A MAJOR PROJECT REPORT ON
SHEILD MASK DETECTION SYSTEM USING DEEP LEARNING

submitted in partial fulfillment of the requirement. for the award of the degree of

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

By

D.Bindu	22P65A0504
K.Shiva Shankar Reddy	22P65A0506
K.Pranay Chandra	22P65A0507

Under the esteemed guidance of

Dr.G.Ashok Kumar
Associate Professor, Dept. of CSE

Department of Computer Science and Engineering

Counselling Code : VBIT
VIGNANA BHARATHI Institute of Technology
(A UGC Autonomous Institution, Approved by AICTE, Accredited by NBA & NAAC-A Grade, Affiliated to JNTUH)
Aushapur Village, Ghatkesar Mandal,Medchal Malkajigiri (District) Telangana-501301

May-2025

DECLARATION

We, **D.Bindu , K.Shiva Shankar Reddy , K.Pranay Chandra ,** bearing hall ticket numbers **(22P65A0504), (22P65A0506), (22P65A0507)** hereby declare that the Major project report entitled "**Sheild Mask Detection System using Deep learning**" under the guidance of **Dr.G.Ashok Kumar, Associate Professor**, Department of Computer Science and Engineering**, Vignana Bharathi Institute of Technology, Hyderabad**, have submitted to Jawaharlal Nehru Technological University Hyderabad, Kukatpally, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

This is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

D.Bindu	22P65A0504
K.Shiva Shankar Reddy	22P65A0506
K.Pranay Chandra	22P65A0507

Counselling Code : VBIT

VIGNANA BHARATHI Institute of Technology

(A UGC Autonomous Institution, Approved by AICTE, Accredited by NBA & NAAC-A Grade, Affiliated to JNTUH)

Aushapur (V), Ghatkesar (M), Hyderabad, Medchal – Dist, Telangana – 501 301.

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Major project titled "**Sheild Mask Detection System using Deep learning**" Submitted by **D.Bindu (2265A0504), K.Shiva Shmakar Reddy (22P65A0506), K.Pranay Chandra (22P65A0507)** B. Tech, IV- II semester, Department of Computer Science & Engineering is a record of the bonafide work carried out by them.

The Design embodied in this report have not been submitted to any other University for the award of any degree.

INTERNAL GUIDE	HEAD OF THE DEPARTMENT
Dr.G.Ashok Kumar	**Dr. Raju Dara**
Associate Professor , CSE Dept.	**Professor,CSE Dept.**

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are extremely thankful to our beloved Chairman, **Dr. N. Goutham Rao** and Secretary, **Dr. G. Manohar Reddy** who took keen interest to provide us the infrastructural facilities for carrying out the project work.

Self-confidence, hard work, commitment and planning are essential to carry out any task. Possessing these qualities is sheer waste, if an opportunity does not exist. So, we whole-heartedly thank **Dr. P.V.S. Srinivas,** Principal, and **Dr. Dara Raju**, Head of the Department, Computer Science and Engineering for their encouragement and support and guidance in carrying out the project.

We would like to express our indebtedness to the Overall Project Coordinator, **Dr.M.Venkateswara Rao,** Professor, and Section Coordinators, **Ms. P. Suvarna Puspha**, Associate Professor, **Ms. A. Manasa**, Associate Professor, Department of CSE, for their valuable guidance during the course of project work.

We thank our Project Guide, **Dr.G. Ashok Kumar, Associate Professor**, Department of Computer Science and Engineering for providing us with an excellent project and guiding us in completing our Major Project successfully.

We would like to express our sincere thanks to all the staff of Computer Science and Engineering, VBIT, for their kind cooperation and timely help during the course of our project. Finally, we would like to thank our parents and friends who have always stood by us whenever we were in need of them.

ABSTRACT

In the wake of global health crises, wearing protective gear such as face masks and face shields has become crucial for minimizing the spread of airborne diseases. This project presents a real-time Shield Mask Detection System that leverages deep learning techniques for classifying whether individuals are wearing a face mask, face shield, both, or none. The system integrates Haar Cascade Classifier for efficient face detection and employs a lightweight Convolutional Neural Network (CNN) architecture using MobileNet for classification tasks.

The MobileNet model, known for its computational efficiency and accuracy, is trained on a custom dataset comprising images labeled into four categories: with mask, with shield, with both, and without protection. The training process involves image preprocessing, augmentation, and transfer learning to enhance model performance. Upon detecting a face using Haar Cascade, the system passes the region of interest to the trained MobileNet model, which predicts the protective status in real time.This deep learning-based detection system is designed to be deployed in public and private spaces to assist in enforcing health compliance measures. Its high accuracy, low computational cost, and real-time processing capability make it suitable for edge devices and embedded systems

Keywords:*Deep Learning,CNN (Convolutional Neural Network),MobileNet,Haar Cascade,Face Detection,Shield Mask Detection*

VISION

To become, a Center for Excellence in Computer Science and Engineering with a focused Research, Innovation through Skill Development and Social Responsibility.

MISSION

DM-1: Provide a rigorous theoretical and practical framework across ***State-of-the-art***

infrastructure with an emphasis on ***software development***.

DM-2: Impact the skills necessary to amplify the pedagogy to grow technically and to meet ***interdisciplinary needs*** with collaborations.

DM-3: Inculcate the habit of attaining the professional knowledge, firm ethical values,

innovative research abilities and societal needs.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO-01: Domain Knowledge: Synthesize mathematics, science, engineering fundamentals, pragmatic programming concepts to formulate and solve engineering problems using prevalent and prominent software.

PEO-02: Professional Employment: Succeed at entry- level engineering positions in the software industries and government agencies.

PEO-03: Higher Degree: Succeed in the pursuit of higher degree in engineering or other by applying mathematics, science, and engineering fundamentals.

PEO-04: Engineering Citizenship: Communicate and work effectively on team-based engineering projects and practice the ethics of the profession, consistent with a sense of social responsibility.

PEO-05: Lifelong Learning: Recognize the significance of independent learning to become experts in chosen fields and broaden professional knowledge.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO-01: Ability to explore emerging technologies in the field of computer science and engineering.

PSO-02: Ability to apply different algorithms indifferent domains to create innovative products.

PSO-03: Ability to gain knowledge to work on various platforms to develop useful and secured applications to the society.

PSO-04: Ability to apply the intelligence of system architecture and organization in designing the new era of computing environment.

PROGRAM OUTCOMES (POs)

Engineering graduates will be able to:

PO-01: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO-02: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO-03: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and cultural, societal, and environmental considerations.

PO-04: Conduct investigations of complex problems : Use research-based knowledge and research methods including design of experiments, analysis and Department of Computer Science and Engineering interpretation of data, and synthesis of the information to provide valid conclusions.

PO-05: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO-06: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO-07: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO-08: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO-09: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO-10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO-11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO-12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Project Mapping Table:

a) PO Mapping:

PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
Title	3	3	3	2	3	2	1	2	2	2	1	3

b) PSO Mapping:

PSO	PSO1	PSO2	PSO3	PSO4
Title	3	3	3	3

List of Figures

List of Tables

Nomenclature

CNN	Convolutional Neural Networks
MobileNetV2	CNN architecture optimized for mobile
Mask	Face Mask worn for respiratory protection
Sheild	Refers to face shield and a transparent protection for face
No Mask/No Shield	Subject is not wearing any protective wear
Bounding Box	A rectangular to used to detect objects(face/mask/shield)
IOU	Intersection Over Union
Accuracy	The percentage of predictions made by model
Transfer Learning	A technique where a model developed for one task is reused as starting point
Preprossessing	Steps taken to prepare raw images for input
Dataset	A collection of labeled images
TP/FP/FN/TN	True Positive/False Positive/False Negative/True Negative
Inference Time	Time taken by the model to process

TABLE OF CONTENTS

CHAPTER – 1
INTRODUCTION

CHAPTER – 1
INTRODUCTION

1.1 INTRODUCTION TO IMAGE AND ITS CLASSIFICATION

The emergence of global pandemics like COVID-19 has underscored the importance of personal protective equipment (PPE) such as face masks and face shields in minimizing the spread of infectious diseases. To ensure public safety, automated surveillance systems have become essential for monitoring compliance with health guidelines. This project aims to develop a real-time shield mask detection system using deep learning techniques. By integrating Haar Cascade classifiers for face detection and a lightweight MobileNet-based CNN model for classification, the system can accurately determine whether individuals are wearing a face mask, a face shield, both, or none. The system is designed for deployment in public areas such as hospitals, airports, shopping malls, and schools to help authorities monitor PPE usage effectively and efficiently.

An image is a two-dimensional picture, which has a similar appearance to some subject usually a physical object or a person.Image is a two-dimensional, such as a photograph, screen display, and as well as a three-dimensional, such as a statue. They may be captured by optical devices—such as cameras, mirrors, lenses, telescopes, microscopes, etc. and natural objects and phenomena, such as the human eye or water surfaces.The word image is also used in the broader sense of any two-dimensional figure such as a map, a graph, a pie chart, or an abstract painting. In this wider sense, images can also be rendered manually, such as by drawing, painting, carving, rendered automatically by printing or computer graphics technology, or developed by a combination of methods, especially in a pseudo-photograph.

Fig 1: Colour image to Gray scale Conversion Process

CLASSIFICATION OF IMAGES

There are 3 types of images used in Digital Image Processing. They are

1. Binary Image
2. Gray Scale Image
3. Colour Image

BINARY IMAGE:

A binary image is a digital image that has only two possible values for each pixel. Typically the two colors used for a binary image are black and white though any two colors can be used. The color used for the object(s) in the image is the foreground color while the rest of the image is the background color.

Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1).This name black and white, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as grayscale images.

GRAY SCALE IMAGE

A grayscale Image is digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray(0-255), varying from black(0) at the weakest intensity to white(255) at the strongest.

Grayscale images are distinct from one-bit black-and-white images, which in the context of computer imaging are images with only the two colors, black, and white (also called bi-level or binary images). Grayscale images have many shades of gray in between. Grayscale images are also called monochromatic, denoting the absence of any chromatic variation.

COLOUR IMAGE:

A (digital) color image is a digital image that includes color information for each pixel. Each pixel has a particular value which determines its appearing color. This value is qualified by three numbers giving the decomposition of the color in the three primary colors Red, Green and Blue. Any color visible to human eye can be represented this way. The decomposition of a color in the three primary colors is quantified by a number between 0 and 255. For example, white will be coded as R = 255, G = 255, B = 255; black will be known as (R,G,B) = (0,0,0); and say, bright pink will be : (255,0,255).

1.2 MOTIVATION

The global health crisis caused by COVID-19 underscored the urgent need for effective, scalable solutions to ensure public compliance with safety measures, especially the use of face masks and shields. While government mandates and awareness campaigns have encouraged their usage, actual enforcement in crowded areas such as markets, public transport, hospitals, and workplaces remains a significant challenge.

Manual monitoring of individuals for mask or shield usage is not only labor-intensive but also exposes security personnel and health workers to potential health risks. This highlights the necessity for a non-contact, automated, and real-time system that can detect whether individuals are wearing proper protective equipment.

The rise of deep learning and computer vision offers a powerful and cost-effective approach to solve this problem. Leveraging these technologies allows the system to operate with high accuracy and speed, making it suitable for integration with existing surveillance infrastructure. Additionally, real-time detection can aid in issuing warnings, triggering alerts, or recording violations—thus contributing to more disciplined and safer public behavior.

This project is motivated by the need to combine technology with public health efforts, aiming not only to support pandemic management but also to pave the way for future systems capable of monitoring various safety gear in industrial or medical environments.

1.3 EXISTING SYSTEM

In recent years, especially during the COVID-19 pandemic, various automated systems have been developed to detect whether individuals are wearing face masks. These systems typically use deep learning models trained on facial images to classify inputs into two categories: mask or no mask. Frameworks such as MobileNet, YOLO, and SSD have been commonly used for this purpose, achieving good accuracy in controlled environments.

However, the majority of these systems are limited in scope, focusing only on face masks and ignoring other protective gear such as face shields. This is a critical gap, as face shields are widely used in many sectors including healthcare, manufacturing, and public services. Furthermore, most existing solutions lack the ability to distinguish between multiple protective states—such as wearing a shield only, a mask only, both, or neither.

Another limitation is that many of these systems are not optimized for real-time deployment, struggle with variations in lighting or camera angles, and are not easily adaptable to different use cases. They also often require high computational power, making them unsuitable for edge devices or real-world surveillance setups.

1.4.PROPOSED SYSTEM

To overcome the limitations of existing systems, this project proposes a Shield Mask Detection System that uses advanced deep learning and computer vision techniques to automatically detect whether a person is wearing a face mask, a face shield, both, or neither. The proposed system is designed to operate in real-time with high accuracy, making it suitable for public surveillance, healthcare facilities, and workplaces.

Key features of the proposed system include:

Face Detection: The first step is to detect human faces from live video feed or images. This is achieved using robust face detection algorithms like Haar Cascade, Dlib, or more advanced models like YOLOv5 or MTCNN.

Image Preprocessing: The detected face regions are cropped, resized, and normalized to ensure consistency and improve model accuracy.

Classification using CNN: A Convolutional Neural Network (CNN) or transfer learning-based model (e.g., MobileNetV2, ResNet50) is trained to classify the input into four categories:

Mask only

Shield only

Both mask and shield

No protective gear

Real-Time Prediction: The trained model is integrated with a webcam or CCTV feed using OpenCV to perform real-time predictions and display the result with bounding boxes and labels.

Alert Mechanism (Optional): Based on the detection result, the system can trigger alerts (e.g., sound, email notification, or log entry) if non-compliance is detected.

This system offers a more holistic approach to PPE compliance monitoring and can be extended for future applications such as detecting other protective gear (gloves, gowns, helmets, etc.) or integrating with access control systems.

1.5 PROBLEM DEFINITION

In the wake of global health crises such as the COVID-19 pandemic, the enforcement of face mask and shield usage has become critical for public safety. However, manual monitoring of personal protective equipment (PPE) compliance is inefficient, prone to human error, and not scalable in densely populated or high-traffic areas.

While some existing systems can detect whether a face mask is worn or not, they often fail to:

Distinguish between face masks, face shields, both, or neither.

Operate efficiently in real-time scenarios.

Perform reliably under varying environmental conditions such as lighting, occlusion, or angles.

Support scalability and deployment on edge devices.

There is a pressing need for an automated, real-time, and multi-class detection system that can accurately identify the type of protective gear worn by individuals. Such a system should be able to assist authorities in monitoring compliance, issuing alerts, and maintaining safety protocols in public or restricted areas.

The goal of this project is to develop a deep learning-based Shield Mask Detection System that addresses these limitations and provides an efficient, accurate, and deployable solution for PPE compliance monitoring.

1.6.OBJECTIVE

The primary objective of this project is to design and implement an automated Shield Mask Detection System using deep learning techniques to accurately classify individuals based on their use of personal protective equipment (PPE). The system should be capable of real-time operation and suitable for deployment in public and private spaces.

- To develop a deep learning-based system capable of detecting and classifying individuals based on the use of face masks, face shields, both, or none.
- To create and preprocess a comprehensive dataset containing images of individuals with different PPE conditions under varied environmental settings.
- To train and evaluate a convolutional neural network (CNN) or transfer learning model (e.g., MobileNetV2) for accurate multiclass classification.
- To implement real-time detection using OpenCV and integrate the trained model with video feeds from webcams or CCTV systems.
- To ensure the system is lightweight, accurate, and scalable for deployment in real-world environments, including edge devices.

1.7.SCOPE

1.Development of a Deep Learning Model

Create and train a deep learning model (e.g., CNN, MobileNetV2) for classifying individuals based on their use of face masks, face shields, both, or neither.

2.Dataset Collection and Preprocessing

Collect a diverse dataset of images with individuals wearing different combinations of face masks and shields, and preprocess these images for model training.

3.System Deployment for Real-World Use

Deploy the trained model on edge devices (e.g., CCTV cameras or Raspberry Pi) to enable practical, real-time use in monitoring public or private spaces.

4.Performance Evaluation and Optimization

Test the system under various real-world conditions (e.g., lighting, angles, occlusion) to ensure high accuracy and robustness, and optimize the model for better performance.

5.Real-Time Detection and Classification

Implement a system that can process live video feeds (from webcams or CCTV) and classify the presence of face masks and shields in real-time.

CHAPTER – 2

LITERATURE SURVEY

CHAPTER – 2

LITERATURE SURVEY

A COMPREHENSIVE STUDY ON SHEILD MASK DETECTION SYSTEM

A Literature Review, is an essential component of any academic or research project. It involves a thorough review of existing research, studies, and publications relevant to the subject or problem that the project aims to address. The primary purpose of conducting a literature survey is to gain an understanding of the current state of knowledge in the field, identify gaps, and build upon the work that has already been done.

In this review paper, Sharma et al. (2021) from India surveyed various deep learning-based approaches for face mask detection in public spaces. The authors reviewed different algorithms such as CNN, YOLO, and SSD and their performance in real-time mask detection systems. The study found that CNN-based models, particularly MobileNetV2 and ResNet, provide high accuracy and real-time efficiency, which is essential for deployment in public areas such as airports, stations, and malls.

Kaur et al. (2021) proposed a real-time face mask detection system using Convolutional Neural Networks (CNN) combined with transfer learning. They utilized pre-trained models like VGG16 and ResNet-50 for transfer learning, which helped in achieving high accuracy while reducing training time. The model was deployed for use in various public environments, such as hospitals and shopping malls, to ensure compliance with mask-wearing protocols during the COVID-19 pandemic.

Rani et al. (2021) from India extended the YOLOv4 (You Only Look Once) model to detect both face masks and face shields in real-time. Their system was trained on a custom dataset containing individuals wearing various combinations of masks and shields, as well as none at all. They demonstrated that YOLOv4 could detect both PPE items efficiently in real-time, achieving high accuracy and fast inference, suitable for surveillance systems in public spaces.

In their research, Patil et al. (2021) focused on using deep learning models for PPE detection in public spaces. They used a combination of Convolutional Neural Networks (CNNs) and YOLOv5 to detect multiple types of PPE, including face masks, shields, gloves, and safety goggles. Their model was trained to identify different types of protective gear in real-time video streams, making it suitable for deployment in high-traffic areas like malls, train stations, and offices.

Gupta et al. (2021) developed a real-time mask detection system using MobileNetV2 and TensorFlow for deployment in crowded public areas. Their system focused on detecting individuals who were either wearing a face mask or not and provided real-time alerts to ensure safety measures were being followed. This study emphasized the importance of edge deployment for such systems, making it feasible for real-world usage on devices like Raspberry Pi and smartphones.

In this study, Singh et al. (2020) proposed a MobileNetV2-based deep learning model to detect face masks and other personal protective equipment (PPE) in real-time. Their research focused on public health surveillance and ensuring compliance in settings like airports and shopping malls. The model was optimized for edge deployment on devices with limited resources, such as Raspberry Pi and smartphones, making it accessible for real-time monitoring.

Jain et al. (2020) utilized the YOLOv3 object detection model for real-time face mask detection. Their model was trained to recognize whether individuals were wearing a mask or not, using a custom dataset that included images captured under different lighting conditions and camera angles. The system was integrated with TensorFlow for efficient inference, making it suitable for deployment in environments with limited resources, such as embedded systems or mobile phones.

No.	Title/Focus	Methodology	Findings	Limitations	Future Work	Advantages
1	Face Mask Detection using Deep Learning and MobileNetV2	MobileNetV2 with transfer learning on face mask dataset	High accuracy with low computational cost	Limited dataset diversity affecting real-time performance	Expand dataset with more variations (e.g., shield, angles, lighting)	Efficient for edge devices like Raspberry Pi or Jetson Nano
2	Real-Time Face Mask and Face Shield Detection Using CNN	CNN from scratch, multi-class classification	Distinguishes between mask, shield, and no protection	Slow inference on low-end devices	Use lightweight models (e.g., MobileNet) for better performance	Useful for public space surveillance during pandemics
3	Face Detection using Haar Cascade Classifier	OpenCV Haar cascades for frontal face detection	Simple and fast face localization	Poor performance under occlusion/lighting variation	Combine with deep learning for improved accuracy	Quick integration in real-time systems
4	COVID-19 Face Mask Detector with OpenCV, Keras/TensorFlow	Fine-tuned CNN + OpenCV for face detection	Real-time detection with decent accuracy	Only binary classification	Add classes like face shield or incorrect mask usage	Easy implementation in surveillance systems
5	Lightweight CNN for Face Mask Detection	Custom lightweight CNN	Fast, power-efficient, reasonable accuracy	Accuracy drops in crowded/low-light scenes	Add preprocessing (e.g., histogram equalization)	Suitable for embedded systems in smart cities

Table 1: Comparison of the related work

CHAPTER – 3
REQUIREMENT ANALYSIS

CHAPTER – 3

REQUIREMENT ANALYSIS

The development of the Shield Mask Detection System involves understanding both the operational goals and the resources needed to achieve them. This section breaks down the key functional capabilities, technical constraints, and resource needs that the system must meet to ensure successful implementation and performance in real-world scenarios.

3.1.FUNCTIONAL REQUIREMENTS

The system's primary objective is to detect and differentiate between individuals wearing face masks, face shields, or neither, using deep learning techniques. The functional requirements are as follows:

Input Acquisition

The system must capture real-time video or accept image input from webcams, CCTV feeds, or stored media files.

Face Localization

It should accurately detect human faces within each frame before analyzing protective gear.

PPE Detection and Classification

Once a face is detected, the model must classify it into one of the following categories:

Mask Detected

Face Shield Detected

No PPE Detected

Real-Time Feedback

The system should be capable of providing instant visual feedback or alerts (e.g., sound notification or visual indicators) when individuals are not properly protected.

3.2. NON FUNCTIONAL REQUIREMENTS

While the above describe **what** the system should do, non-functional requirements address **how well** it should perform:

Efficiency: The system must be able to process video input in real-time or near-real-time with minimal latency.

Accuracy: High detection accuracy is essential for reliable PPE compliance enforcement.

Robustness: Should perform well under different lighting conditions, angles, and partial occlusions.

Portability: Ideally, the system should be deployable on edge devices (e.g., Raspberry Pi, Jetson Nano) or traditional desktops.

Maintainability: Code should be modular, well-documented, and easy to update or retrain with new data.

3.3.HARDWARE REQUIREMENTS

To support real-time deep learning inference, the following hardware is recommended:

Camera: Standard webcam or IP camera (HD recommended)

Processor: Minimum Intel i5 / Ryzen 5 or ARM Cortex-A72 (for embedded use)

RAM: At least 4 GB (8 GB preferred)

GPU (Optional): NVIDIA GPU (e.g., GTX 1050 Ti or higher) for accelerated inference

Storage: Sufficient space (~20GB) for models, logs, and datasets.

3.4.SOFTWARE REQUIREMENTS

The system development will rely on the following software stack:

Operating System: Windows 10/11, Ubuntu, or Raspberry Pi OS

Programming Language: Python 3.x

Libraries & Tools:

OpenCV (for image processing)

TensorFlow or PyTorch (for deep learning models)

Keras (if using TensorFlow backend)

NumPy, Pandas (for data manipulation)

Flask/Streamlit (for optional web interface)

3.5.SYSTEM ANALYSIS

System analysis is a fundamental phase in the development of any software or intelligent system. It involves examining the current scenario, understanding the requirements, identifying potential challenges, and defining how the proposed system will function within its intended environment. In the context of the Shield Mask Detection System, system analysis focuses on evaluating the need for automated personal protective equipment (PPE) detection, especially in public and high-risk areas, and determining the most effective way to implement such a system using modern deep learning techniques.

CHAPTER - 4
SYSTEM ANALYSIS & DESIGN

CHAPTER – 4

SYSTEM ANALYSIS & DESIGN

TECHNICAL BLUEPRINT OF SHEILD MASK DETECTION SYSTEM

The Shield Mask Detection System using deep learning addresses the increasing need for automated public health monitoring, especially in the context of enforcing the use of personal protective equipment such as face masks and face shields. The primary motivation behind this system is the inefficiency and inaccuracy of manual surveillance in high-traffic areas like hospitals, schools, and public transport hubs. Existing systems are either limited to detecting only face masks or suffer from poor performance in real-time scenarios and under varying environmental conditions.

To overcome these limitations, the proposed system leverages advanced computer vision techniques, specifically convolutional neural networks (CNNs), to detect and classify individuals based on whether they are wearing a mask, a face shield, or no protective gear at all. This classification is performed on live video input, allowing real-time monitoring and decision-making. By classifying faces as mask, shield, or none, the system can effectively help enforce safety protocols in a variety of environments.

The system is technically feasible with the support of modern deep learning frameworks such as TensorFlow or PyTorch and can be implemented using readily available hardware like webcams or CCTV systems. It is also economically viable, as it utilizes open-source tools and requires minimal infrastructure upgrades, making it a cost-effective solution for both small and large-scale deployments.

Functionally, the system must be capable of detecting faces in a video stream, classifying them accurately, and overlaying the classification results on the display. Additionally, it can include alert mechanisms for non-compliance, such as notifying security personnel or displaying a warning message. These functionalities ensure that the system serves its intended purpose effectively.

Non-functional requirements such as high accuracy, low latency, scalability for multi-camera setups, and data security are essential to ensure the system is both effective and practical for real-world deployment. The system should be able to process video in real-time (≥15 FPS) and maintain a high classification accuracy of over 90%. This guarantees that it can function smoothly and provide reliable results in a variety of use cases.

Overall, the Shield Mask Detection System represents a modern, intelligent solution to enhance safety and health compliance in shared environments. With its combination of deep learning, computer vision, and real-time monitoring, the system promises to make public spaces safer by automating and streamlining mask and shield detection.

UML DIAGRAMS

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

- Actors
- Business processes
- (Logical) Components
- Activities
- Programming language statements
- Database schemas, and
- Reusable software components.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies· UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

4.1.SEQUENCE DIGRAM

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

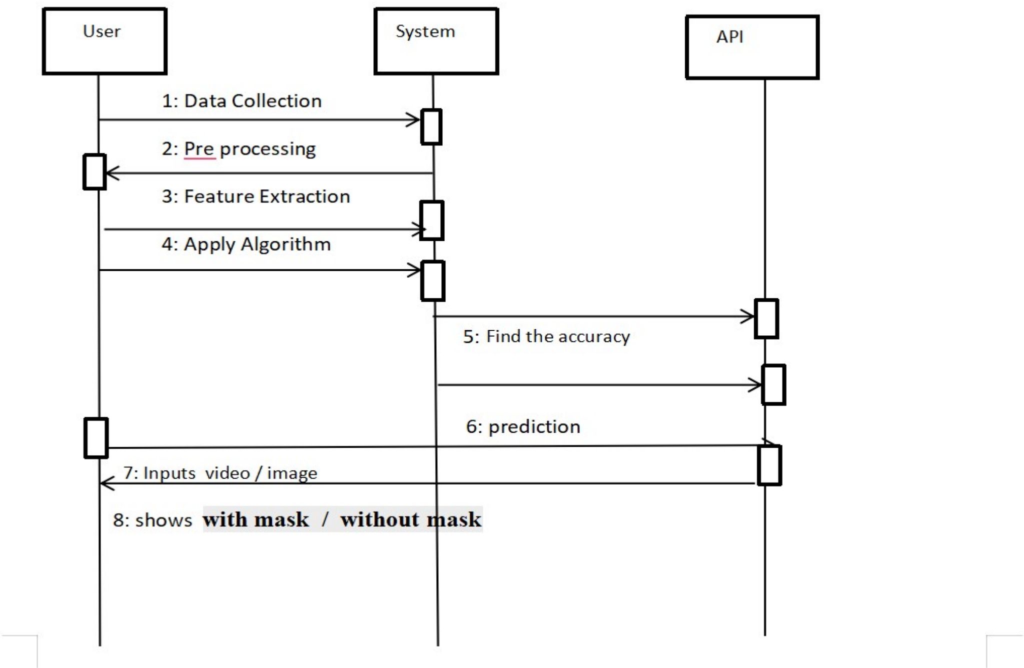


Fig 2: Sequence Diagram

4.2.ACTIVITY DIGRAM

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

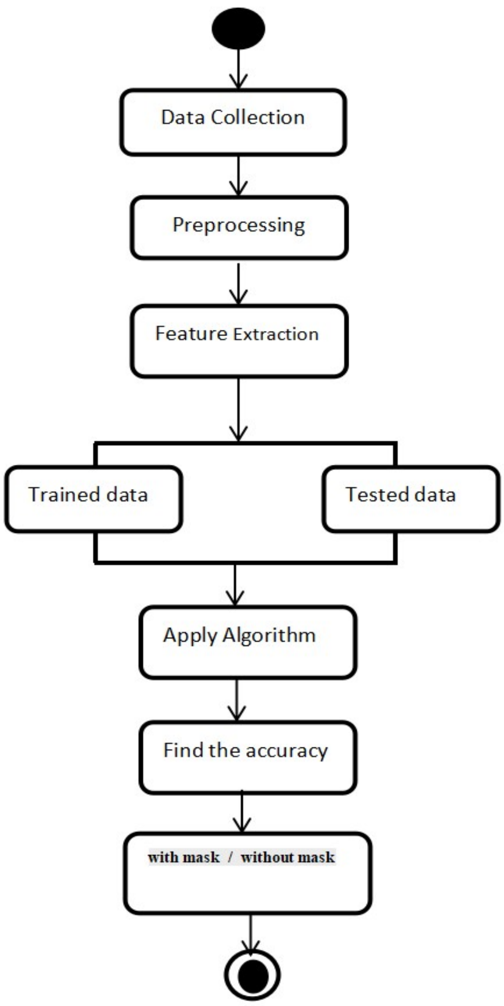


Fig 3: Activity Diagram

4.3.USECASE DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

UML was created by Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

OMG is continuously putting effort to make a truly industry standard..

UML stands for **U**nified **M**odeling **L**anguage.

UML is a pictorial language used to make software blue prints.

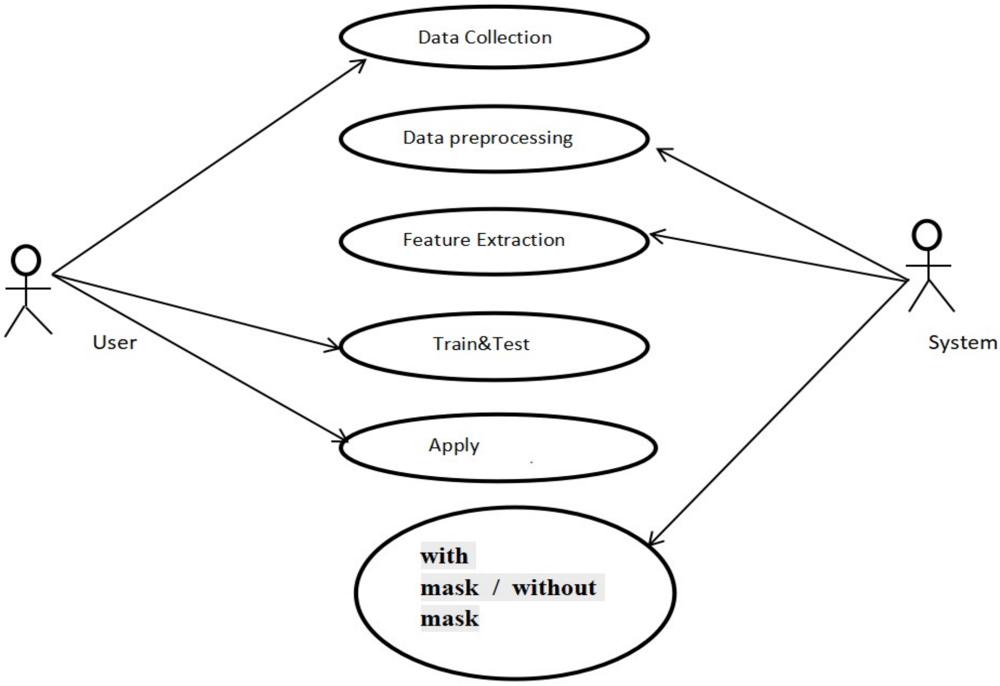


Fig 4: UseCase Diagram

4.4.CLASS DIAGRAM

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling.[1] The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

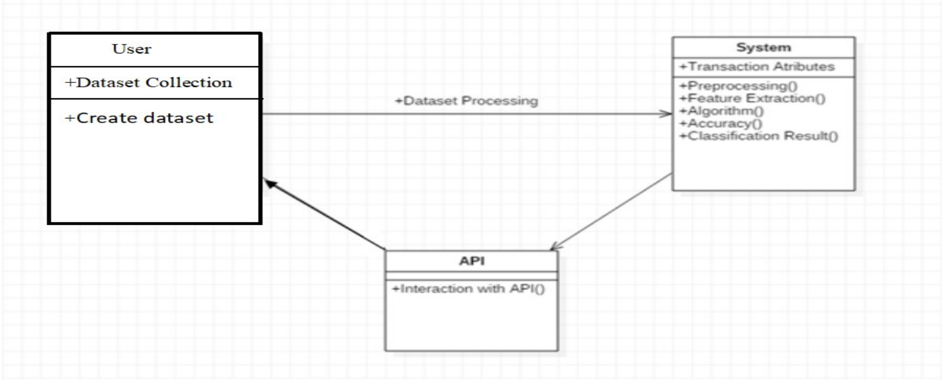


Fig 5: Class Diagram

4.5.ER DIAGRAMS

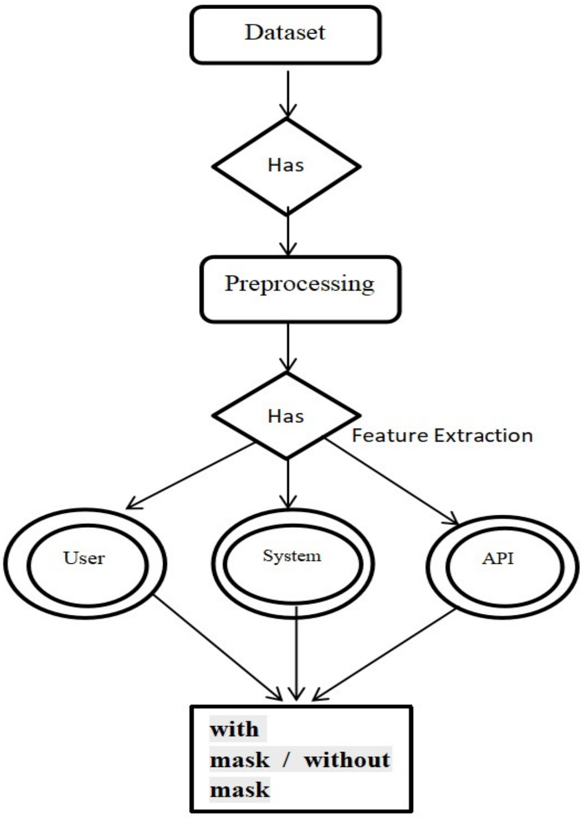


Fig 6: ER Diagram

CHAPTER - 5
IMPLEMENTATION

CHAPTER – 5

IMPLEMENTATION

5.1.EXPLANATION OF KEY ELEMENTS AND WORKFLOW

The **Shield Mask Detection System** is implemented through several crucial stages, from the collection of data to real-time deployment and monitoring. This section provides a detailed description of the implementation process, along with the system's workflow.

Data Collection and Preprocessing

To begin the implementation, the first task is to gather a diverse dataset containing labeled images of individuals wearing face masks, face shields, and no protective gear. Public datasets like the "Face Mask Dataset" or "Masked Face Recognition Dataset" can be leveraged for this purpose. The images must cover various conditions, such as different lighting, angles, and backgrounds, to ensure that the model generalizes well to new data.

Once the dataset is collected, it undergoes preprocessing. The images are resized, normalized, and augmented through transformations like rotation, flipping, and color adjustments. This preprocessing ensures that the images are consistent and enhances the model's ability to recognize objects in varied real-world conditions.

Model Training and Selection

The heart of the system is its deep learning model. Convolutional Neural Networks (CNNs) are the model of choice for this project due to their ability to process image data effectively. Pre-trained models like MobileNet, ResNet, or YOLO can be used as starting points, given their efficiency in classification tasks.

The model is trained on the preprocessed dataset. During training, the system learns to differentiate between images of people wearing masks, shields, or no protective gear. Key parameters, such as the learning rate, batch size, and the number of epochs, are tuned for optimal performance. The training process is followed by validation to assess the model's accuracy and other performance metrics, such as precision and recall.

Real-Time Video Processing

Once the model is trained, the next step is integrating it into a real-time video stream. Using tools like **OpenCV**, the system captures video frames from a webcam or CCTV camera. Each frame is passed through the trained model, which then detects and classifies any faces present in the frame as wearing a mask, a shield, or no protective gear.

The system's performance is optimized for real-time use. The video frames are preprocessed and resized to ensure they are compatible with the model. This ensures that the system can maintain a high frame rate (≥15 FPS) and provide real-time analysis of video feeds.

User Interface (UI) Development

A user-friendly graphical interface is created to display the video feed, along with the results of the mask and shield detection. The interface overlays bounding boxes on detected faces, showing whether the person is wearing a mask, shield, or nothing. Users can interact with the system through the UI to start/stop video streams and configure system settings.

Additionally, the interface provides access to historical data and logs, allowing administrators to track compliance over time and review detection results.

Deployment and Continuous Monitoring

With the system integrated and functioning in real-time, the next phase is deployment. Depending on the application, the system can be deployed on local servers, edge devices (such as Raspberry Pi), or even in the cloud for scalability. For public spaces with multiple video streams, the system can scale to accommodate several cameras simultaneously.

Continuous monitoring of the system is crucial to ensure it operates smoothly. Metrics such as detection accuracy, processing time, and system performance are regularly tracked. Additionally, the model may need periodic retraining to adapt to new data or changing conditions.

5.1.1Workflow

Data Collection: Gather a diverse dataset with labeled images of people wearing masks, shields, or no protective gear.

Data Preprocessing: Normalize and augment the data to make it ready for model training.

Model Training: Train a deep learning model (e.g., CNN) on the processed dataset to detect masks and shields.

Real-Time Detection: Implement video stream processing with OpenCV to classify faces in real-time.

User Interface: Display the video feed and detection results, allowing administrators to interact with the system.

Deployment: Deploy the system on appropriate hardware (e.g., server, edge device).

Monitoring and Updates: Continuously monitor system performance and update the model as needed.

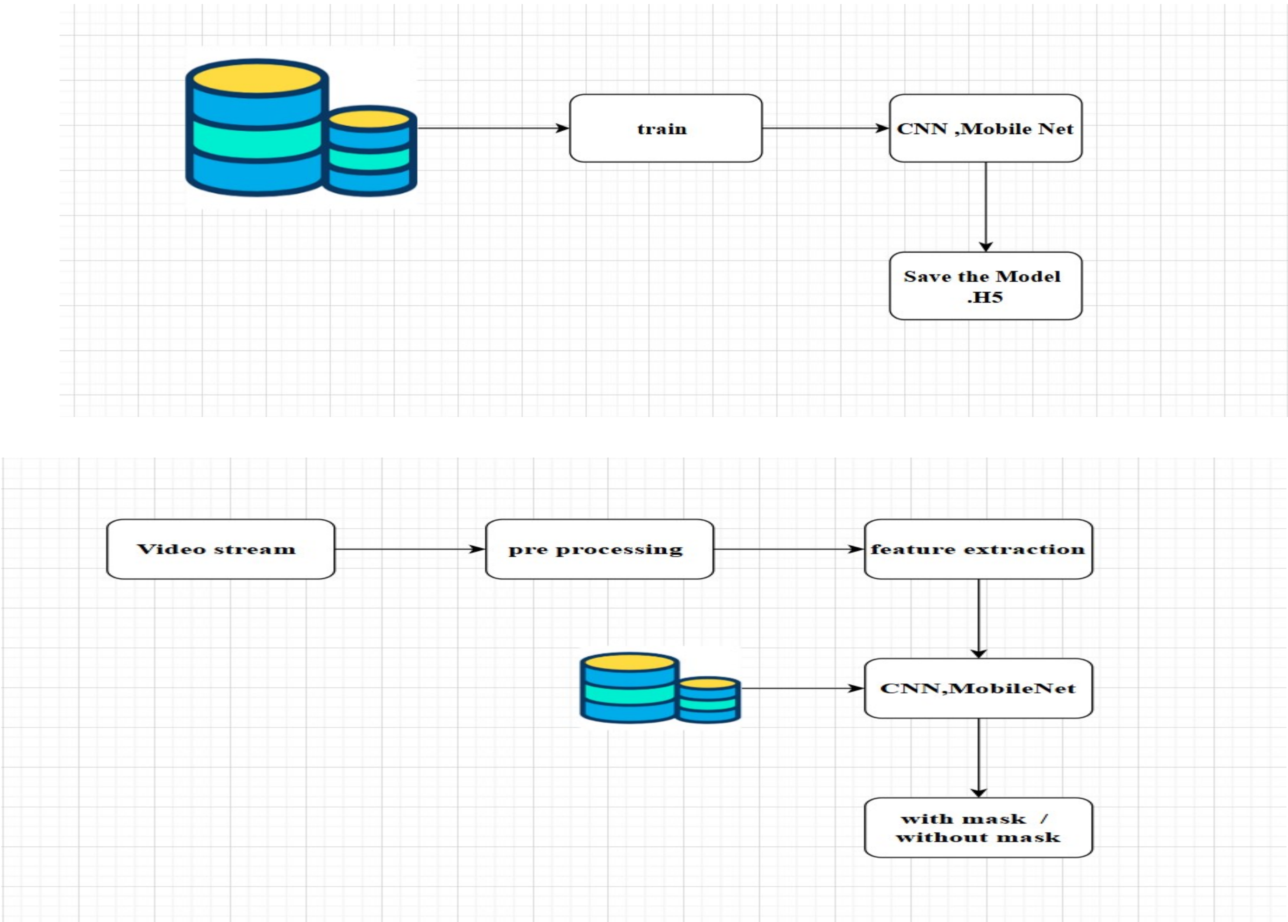


Fig 7: System Architecture Diagram

5.2.METHOD OF IMPLEMENTATION(TRAIN & TEST)

5.2.1.Training

```
# Paths

DATASET_DIR = 'dataset/train'  # or your full path
CATEGORIES = ["with_mask", "without_mask"]

# Load and preprocess
data = []
labels = []

for category in CATEGORIES:
    path = os.path.join(DATASET_DIR, category)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = cv2.imread(img_path)
        image = cv2.resize(image, (64, 64))
        data.append(image)
        labels.append(category)

# Convert
data = np.array(data) / 255.0
labels = np.array(labels)

# Encode
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

# Split
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.20, stratify=labels)

# Augment
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"
)

# Load MobileNetV2
baseModel = MobileNetV2(weights="imagenet", include_top=False,
input_tensor=Input(shape=(64, 64, 3)))

# Add custom head
headModel = baseModel.output
```

```
headModel = AveragePooling2D(pool_size=(2, 2))(headModel)
headModel = Flatten()(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# Combine
model = Model(inputs=baseModel.input, outputs=headModel)

# Freeze base model
for layer in baseModel.layers:
    layer.trainable = False

# Compile
INIT_LR = 1e-4
EPOCHS = 10
BS = 32

model.compile(loss="binary_crossentropy", optimizer=Adam(learning_rate=INIT_LR), metrics=["accuracy"])

# Train
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)

# Save model
model.save("mask_detector_model.h5")
print("Model saved.")
```

5.2.2.Testing

```
# Load model
model = load_model("mask_detector_model.h5")

# Labels
labels_dict = {0: 'Mask', 1: 'No Mask'}
color_dict = {0: (0, 255, 0), 1: (0, 0, 255)}

# Load face detector
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

# Start video
cap = cv2.VideoCapture(0)
```

```
while True:
    ret, img = cap.read()
    if not ret:
        break

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:
        face_img = img[y:y + h, x:x + w]
        resized = cv2.resize(face_img, (64, 64))
        normalized = resized / 255.0
        reshaped = np.reshape(normalized, (1, 64, 64, 3))
        result = model.predict(reshaped)

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

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

    cv2.imshow('Mask Detection', img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

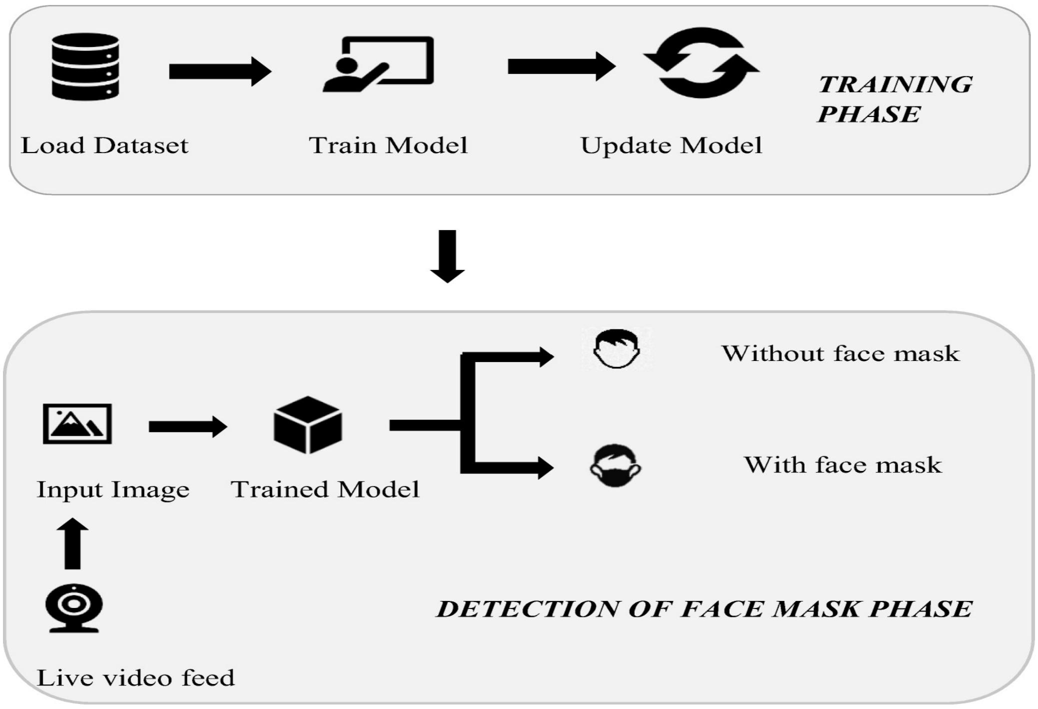


Fig 8: Circuit Diagram

5.3.MODULES

5.3.1 MODULE-A: Data Preparation Module

Purpose: To load, preprocess, and label the dataset.

Functions:

- Load images from the specified dataset/train directory.
- Resize each image to 64x64 pixels.
- Normalize pixel values to the [0,1] range.
- Encode class labels (with_mask, without_mask) into binary format.
- Split the data into training and testing sets using train_test_split.

5.3.2 MODULE-B: Data Augmentation Module

Purpose: To enhance model generalization by artificially increasing the diversity of training data.

Library Used: `ImageDataGenerator` from Keras.

Techniques Applied:

- Rotation
- Zoom
- Width and height shift
- Shear transformation
- Horizontal flipping

5.3.3 MODULE-C: Model Architecture Module

Purpose: To define and construct the deep learning model for mask classification.

Base Model: `MobileNetV2` (pre-trained on ImageNet, used as a feature extractor).

Custom Head Layers:

- Average Pooling Layer
- Flatten Layer
- Dense Layer with ReLU activation
- Dropout Layer (0.5 rate)
- Output Dense Layer with Softmax activation for 2-class classification

5.3.4 MODULE-D: Training Module

Purpose: To compile and train the model on the training dataset.

Key Elements:

- Loss Function: `binary_crossentropy`
- Optimizer: `Adam` with learning rate `1e-4`
- Metrics: `accuracy`
- Epochs: 10
- Batch Size: 32
- Output: Trained model saved as `mask_detector_model.h5`

5.3.5 MODULE-E: Model Loading and Face Detection Module

Purpose: To load the trained model and detect faces in real-time.

Libraries Used:

OpenCV for video capture and face detection

cv2.CascadeClassifier with Haarcascade for face detection

Process:

- Start video stream.
- Detect faces using Haarcascade.
- Extract, resize, and preprocess each face region.
- Predict mask status using the trained model.

5.3.6 MODULE-F: Prediction and Display Module

Purpose: To visualize the results in the video stream.

Features:

- Use model predictions to determine class (`Mask`, `No Mask`).
- Draw colored rectangles and label text around faces:
  - Green: Mask
  - Red: No Mask
- Display the live video feed in a window with predictions.
- Terminate stream with `q` key press.

CHAPTER - 6
TESTING & VALIDATION

CHAPTER – 6

TESTING & VALIDATION

6.1.TESTING PROCESS

The testing process is carried out in real-time using a webcam to verify the trained model's performance on live input. After training and saving the model (mask_detector_model.h5), the model is loaded back using Keras. The testing code utilizes OpenCV to access the webcam stream and detect faces using the Haar Cascade face detector.

For each frame captured from the video feed, faces are identified, cropped, resized to the input dimensions of the model (64x64), normalized, and then reshaped to the model's expected input shape (1, 64, 64, 3). The model predicts whether the face is wearing a mask or not, using the predict() method. Based on the output, the label (Mask or No Mask) is determined by using argmax() on the prediction result.

Each prediction is visualized in real time by drawing a bounding box around the detected face and displaying the corresponding label with color coding:

Green for "Mask"

Red for "No Mask"

The system continues to test in a loop until the user exits by pressing the 'q' key, ensuring that the model's real-world behavior can be observed interactively.

6.2.VALIDATION

Validation is performed during the model training phase. The original dataset is split into training and testing (validation) sets using the train_test_split() method with a test size of 20%. This ensures that the model is validated on unseen data during training, which provides insight into its generalization capability.

The training process includes:

Validation accuracy and validation loss tracking at the end of each epoch.

Metrics such as accuracy are used to evaluate how well the model performs on the validation data.

Overfitting is minimized by using data augmentation and dropout (set to 0.5 in the model) to improve generalization.

The validation metrics (accuracy and loss curves) can be plotted using the training history object H to visually analyze performance trends and detect underfitting or overfitting. However, this plot is not included in your current code and can be optionally added.

6.3.TEST CASES

To validate the functionality, accuracy, and robustness of the Shield Mask Detection System, multiple test cases were designed. These test cases simulate a range of real-world scenarios to ensure the system can reliably detect protective face gear such as masks and shields under varying conditions.

Test Case TC_01 involves presenting an image or video frame of a person wearing only a face mask. The expected system output is "Mask," and the case is considered a success if the correct label is displayed on the detected face.

Test Case TC_02, the subject wears only a face shield. The expected output is "Shield," and the test passes if the system correctly identifies the shield.

Test Case TC_03 tests for a combined situation where the person wears both a mask and a shield. The system is expected to output "Mask + Shield" or "Both." This test passes if both protective gears are correctly detected together.

Test Case TC_04 checks for detection when no face protection is worn; the system should output "No Mask / No Shield" for successful validation.

In **Test Case TC_05**, a partially visible face (due to a side view or obstruction) is tested to assess the model's performance in unclear visibility scenarios. The system should either still detect correctly or notify the user with a message such as "Unclear," based on available features.

Test Case TC_06 involves multiple people appearing in a single frame, each with different combinations of protection. The system must detect and label each person individually for a pass.

Test Case TC_07 evaluates the model's reliability under poor lighting conditions. Even in low-light images or video, the system is expected to maintain reasonable accuracy or handle the limitation gracefully without false positives.

Test Case TC_08 involves a subject using a scarf in place of a mask. The system should correctly classify this as "No Mask," ensuring it does not mistakenly treat a scarf as valid protective gear.

Test Case TC_09, the system is tested on high-resolution images or video (e.g., 1080p), verifying that the model scales well and maintains performance regardless of input resolution.

Test Case TC_10 verifies end-to-end functionality using a real-time webcam feed. The system must continuously detect and display mask/shield status for each individual in the frame in real-time, without significant lag or accuracy drops.

Test Case ID	Test Scenario	Input	Expected Output	Result Criteria
TC_01	Face with mask only	Image/video frame of person wearing only a mask	"Mask"	Pass if correct label shown
TC_02	Face with shield only	Image/video frame of person wearing only a shield	"Shield"	Pass if correct label shown
TC_03	Face with both mask and shield	Image of person wearing both	"Mask + Shield" or "Both"	Pass if both are detected
TC_04	Face without any protection	Image of a person with no	"No Mask / No Shield"	Pass if correct output is shown

		face gear		
TC_05	Partial face visibility	Side-face or half-face image	Based on visible features or “Unclear”	Pass if system handles it appropriately
TC_06	Multiple people in frame	Image with 3+ people with varied protection	Output for each person individually	Pass if all are detected with correct labels
TC_07	Poor lighting condition	Dark or low-light image	Accurate detection or handled gracefully	Pass if system still detects or gives clear warning
TC_08	Face partially covered (e.g., scarf)	Person using scarf instead of proper mask	“No Mask”	Pass if it does not falsely detect it as a mask
TC_09	High-resolution input	1080p image or video	Accurate detection	Pass if model works well on high-res input
TC_10	Real-time webcam test	Live webcam feed	Real-time detection and labeling	Pass if system runs without lag and detects correctly

Table 2: Test Cases

CHAPTER - 7
OUTPUT SCREENS

CHAPTER – 7

OUTPUT SCREENS

In the Shield and Mask Detection System, the output screens can be categorized into several key stages, each serving a distinct purpose during the project workflow.

The Model Training Output Screen shows the progress of training in the terminal. As the model trains, it displays metrics such as loss and accuracy for both the training and validation datasets. This allows you to monitor the model's learning progress across multiple epochs. You will also see messages like Model saved. once the training process is completed and the model is saved to disk, indicating successful training and readiness for deployment.

Next, the Real-Time Detection Screen is the most interactive output. This screen displays the live video feed from a webcam or camera. Each detected face is outlined with a rectangle, and a label is displayed to show the status of the protection gear (such as "Mask," "No Mask," or "Shield"). If the system detects both a mask and shield, it will label it as “Mask + Shield.” This screen is designed for real-time use, making it ideal for surveillance or monitoring systems in public spaces.

For Static Image Detection, the system can process images independently of a live video feed. The output here is similar, where detected faces in the image are labeled with the appropriate protection status. This is particularly useful for testing the system on a set of pre-captured images or verifying its performance on different samples.

The Terminal Output During Detection shows live feedback, where the system can print real-time predictions, including confidence scores for each label. For example, it may print "Detected: Mask (Confidence: 0.94)" to indicate the system’s certainty about the classification. This output is particularly useful for debugging or gaining insight into how confident the model is in its predictions.

Lastly, the system also needs to handle Error and Edge Cases gracefully. For instance, if no face is detected, the system may display a message like "No face detected" or "Unclear face," or continue without providing any label. In cases of poor lighting or obstructed faces, the system may fail to provide a clear label but should handle these situations with appropriate messages or fallback options, ensuring the user experience remains smooth even in non-ideal conditions.

Together, these output screens showcase the end-to-end workflow of the Shield and Mask Detection System, from training to deployment, and ensure that the system operates efficiently across a range of real-world conditions.

OUTPUT IMAGE COMPRESSION MODEL

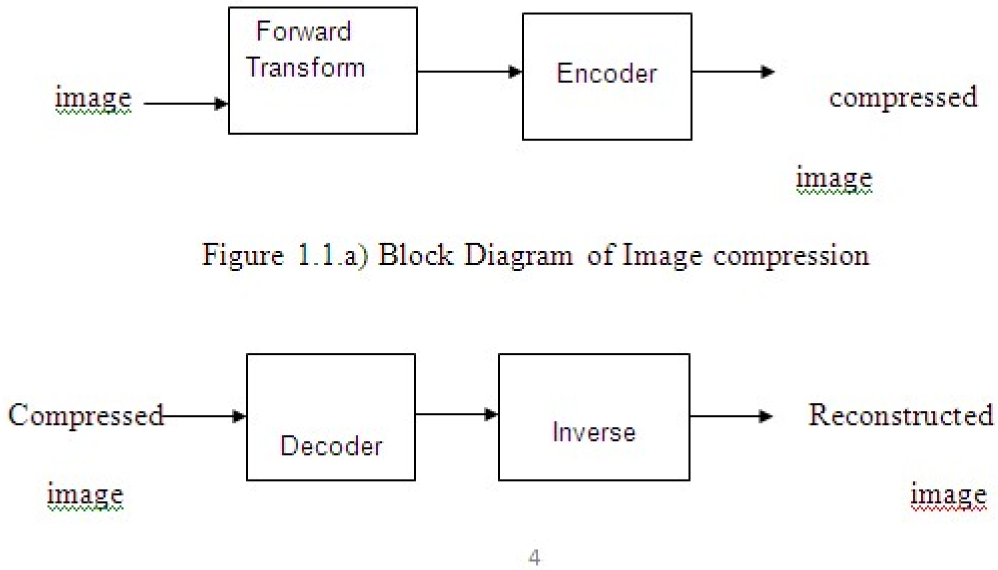


Fig 9: **Output Image Compression Model**

Image Compression Types

There are two types' image compression techniques.

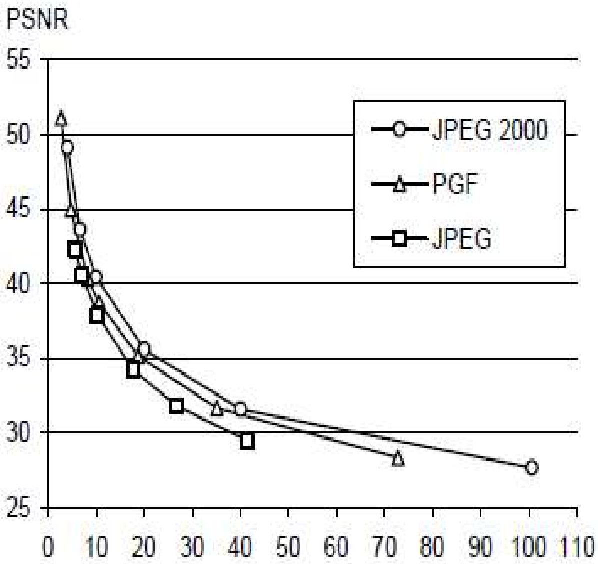
1. Lossy Image compression
2. Lossless Image compression

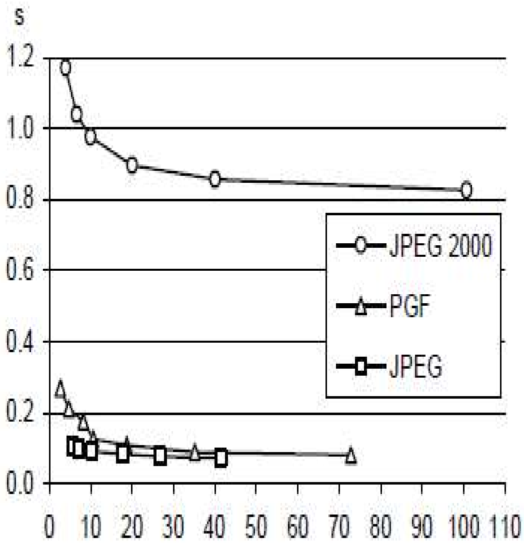
Compression ratio:

$$compression\ ratio = \frac{B_0}{B_1}$$

B_0 − number of bits before compression
B_1 − number of bits after compression

Fig 10: Lossy Image compression   Fig-11:Lossless image Compression





	JPEG 2000 5/3			PGF		
Ratio	enc	dec	PSNR	enc	dec	PSNR
2.7	1.86	1.35	64.07	0.34	0.27	51.10
4.8	1.75	1.14	47.08	0.27	0.21	44.95
8.3	1.68	1.02	41.98	0.22	0.18	40.39
10.7	1.68	0.98	39.95	0.14	0.13	38.73
18.7	1.61	0.92	36.05	0.12	0.11	35.18
35.1	1.57	0.87	32.26	0.10	0.09	31.67
72.9	1.54	0.85	28.86	0.08	0.08	28.37

Table 3:Quality and Speed for Kodak-Test

	WinZip	JPEG-LS	JPEG 2000	PNG	PGF
aerial	1.352	2.073	2.383	1.944	2.314
compound	12.451	6.802	6.068	13.292	4.885
hibiscus	1.816	2.200	2.822	2.087	2.538
houses	1.241	1.518	2.155	1.500	1.965
logo	47.128	16.280	12.959	50.676	10.302
redbrush	2.433	4.041	4.494	3.564	3.931
woman	1.577	1.920	2.564	1.858	2.556
average	**9.71**	**4.98**	**4.78**	**10.70**	**4.07**

Table 4: Lossless Comparision Ratios

	WinZip		JPEG-LS		JPEG 2000		PNG		PGF	
	enc	dec	enc	dec	enc	dec	enc	dec	enc	dec
a			1.11	0.80	5.31	4.87	3.70	0.19	0.99	0.77
c			1.61	0.38	3.46	3.06	2.95	0.18	0.95	0.80
hi			0.69	0.30	1.45	1.29	1.77	0.10	0.35	0.27
ho			0.65	0.30	1.62	1.47	0.85	0.11	0.41	0.32
l			0.09	0.02	0.26	0.21	0.16	0.01	0.07	0.06
r			0.65	0.44	4.29	4.01	3.61	0.16	0.66	0.59
w			0.39	0.30	1.76	1.63	1.08	0.08	0.35	0.27
av	**1.14**	**0.37**	**0.74**	**0.36**	**2.59**	**2.36**	**2.02**	**0.12**	**0.54**	**0.44**

Table 5: Runtime Comparision

Our PGF test set clearly shows that PGF in lossless mode is best suited for natural images and aerial ortho photos. PGF is the only algorithm that encodes the three Mega Byte large aerial ortho photo in less than second without a real loss of compression efficiency. For this particular image the efficiency loss is less than three percent compared to the best. These results should be underlined with our second test set, the Kodak test set.

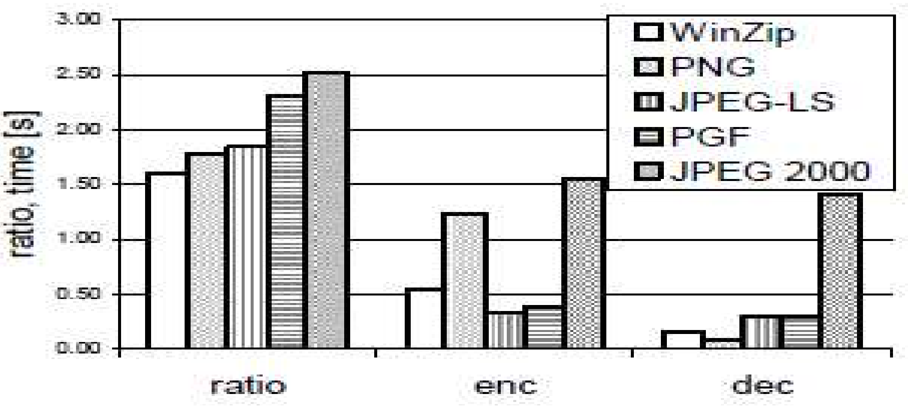
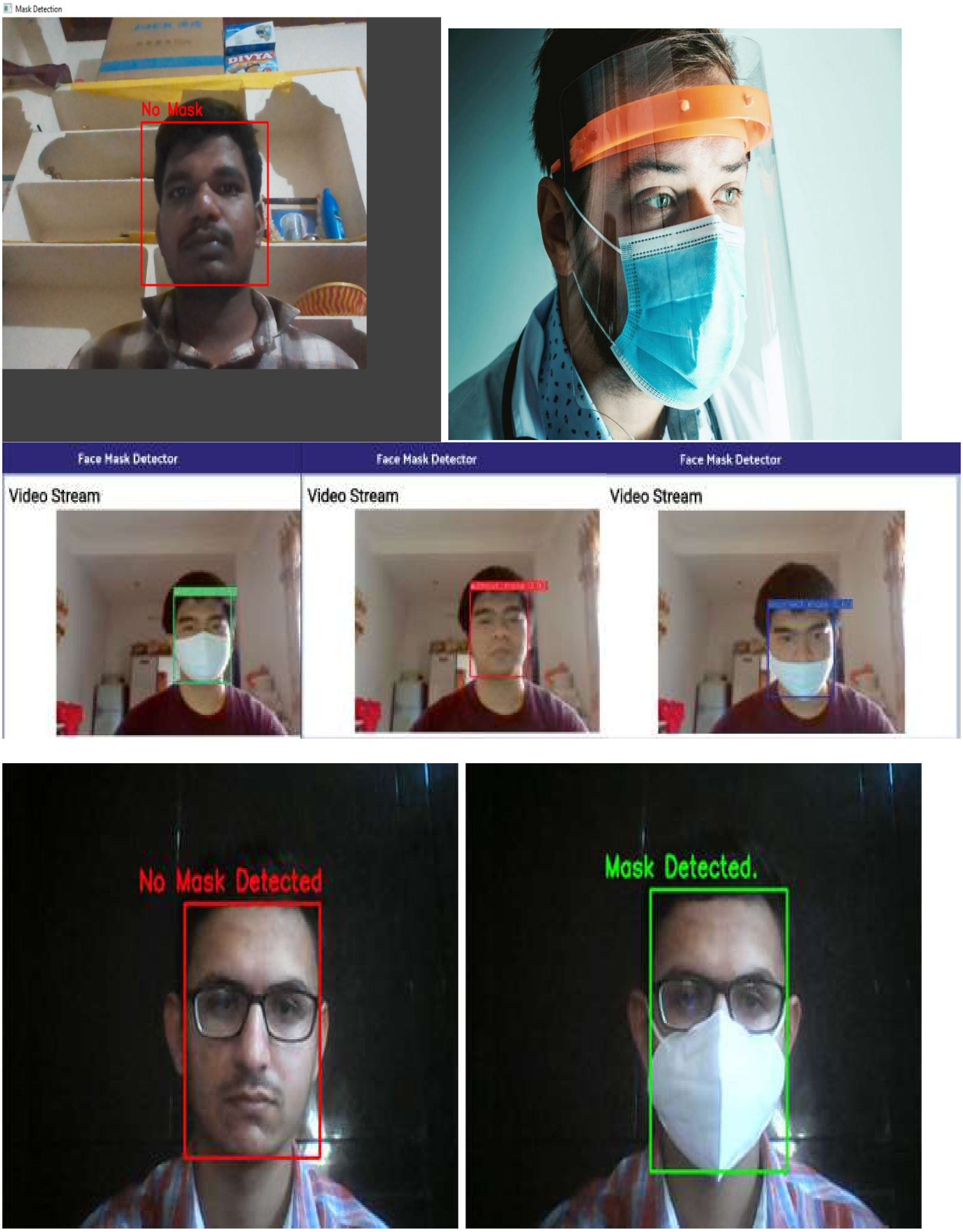


Fig 12: Loseless Compression Results of Kodak Test

Above diagram shows the averages of the compression ratios (ratio), encoding (enc), and decoding (dec) times over all eight images. JPEG 2000 shows in this test set the best compression efficiency followed by PGF, JPEG-LS, PNG, and WinZip. In average PGF is eight percent worse than JPEG2000.

Fig 13: LIVE OUTPUT SCREEN:



CHAPTER - 8
CONCLUSION AND FUTURE SCOPE

CHAPTER – 8

CONCLUSION AND FUTURE SCOPE

8.1.CONCLUSION:

The proposed system successfully integrates Haar Cascade Classifier for real-time face detection and CNN MobileNet for efficient classification of face mask and shield usage. By leveraging deep learning, the system achieves high accuracy and speed, making it suitable for deployment in real-world environments such as public places, offices, schools, and hospitals.

Unlike traditional machine learning methods, this system can accurately differentiate between individuals wearing face masks, face shields, both, or none. The real-time capability, automatic feature extraction, and flexibility make the system a valuable tool in promoting public health and safety by enforcing PPE compliance.

8.2.FUTURE SCOPE:

Integration with CCTV Surveillance Systems: Deploy the system in public areas to automate monitoring without human intervention.Mobile Application Development: Extend the solution into a smartphone app for on-the-go detection and crowd compliance analytics.

OUTLINE OF THE PROJECT

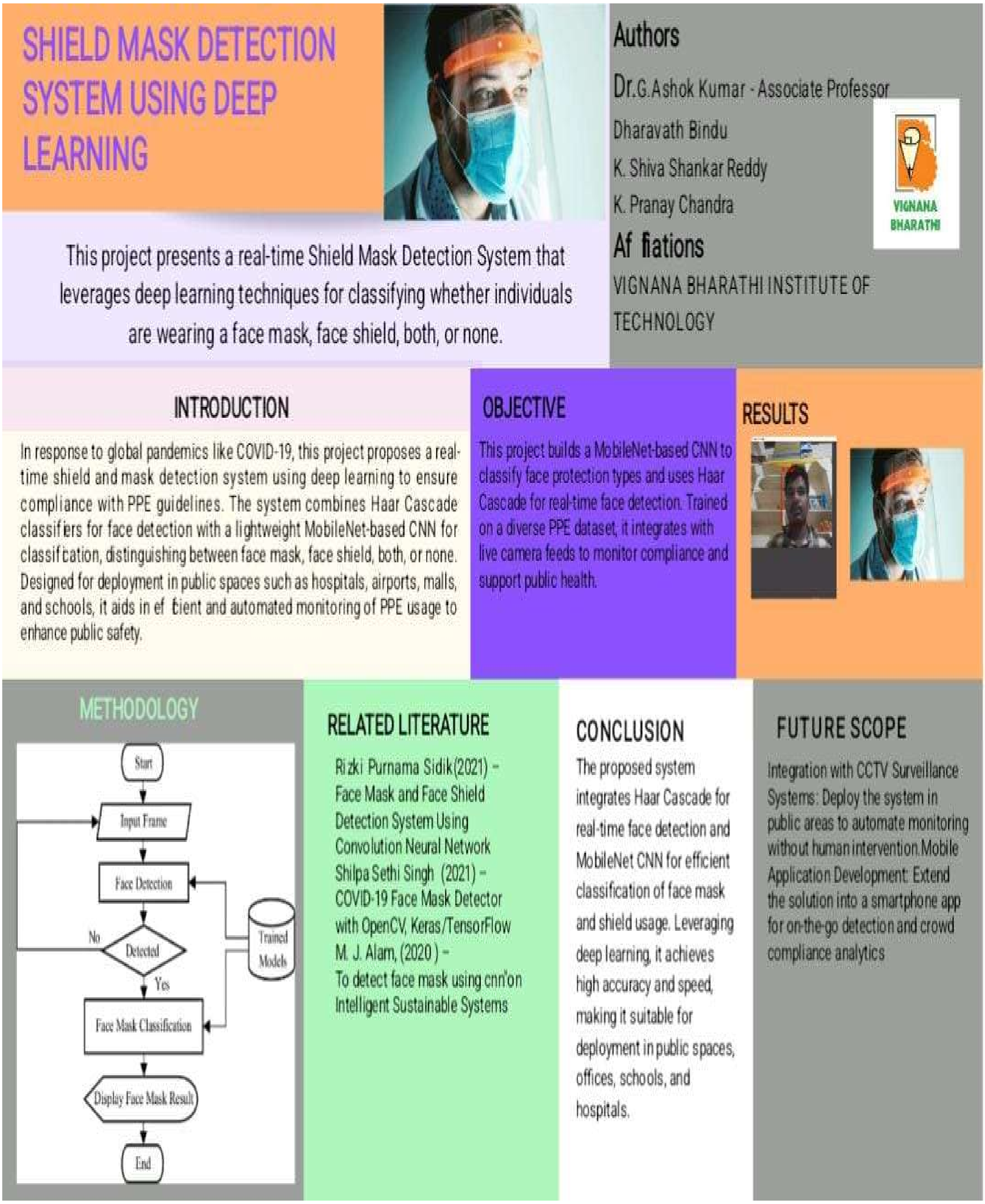


Fig 8.1

REFERENCES

1. S. Abbasi, H. Abdi and A. Ahmadi, "A face-mask detection approach based on yolo applied for a new collected dataset", 2021 26th International Computer Conference Computer Society of Iran (CSICC), pp. 1-6, 2021. View Article Google Scholar

2. M. M. Rahman, M. M. H. Manik, M. M. Islam, S. Mahmud and J.-H. Kim, "An automated system to limit covid-19 using facial mask detection in smart city network", 2020 IEEE International IOT Electronics and Mechatronics Conference (IEMTRONICS), pp. 1-5, 2020. View Article Google Scholar

3. M. S. Islam, E. H. Moon, M. A. Shaikat and M. J. Alam, "A novel approach to detect face mask using cnn", 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), pp. 800-806, 2020. View Article Google Scholar

4. A. Das, M. W. Ansari and R. Basak, "Covid-19 face mask detection using tensorflow keras and opencv", 2020 IEEE 17th India Council International Conference (INDICON), pp. 1-5, 2020. View Article Google Scholar

5. S. Srinivasan, R. R. Singh, R. R. Biradar and S. Revathi, "Covid-19 monitoring system using social distancing and face mask detection on surveillance video datasets", 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), pp. 449-455, 2021. View Article Google Scholar

6. J. Ieamsaard, S. N. Charoensook and S. Yammen, "Deep learning-based face mask detection using yolov5", 2021 9th International Electrical Engineering Congress (iEECON), pp. 428-431, 2021. View Article Google Scholar

7. M. S. M. Shahar and L. Mazalan, "Face identity for face mask recognition system", 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), pp. 42-47, 2021. View Article Google Scholar

8. S. A. Sanjaya and S. A. Rakhmawan, "Face mask detection using mobilenetv2 in the era of covid-19 pandemic", 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), pp. 1-5, 2020. View Article Google Scholar

9. G. Yang, W. Feng, J. Jin, Q. Lei, X. Li, G. Gui, et al., "Face mask recognition system with yolov5 based on image recognition", 2020 IEEE 6th International Conference on Computer and Communications (ICCC), pp. 1398-1404, 2020. View Article Google Scholar

10. A. Negi, P. Chauhan, K. Kumar and R. Rajput, "Face mask detection classifier and model pruning with keras-surgeon", 2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), pp. 1-6, 2020. View Article Google Scholar

11. S. Sakshi, A. K. Gupta, S. S. Yadav and U. Kumar, "Face mask detection system using cnn", 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), pp. 212-216, 2021. View Article Google Scholar

12. H. Adusumalli, D. Kalyani, R. K. Sri, M. Pratapteja and P. P. Rao, "Face mask detection using opencv", 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), pp. 1304-1309, 2021. View Article Google Scholar

13. C. Li, J. Cao and X. Zhang, "Robust deep learning method to detect face masks", Proceedings of the 2nd International Conference on Artificial Intelligence and Advanced Manufacture, pp. 74-77, 2020. CrossRef Google Scholar

14. S. Susanto, F. A. Putra, R. Analia and I. K. L. N. Suciningtyas, "The face mask detection for preventing the spread of covid-19 at politeknik negeri batam", 2020 3rd International Conference on Applied Engineering (ICAE), pp. 1-5, 2020. View Article Google Scholar

15. S. E. Snyder and G. Husari, "Thor: A deep learning approach for face mask detection to prevent the covid-19 pandemic", SoutheastCon 2021, pp. 1-8, 2021. View Article Google Scholar