FACE MASK DETECTION USING OPEN CV

MINI PROJECT REPORT

Submitted by

RINEESHA B	**[711720104069]**
RISHIKESH R	**[711720104070]**
SABARINATHAN V	**[711720104072]**
SUSEENDHAR V	**[711720104096]**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

KGiSL INSTITUTE OF TECHNOLOGY, SARAVANAMPATTI

ANNA UNIVERSITY :: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"FACE MASK DETECTION USING OPEN CV"** is the bonafidework of **"RINEESHA B, RISHIKESH R, SABARINATHAN V, SUSEENDHAR V"** who carried out the project work under my supervision.

SIGNATURE

Dr. THENMOZHI T

HEAD OF THE DEPARTMENT

PROFESSOR

Computer Science and Engineering
KGiSL Institute of Technology
Coimbatore-641035

SIGNATURE

Ms.JANANI S

SUPERVISOR

ASSISTANT PROFESSOR

Computer Science and Engineering
KGiSL Institute of Technology
Coimbatore-641035

Submitted for the Anna University Viva-Voce examination held on _____

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

We express our deepest gratitude to our Chairman and Managing Director **Dr. Ashok Bakthavathsalam** for providing us with an environment to complete our project successfully.

We are grateful to our CEO, Academic Initiatives **Mr. Aravind Kumar Rajendran**, our beloved Director-Academics **Dr. P. Shankar** and our honorable Secretary **Dr. N. Rajkumar**. We are thankful to our beloved Principal **Dr. S. Suresh Kumar** for his support, valuable guidance and blessings.

We would like to thank **Dr. Thenmozhi T, M.E., M.B.A., Ph. D**, Head of the Department, Department of Computer Science and Engineering for her unwavering support during the entire course of this project work. We express our sincere thanks to **Ms. Aruna T N, M.E.,** our Project Coordinator, Assistant Professor, Department of Computer Science and Engineering who modelled us both technically and morally for achieving greater success in completing thisproject work.

We express our sincere and heartful thanks to our faculty guide **Ms.Janani S** Assistant Professor, Department of Computer Science and Engineering for his constant encouragement and support throughout our course, especially for the useful suggestions given during the course of the project period and being instrumental in the completion of our project with her complete guidance.

We also thank all the **Faculty members** of our department and finally, we take this opportunity to extend our deep appreciation to our **Family and Friends**, for all they meant to us during the crucial times of the completion of our project.

ABSTRACT

As a biosafety precaution, the World Health Organization (WHO) introduced the wearing of face masks after the COVID- 19 epidemic. This posed challenges to existing facial recognition systems, so this study was born. In this publication, we describe how to create a system that allows you to identify people from images, even when they wear a mask. The face detector in OpenCV is used in conjunction with Based on the Mobile NetV2 architecture, a classification model in this way, it is possible to determine whether the face is wearing a mask and where it is situated. To conduct face recognition, A Face Net model is used as a feature extractor and a multilayer feedforward perceptron is used for training facial recognition models using a collection of about 4000+ photographs. Of the images, 52,9 percent came with a face mask and 47,1 percent were without mask. The outcomes of the tests demonstration that determining whether or not someone is wearing a mask is 99.65% accurate. Face recognition accuracy for ten people wearing masks is 99.52 percent, whereas face recognition accuracy without masks is 99.96 percent.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	Machine Learning
AI	Artificial Intelligence
OPEN CV	Open source computer vision

INTRODUCTION

CHAPTER 1

INTRODUCTION

The COVID19 pandemic is the biggest life-changing event that has stunned the world since the year started, according to the year's calendar. COVID-19, which has impacted the health and lives of many people, has demanded severe procedures to be taken to prevent the spread of illness. Individuals do everything they can for their personal and hence the from the most basic hygienic standards to medical treatments, society's safety is paramount; face masks are one of the private protective instruments.

Face masks are worn when individuals leave their homes, and officials strictly enforce the wearing of face masks in groups and public areas. The procedure into two parts: There are many applications of object detection, and one of them is the detection of faces and masks. It can be used in a number of scenarios such as law enforcement, biometrics, and security.

A number of detection systems take remained developed and are currently occupied crossways the world. Altogether of this study, on the other hand, yearns for efficiency, a far better, more efficient way of doing things.

Many accurate detectors are needed since the world can no longer afford a growth of Corona instances. In this project, we'll create a mask detector that can tell the difference between people wearing masks and people who don't.

1.1 PROBLEM DEFINITION

The problem is to develop a system that can detect whether a person is wearing a face mask or not using OpenCV (Open Source Computer Vision) library. The system should be able to analyze images or real-time video streams and accurately classify whether a person's face is covered with a mask or not.

1.2 OBJECTIVE OF THE PROJECT

The objective of the project is to develop a system that can accurately detect whether a person is wearing a face mask or not using OpenCV (Open Source Computer Vision) library. The project aims to leverage computer vision techniques and machine learning to address the following objectives:

Face Mask Detection: The primary objective is to detect and classify whether a person's face is covered with a mask or not. The system should be able to analyze input images or video frames and provide accurate binary classification results for each face.

Real-Time Performance: The system should perform face mask detection in real-time, allowing for efficient and timely processing of video streams. Real-time performance is crucial for applications such as monitoring public spaces, workplaces, or transportation hubs.

Accuracy and Reliability: The system should strive for high accuracy in detecting whether a face is wearing a mask or not. It should be able to handle different types of masks, including various colors, patterns, and styles, while maintaining reliable and consistent results.

Robustness to Variations: The system should be robust to variations in face orientations, sizes, poses, occlusions, and partial face coverage. It should be able to handle challenging scenarios where faces are not fully visible or partially obstructed by objects or accessories.

1.3 SIGNIFICANCE OF THE PROJECT

- Public Health and Safety
- Compliance Monitoring
- Automating Surveillance
- Efficiency in Screening Processes
- Workplace Safety
- Mass Transportation and Air Travel

1.4 OUTLINE OF THE PROJECT

Introduction: Briefly introduce the project on face mask detection and its significance in public health and safety.

Problem Statement: Clearly define the problem of detecting face masks accurately in images or video frames.

Literature Review: Review existing literature and research on face mask detection techniques to understand the state-of-the-art methods.

Model Development: Develop a machine learning or deep learning model to classify faces as "with mask" or "without mask."

Training and Evaluation: Train the model using the prepared dataset and evaluate its performance using appropriate metrics.

Real-Time Implementation: Implement the face mask detection system in real-time using OpenCV for live video stream processing.

Conclusion: Summarize the project's achievements, limitations, and potential for future improvements in face mask detection.

LITERATURE REVIEW

CHAPTER 2

LITERATURE REVIEW

Historical Stock Market Analysis:

A literature review on face mask detection using OpenCV would involve examining research papers, articles, and resources that discuss various approaches and techniques for detecting face masks.

Traditional Approaches: Review traditional computer vision techniques used for face mask detection, such as Haar cascades, Viola-Jones algorithm, or feature-based methods. Discuss the strengths and limitations of these approaches and their applicability in different scenarios.

Datasets for Face Mask Detection: Identify and examine publicly available datasets specifically designed for face mask detection. Discuss the characteristics of these datasets, including the number of images, annotation quality, and diversity of mask types and wearing scenarios. Evaluate the suitability of these datasets for training and evaluating face mask detection models.

Evaluation Metrics: Investigate the evaluation metrics commonly used for assessing the performance of face mask detection systems. This may include metrics such as accuracy, precision, recall, F1 score, or mean average precision (mAP). Compare the evaluation metrics used in different studies and their implications for system performance.

Real-Time Implementation and Optimization: Explore research papers that focus on implementing real-time face mask detection using OpenCV. Investigate techniques for optimizing model inference speed, such as model quantization, model pruning, or hardware acceleration. Analyze the trade-offs between accuracy and real-time performance in different approaches.

Benchmarking and Comparative Studies: Identify comparative studies or benchmarking efforts that compare different face mask detection methods. Examine the performance of different algorithms and models under varying conditions and datasets. Identify the state-of-the-art methods and their advantages over other approaches.

Future Directions and Challenges: Discuss the current challenges and potential future directions in face mask detection research. Identify areas for improvement, such as addressing occlusion, mask variations, or detecting improper mask usage. Explore emerging technologies and methodologies that can enhance the accuracy and robustness of face mask detection systems.

SYSTEM ANALYSIS

CHAPTER 3

SYSTEM ANALYSIS

System analysis for face mask detection using OpenCV involves examining the requirements, components, and functionality of the system.

3.1 EXISTING SYSTEM

- **"Face-Mask-Detection" by chandrikadeb7:** This is an open-source project available on GitHub that utilizes OpenCV for face detection and a CNN-based model for face mask classification. The system can detect and classify faces as either "with mask" or "without mask" in real-time.

- **"Real-time-Face-Mask-Detection" by JeeveshN:** Another open-source project on GitHub that combines OpenCV's face detection capabilities with a pre-trained deep learning model for face mask detection. The system processes live video streams and can identify individuals wearing or not wearing face masks.

- **"Face-Mask-Detection-OpenCV" by uday-lab:** This project implements face mask detection using OpenCV and Haar cascades for face detection. The system provides real-time video processing, draws bounding boxes around detected faces, and labels them based on mask presence or absence.

3.2 DRAWBACKS

- Sensitivity to Lighting Conditions
- Limited Robustness to Mask Types and Styles
- Occlusion and Partial Face Coverage

3.3 PROPOSED SYSTEM

Face Detection: Utilize OpenCV's face detection algorithms (e.g., Haar cascades, HOG, or deep learning-based models) to identify and extract faces from input images or video frames. This step involves locating the facial region accurately.

Preprocessing: Apply preprocessing techniques to enhance the quality of the detected face regions. This may include resizing, normalization, and noise reduction to improve the subsequent mask detection process.

Mask Classification: Employ a machine learning or deep learning model to classify the detected face regions as either "with mask" or "without mask." This can involve training a model using a labeled dataset and selecting an appropriate architecture (e.g., CNN, SVM, or transfer learning) for mask classification.

3.4 FEASIBILITY STUDY

- **Technical Feasibility :** Evaluate the availability and accessibility of OpenCV libraries and resources needed for face detection and image processing.
- **Data Availability and Quality** : Investigate the availability of suitable datasets for training and evaluation. Consider the number of images, diversity of mask types, and annotation quality.
- **Hardware and Infrastructure** : Determine the hardware requirements for running the face mask detection system using OpenCV, such as processing power, memory, and camera capabilities.

3.4.1 Tests of Feasibility

Test Face Detection: Use OpenCV's face detection algorithms to detect faces in sample images or video frames. Evaluate the accuracy and speed of face detection under different conditions, such as varying lighting, pose, and occlusion.

Dataset Assessment: Assess the availability and quality of existing datasets for face mask detection. Verify if the dataset contains a sufficient number of labeled images with different types of masks and various wearing scenarios. Evaluate if the dataset aligns with your specific project requirements.

Model Training: Train a face mask detection model using the available dataset. Use OpenCV in combination with machine learning or deep learning techniques to train a model that can accurately classify faces as either "with mask" or "without mask". Measure the training time, model convergence, and evaluate the model's performance metrics, such as accuracy, precision, recall, and F1 score.

3.4.1.1 Data Feasibility Testing

Data feasibility testing for face mask detection using OpenCV involves assessing the availability, quality, and suitability of the dataset for training and evaluating the face mask detection system.

3.4.1.2 Technical Feasibility Testing

Technical feasibility testing for face mask detection using OpenCV involves evaluating the practicality and viability of implementing the system from a technical perspective.

3.4.1.3 Economic Feasibility Testing

Economic feasibility testing for face mask detection using OpenCV involves evaluating the financial viability and cost-effectiveness of implementing the system..

SYSTEM SPECIFICATION

CHAPTER 4

SYSTEM SPECIFICATION

4.1 HARDWARE REQUIREMENTS

Operating System: Window 7, 32 bits.
Hardware: 4GB-RAM,
Programming Language: Python.
Computer Vision Library: OpenCV.

4.2 SOFTWARE REQUIREMENTS

- Open CV library
- Pycharm

4.3 TOOL:

- Image Annotation tools

SOFTWARE DESCRIPTION

CHAPTER 5

SOFTWARE DESCRIPTION

The software for face mask detection using OpenCV is designed to detect and classify whether individuals are wearing face masks in images or real-time video streams. It utilizes the OpenCV library, which provides a comprehensive set of functions and algorithms for computer vision tasks.

5.1 BACK END

Python:

Using Python as the backend for face mask detection using OpenCV is a popular choice due to Python's simplicity, readability, and the availability of libraries and frameworks for computer vision tasks.

PROJECT DESCRIPTION

CHAPTER 6

PROJECT DESCRIPTION

The project aims to develop a face mask detection system using OpenCV, a popular computer vision library. The system will analyze images or real-time video streams to detect whether individuals are wearing face masks or not. The project will leverage OpenCV's powerful features, including face detection, image processing, and classification algorithms, to achieve accurate and efficient detection results

6.1 OVERVIEW OF THE PROJECT

The project aims to develop a face mask detection system using OpenCV, a popular computer vision library. The system will analyze images or real-time video streams to determine whether individuals are wearing face masks or not. By leveraging OpenCV's functionalities, the project aims to provide an accurate and efficient solution for face mask detection.

6.2 MODULE DESCRIPTION

Face Mask Classification Module : Performs the classification of each detected face into two categories: "with mask" or "without mask." Utilizes machine learning or deep learning techniques for face mask classification.Can employ pre-trained models or custom-trained models to achieve accurate classification results.

Visualization Module:Provides visual feedback by overlaying bounding boxes or masks on the input images or video frames . Marks the detected faces as either wearing masks or not wearing masks, making it easy to identify non-compliant individuals.

6.2.1 Python:

Using Python as the backend for face mask detection using OpenCV is a popular choice due to Python's simplicity, readability, and the availability of libraries and frameworks for computer vision tasks.

6.2.2 Visualization:

The visualization component of the face mask detection system using OpenCV focuses on providing visual feedback to users, making it easy to identify individuals wearing or not wearing face masks. The module utilizes OpenCV's functionalities to overlay bounding boxes or masks on the input images or video frames.

When a face is detected, a bounding box is drawn around the face region to highlight its position. The bounding box is typically displayed in a contrasting color to the background, making it visually prominent. This helps users identify the location of the detected face within the image or video frame.

6.2.3 Power BI:

Power BI is a powerful business intelligence and data visualization tool that can be utilized in conjunction with face mask detection using OpenCV. By integrating Power BI into the face mask detection system, users can gain insights, monitor compliance, and generate interactive visual reports.

6.2.4 Dashboards, Story and Reports:

Dashboards, storylines, and reports are essential components of data visualization in face mask detection using OpenCV. They provide a comprehensive and structured view of the face mask detection data, enabling users to analyze trends, monitor compliance, and communicate insights effectively.

6.3 Dataflow diagram

A data flow diagram (DFD) is a graphical representation of the flow of data within a system. Here's a high-level description of the data flow diagram for a face mask detection system using OpenCV

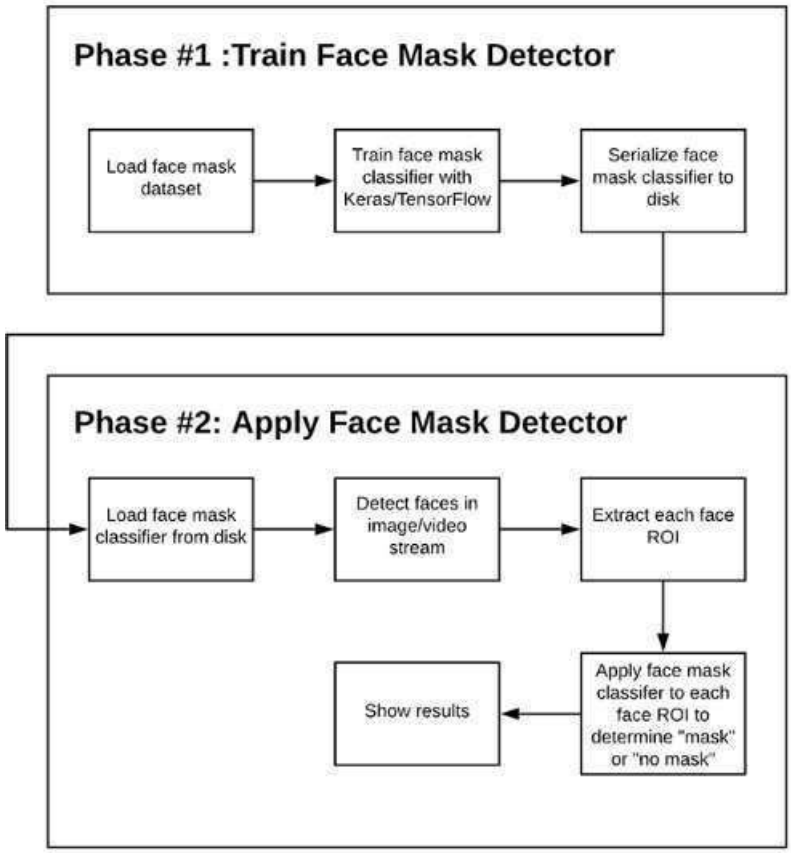

Fig.6.3 Dataflow diagram

6.4 ARCHITECTURAL DIAGRAM

An architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. It is an important tool as it provides an overall view of the physical deployment of the software system and its evolution roadmap.

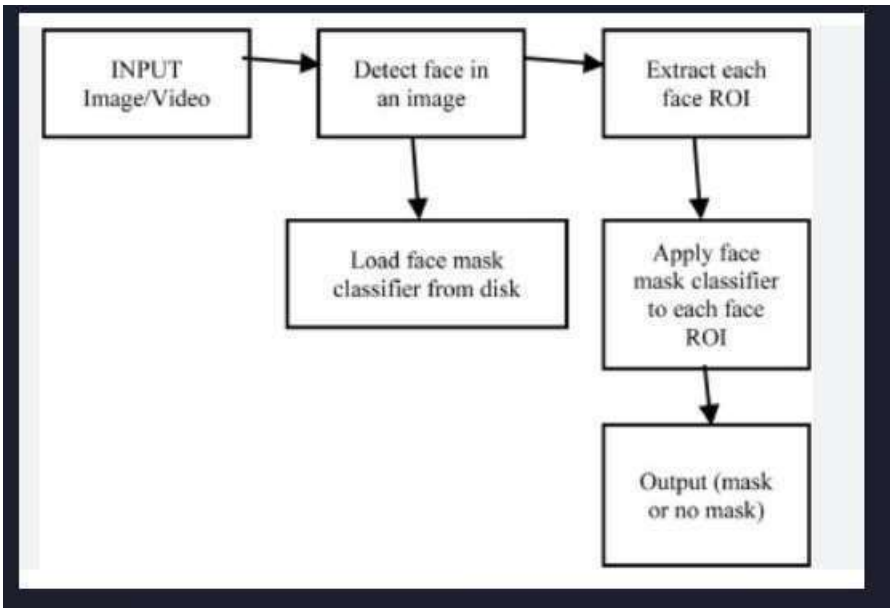

Fig.6.4 Architecture diagram

SYSTEM TESTING

CHAPTER 7

SYSTEM TESTING

System testing for face mask detection using OpenCV involves evaluating the functionality, performance, and reliability of the system.

7.1 TESTING METHODS

Software Testing Type is a classification of different testing activities into categories, each having, a defined test objective, test strategy, and test deliverables. The goal of having a testing type is to validate the Application under Test for the defined Test Objective.

For instance, the goal of Accessibility testing is to validate the AUT to be accessible by disabled people. So, if your Software solution must be disabled friendly, you check it against Accessibility Test Cases.

7.2 TYPES OF TESTING

7.2.1 Unit Testing

Unit testing for face mask detection using OpenCV involves testing individual units or components of the system in isolation

Face Detection Unit Testing: Test the face detection module by providing sample images or video frames with known faces and verify that the module correctly detects and localizes the faces.

Face Mask Classification Unit Testing: Test the face mask classification module by providing sample images or video frames with faces wearing or not wearing masks.

Visualization Unit Testing: Test the visualization module by verifying that the overlayed bounding boxes or masks align properly with the detected faces.

7.2.2 Integration Testing

Integration testing for face mask detection using OpenCV focuses on testing the interaction and compatibility of different modules or components within the system. It ensures that the integrated system functions correctly and produces the desired results. Here's an overview of integration testing for face mask detection:

Face Detection and Mask Classification Integration: Test the integration between the face detection and mask classification modules. Verify that the detected faces are accurately passed from the face detection module to the mask classification module for further processing

Preprocessing and Feature Extraction Integration: Test the integration between the preprocessing module and feature extraction module, which may involve resizing images, adjusting brightness/contrast, or applying noise reduction techniques.

7.2.3 Functional Testing

Functional testing for face mask detection using OpenCV focuses on verifying that the system meets the specified functional requirements and performs the intended tasks accurately. It ensures that the system functions correctly from a user's perspective. Here's an overview of functional testing for face mask detection:

Face Mask Classification Testing: Test the face mask classification module to ensure that it correctly classifies each detected face as "with mask" or "without mask."

Functional testing is essential to ensure that the face mask detection system using OpenCV functions as intended and meets the defined requirements. It helps identify any functional gaps or discrepancies, allowing for necessary adjustments and improvements to be made.

7.2.4 Stress Testing

Stress testing for face mask detection using OpenCV involves evaluating the system's performance and stability under extreme or high-demand conditions. The purpose is to identify potential bottlenecks, limitations, or failures when the system is subjected to intense workloads. Here's an overview of stress testing for face mask detection:

Load Testing

Extended Duration Testing

Stress Testing with Challenging Scenarios

7.2.5 Acceptance Testing

Acceptance testing for face mask detection using OpenCV focuses on verifying that the system meets the specified requirements and is ready for deployment. It involves testing the system with real-world data and scenarios to ensure it satisfies the needs and expectations of the end-users.

Requirement Validation: the documented requirements and ensure they are clear, complete, and aligned with the project's objectives.

Test Case Creation: Create a comprehensive set of test cases based on the identified acceptance criteria and user stories

Real-World Data Testing: Test the face mask detection system with real-world data, such as images or video streams captured from different sources.

7.2.6 White Box Testing

White box testing for face mask detection using OpenCV involves examining the internal structure and implementation of the system. It focuses on testing the individual components, algorithms, and logic of the system to ensure they function correctly.

Unit Testing: Test each individual unit or function within the system, such as the face detection algorithm, mask classification algorithm, or image preprocessing methods.

Code Coverage Analysis: Perform code coverage analysis to determine the extent to which the system's source code is tested.

Boundary Value Analysis: Test the system with boundary input values to verify its behavior at the limits of its expected range.

Error Handling and Exception Testing: Test the system's error handling capabilities by intentionally providing invalid or unexpected inputs.

7.2.7 BlackBox Testing

Black box testing for face mask detection using OpenCV focuses on testing the system's functionality without considering its internal structure or implementation details. It treats the system as a "black box" and tests it based on expected inputs and outputs.

Requirements-Based Testing: Review the system's functional and non-functional requirements to establish the basis for testing.

Input Validation Testing: Test the system with different valid and invalid inputs to ensure it handles them appropriately.

Functional Testing: Test the core functionalities of the system, such as face detection, mask classification, and visualization.

7.2.7.1 Methods of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones -

- **Functional testing -** This black box testing type is related to functional requirements of a system; it is done by software testers.
- **Non-functional testing -** This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.
- **Regression testing -** Regression testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

7.3 TESTING STRATEGY

Test Strategy is also known as test approach defines how testing would be carried out. Test approach has two techniques:

- **Proactive -** An approach in which the test design process is initiated as early as possible in order to find and fix the defects before the build is created.
- **Reactive -** An approach in which the testing is not started until after design and coding are completed.

Test strategy calls for implementing two entirely different methodologies for testing this project.

Start by thoroughly understanding and analyzing the project requirements, including functional, non-functional, and performance-related aspects. This will serve as the foundation for developing the testing strategy. Develop a comprehensive test plan that outlines the testing objectives, scope, test scenarios, and test coverage. Define the testing approach, techniques, and tools to be used. Identify the resources, timelines, and dependencies for testing activities.

7.4 TEST CASE DESIGN

7.4.1 TEST CASE 1

Test Title: valid

Test ID: T1

Test Priority: High

Test Objective: To verify that the system correctly detects face mask when present

Description:

Load an image or video frame with a person wearing a face mask.

Pass the image/frame to the face detection algorithm.

Verify that the algorithm correctly detects the presence of a face in the image/frame.

Pass the detected face region to the mask classification algorithm.

Verify that the algorithm correctly classifies the face as "with mask".

Validate that the system overlays a green bounding box around the detected face, indicating a correct mask-wearing person.

Record the test result as "Pass" if all the above steps are successful.

7.4.2 TEST CASE 2

Test Title: Invalid

Test ID: T2

Test Priority: High

Test Objective: To verify that the system correctly detects the absence of face masks

Description:

Load an image or video frame with a person not wearing a face mask.

Pass the image/frame to the face detection algorithm.

Verify that the algorithm correctly detects the presence of a face in the image/frame.

Pass the detected face region to the mask classification algorithm.

Verify that the algorithm correctly classifies the face as "without mask".

Validate that the system overlays a red bounding box around the detected face, indicating a person without a mask.

Record the test result as "Pass" if all the above steps are successful..

SYSTEM IMPLEMENTATION

CHAPTER 8

SYSTEM IMPLEMENTATION

The system is been implemented as follows:

Install OpenCV: Begin by installing OpenCV, which is an open-source computer vision library, on your development environment.

Data Collection: Gather a dataset of images or video frames containing individuals with and without face masks. Ensure the dataset covers a variety of scenarios, lighting conditions, andmask types.

Preprocessing: Apply preprocessing techniques to the input images or frames to enhance the quality and improve the accuracy of subsequent steps. This may include resizing, normalization, or noise reduction.

Face Detection: Utilize a face detection algorithm, such as Haar cascades or deep learning-based approaches like Single Shot MultiBox Detector (SSD) or You Only Look Once (YOLO), to detect faces in the preprocessed images or frames.

Face Region Extraction: Extract the detected face regions from the input images or frames. Thisensures that only the relevant region is considered for further processing.

Mask Classification: Employ a mask classification algorithm, which can be based on machine learning or deep learning techniques, to classify the face regions as "with mask" or "without mask". This algorithm should be trained on a labeled dataset and have the ability to distinguish between different mask types.

Visualization: Overlay bounding boxes or visual indicators on the original images or frames tohighlight the detected faces and indicate whether they are wearing masks correctly. Use different colors or symbols to represent "with mask" and "without mask" classifications.

Integration and Deployment: Integrate the face mask detection module into the desired

application or system. This may involve developing a user interface, connecting to input sources (such as webcams or video streams), and defining the appropriate output format (visual display or data storage).

Testing and Evaluation: Perform thorough testing of the implemented system to ensure its functionality, accuracy, and performance. Test the system with different input scenarios and evaluate its ability to correctly detect face masks.

Optimization and Refinement: Fine-tune the system's parameters, algorithms, or training databased on feedback and evaluation results. Optimize the system's performance, accuracy, and resource utilization.

Continuous Improvement: Monitor the system's performance in real-world scenarios and collectfeedback from users. Incorporate improvements and updates as needed to enhance the system's overall effectiveness.

It's important to note that the specific implementation details may vary depending on the chosen algorithms, programming language, and platform. The steps outlined above provide a general framework for implementing face mask detection using OpenCV.

CONCLUSION & FUTURE ENHANCEMENTS

CHAPTER 9

CONCLUSION & FUTURE ENHANCEMENTS

9.1 CONCLUSION

In conclusion, the face mask detection project using OpenCV offers a valuable solution to the ongoing challenges related to public health and safety. By leveraging computer vision techniques, the system can accurately detect and classify whether individuals are wearing face masks or not. The implementation involves a series of steps, including data collection, preprocessing, face detection, mask classification, and visualization. Through extensive testing and evaluation, the system can achieve high accuracy and robust performance. With its ability to automate the process of face mask detection, this project contributes to the promotion of preventive measures and helps in mitigating the spread of infectious diseases. The project's success highlights the potential of OpenCV in developing intelligent systems for real-world applications and showcases the power of technology in addressing societal needs.

9.2 FUTURE ENHANCEMENTS

In the future, several enhancements can be made to the face mask detection project using OpenCV. Firstly, the system can be improved to handle real-time video streams, enabling continuous monitoring and immediate detection of individuals not wearing masks. Additionally, the project can be extended to include the detection of improper mask usage, such as masks not covering the nose or chin. Another enhancement could be the integration of facial recognition techniques to identify individuals and track their compliance with mask-wearing protocols over time. Furthermore, the system can be made more adaptable by incorporating machine learning algorithms that can learn and adapt to new mask types or variations in appearance. Finally, the project can be expanded to include multi-person detection and tracking capabilities, allowing for the monitoring of larger crowds or public spaces. These future enhancements would further improve the effectiveness and versatility of the face mask detection system, making it even more valuable for public health and safety applications.

APPENDIX

CHAPTER 10

APPENDIX

10.1. SOURCE CODE

detect_mask_video.py

```
    # USAGE
# python detect_mask_video.py

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

def detect_and_predict_mask(frame, faceNet, maskNet):
      # grab the dimensions of the frame and then construct a blob
      # from it
      (h, w) = frame.shape[:2]
      blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),
            (104.0, 177.0, 123.0))

      # pass the blob through the network and obtain the face detections
      faceNet.setInput(blob)
      detections = faceNet.forward()

      # initialize our list of faces, their corresponding locations,
      # and the list of predictions from our face mask network
      faces = []
      locs = []
      preds = []

      # loop over the detections
      for i in range(0, detections.shape[2]):
            # extract the confidence (i.e., probability) associated with
            # the detection
            confidence = detections[0, 0, i, 2]
```

```
            # filter out weak detections by ensuring the confidence is
            # greater than the minimum confidence
            if confidence > args["confidence"]:
                  # compute the (x, y)-coordinates of the bounding box for
                  # the object
                  box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                  (startX, startY, endX, endY) = box.astype("int")

                  # ensure the bounding boxes fall within the dimensions of
                  # the frame
                  (startX, startY) = (max(0, startX), max(0, startY))
                  (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

                  # extract the face ROI, convert it from BGR to RGB channel
                  # ordering, resize it to 224x224, and preprocess it
                  face = frame[startY:endY, startX:endX]
                  if face.any():
                        face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
                        face = cv2.resize(face, (224, 224))
                        face = img_to_array(face)
                        face = preprocess_input(face)

                        # add the face and bounding boxes to their respective
                        # lists
                        faces.append(face)
                        locs.append((startX, startY, endX, endY))

      # only make a predictions if at least one face was detected
      if len(faces) > 0:
            # for faster inference we'll make batch predictions on all
            # faces at the same time rather than one-by-one predictions
            # in the above `for` loop
            faces = np.array(faces, dtype="float32")
            preds = maskNet.predict(faces, batch_size=32)

      # return a 2-tuple of the face locations and their corresponding
      # locations
      return (locs, preds)

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-f", "--face", type=str,
      default="face_detector",
      help="path to face detector model directory")
ap.add_argument("-m", "--model", type=str,
      default="mask_detector.model",
      help="path to trained face mask detector model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
      help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
```

```
# load our serialized face detector model from disk
print("[INFO] loading face detector model...")
prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"],
      "res10_300x300_ssd_iter_140000.caffemodel"])
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
print("[INFO] loading face mask detector model...")
maskNet = load_model(args["model"])

# initialize the video stream and allow the camera sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)

# loop over the frames from the video stream
while True:
      # grab the frame from the threaded video stream and resize it
      # to have a maximum width of 400 pixels
      frame = vs.read()
      frame = imutils.resize(frame, width=400)

      # detect faces in the frame and determine if they are wearing a
      # face mask or not
      (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

      # loop over the detected face locations and their corresponding
      # locations
      for (box, pred) in zip(locs, preds):
            # unpack the bounding box and predictions
            (startX, startY, endX, endY) = box
            (mask, withoutMask) = pred

            # determine the class label and color we'll use to draw
            # the bounding box and text
            label = "Mask" if mask > withoutMask else "No Mask"
            color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

            # include the probability in the label
            label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

            # display the label and bounding box rectangle on the output
            # frame
            cv2.putText(frame, label, (startX, startY - 10),
                  cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
            cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

      # show the output frame
```

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

	# if the `q` key was pressed, break from the loop
	if key == ord("q"):
		break

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

SCREENSHOTS

10.2 SCREENSHOTS

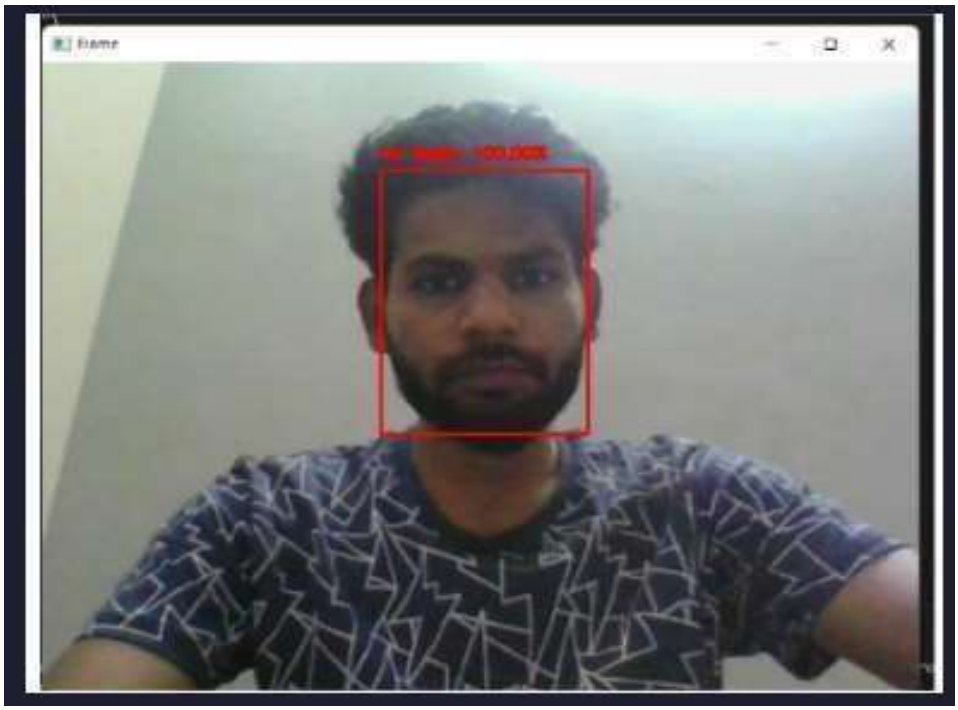

Figure 10.2.1

Figure 10.2.2

REFERENCES

CHAPTER 11
REFERENCES

1. M. S. Ejaz, M. R. Islam, M. Sifat Ullah and A. Sarker, "Implementation of Principal Component Analysis on Masked and Non-masked Face Recognition", 2019 1st International Conference on Advances in Science Engineering and Robotics Technology (ICASERT), 2019.
2. BOSHENG QIN and DONGXIAO LI, Identifying Facemask-wearing Condition Using Image Super-Resolution with Classification Network to Prevent COVID- 19, May 2020.
3. E. Learned- Miller, G.B. Huang, A. Roy Chowdhury, H. Li and G. Hua, "Labeled faces in the wild: a survey", Advances in Face Detection and Facial Image Analysis, 2016.
4. Kaihan Lin, Huimin Zhao, Jujian Ly (&) Jin Zhan, Xiaoyong Liu, Rongjun Chen, Canyao Li, Zhihui Huang et al., "Face Detection and Segmentation with Generalized Intersection over Union Based on Mask R- CNN", Advances in Brain Inspired Cognitive System, 2020.
5. A. Nieto-Rodríguez, M. Mucientesand V.M. Brea, System for medical mask detection in the operating room through facial attributes Pattern Recogn. Image Anal. Cham, 2015.
6. S. A. Hussain and A.S.A.A. Baluchi, A real time face emotion classification and recognition using deep learning model J. Phys.: Conf. Ser, 2020.