spreading of this virus is the direct contact of people with each other. There are many precautionary measures to reduce the spread of this virus; however, the major one is wearing face masks in public places. Detection of face masks in public places is a real challenge that needs to be addressed to reduce the risk of spreading the virus. To address these challenges, an automated system for face mask detection using deep learning (DL) algorithms has been proposed to control the spreading of this infectious disease effectively.

3.3 EFFECTS

spread of the Covid-19 virus has become an emerging health problem all around the world, to slow down its devastating effects on societies and economies, World Health Organization (WHO) has imposed many guidelines. These guidelines include wearing face masks, maintaining social distancing, adopting of virtual work culture, and many more. Among all these guidelines, face mask detection is one of the innovative technologies which can help in identifying the number of people wearing a face mask, regulating proper social distancing, and preventing a huge mass from the severity of the infection. The facemask detection technology is operated with the use of different programming languages and software such as Python, Deep-Learning, Tensor Flow, Keras, OpenCV, and many more.

3.4 REQUIREMENTS

3.4.1 Software Requirements

• Python 3.5 in Google Colab is used for data pre-processing, model training and prediction

• Operating System: windows 7 and above or Linux based OS or MAC OS

• Coding Language: Python.

3.4.2 Hardware Requirements

• RAM: 4 GB

• Storage: 500 GB

• CPU: 2 GHz or faster

• Architecture: 32-bit or 64-bit

3.5 OBJECTIVE

The main objective of “Face mask detection” project is to provide some effective technology for preventing the spread of Corona virus. Primary objectives behind the development of this system are as follows: -

• Prevent the spread of Corona virus by promoting the use of face masks with the help of effective technology to detect the face mask.

• Help to take necessary precautions for the safety of society by predicting the future outbreaks of COVID-19.

• Ensure a safe working environment.

• Save the lives of people.

Chapter 4

SYSTEM ANALYSIS & DESIGN

4.1 FEATURES

Key Features of a Face Mask Detection System

Keep your visitors safe without approaching them — Deep Sight software offers real-time face recognition from video streams and images, powered by computer vision and deep learning.

Intelligent Alerts

Send custom messages to people who are not following the rules or configure public announcements to be auto-displayed, across a variety of mediums.

Facial Recognition

State-of-the-art facial recognition algorithm, pre-trained, and tested across a wide range of datasets to ensure high accuracy rates

Camera Agnostic

Compatible with all modern CCTV, USB, and IP camera systems, connected to the Internet. Run the software on the equipment you already have.

Easy Implementation

Non-invasive and light-weight, our pre-trained algorithm can be fully operational in a matter of days with remote support from our ML specialists.

Staff Friendly

Avoid workplace tensions and in-person confrontations with a system designed to proactively nudge people who are not wearing masks

No New Hardware Needed

Our cloud-based system can be connected remotely to any on-premises camera systems and computers, so you don’t need to invest in anything extra.

4.2 SPECIFICATION

4.2.1 FUNCTIONAL SPECIFICATION

The primary purpose of computer results is to deliver processing results to users. They are also employed to maintain a permanent record of the results for future use.

In general, the following are many types of results:

External results are those that are exported outside the company.

Internal results, which are the main user and computer display and have a place within the organization.

Operating results used only by the computer department. User-interface results that allow the user to communicate directly with the system. Understanding the user's preferences, the level of technology and the needs of his or her business through a friendly questionnaire.

4.2.1 SYSTEM CONFIGURATION

This project can run on commodity hardware. We ran entire project on an Intel I5 processor with 8 GB Ram, 2 GB Nvidia Graphic Processor, it also has 2 cores which runs at 1.7 GHz, 2.1 GHz respectively. First part of the is training phase which takes 10-15 mins of time and the second part is testing part which only takes few seconds to make predictions and calculate accuracy.

4.3 USER INTERFACE

A friendly Graphical User Interface (GUI) is necessary for all individuals irrespective of their technical knowledge to operate, hence we tried to create a GUI based system using tkinter which consists of primary six blocks containing images for its respective use case. The blocks include options such as “Facemask”, “Social Distancing”, “Facemask & Social Distancing”, “Manual inform”, “Automatic inform” and “Quit”.

Chapter 5

IMPLEMENTATION

5.1 MODULES

1.Creating image datasets a data-loaders for train and test using the experiments folder split

● Training Dataset: A dataset that we feed into our algorithm to train our model.

● Testing Dataset: A dataset that we use to validate the accuracy of our model but is not used to train the model. It may be called the validation dataset.

2.Training the model

3. Visualizing images

5.2 IMPLEMENTATION

5.2.1 Preparing dataset

Two image datasets were collected from the kaggle website for the model's training and testing purposes. (Kaggle 2022.) Following that, image dataset was split into two groups: one Images_with_mask and one Images_ without mask. More images have also been added to the dataset, which were taken with the laptop's default camera. Using these datasets, it is possible to develop a model that can distinguish between people wearing masks and people who are not wearing masks. These datasets contain around 1400 images from the folders "Images_with_mask" and "Images_without_mask". Images with mask and without mask can be seen in FIGURE.

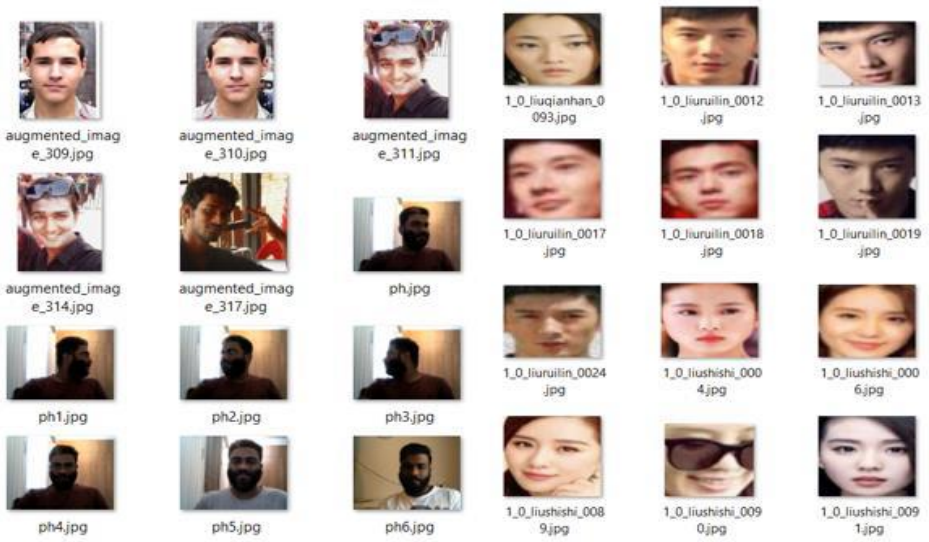

Figure 5.1 Without mask

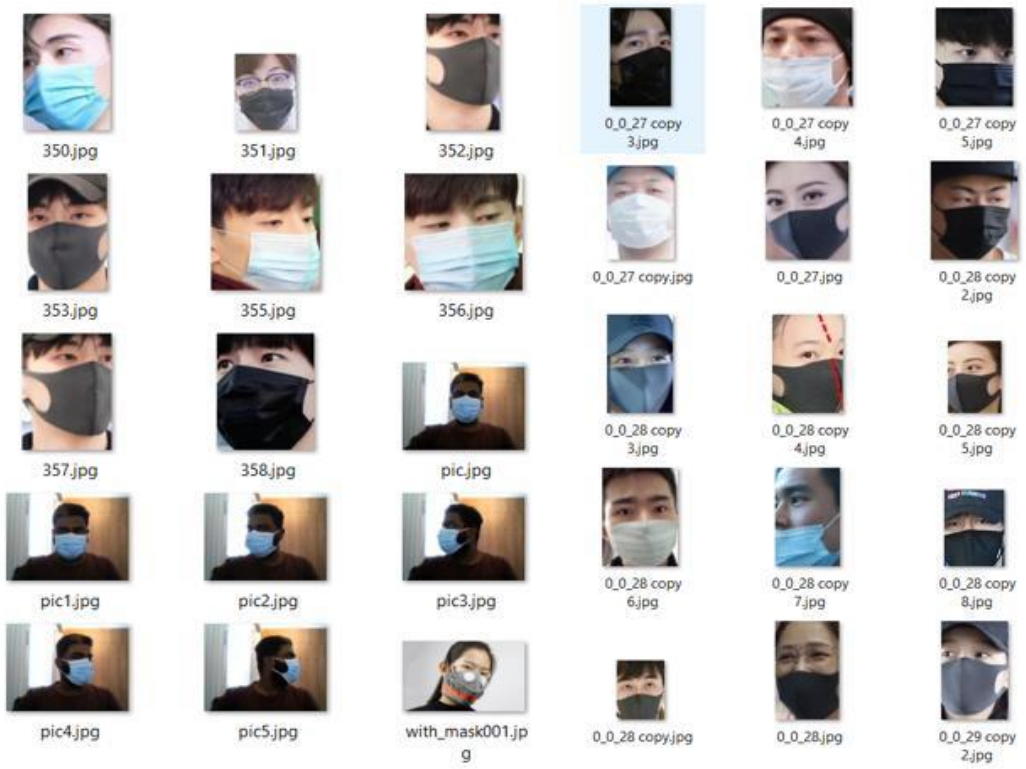

Figure 5.2 With mask

5.2.2 Train neural network model

To begin, the libraries needed for the implementation process were imported, including tensorflow, numpy, imutils, keras, opencv, scipy, and matplotlib. Datasets were also imported using the import technique. Model's directory has been created. In this area, there is category. Within the category, there are two values: one with mask and the other without. Through these two categories, a loop function has been applied. Because the learning rate has been used less than the loss, the loss has been appropriately estimated. As a result, the level of accuracy was high.

There were no items on either list. All image arrays were concatenated with the data. All the corresponding images were appended to the labels list whether they have a mask or not. They effectively served as a label for those images, indicating whether they had mask or not. After that, the for-loop function was used to loop through the categories. A link to the route with the directory and category has been generated. 'Os. lisdir' produces a list of all the images paths inside the loop. Following that, the Load image function is utilized. The load image is handled by the imported library "keras.preprocessing.image". The image path has been loaded withload image, and the image size target is (224, 224) pixels. The dimensions of an image are its width and height. By loading the images, they were saved into an array. The "img_to_array" method was used to convert the images to an array. The "keras.preprocessing.image" package was used to import the image into the array module. After converting the images to an array, a library called "preprocess_input" function was utilized. This model was created with MobileNet. Following that, all the images were appended to the data list and labelled within the labels list. The "preprocess_input" method was used because of the MobileNet library. When utilizing the MobileNet library, the "preprocess_input" function is necessary. FIGURE below shows the entire operation of this part.

```
37 for category in CATEGORIES:
38     path = os.path.join(DIRECTORY, category)
39     for img in os.listdir(path):
40         img_path = os.path.join(path, img)
41         image = load_img(img_path, target_size=(224, 224))
42         image = img_to_array(image)
43         image = preprocess_input(image)
44 
45         data.append(image)
46         labels.append(category)
```

Figure 5.3 For Loop function has applied through categories

Training and testing data were separated into trainX and testY, as well as trainY and testY. Within the classification process, there were two types of datasets: training and testing datasets. The training set was used to train a model, whereas the test set was used to test the model that have been trained. As shown in FIGURE, 30% of the testing sets were used, while the remaining 80% was used for additional training. In the labels stratify was utilized and in the random state value was used 42. Because the datasets were separated into training and testing, data preparation was carried out. The "sklearn.model_selection import train_test_split" package was used to split the data into train and test. MobileNet was utilized for image processing in addition to the convolution neural network. After being processed, the images were sent to MobilNet. Then s fully connected layer was created via max-pooling.As a result, the results appeared.

```
55 (trainX, testX, trainY, testY) = train_test_split(data, labels,
56     test_size=0.30, stratify=labels, random_state=52)
```

Figure 5.4 Data splitting

The "model. predict" method was used to evaluate the model network, which can be seen in FIGURE. The label's index was determined using the highest predicted probability for each image in the testing set. As shown in FIGURE, the command "np.argmax" was used to do so. The classification report was formatted well formatted, and the model was saved at the end, as seen in FIGURE. FIGURE also shows the accuracy and metrics plotted using Matplotlib. The image was also saved using matplotlib. As a result, two files appeared on the desktop. One was a model file, while the other was a matplotlib plotting file called "plot.png".

```
print("evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

predIdxs = np.argmax(predIdxs, axis=1)

print(classification_report(testY.argmax(axis=1), predIdxs,
    target_names=lb.classes_))

print("saving mask detector model...")
model.save("mask_detector.model", save_format="h5")

N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")
```

Figure 5.5 The network for evaluating

5.2.3 Test the model

To run the model, the computer's command prompt was opened and navigated to the location where the relevant training file was located. The python keyword was typed, followed by the file name train_mask.py, and the enter key was hit to start it. After that, the training phase of this model started. All the images had to be trained, which took a lengthy time. The "plot.png" image of the findings was stored in the project folder after the mask detector model was trained. The accuracy and training loss were plotted. The accuracy and loss have been precisely conveyed. The model appeared to be in good form. The epochs down were given up to 20 epochs. The model's functioning performance was excellent, as shown in FIGURE below. The "plot.png" and mask detector files have been saved on the local desk. Face detection was performed using two files that were downloaded and saved in the face-detector folder. For the camera operation, OpenCV was used.

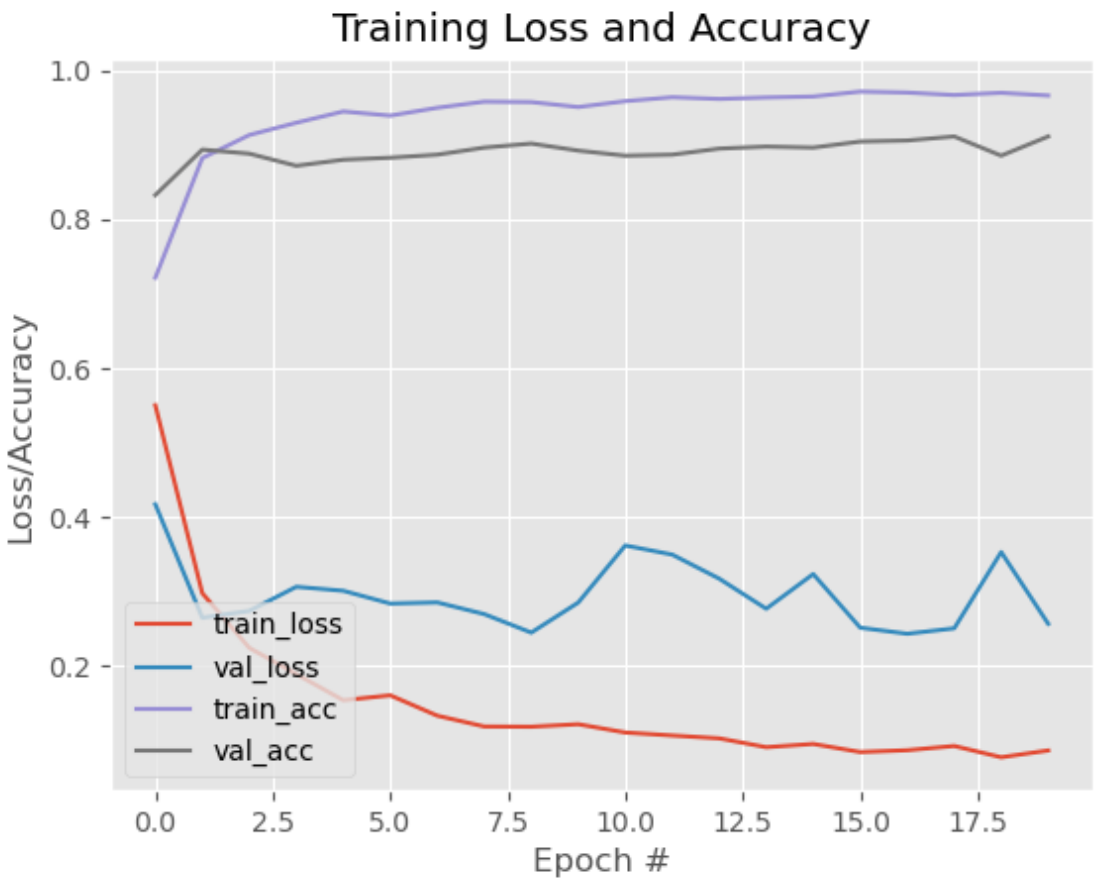

Figure 5.6 Visualization through matplotlib

5.2.4 Mask detection process

The model was loaded with “load_model”, and the camera was loaded with the “Video Stream” method, as shown in FIGURE 30. The load model was used to detect masks. There were two different models for detecting masks and faces. Within the “Video Stream” method, there was a function called src that represented the camera. And src=0 indicated the primary camera start function, which was used to start loading the camera.

```
52 maskNet = load_model("mask_detector.model")
53
54 print("[ starting video stream...")
55 vs = VideoStream(src=0).start()
```

Figure 5.7 load model and VideoStream

The order in which the images were displayed was constructed here, as was the frame presentation. The while loop was finally broken here, as shown in FIGURE 36. A configuration was saved with the letter ‘q’ for breaking from the loop, and the loop was broken by hitting the q button. All the windows were eventually eliminated, and the video was turned off.

```
76      cv2.imshow("Frame", frame)
77      key = cv2.waitKey(1) & 0xFF
78
79      if key == ord("q"):
80          break
81
82  cv2.destroyAllWindows()
83  vs.stop()
```

Figure 5.8. Image Frame, breaking loop, and turn off the windows

5.3 CODE

```
import cv2,os
data_path='dataset'
categories=os.listdir(data_path)
labels=[i for i in range(len(categories))]
label_dict=dict(zip(categories,labels))
print(label_dict)
print(categories)
print(labels)

img_size=100
data=[]
target=[]



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

    for img_name in img_names:
        img_path=os.path.join(folder_path,img_name)
```

```
        img=cv2.imread(img_path)

        try:
            gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            #Coverting the image into gray scale
            resized=cv2.resize(gray,(img_size,img_size))
            #resizing the gray scale into 100x100, since we need a fixed common size for all the
images in the dataset
            data.append(resized)
            target.append(label_dict[category])
            #appending the image and the label(categorized) into the list (dataset)

        except Exception as e:
            print('Exception:',e)
import numpy as np

data=np.array(data)/255.0
data=np.reshape(data,(data.shape[0],img_size,img_size,1))
target=np.array(target)

from keras.utils import np_utils

new_target=np_utils.to_categorical(target)

np.save('data',data)
np.save('target',new_target)



from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D

data = np.load('data.npy')
```

```
target = np.load('target.npy')
model = Sequential()
model.add(Conv2D(200,(3,3),input_shape=data.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(100,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(50,activation='relu'))
model.add(Dense(2,activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics = ['acc'])
model.summary()
```

```
from sklearn.model_selection import train_test_split

train_data, test_data, train_target, test_target = train_test_split(data, target, test_size=0.1)
from keras.callbacks import ModelCheckpoint

checkpoint=ModelCheckpoint('model-{epoch:03d}.model', monitor='val_loss', verbose = 0,
save_best_only = True,mode='auto')
history = model.fit(train_data,train_target,epochs = 20, callbacks = [checkpoint],
validation_split = 0.2)

import matplotlib.pyplot as plt

plt.plot(history.history['acc'],'r',label='training accuracy')
plt.plot(history.history['val_acc'],'b',label='validation accuracy')
plt.xlabel('epochs')
```

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



print(model.evaluate(test_data,test_target))
from keras.models import load_model

import cv2
model = load_model('model-010.model')

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

source=cv2.VideoCapture(0)

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

from cv2 import (VideoCapture, namedWindow, imshow, waitKey, destroyWindow, imwrite)
while(True):
    ret,frame =source.read()
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_clsfr.detectMultiScale(gray,1.3,5)

    for x,y,w,h in faces:
        face_img = gray[y:y+w,x:x+h]
        resized = cv2.resize(face_img,(100,100))
```

```
        normalized = resized/255.0
        reshaped = np.reshape(normalized,(1,100,100,1))
        result = model.predict(reshaped)

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

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

    cv2.imshow('Live',frame)
    key=cv2.waitKey(1)

    if(key==27):
        break;

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

Chapter 6

RESULT & DISCUSSION

the model can accurately identify face photos (mask or no mask) with promising precision and recall is demonstrated. As a result of the model's strong recall and precision findings in the experiment, it has a great deal of promise for minimizing the number of false positives and negatives in facial mask detection applications during the COVID-19 pandemic. For real-time face photos, the model's average classification time of 1,020 samples was 2.5 s, making it suitable for mask and no-mask classification.

Real Time Input

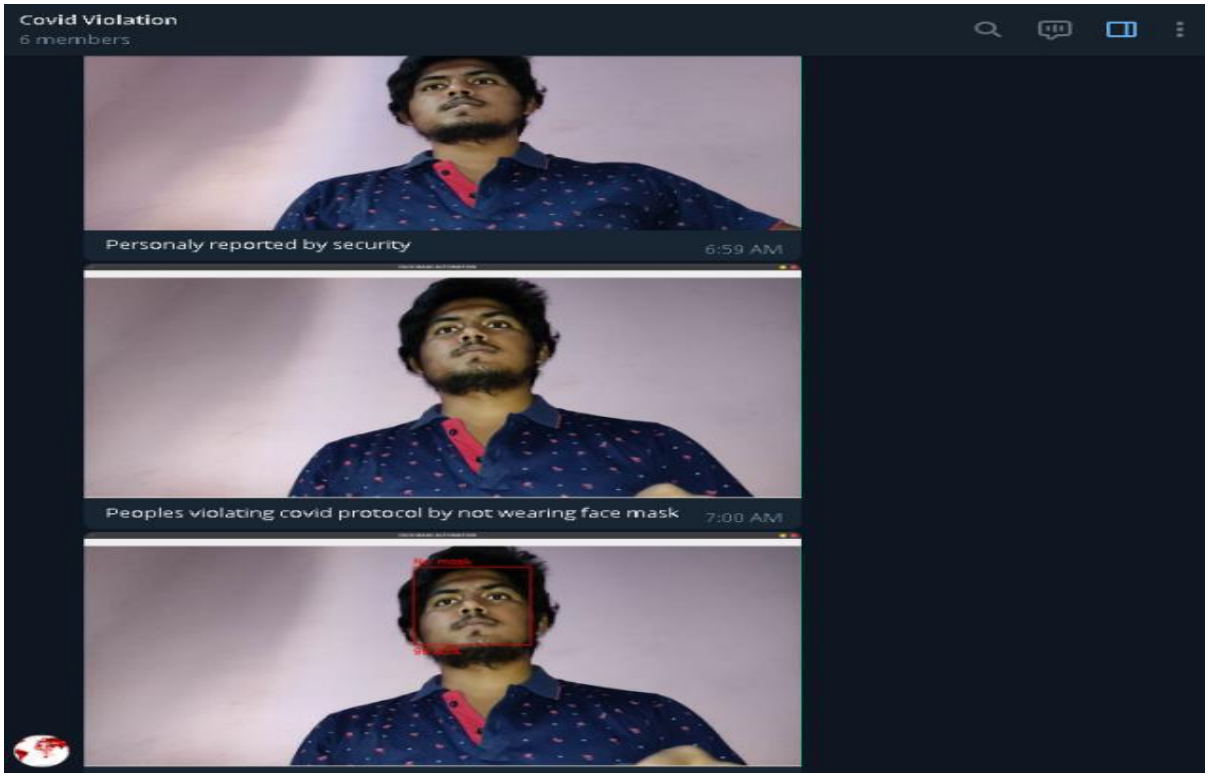

Figure 6.1 without mask

Real Time Output

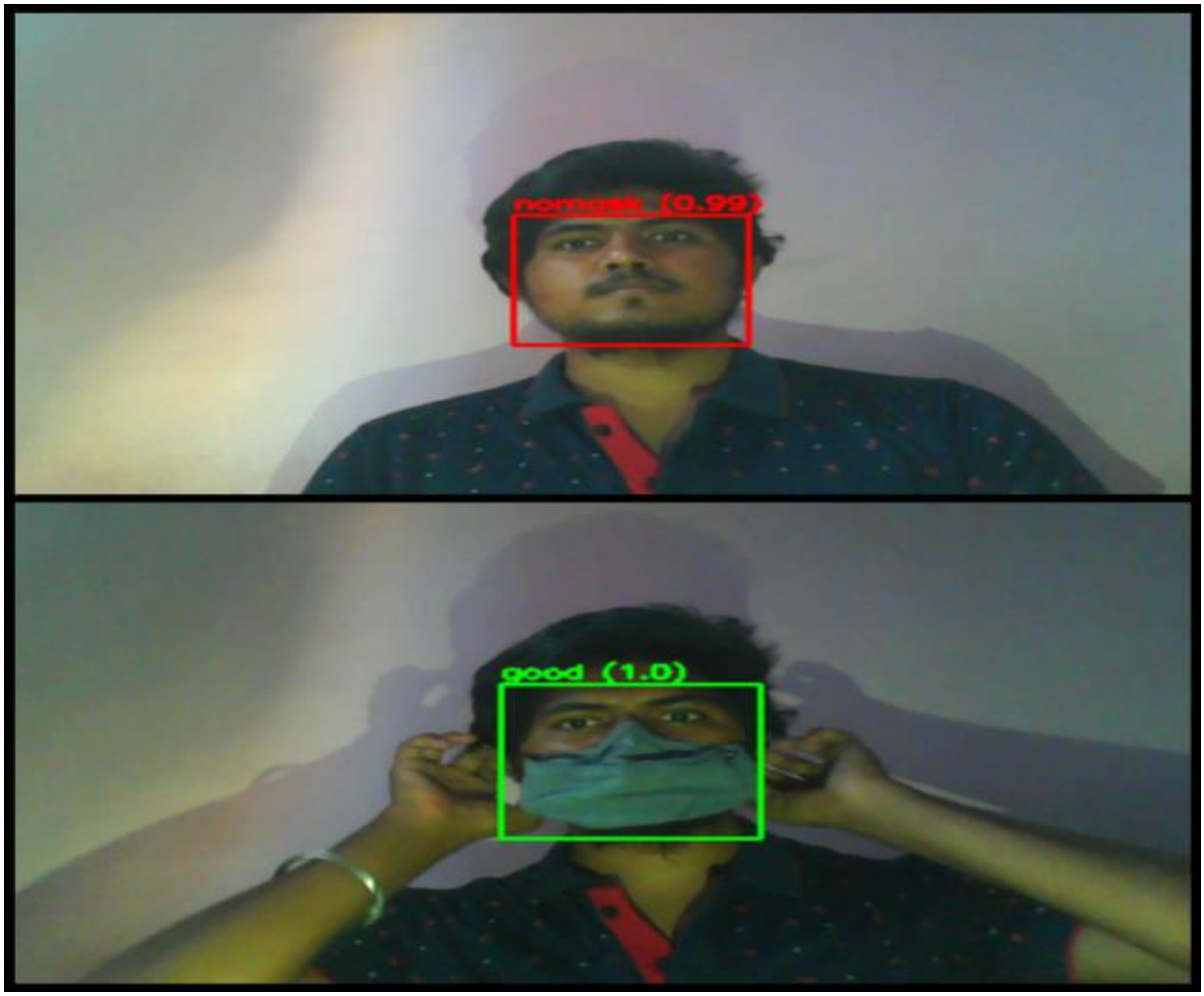

Figure 6.2 with mask and no mask output

Chapter 7

CONCLUSION, LIMITATION & FUTURE SCOPE

7.1 CONCLUSION

As the technology are blooming with emerging trends the availability so we have novel face mask detector which can possibly contribute to public healthcare. The architecture consists of Mobile Net as the backbone it can be used for high and low computation scenarios. In order to extract more robust features, we utilize transfer learning to adopt weights from a similar task face detection, which is trained on a very large dataset. We used OpenCV, tensor flow, and NN to detect whether people were wearing face masks or not. The models were tested with images and real-time video streams. The accuracy of the model is achieved and, the optimization of the model is a continuous process and we are building a highly accurate solution by tuning the hyper parameters. This specific model could be used as a use case for edge analytics. Furthermore, the proposed method achieves state-of-the-art results on a public face mask dataset. By the development of face mask-detection we can detect if the person is wearing a face mask and allow their entry would be of great help to the society.

7.2 LIMITATION

Lack of control over their personal information Critics of mask recognition also think that this new technology could be prone to some of the same pitfalls as facial recognition. Many of the training datasets uses for facial recognition are dominating by light-skinned individuals.

7.3 FUTURE SCOPE

Objects are usually recognized by their unique features. There are many features in a human face, which can be recognized between a face and many other objects. It locates faces by extracting structural features like eyes, nose, mouth etc. And then uses them to detect a face. Typically, some sort of statistical classifier qualified then helpful to separate between facial and non-facial regions Human faces have particular textures which can be used to differentiate

between a face and other objects. Moreover, the edge of feature scan helps to detect the objects from the face. The above-mentioned use cases are only some of the many features that were incorporated as part of this solution. We assume there are several other cases of usage that can be included in this solution to offer a more detailed sense of safety.

Several of the currently under development features

are listed below in brief:

• Coughing and Sneezing Detection: Chronic coughing and sneezing is one of the key symptoms of COVID-19 infection as per WHO guidelines and also one of the major routes of disease spread to non-infected public. Deep learning-based approach can be proved handy here to detect & limit the disease spread by enhancing our proposed solution with body gesture analysis to understand if an individualist coughing and sneezing in public places while breaching facial mask and social distancing guidelines and based on outcome enforcement agencies can be alerted.

• Temperature Screening: Elevated body temperature is another key symptom of COVID-19 infection, at present scenario thermal screening is done using handheld contactless IR thermometers where health worker need to come in close proximity with the person need to be screened which makes the health workers vulnerable to get infected and also its practically impossible to capture temperature for each and every person in public places, the proposed use-case can be equipped with thermal cameras based screening to analyse body temperature of the peoples in public places that can add another helping hand to enforcement agencies to tackle the pandemic effectively.

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)

[1] 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

[1] 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.

[4] 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.

[5] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," CoRR abs/1411.7923, 2014.

[6] 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.

[2] 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.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. KarpatEE, 2013.

[7] 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.

[8] 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.

[9] Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. CoRR, abs/1406.4773, 2014. 1, 2, 3

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. Nature, 1986. 2, 4