CAPSTONE PROJECT REPORT

(Project Term January-May 2022)

FACE MASK DETECTION USING DEEP LEARNING

Submitted by

NAME	REGESTRATION NUMBER
SUBROJIT ROY	**11801400**
SOURAV DUTTA	**11801258**

Project Group Number CSERGC0154

Course Code CSE445

Under the
Guidance of
Puneet Kumar
Assistant professor

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

TOPIC APPROVAL PERFORMA

School of Computer Science and Engineering (SCSE)

Program : P132::B.Tech. (Computer Science & Engineering)

COURSE CODE : CSE445 **REGULAR/BACKLOG :** Regular **GROUP NUMBER :** CSERGC0154

Supervisor Name : Puneet Kumar **UID :** 26159 **Designation :** Assistant Professor

Qualification : _____ **Research Experience :** _____

SR.NO.	NAME OF STUDENT	Prov. Regd. No.	BATCH	SECTION	CONTACT NUMBER
1	Kishan Kumar	11813123	2018	K18VQ	9521411783
2	Subrojit Roy	11801400	2018	K18BY	8399030428
3	Sourav Dutta	11801258	2018	K18FG	7003723280

SPECIALIZATION AREA : Database Systems-I **Supervisor Signature:** _____

PROPOSED TOPIC : Face Mask Detection

Qualitative Assessment of Proposed Topic by PAC		
Sr.No.	**Parameter**	**Rating (out of 10)**
1	Project Novelty: Potential of the project to create new knowledge	5.95
2	Project Feasibility: Project can be timely carried out in-house with low-cost and available resources in the University by the students.	6.58
3	Project Academic Inputs: Project topic is relevant and makes extensive use of academic inputs in UG program and serves as a culminating effort for core study area of the degree program.	7.11
4	Project Supervision: Project supervisor's is technically competent to guide students, resolve any issues, and impart necessary skills.	6.95
5	Social Applicability: Project work intends to solve a practical problem.	6.05
6	Future Scope: Project has potential to become basis of future research work, publication or patent.	6.05

PAC Committee Members		
PAC Member (HOD/Chairperson) Name: Kewal Krishan	UID: 11179	Recommended (Y/N): Yes
PAC Member (Allied) Name: Dr.Gurpreet Singh	UID: 17671	Recommended (Y/N): Yes
PAC Member 3 Name: Dr. Amritpal Singh	UID: 17673	Recommended (Y/N): Yes

Final Topic Approved by PAC: **Face Mask Detection**

Overall Remarks: Approved

PAC CHAIRPERSON Name: 14770::Sawal Tandon **Approval Date:** 16 Mar 2022

DECLARATION

We hereby declare that the project work entitled FACE MASK DETECTION USING DEEP LEARNING is an authentic record of our own work carried out as requirements of the Capstone Project for the award of a B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Puneet Kumar, during February to April 2022. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Project Group Number: CSERGC0154

Sourav Dutta
11801258

Subrojit Roy
11801400

(SIGNATURE OF STUDENT 1)

(SIGNATURE OF STUDENT 2)

CERTIFICATE

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfilment of the conditions for the award of a B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara.

Signature and Name of the Mentor

Designation

School of Computer Science and Engineering,
Lovely Professional University
Phagwara, Punjab, India.

Date:

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to the Board of Management of Lovely Professional University for their assistance and motivation in finishing this project.

They have my gratitude.

My heartfelt and profound appreciation to **Puneet Kumar**, my Project Guide, for his invaluable assistance, recommendations, and consistent support, which paved the path for the successful completion of my project work.

I would like to thank all of the Department of Computer Science and Engineering's teaching and non-teaching staff members who contributed to the project's success in a variety of ways.

Table of Contents

ABSTRACT

Chapter No.	Title	Page No.
1	**Introduction**	**1**
2	**Literature Review**	**5**
3	**Scope and Objective**	**7**
	3.1 Objective of the Project	7
	3.2 Scope of the Project	7
	3.2 Feasibility Study	7
4	**System Analysis**	**9**
	4.1 Benefits of the Current System	9
	4.2 Advantages	9
	4.3 Motivation	9
	4.4 Requirement Specification	9
	4.5 Software Description	10
	4.6 Proposed System	16
	4.7 Planning and Scheduling	17
5	**System Design and Methodology**	**19**
	5.1 Training the Dataset	19
	5.1.1 Dataset Visualization	19
	5.1.2 Pre-processing of Image	21
	5.1.3 Training of the Model	22
	5.2 Detection	23

6	**Result**	**26**
	6.1 Result	26
	6.2 Detection of Mask	27
	6.3 Screenshot of Detection of Mask	27
7	**Conclusion and Future Enhancement**	**28**
	7.1 Conclusion	28
	7.2 Future Enhancement	28

APPENDIX

A. SOURCE CODE **31**

ABSTRACT

The year 2020 has shown humanity an incredible variety of occasions among them COVID19 pandemic is the most dangerous change. It surprised the world from the beginning of the year and also affected the happiness and life of the masses. COVID19 has called for serious measures to be taken to prevent the spread of the disease. Cleanliness is an essential instruction for medication use in the emergency room, individuals give their best and public welfare; Face coverings are one of the personal defence equipment. People cover their faces when they leave their house, also authorities strictly ensure that individuals wear a face mask when they are gathering in an open point. To confirm that people are complying with these need of welfare rules, a method must be made. A face coverage indicator frame can be made to consider this. Face Covering identification means knowing whether an individual has a mask or not. The primary move to distinguish the existence of a face mask on a human's face is a person's facial recognition, this process is basically divided into two different segments: the first one is face recognition and then mask detection on the recognised faces. Face recognition is a use case of item recognition and can be used in numerous fields such as security and surveillance, law enforcement and policing, biometrics and many more. Many identity frameworks are created around the world and running. By all means, this science needs to be improved; superior, more precise identifiers, because the world can't handle the cost of expanding corona cases further.

CHAPTER 1
INTRODUCTION

Overview: More than 512 million people in the world have been already contaminated by the Covid-19 Virus, also known as Novel Coronavirus (SARS-CoV-2), coming about in over 6 million fatalities, started in December 2019 till now this virus is expanding throughout the world. Many side effects are reported in different parts of the world by the affected patients, extending from mellow signs to a major ailments. The most common among all of them is respiratory issues. It leads to shortness of breath or inconvenient breathing. COVID-19 disease can cause major results in elderly patients with a lung infection since they show up to be at a better risk. Though there are some finite numbers of species in the coronavirus family that contaminate individuals all over the world. But infections of other viruses in the Coronavirus family taint animals first and then they alter themselves to infect human beings. Those who have respiratory issues can disseminate irresistible globules to anybody who comes into contact with them. Droplets carrying the virus may land on a tainted individual's surrounding surfaces, resulting in contact transmission. Some airborne infectious diseases, including COVID-19, necessitate the use of a clinical mask. The wider populace must be informed of whether they should wear a mask or not to control the spreading of the virus. The use of masks has the potential to reduce vulnerability from the danger of an already affected person specifically during the "pre-symptomatic" stage, as well as stigmatise those who use masks to prevent the transmission of the virus. WHO and governments all around the world emphasized the usage of the mask. Thus, identifying face masks has become a serious challenge in today's world community.

Face Recognition is a method that utilizes saved models of each and every person's face in a group of individuals to recognize anyone just based on certain features of their face. Face recognition is a straightforward means of identifying and confirming individuals Face recognition is an important aspect of everyday living and activities. Personal safety and authentication are essential in any company or organization. As a result, computerised facial recognition utilising computer systems or gadgets for identity

confirmation all around clock or even remotely has stirred up a lot of interest in the present world. Face recognition has considered as one of pattern recognition and image processing's most difficult and fascinating problems. With this form of technology, one can effortlessly find somebody's face using a dataset with a comparable matching look. Python and a Convolutional Neural Network in deep learning are by far the most efficient techniques for recognising someone's face. The military, defence, schools, colleges, universities, airlines, banking, online internet apps, gaming, and many other areas all benefit from this methodology. Face masks are currently being widely utilised as a part of recommended virus-prevention measures, namely during the Covid-19 virus pandemic. Many people or companies need to be able to tell whether or not someone in a certain place or time is using a facial mask. The requirements for this data must be real-time and automated. Face identification is challenging because of the considerable variation in faces, which includes form, texture, colour, and the addition of a beard, moustache, spectacles, and even a mask. The suggested CNN and Python set of rules appears to be highly efficient and correct in determining the facial recognition and spotting of persons, based on experiments.

COVID-19 Signs and Symptoms: Dry hack, fever, windedness, pain, and migraine are the most well-known symptoms of COVID. COVID's rapid delivery causes severe muscle aches and pains, allowing those with weakened immune systems to be trained without difficulty. The terrible stage of COVID-19 leads to death in a variety of people groups due to severe lung and organ failure. Different types of drugs are being studied by doctors all over the world in order to find an efficient way to prevent an illness from spreading to its most severe state.

Face Mask Detection: Artificial Intelligence is used by the Face Mask Detection Platform to check if a person is wearing a face mask or not. To identify persons with or without a mask, the programme may be connected to any present or new IP camera. The picture capture with a camera is the first step in the face mask detection procedure. The imaging gadget and modules were made utilizing TensorFlow and Open-CV programming to recognise the face and assess all points in order to check if a person is wearing a face mask or not. On condition an individual wears a mask, they are

within the secure zone, which shows up as a green rectangle-box with a secure caution, in any case in case, they do not, it shows up as a red rectangle-box with an alarm message.

Face detection can be done in two ways:

1. Feature Based Approach: It finds faces by dispensing with major highlights such as eyes, nose, and mouth, and after that utilizing them to recognize a face. Typically, a factual classifier is qualified and then accommodated to distinguish between face and non-facial parts. Furthermore, human faces have distinct surfaces that may be used to distinguish between a face and various products. Additionally, the border of highlights can help identify things from the face. We will use OpenCV to implement a component-based technique in the next section.

2. Image Based Approach: Various computations are used in this approach such as HMM, SVM, and AdaBoost learning. In the next section, we will be using Multi-Task Cascaded Convolutional Neural Network (MTCNN) to recognise human faces, this is an image-based approach.

CHAPTER 2
LITERATURE REVIEW

A cumulative article was presented, counting the different sorts of COVID datasets that are accessible. These are open-source datasets that are freely available. The specified data sequence includes X-ray pictures and CT scan images, and in a few circumstances where the health background is limited, MRI pictures are moreover utilized. Textual data based on debates, medical advice made on social media, and so on are another group of datasets utilised for COVID analysis. Clinical test discoveries and reports are also taken into consideration in a few dataset windows utilized for diagnostic procedure investigation.

Face identification and classification were evaluated using a deep learning technique. Using a pre-trained facial dataset, distinct faces are clustered. The suggested model is trained and tested using the FDDB dataset. The suggested model is tweaked in order to achieve uniqueness and excellent prediction performance. The accuracy of the convolution neural network is stated to be high, and the next challenges are announced by employing real-time facial photos and live video capture.

Multi-face identification approach based on machine learning methods Haar boost, Ada-Boost, Gradient boost and SVM (support vector machine) is described in depth. These strategies are taught in detail in order to attain high accuracy when used in conjunction with one another. The researcher's main problem is an increase in the percentage of false positives in particular facial data.

Several analysts and researchers have worked on face technology detection throughout the last few decades. Viola-Jones proposed an incredible face detection method with one of his notable face detectors which are real-time. However, it ran into a few issues, such as face angles and brightness, which made it difficult to intercept. Ramanan introduced a random forest tree model technique for recognising masked faces, that predicts face positions and structures. Yang created a face detection model with a facial

feature aggregation approach in 2015. The facial detection models learn directly from the data then various machine learning methods are applied to it. In order to overcome facial provided by the user in the categorization based on convolutional neural networks, and occlusion concerns, Opitz-Waltner devised a grid loss in 2016. The COVID-19 epidemic highlighted the need for masked face detection, which tries to reduce viral propagation by recognising whether or not an individual is wearing a mask. Jiang- Fan-Yan has announced Retina Mask, a one-step general-purpose detector that seeks to extract robust functionality while taking contextual information into account. Militante-Dionisio suggested a model in which the input photos are segmented, and end outcomes are predicted using VGG16 models.

Face recognition and face mask detection methods like the ones above are aimed at recognising face masks or clean faces, however, they neglect differing fine-grained mask-wearing situations. On a real dataset, the proposed model aims to tackle all of the concerns raised in earlier studies with high average precision, demonstrating that it has the capacity to construct a distinct feature map.

CHAPTER 3
SCOPE AND OBJECTIVE

3.1 OBJECTIVE OF THE PROJECT

Face mask detection is the process of determining whether or not someone is wearing a mask. In reality, the subject is reverse engineering of face detection, in which the face is identified using various machine learning algorithms for security, authentication, and surveillance purposes.

3.2 SCOPE OF THE PROJECT

Video captioning intends to automatically identify a person's face mask and transmit a notice to the appropriate authorities if the mask is missing. This will aid in the prevention of the pandemic's spread. This can be employed in a number of public meeting places, including theatres, shopping malls, convention centres, and educational institutions.

3.3 FEASIBILITY STUDY

The implementation of this policy cannot be tracked manually. The central point here is technology. We offer a Deep Learning-based strategy for recognizing occurrences of disgraceful utilization of face covers. Our framework employments a two-stage Convolutional Neural Network (CNN) engineering that can perceive both unmasked and masked faces and is congruous with previously installed cameras. It will help with the following security breaches, the support of face cover utilization, and the advancement of a secure working environment. The research is carried out by dissecting the specified advances utilized in earlier studies and building a successful model that helps individuals in real-time.

With the assistance of Computer Vision and Deep Learning, we propose a Python-based

image processing and machine learning strategy to realize the vigorous structure and the possibility of considering an assortment of cases utilizing the proposed strategy with a tall level of certainty and precision in social distancing checking and risk evaluation.

CHAPTER 4
SYSTEM ANALYSIS

4.1 BENEFITS OF THE CURRENT SYSTEM

1. Intelligent notifications
2. Camera agnostic
3. Facial Recognition
4. Simple to implement
5. Simple to operate
6. There is no need for new hardware.

4.2 ADVANTAGES

1. Identifying those people are wearing a mask.
2. Remind visitors to wear masks with digital screens.
3. When no masks are detected, notify the management.
4. Compatible with existing cameras.

4.3 MOTIVATION

The model can recognise people who are not wearing a mask and a notification will be sent to the system's controller.

4.4 REQUIREMENT SPECIFICATION

SPECIFICATIONS FOR THE HARDWARE:

RAM: 2GB

Hard Disk: 20GB

Camera: CCTV/ Webcam / Mobile Camera

Keyboard: 104 keys

System: PC, Mobile.

SOFTWARE REQUIREMENT SPECIFICATION:

Operating System: Any Windows 7, 8 etc.

Browser: Mozilla, Chrome etc.

Software Interfaces: Python 3.9

Programming Language: Python

4.5 SOFTWARE DESCRIPTION

Machine Learning: The study of computer algorithms that progresses themselves over time is known as Machine Learning (ML). It is a branch of AI (Artificial intelligence). A mathematical model on the basis of test data alluded to as "training data," is created in arrange to create forecasts or judgments by Machine Learning algorithms.

Machine learning algorithms are used in a wide run of applications, counting e-mail filtering and computer vision, when creating conventional algorithms to do the desired assignments is troublesome or inconceivable. Computational statistics, which centres on making forecasts with computers, is closely associated with machine learning. The teaching of machine learning benefits from thinking about numerical optimization since it gives devices, hypotheses, and application fields. Data mining may be a comparative department considering that centres on unsupervised learning for exploratory data analysis.

Depending on the sort of the "signal" or "feedback" provided to the learning system, machine learning frameworks are generally categorised into three major categories:

Supervised Learning:

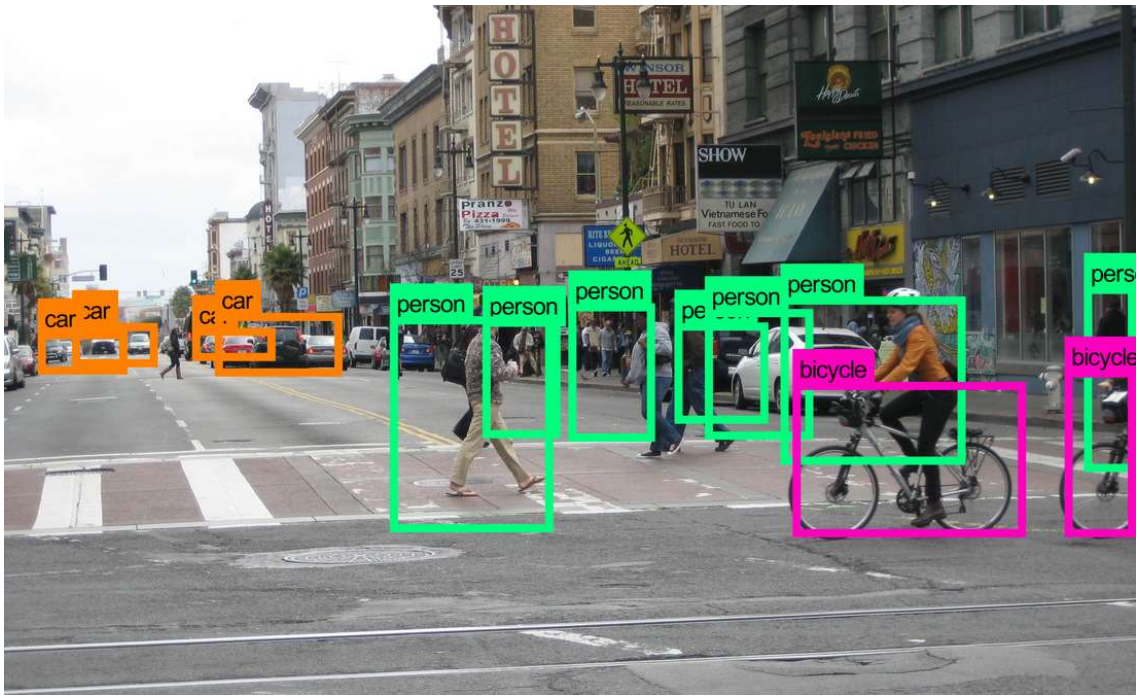
A "teacher" presents the computer with test inputs and wanted outputs, for the reason of learning a common rule which maps inputs to outputs.

Unsupervised learning: When the learning algorithm is not given labels and is left to uncover structure in the data on its own is known as Unsupervised Learning. This can be an objective in and of itself (finding covered up patterns in data) or

implies to a conclusion (finding covered up patterns in data) feature learning.

Reinforcement learning: This is a type of machine learning in which a computer program interacts with a dynamic environment to achieve a certain objective (such as driving a car or playing a game with a rival). The program receives input in a motivational manner as it navigates the theme space it is moving forward.

Computer Vision: Computer vision is an intriguing scientific subject that reflects how computers can recognize digital images and recordings at a high level. Computer vision tasks include strategies for protecting, handling, investigating, and understanding digital images, as well as strategies for extracting numerical and symbolic information from high-dimensional data in the actual world, such as in the form of a choice to supply. Understanding in this setting implies changing over a visual picture (retinal input) into numerical or typical data, e.g., within the shape of a choice. Illustrating typical data from visual information utilizing models built utilizing geometry, physics, statistics and learning hypotheses can be considered image comprehension.



The hypothesis supporting artificial frameworks that extricate data from pictures is the subject of computer vision, a scientific discipline. Video groupings, different camera points of view, or medical checking hardware are all cases of picture data. Computer vision is a fascinating study that reflects how computers are programmed to display digital images and recordings at a high level. This suggests automating what the human visual system can do from a design angle.

Convolutional Neural Network (CNN): The convolutional neural network (CNN or ConvNet) is a sort of deep neural network utilized to examine visual pictures. Changeinvariant or spatially invariant artificial neural networks (SIANN) are built on a weightsharing design of convolution kernels that slides on the input highlights and gives a proportionate interpretation reaction named a feature map. Shockingly, most convolutional neural networks are as they were equivariant beneath the interpretation, instead of invariant. They're used in recommender frameworks, image and video recognition, picture classification, image splitting, medical picture investigation, natural language processing, brain-computer interfacing, and financial time series and interfacing, to title a couple of applications. The convolutional layer is the basic building component of the CNN process, and it is responsible for the majority of the computations. The CNN normally recognises fundamental information from the input picture in this layer, such as the geometry of edges. Three convolutional layers were used in this work to attain the best model performance. The main layer is responsible for gathering low-level information including colours, gradient modifications, edges, and angles in a conservative manner. The main layer's output becomes the second layer's input when the design travels through another convolutional layer. As more layers are added, the design evolves to encompass higher-level elements such as a mix of curved and straight edges.

Kernel size and stride were two more parameters for each convolutional layer. The kernel size was adjusted to 5x5 in this experiment, which aids in recognising differently sized features in the input picture, resulting in varied sized feature maps. Similarly, the stride is the amount of movement between filter applications to the input picture, and it is virtually always balanced in height and breadth proportions. As a result, the stride

dimensions were set to 1 in this experiment. ReLU has been utilised after each convolutional stage. Rectified Linear Unit (ReLU) refers to a non-linear operation. Each pixel must be subjected to ReLU, which is an element-by-element procedure. ReLU zeros out all negative pixel values in the feature map. Since most real-world data is non-linear, the objective of ReLU is to show non-linearity within the CNN model. The design advances through enactment maps that make progressively complicated features as the show is built, performing all of these forms and including extra convolutional layers to comprehend the perspectives of human faces.

Max pooling: 2x2 max-pooling was utilized to diminish the three-dimensional measure of the convoluted feature in this proposal to create the convolutional neural network. Through dimensionality reduction, 2x2 max-pooling is advantageous for reducing the computational power required to analyse the data. It's also useful for extracting major features, which keeps the model's training process running smoothly. The most prominent esteem from the image's area that was secured by the kernel is moreover returned by max pooling. Max pooling can offer assistance in decreasing dimensionality and evacuating boisterous actions. The convolution layer and pooling layer come together to help the model understand the properties of the image.

Fully Connected Layer: This strategy included two more fully mapped layers that supported learning a high level of non-linear blending of facial features derived from kernel handles. After the fully mapped layer, the SoftMax activation work was used. The task of SoftMax is to convert a list of integers into a list of probabilities. As a result of using SoftMax, the outcomes are normalised and converted into probabilities that must sum up to 1.0. Image flattening should be performed after the input image has been converted to a multi-layer neural network. After the flattened output is fed to the feedforward neural network, backpropagation is performed at each step of the training process. The main goal of utilising a fully connected layer is to categorise the input image into individuals wearing masks or people not wearing masks using high-level characteristics.

Training: Backpropagation is another term for the training process. The training phase is important for the model to determine the weights that most accurately represent the

input data and tune its comparative output class. As a result, their weights are updated on a regular basis and shifted closer to their ideal output class. For this task, we prepared a CNN model using the face cover detection dataset. During training preparation, the training data was split into 100 batches. Dividing the entire dataset into smaller chains of data and feeding them into the model one at a time is called batch estimation. Part of batch preparation of datasets can speed up model training and control the accuracy of gradient errors. Similarly, a learning rate of 0.001 was utilized to decide the size of a step toward lessening the loss function. To train the model utilizing the optimization strategy, the labelled data, loss function, forward pass, backward pass and weights updates are utilized.

The CNN model was trained using the Adam optimizer during this paper's training phase. Adam is an adaptive learning rate optimization technique designed specifically for deep neural network training. The Adam optimizer calculates each learning rate based on the model's distinct parameters. The Adam optimization strategy demonstrated profoundly beneficial for this strategy, as it uses the values of the first and second gradients to adjust the learning rate of the weights of each neural network.

Python: Python is a programming language that may be used in a variety of ways. Object-oriented and structured programming are completely supported, and functional programming is supported by many of the features. Many other paradigms, such as contractual design and logic programming, are supported by extensions.

Python manages memory using a combination of dynamic typing, reference counting, and a cycle-aware garbage collector. There is also late binding (dynamic name resolution). Python's architecture includes some support for Lisp-style functional programming. It includes list comprehensions, sets, dictionaries, generator expressions, and also maps, filters, and reduce functions. Python's creators have made it a priority to maintain the language enjoyable to use. Python was created with the intention of being extremely extendable. To delimit blocks, Python uses blank indentation instead of curly braces and keywords.

To get started with Python development, you'll need a platform or framework. Remember to think about the size of your application or project when choosing a framework. Python supports a large selection of frameworks that are commonly used in application development.

Python Programming Language Benefits:

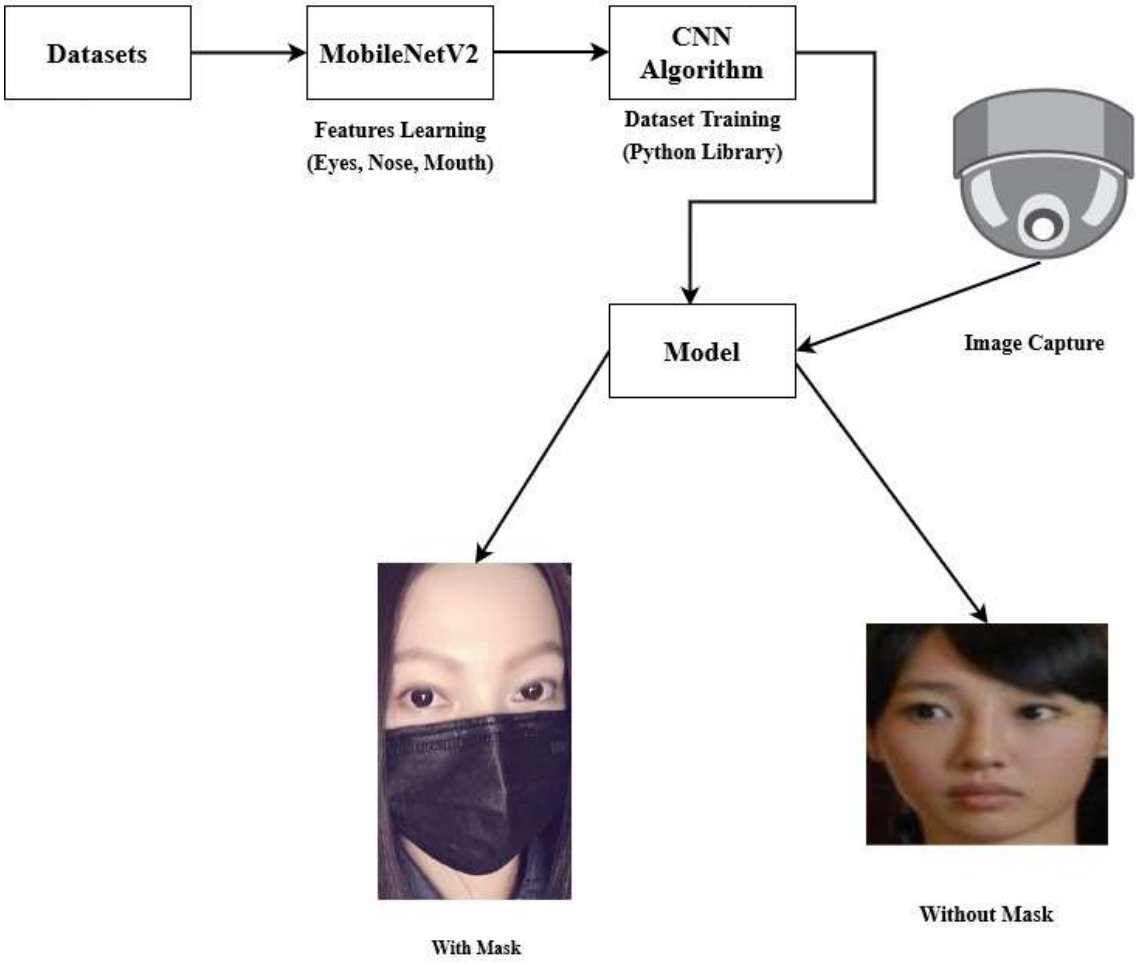
- Third-party modules are available
- Extensive support libraries available (Pandas for data analysis, NumPy for numerical calculations etc.)
- Open source and community development are encouraged
- Convenient data structures are available.
- High-level Object-oriented language
- Portable and interactive language
- Dynamically typed language (you don't have to specify the data type according to the value provided)

4.6 PROPOSED SYSTEM

Real-Time Implementation of AI-Based Face Mask Detection:
TensorFlow, Keras, and OpenCV are three Python libraries used to develop and build the model presented here. The convolutional neural network model we utilised was MobileNetV2. Transfer Learning is the way of employing MobileNetV2. Transfer learning is the process of utilising a previously trained model to train your current model and obtain a prediction, which saves time and simplifies the process of training various models. The hyperparameters: number of epochs, learning rate, and batch size are used to fine-tune the model. The model is trained on a set of photos divided into two categories: with and without a mask. There are 1915 photos with masks and 1918 images without masks in the collection.

- The dataset was utilized to prepare the model.
- Putting the model into action.

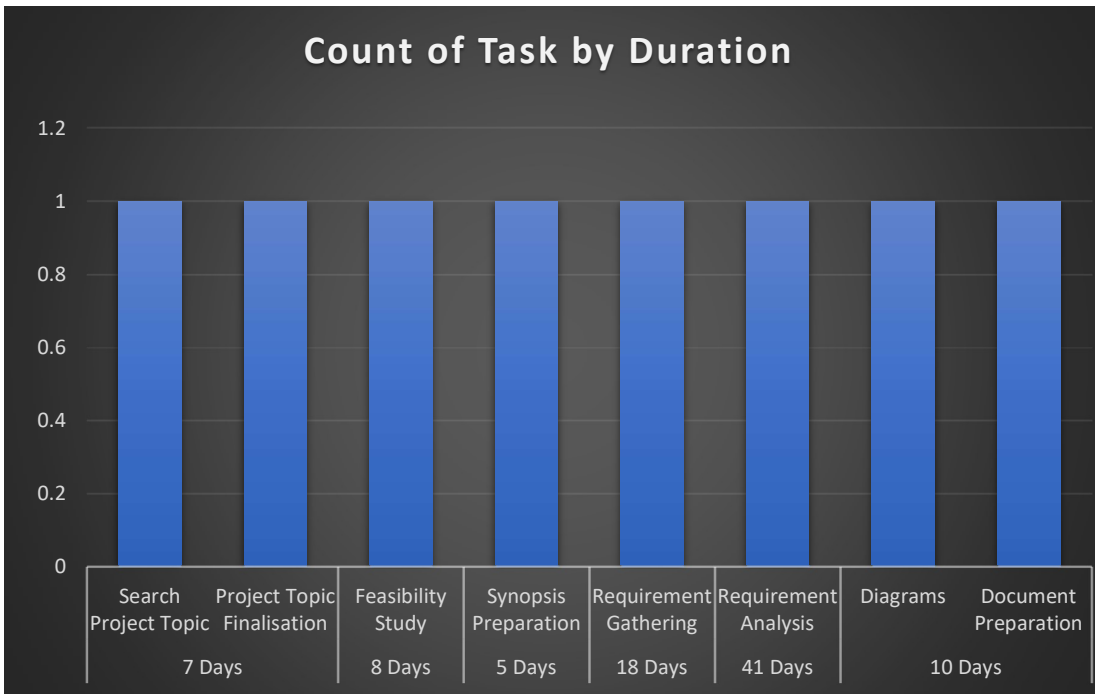
We used the above-mentioned libraries to create a model for the paper. We evaluated the model under various settings and with various hyperparameters, and the findings are shown in the next section. We start by feeding the dataset into the model and then running the training algorithm, which trains the model on the provided data. Then, using the object detection method, we execute the detection software, which turns on the video stream and grabs the frames continually from the video stream with an anchor box. This data is sent via the MobileNetV2 model layer, which determines if the image contains a mask. If you are wearing a mask, you will see a green box, and if you are not wearing a mask, you will see a red box. Same accuracy as shown in the box. The flow of the Face Mask Detection model utilised in this work is depicted in Figure.



4.7 PLANNING AND SCHEDULING

Gantt Chart: A Gantt chart is a visual depiction of a project's progress. This graph depicts the starting and ending dates of each project activity. It displays the number of weeks, months, or quarters needed to complete each activity. It's also referred to as a timeline chart. It uses horizontal bars to provide information about activities. It's made out of graph paper. If it's complicated, we use a programme like Microsoft Excel to prepare it.

Task	Date	Duration
Search Project Topic	21/01/2022	7 Days
Project Topic Finalisation	28/01/2022	7 Days
Feasibility Study	05/02/2022	8 Days
Synopsis Preparation	10/02/2022	5 Days
Requirement Gathering	28/02/2022	18 Days
Requirement Analysis	10/04/2022	41 Days
Diagrams	20/04/2022	10 Days
Document Preparation	30/04/2022	10 Days



CHAPTER 5
SYSTEM DESIGN AND METHODOLOGY

5.1 Training the Dataset

5.1.1 Dataset Visualization: Only a few datasets are available for face mask identification, and the majority of them are either intentionally generated or full of noise with incorrect labels. Hence, we developed a useful dataset for training the model, which gave positive results in the end. The dataset, which included photos from a variety of data sources such as MAFA, RMFD, CelebA, and other photographs retrieved from the internet, was cleaned by personal examination. Finally, a dataset of 4000 photos with the labels "with mask" and "without mask" was constructed, and the distribution is shown in Figure 1.

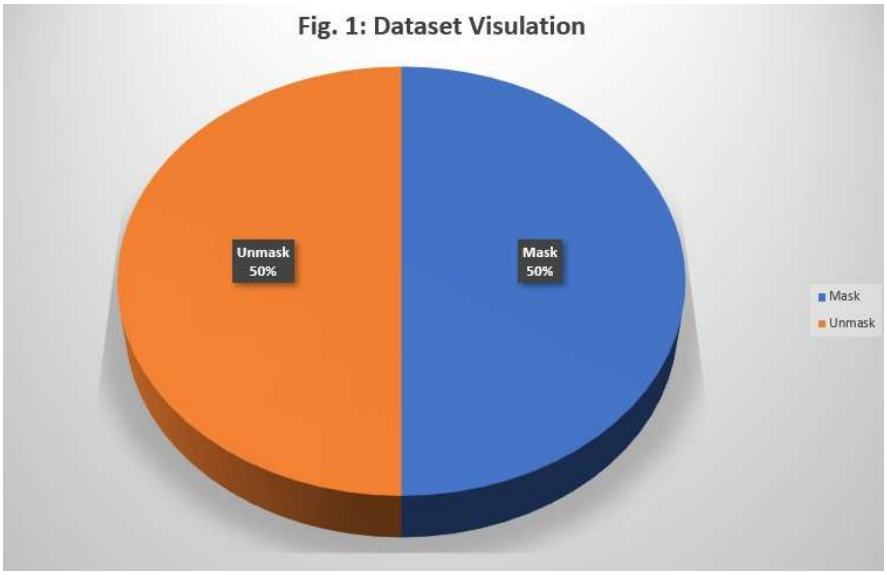


Figure 2: With Mask

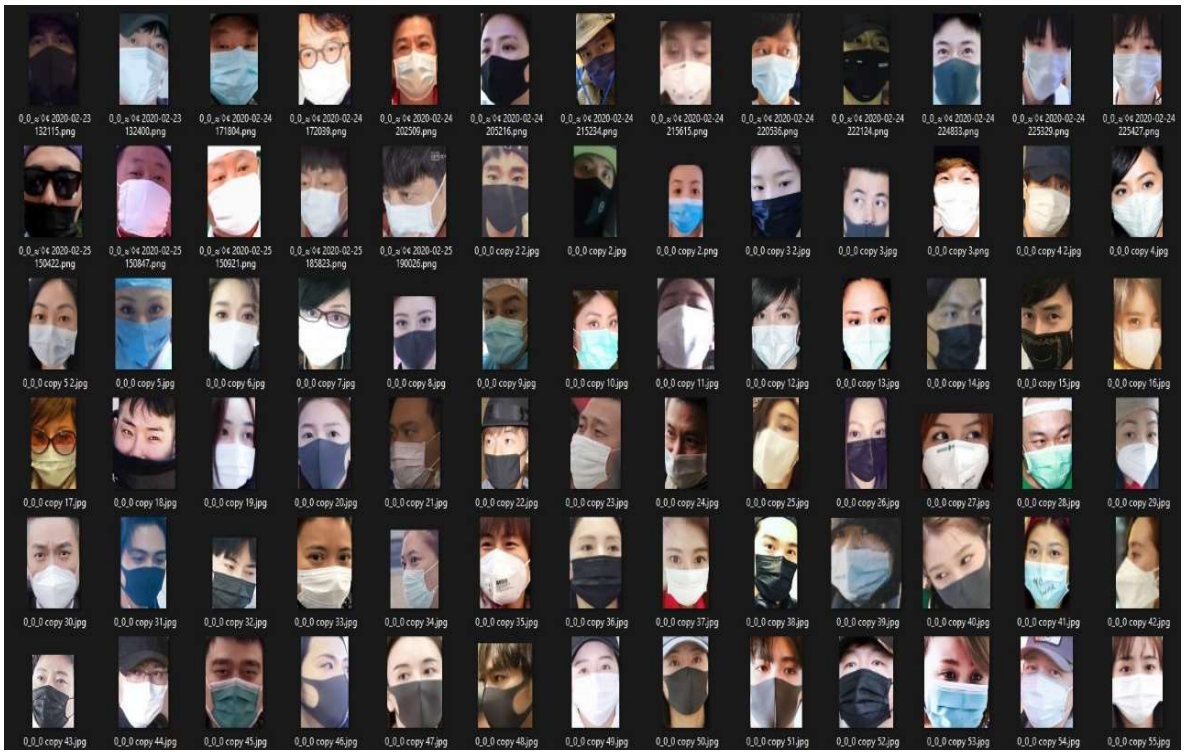
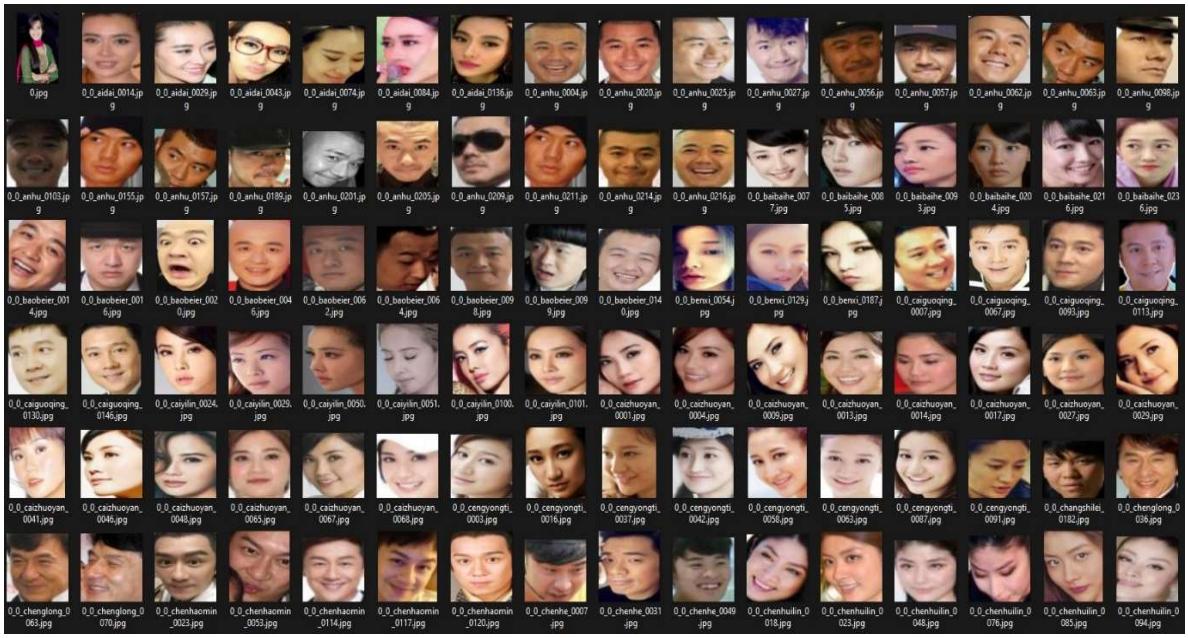
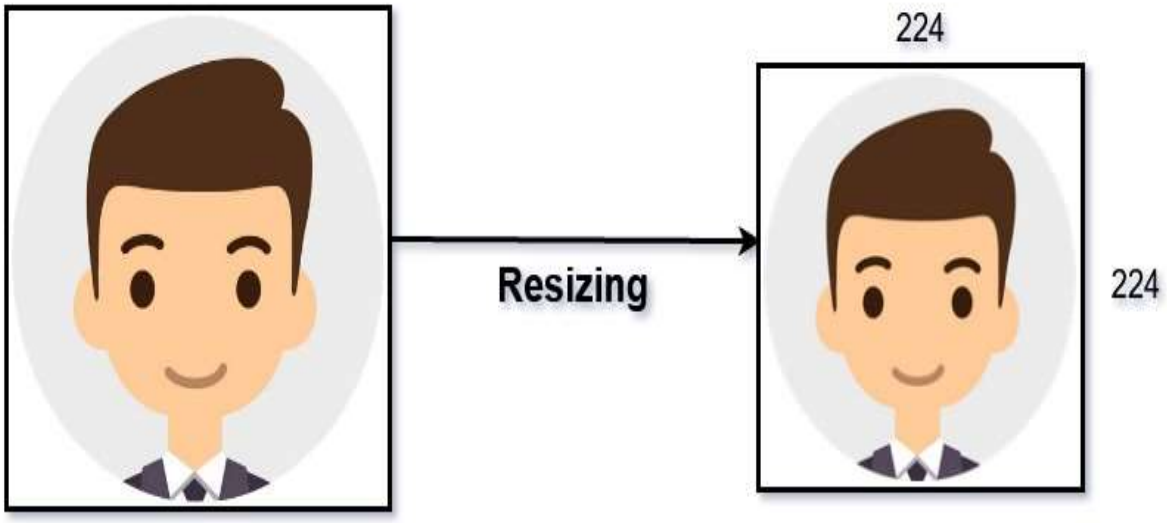


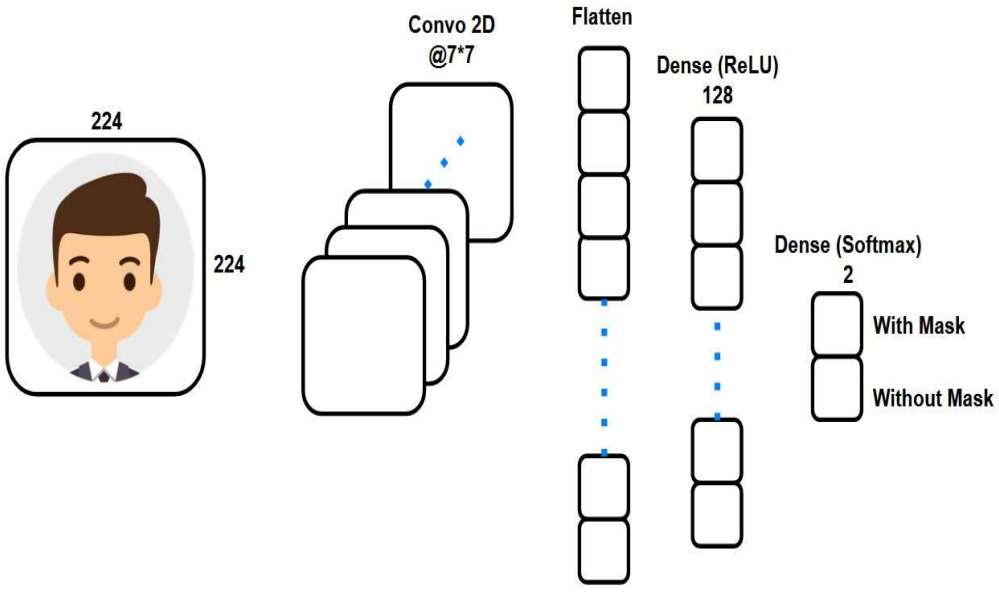
Figure 3: Without Mask



5.1.2 Pre-processing of images: Pre-processing the picture is a critical step before the actual training for greater model accuracy. Before using face detection and matching algorithms, the input picture must be pre-processed. Noise reduction, eye and mask identification, and hole filling procedures are all part of the pre-processing process. Noise reduction and hole filling assist to eliminate erroneous face/face detection. The facial picture is cropped and re-localized after pre-processing. To increase the quality of the pre-processed image, histogram normalisation is used. We employed Convolutional Neural Networks in this article, hence the dataset pictures were downsized to 224*224-pixel resolution, as shown in Figure 2. Following resizing, the photos were pixelized (that is, transformed into an array) using OpenCV library methods. With TensorFlow and Keras library functions, the pictures are labelled as 'with mask' or 'without mask' after they have been pixelized. For training, a master array comprising all pixelized pictures with labels was created.



5.1.3 Training of the Model: The Convolutional Neural Network, i.e., MobileNetV2 is used in the last phase of training the model, as illustrated in Figure 3. As seen in Figure 3, both models were given identical parameters for training. The dataset has been separated into two parts: 80 percent for training and 20% for validation. Both models employ accuracy metrics to validate the data, as it evaluates the model after each epoch. In this study, we utilised photographs with a batch size of 32 and input sizes of 224*224 as width and height, respectively for the CNN model. After testing, the parameters for the ADAM optimizer were adjusted based on the accuracy, which was found to be optimum at 20 epochs and a learning rate of 1e-4. As a result, the Image Data Generator is used to expand the dataset by rotating, rescaling, zooming, and flipping the photos. Because picture resolutions are lowered when features are extracted, the dropout rate for both models is set at 50 to avoid overfitting.



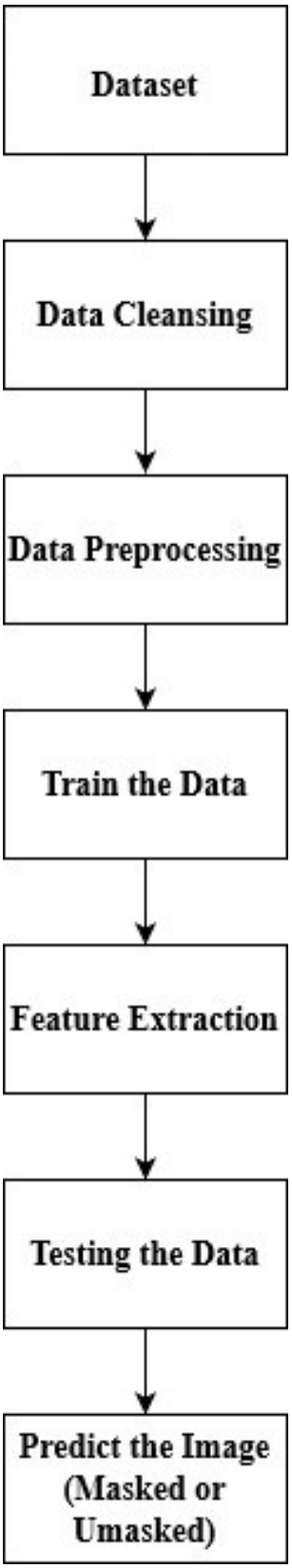
5.2 Detection

I. **Loading of Trained Model:** After some training, a model extension file containing the learned model classifier will be generated. This training will aid in the identification of masks from faces that have been identified.

II. **Face Recognition in a Live Video Stream:** When an object (human) passes in front of the real-time camera, a frame is taken and analysed, and the processing leads to face identification and extraction.

III. **Extraction of Features from Each Face:** The features are retrieved from the cropped facial picture identified by the camera in this stage. These qualities aid in increasing the mask detection accuracy rate.

IV. **Output:** The final stage is to compile all of the data, including face extraction to detect, and check whether the person is wearing a mask.

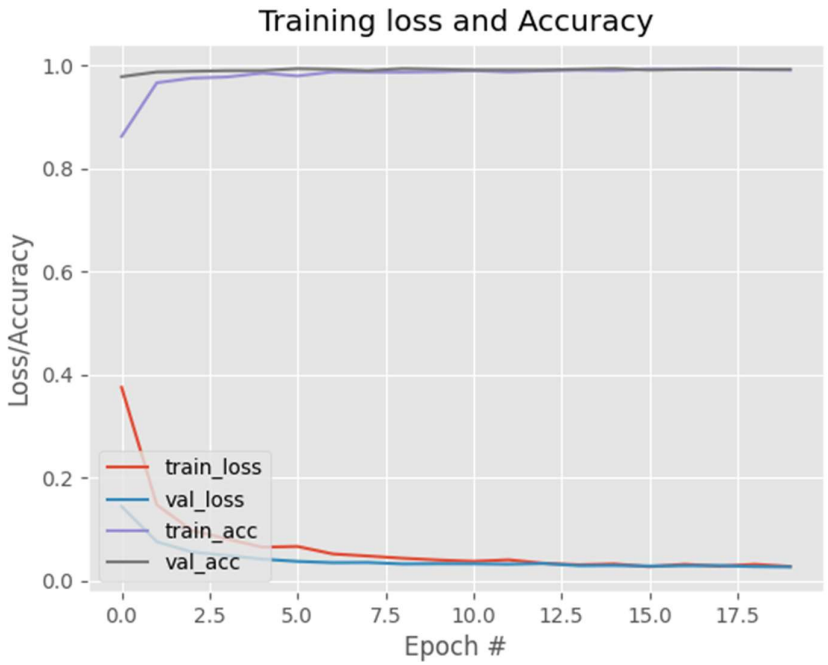

Dataset
Data Cleansing
Data Preprocessing
Train the Data
Feature Extraction
Testing the Data
Predict the Image (Masked or Umasked)

CHAPTER 6

RESULT

6.1 RESULT

After 20 epochs of training and a batch size of 32, the validation accuracy of this model is 99.8 percent for MobileNetV2.



Epochs	Accuracy
5	98.06
10	99.22
15	99.35
20	99.81

6.2 Detection of Mask

The face mask identification system employments artificial intelligence to decide whether an individual is wearing a mask or not. It may be linked to any surveillance system you have placed on your property to be sure they're who they say they are. If someone enters the premises without wearing a face mask, the system sends an alarm message to the designated individual. Detecting a person wearing a face mask is 97-99 percent accurate, depending on the digital capabilities. The data has been automatically transmitted and kept in the system, allowing you to run reports whenever you wish.

6.3 SCREENSHOTS OF DETECTION OF MASK

Fig 1: Where Sourav is not wearing a mask

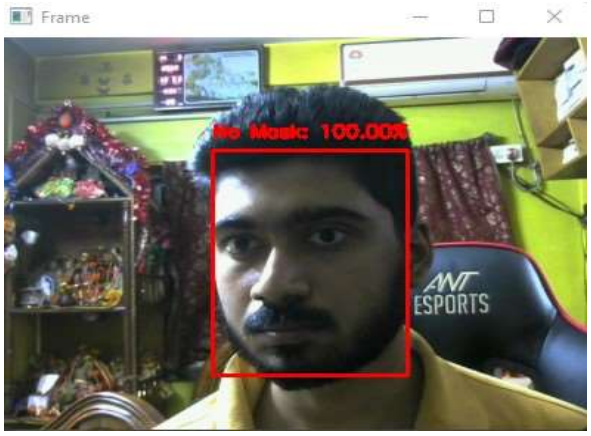


Fig 2: Where Sourav is wearing a mask

Fig 3: Where Subrojit is wearing a mask on his forehead

Fig 4: Where Subrojit is wearing a mask but the nose is not covered

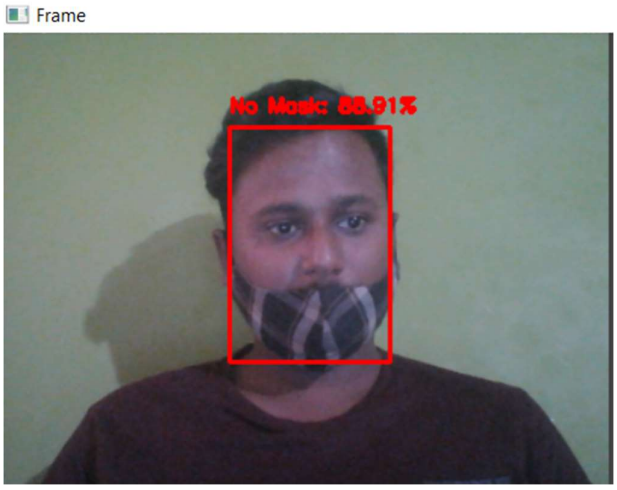
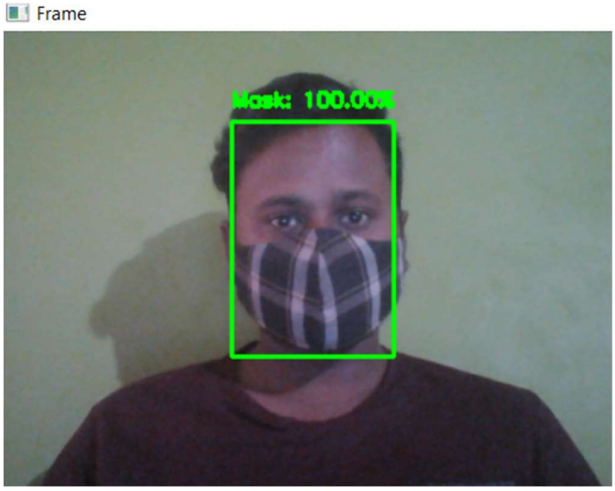


Fig 5: Where Subrojit is wearing a mask properly



CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION:

Wearing a face mask all the time is a demanding and tedious duty, but it has been mandatory since the Covid-19 crisis since wearing a face mask can help restrict the virus's spread. Some public service providers require customers to wear masks in order to be able to provide services at the beginning of this paper, we briefly talked about the work's inspiration. The model's learning and execution tasks were at that point illustrated. The strategy has achieved a sensible level of precision by utilizing fundamental machine learning tools and disentangled strategies.

7.2 FUTURE ENHANCEMENT:

In the future, the model can be extended to determine if a person wears the mask correctly (recommended by WHO) and the type of mask. This trained model can be used in public places using an IP camera to detect if a person is wearing a mask and is a valuable tool in the fight against the COVID 19 virus. We can utilise another source of information for future investigation. This model may be integrated with an alarm system, a social distancing system, and an SMS alert system. This model may also be evaluated with various optimizers and adaptive learning approaches can be used.

REFERENCES

1. W.H.O., “Coronavirus disease 2019 (covid-19): situation report, 205”. 2020
2. “Coronavirus Disease 2019 (COVID-19) – Symptoms”, Centers for Disease Control and Prevention, 2020. [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/symptomstesting/symptoms.html.2020
3. F. Hohman, M. Kahng, R. Pienta and D. H. Chau, “Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers,” in IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 8, pp. 2674-2693, 1 Aug. 2019, DOI: 10.1109/TVCG.2018.2843369.
4. Bosheng Qin and Dongxiao Li, “Identifying Facemask-Wearing Condition Using Image Super-Resolution with Classification Network to Prevent COVID-19”, Sensors, September - 2020, Vol.20, No.18, pp.5236.
5. C. Kanan and G. Cottrell, “Color-to-Grayscale: Does the Method Matter in Image Recognition?”, PLoS ONE, vol. 7, no. 1, p. e29740, 2012. Available: 10.1371/journal.pone.0029740.
6. Shou Yang, Ping Luo, Chen Change Loy, Xiaoou Tang, “From Facial Parts Responses to Face Detection: A Deep Learning Approach”, www.arXiv.org, September – 2015, arXiv: 1509.06451
7. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu and Alexander C Berg, “SSD: Single Shot MultiBox Detector”, European Conference on Computer Vision, Springer, 2016, pp. 21-37.
8. Hoanh Nguyen, “Fast Object Detection Framework based on MobileNetV2 Architecture and Enhanced Feature Pyramid”, Journal of Theoretical and Applied Information Technology, March – 2020, Vol.98, No.05.

9. Nwankpa, C., Ijomah, W., Gachagan, A. and Marshall, S., 2020. Activation Functions: Comparison of Trends In Practice And Research For Deep Learning. [online] arXiv.org. Available at: https://arxiv.org/abs/1811.03378v1

10. **https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html**

APPENDIX

A. SOURCE CODE

FILENAME: - FaceMask.py

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import numpy as np
import os

# where the dataset is actually present
dir = r"E:\python project\face mask detection\dataset"
# the data in the category is actually the folder names that are present in the directory path
category = ["with_mask", "without_mask"]

# so that we have context on what is happening
```

```python

# so that we have context on what is happening
print(" Stay Clam. The images are being loaded.....")

# append all the image array inside the data list
data = []
# append the labels with or without mask corresponding to the data list
label = []

# looping through the categories
for category in category:
    # joining the directory and category
    # first looping through with mask then without mask
    path = os.path.join(dir, category)
    # list all the images inside the directory
    for img in os.listdir(path):
        # join the path of particular with mask to the corresponding image
        image_path = os.path.join(path, img)
        # loading the image and setting the height and width of all the images as 224, 224 as in the target size
        # coming from keras.preprocessing.image
        image = load_img(image_path, target_size=(224, 224))
        # convert the image to array's
        # coming from keras.preprocessing.image
        image = img_to_array(image)

        image = preprocess_input(image)
```

```
            image = preprocess_input(image)

            # appending the array to the data list
            data.append(image)
            # appending label array to the label list
            label.append(category)

    # we have all the data as numerical values but the labels are not
    # that's why converting the alphabetical labels to array using one hot encoding
    # using the label binarizer method which is coming from sklearn.preprocessing
    LB = LabelBinarizer()
    label = LB.fit_transform(label)
    # converting the label values to categorical values
    label = to_categorical(label)

    # ones the label values are converted to numerical values
    # now we need to convert those values into numpy arrays
    # also the data values to numpy arrays
    # because only with arrays our deep learning model will work
    data = np.array(data, dtype="float32")
    label = np.array(label)

    # splitting the training and testing data

    trainX, testX, trainY, testY = train_test_split(data, label,
```

```
    trainX, testX, trainY, testY = train_test_split(data, label,
                                                    test_size=0.20, stratify=label, random_state=42)

    # specifying the learning rate, epochs and the batch size
    # When the learning rate is less, our loss gets calculated properly
    # here the learning rate is 0.0001
    learningRate = 1e-4
    epochs = 20
    BS = 32

    # image data generator is basically data augmentation
    # it creates many images from a single image by adding various properties like rotating the image,
    # shifting the image, changing height and width, etc. so that we can create more data with this

    aug = ImageDataGenerator(rotation_range=20,
                             zoom_range=0.15,
                             width_shift_range=0.2,
                             height_shift_range=0.2,
                             shear_range=0.15,
                             horizontal_flip=True,
                             fill_mode="nearest")

    # using the mobilenetv2 we care basically creating a base model here here imagenet is there are some pretrained
    # models only for images, so that when we use imagenet those weights will be initialized for us and it will give us
    # better results include top is whether include fully connected layer at the top of our network input tensor is the
```

```
                            fill_mode="nearest")

    # using the mobilenetv2 we care basically creating a base model here here imagenet is there are some pretrained
    # models only for images, so that when we use imagenet those weights will be initialized for us and it will give us
    # better results include top is whether include fully connected layer at the top of our network input tensor is the
    # shape of the image that is going through and 3 is the three channels f the image that is the RGB becuase we are
    # inputting colored images
    baseModel = MobileNetV2(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))

    headmodel = baseModel.output
    # average 2d pooling reduce the size of data, number of parameters, amount of computation needed and also controls
    # overfitting
    headmodel = AveragePooling2D(pool_size=(7, 7))(headmodel)
    # flatten will flatten the multi dimensional input tensor into a single dimension
    headmodel = Flatten(name="flatten")(headmodel)
    # dense is used to create fully connected layers in which every output depends on every input
    # relu is a goto activation function for non linear use cases
    # dense layer is added using 128 neurons
    headmodel = Dense(128, activation="relu")(headmodel)
    # drop out is used to avoid overfitting of our model
    headmodel = Dropout(0.5)(headmodel)
    # this is the final output layer so 2 is used one is for with mask and one is for without mask
    # here softmax activation function is used
    headmodel = Dense(2, activation="softmax")(headmodel)
    # place the head FC model on top of the base model (this will become the actual model we will train)
    # it accepts two parameters one is the input which will be the base model input
```

```
    # place the head FC model on top of the base model (this will become the actual model we will train)
    # it accepts two parameters one is the input which will be the base model input
    # and one will be the output which will be the head model
    model = Model(inputs=baseModel.input, outputs=headmodel)
    # initially we need to freeze the layers in the base model so that they will not be updated on the first training
    # process. because they are just a replacement for our convolutional neural network
    for layer in baseModel.layers:
        layer.trainable = False

    # optimizing the model
    # adam is a goto optimizer just like relu
    optimizer = Adam(lr=learningRate, decay=learningRate/epochs)
    # giving the parameters for compiling the model using the binary crossentropy as the loss function
    # and here we are going to track only the accuracy metrics
    model.compile(loss="binary_crossentropy", optimizer=optimizer, metrics=["accuracy"])

    # training the head of the network
    print("training head....")
    H = model.fit(
        aug.flow(trainX, trainY, batch_size=BS),
        steps_per_epoch=len(trainX) // BS,
        validation_data=(testX, testY),
        validation_steps=len(testX) // BS,
        epochs=epochs
    )

```

```
    )

    predIdxs = model.predict(testX, batch_size=BS)

    # for each image in the testing set we need to find the index of the label with corresponding largest
    # predicted probability
    predIdxs = np.argmax(predIdxs, axis=1)

    # printing the classification report
    print(classification_report(testY.argmax(axis=1), predIdxs, target_names=LB.classes_))

    # H5 is a file format to store structured data, it's not a model by itself. Keras saves models in this format as it
    # can easily store the weights and model configuration in a single file.
    print("saving mask detector model....")
    model.save("mask_detector.model", save_format="h5")

    # now at last plotting the training loss and accuracy
    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 #")
```

```
    # now at last plotting the training loss and accuracy
    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")
```

FILE NAME: - DetectMask.py

```python
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 cv2


def detect_and_predict_mask(frame, faceNet, maskNet):
    # here we are grabing the dimension of the frame and then constructing a blob from it
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224), (104, 177, 123))

    # passing the blob through the network and obtain the face detections
    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)
    # initializing the list of faces, their locations, and the prediction of our face mask networks
    faces = []
    locations = []
    predctions = []

    # looping over the detections
    for i in range(0, detections.shape[2]):
        # here we are extracting the confidence which is nothing but the
```

detect_and_predict_mask() › for i in range(0, detections.sh…

```python
    for i in range(0, detections.shape[2]):
        # here we are extracting the confidence which is nothing but the
        # probability associated with the detections
        confidence = detections[0, 0, i, 2]
        # here we are filtering out the weak detections and ensuring that the confidence is greater than minimum
        if confidence > 0.5:
            # here we are computing the x and 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")

            # here we are ensuring the bounding Boxes fall within the dimensions of teh frame
            startX, startY = (max(0, startX), max(0, startY))
            endX, endY = (min(w - 1, endX), min(h - 1, endY))

            # extracting the face ROI, converting it from BGR to RGB channel
            # then we are resizing it pre-processing it and ordering 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)

            # now adding the face and bounding Boxes to their respective lists
            faces.append(face)
            locations.append((startX, startY, endX, endY))
    # this line is for we will only make a prediction if at least one face is detected
```

detect_and_predict_mask() › for i in range(0, detections.sh…

```
                locations.append((startX, startY, endX, endY))
        # this line is for we will only make a prediction if at least one face is detected
        if len(faces) > 0:
            # for faster influence we will 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")
            predctions = maskNet.predict(faces, batch_size=32)
        # return a 2-tuple of the face locations and their corresponding locations
        return locations, predctions


# storing the path of  the face_detector model to a variable
prototxtPath = r"E:\python project\face mask detection\face_detector\deploy.prototxt"
weightsPath = r"E:\python project\face mask detection\face_detector\res10_300x300_ssd_iter_140000.caffemodel"
# here we are using a method call readNet which is under cv2.dnn and dnn stands for deep neural network
# here we are using the faceNet to detect the face
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# loading the mask detector model from the disk that we have made before
maskNet = load_model("mask_detector.model")

print("Starting video stream....")
# here using the video stream we are accessing the camera and using it
# here src which is source is 0 because we are using the integrated camera of our system
# but if we have more than one webcam than we have to play around with this
vs = VideoStream(src=0).start()
```

detect_and_predict_mask() › for i in range(0, detections.sh…

```
# but if we have more than one webcam than we have to play around with this
vs = VideoStream(src=0).start()

# as we all know that every frame is an image
# and that's why we are looping through each and every frame from the video or live recording
while True:
    # now here we are grabing the frame from the video and resizing it
    # here the max width of a frame will be 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)
    # here we are sending all the values to the function and detecting whether
    # a person is wearing a mask or not
    (locations, predctions) = detect_and_predict_mask(frame, faceNet, maskNet)
    # looping through the detected face locations and their corresponding locations
    for (Box, pred) in zip(locations, predctions):
        # unpacking the bounding Box and the predictions
        startX, startY, endX, endY = Box
        mask, without_mask = pred
        # giving the color of the Box or rectangle that we are going to draw
        # that is red for no mask and green for mask
        label = "Mask" if mask > without_mask else "No Mask"
        # here we have used RGB that is why if mask is present then the green value is max and rest is min
        # and vice versa
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
        # adding the probability in  the label
        label = "{}: {:.2f}%".format(label, max(mask, without_mask) * 100)
```

detect_and_predict_mask() › for i in range(0, detections.sh…

```
96            color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
97            # adding the probability in  the label
98            label = "{}: {:.2f}%".format(label, max(mask, without_mask) * 100)
99
100           # displaying the text and the rectangle here frame is the frame that we capture, label is the
101           # with mask and without_mask, startX and startY are the coordinated where we want to put the text
102           # the we gave the font style then the font scale then color that we had defined before and last the
103           # thickness
104           cv2.putText(frame, label, (startX, startY - 10),
105                       cv2.FONT_HERSHEY_SIMPLEX, 0.45,
106                       color, 2)
107           # here 2 is the thickness of the rectangle
108           cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
109
110           # showing the output
111           cv2.imshow("Frame", frame)
112           key = cv2.waitKey(1) & 0xFF
113
114           # is q is pressed then it will stop the program
115           if key == ord("q"):
116               break
```

detect_and_predict_mask() › for i in range(0, detections.sh…