FACE MASK DETECTION USING CNN

A Project report submitted in partial fulfillment of requirements for the award of degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
(AI & ML)

by

K. HARIKA (219X1A3313)

B. LEENA SRAVANTHI (219X1A3304)

C. RAJESH (219X1A3338)

Under the esteemed guidance of

Smt. O. ROOPA DEVI
Assistant Professor
Department of ECS.

Department of Emerging Technologies in Computer Science

G. PULLA REDDY ENGINEERING COLLEGE (Autonomous): KURNOOL

(Affiliated to JNTUA, ANANTAPURAMU)

2024 – 2025

Department of

Emerging Technologies in Computer Science

G. PULLA REDDY ENGINEERING COLLEGE (Autonomous):

KURNOOL

(Affiliated to JNTUA, ANANTAPURAMU)

CERTIFICATE

This is to certify that the Project Work entitled ‘‘FACEMASK DETECTION USING CNN’ **is a bonafide record of work carried out by**

K. HARIKA (219X1A3313)

B. LEENA SRAVANTHI (219X1A3304)

C. RAJESH (219X1A3338)

Under my guidance and supervision in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING (AI & ML)

Smt. O. Roopa Devi
Assistant Professor,
Department of ECS.,
G. Pulla Reddy Engineering College,
Kurnool.

Dr. R. Praveen Sam
Professor & Head of the Department,
Department of ECS.,
G. Pulla Reddy Engineering College,
Kurnool.

Signature of the External Examiner : .....

DECLARATION

We hereby declare that the project titled **"FACEMASK DETECTION USING CNN"** is an authentic work carried out by us as the students of **G. PULLA REDDY ENGINEERING COLLEGE (Autonomous) Kurnool,** during 2024-25 and has not been submitted elsewhere for the award of any degree or diploma in part or in full to any institute.

K. HARIKA
(219X1A3313)

B. LEENA SRAVANTHI
(219X1A3304)

C. RAJESH
(219X1A3338)

ACKNOWLEDGEMENT

We wish to express our deep sense of gratitude to our project guide **Smt. O. Roopa Devi,** Assistant Professor of Emerging Technologies in Computer Science Department, G. Pulla Reddy Engineering College, for her immaculate guidance, constant encouragement and cooperation which have made possible to bring out this project work.

We are grateful to our project in charge **Sri. U.U Veerendra,** Assistant Professor of Emerging Technologies in Computer Science Department, G. Pulla Reddy Engineering College, for helping us and giving us the required information needed for our project work.

We are thankful to our Head of the Department **Dr. R. Praveen Sam Garu**, for his whole hearted support and encouragement during the project sessions.

We are grateful to our respected Principal **Dr. B. Sreenivasa Reddy Garu**, for providing requisite facilities and helping us in providing such a good environment.

We wish to convey our acknowledgements to all the staff members of the Emerging Technologies in Computer Science department for giving the required information needed for our project work.

Finally, We wish to thank all our friends and well wishers who have helped us directly or indirectly during the course of this project work.

ABSTRACT

Face recognition is a key computer vision technology used in applications like passport control, smart access, voter ID, and public security. However, the COVID-19 pandemic posed challenges due to widespread mask usage, which obscures facial features and reduces traditional algorithm accuracy. To address this, enhanced systems were developed to detect and classify masked faces through three main stages: image pre-processing, face detection, and image classification. Using CCTV or webcam footage, these systems monitor mask compliance in real time. Powered by Convolutional Neural Networks (CNNs) and Caffe models, they ensure fast, accurate detection, improving public safety and health security.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

1. INTRODUCTION

1.1 Introduction:

Face mask detection using deep learning and computer vision offers a practical solution to this problem. By leveraging Convolutional Neural Networks (CNNs), which are highly effective in image classification and object detection tasks, we can build an intelligent system that can identify masked and unmasked faces in real-time from video feeds or images.

The COVID-19 pandemic has had a significant impact on public health and safety, making face masks a crucial preventive measure against the virus's spread. As mask-wearing became mandatory in many places, traditional face recognition systems began to face challenges in identifying individuals with partially covered faces. This created an urgent need for systems that can automatically detect whether a person is wearing a face mask or not, especially in crowded public areas like airports, hospitals, schools, offices, and transportation hubs.

The goal of this project is to develop a robust and accurate face mask detection system using CNN architecture. The system processes images through three main stages: image pre-processing, face detection, and classification. Once a face is detected, the CNN model classifies it into two categories: "Mask" or "No Mask".

This project has wide-ranging applications in enhancing public safety, automating surveillance systems, and ensuring compliance with health guidelines in both public and private spaces.

1.2 Motivation:

Public Health Importance

Face masks are a key measure to prevent the spread of COVID-19 and similar airborne diseases.

Enforcing mask-wearing in public spaces is critical to reduce transmission rates.

Manual Monitoring Limitations

Relying on humans to monitor mask usage in crowded places is inefficient and prone to errors.

It is not feasible to have security personnel constantly monitor large populations.

Role of Automation

Automation can help ensure consistent and real-time monitoring.

A face mask detection system reduces the need for manual intervention.

Advancement in Deep Learning

CNNs (Convolutional Neural Networks) provide highly accurate results in image classification tasks.

Deep learning enables the development of reliable, scalable, and real-time detection systems.

Real-World Applications

Useful in airports, schools, malls, offices, and other public areas.

Can be integrated into CCTV systems, smart doors, and access control systems.

Societal Impact

Helps enforce public safety measures.

Contributes to controlling the spread of infectious diseases.

Encourages responsible behavior in society**.**

1.3 Objective of the project:

The primary objective of this project is to design and develop an automated system that can accurately detect whether a person is wearing a face mask or not using Convolutional Neural Networks (CNN). The system should be capable of analyzing images or real-time video feeds from cameras to classify faces into two categories: **"Mask"** and **"No Mask"**.

Specific Objectives

Face Detection

Detect human faces in images or video streams using computer vision techniques.

Mask Classification

Classify detected faces as either wearing a mask or not using a CNN-based deep learning model.

Real-Time Monitoring

Enable the system to process live camera feeds for continuous monitoring in public or private areas.

High Accuracy and Speed

Achieve fast and reliable detection with high accuracy and minimal false results.

Easy Integration

Ensure the system can be easily integrated with existing CCTV surveillance, attendance systems, or access control mechanisms**.**

Public Safety Support

Assist authorities in monitoring mask compliance to enhance health and safety during pandemics or outbreaks.

SYSTEM SPECIFICATIONS

2. SYSTEM SPECIFICATIONS

2.1 Software specification:

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application.

These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

The system should be able to interface with the existing system

- The system should be accurate
- The system should be better than the existing system

2.2 Hardware specifications:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list, especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture.

The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored.

HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENTS

- RAM: 8GB
- Processor: Intel ICore
- Hard disk: 100 GB

SOFTWARE REQUIREMENTS

- Operating System: Windows, macOS, or Linux
- Code Editors: Google Colab
- Libraries: Numpy, Pandas, Scikitlearn, Matplotlib,TensorFlow/Keras,OpenCV
- Language: Python

LITERATURE SURVEY

3. LITERATURE SURVEY

3.1 Existing Systems:

Title: Face mask recognition system using CNN model

Author: Gagandeep Kaur, Ritesh Sinha, Puneet Kumar Tiwari

Abstract:

COVID-19 epidemic has swiftly disrupted our day-to-day lives affecting the international trade and movements. Wearing a face mask to protect one's face has become the new normal. In the near future, many public service providers will expect the clients to wear masks appropriately to partake of their services. Therefore, face mask detection has become a critical duty to aid worldwide civilization. This paper provides a simple way to achieve this objective utilising some fundamental Machine Learning tools as TensorFlow, Keras, OpenCV and Scikit-Learn. The suggested technique successfully recognises the face in the image or video and then determines whether or not it has a mask on it. As a surveillance job performer, it can also recognise a face together with a mask in motion as well as in a video. The technique attains excellent accuracy. We investigate optimal parameter values for the Convolutional Network model model (CNN) in order to identify the existence of masks accurately without generating over-fitting.

Methodology Used

We used python script, tensor flow, and CNN as deep learning architecture to develop an efficient network for recognising facemasks. Our objective is to train a specialised CNN model to detect whether or not someone is wearing a mask. This project can instantly recognise the faces of the mask from any angle. It generates output from an RGB input image of any orientation. The primary responsibility of this function is to extract characteristics from photographs and predict which class they belong to. The feature extraction approach sketches the image and transforms it into a new image, which is more efficient than the previous image. The dimensionality of photographs is reduced to an efficient representation in this section. In our recommended concept, the camera might be utilised to recognise the mask face. To begin, resize the input image to 100*100 then extract and forecast features. We are provided some model data with their accuracy level when the training phase is done.

The implementation of the project is carried out in python notebook. Libraries like pandas, NumPy, matplotlib, sklearn, etc. are used. To train the CNN model and to run the python code for the project the following libraries with the given or higher version is required.

Title: A Systematic Review On CNN For Face Mask Detection

Authors: Shakti Punj, Dr. Lavkush Sharma, Dr. Brajesh Kumar Singh

Overview

The COVID-19 pandemic has necessitated the development of face mask detection systems to enforce safety measures and monitor compliance. This abstract provides an overview of various datasets and approaches used for face mask detection using Convolutional Neural Networks (CNN). The datasets described include NO-MASK, UMD Faces, Annotated facial landmarks in the wild, UMD Masks, FDNA, and more. Researchers have created large-scale datasets such as NO-MASK, UMD Masks, and UMD Faces, which provide annotated images for training deep networks. These datasets have played a vital role in advancing face mask detection algorithms. Additionally, datasets like COVID-19 screening on chest X-ray images have been used for anomaly detection and screening purposes. Numerous deep learning approaches have been proposed for face mask detection. Techniques such as neural networks with augmentation, vision-based frameworks, deep learning-based anomaly are used

Methodology Used

1. Data Collection:

Face detection is the initial step in the process, aiming to locate and identify faces in an image or video frame. Haar cascades and the Viola-Jones algorithm are commonly used techniques for face detection. These methods utilize a set of pre-trained classifiers that are trained to identify specific facial features, such as the presence of eyes, nose, and mouth. By scanning the image or frame with these classifiers, potential face regions are identified.

2. Data Preprocessing:

Once the faces are detected, the next step is to extract meaningful features from the face regions. Various feature extraction techniques have been used in traditional computer vision approaches for facemask detection. Some common methods include:

a. Histogram of Oriented Gradients (HOG): HOG computes the distribution of gradient orientations within local image patches. It captures shape and edge information and has been widely used for object detection tasks, including face-related applications.

b. Local Binary Patterns (LBP): LBP encodes the relationship between pixels in a local neighbourhood. It describes texture information and has been employed for various pattern recognition tasks, including facial analysis.

c. Scale-Invariant Feature Transform (SIFT): SIFT detects and describes distinctive local features within an image. It is robust to changes in scale, rotation, and illumination and has been utilized for face recognition and feature matching tasks. These feature extraction techniques aim to capture relevant information from the face regions, which can later be used for classification.

3. Classification:

After feature extraction, a classification algorithm is employed to determine whether a person is wearing a mask or not. Commonly used classifiers in traditional computer vision approaches include Support Vector Machines (SVM), Random Forests, and other classical machine learning algorithms. These classifiers are trained on labeled datasets, where the extracted features are associated with corresponding labels (mask or no-mask). Once trained, the classifier can predict the presence or absence of a facemask based on the extracted features.

Advantages of traditional computer vision approaches for facemask detection include their interpretability and relatively low computational requirements. These methods can provide reasonable performance in scenarios with limited variations in lighting conditions and mask types. They have been extensively used in research and practical applications. Additionally, these approaches are based on well-established techniques and do not require large amounts of training data.

However, traditional computer vision approaches have limitations. They often struggle with handling occlusions, where the face is partially covered by objects or other people. They can also face challenges when dealing with variations in mask types, such as different colors, styles, or patterns. Complex backgrounds and variations in lighting conditions can further impact the performance of these methods. Additionally, feature engineering in traditional approaches can be a time-consuming and labor-intensive process, requiring domain knowledge

and expertise to design effective features.

As computer vision advances, modern deep learning approaches, such as convolutional neural networks (CNNs) and transfer learning, have gained popularity for facemask detection due to their ability to automatically learn discriminative features from data. These approaches often outperform traditional methods, especially in more challenging scenarios with diverse mask types and occlusions. Random Forest Classifier: Ensemble method using multiple decision trees and majority voting.

3.2 Disadvantages of existing systems:

Real-Time Performance and Latency

- Real-time deployment, especially on edge devices or low-resource systems, can be limited due to high computation needs.
- Complex deep learning models may cause latency, making them unsuitable for high-speed detection in dynamic environments.

Privacy Concerns

- Use of facial recognition in public settings raises privacy and ethical issues.
- Tracking individuals based on facial features can lead to surveillance concerns**.**

Generalization to New Mask Types or Styles

- Models trained on common mask types may fail to detect novel or designer masks.
- Transparent, patterned, or loose-fitting masks may not be accurately detected.

Susceptibility to Adversarial Attacks

- Deep learning models can be tricked by subtle perturbations or adversarial inputs, leading to incorrect classification.
- Attackers could intentionally modify appearance to bypass detection.

Dependence on High-Quality Datasets

- The effectiveness of models heavily relies on the quality, diversity, and size of training datasets. Many real-world scenarios (e.g., poor lighting, motion blur) are underrepresented.
- Biased datasets can lead to poor generalization across ethnicities, age groups, or different mask types**.**

Difficulty with Partial or Incorrect Mask Usage

- Detecting improperly worn masks (e.g., under the nose) remains challenging.
- Most models classify binary categories — with mask vs. without mask — and do not assess mask correctness**.**

Environmental Variability

- Models may struggle in uncontrolled environments (e.g., crowded places, night settings, variable lighting).
- Obstructions like sunglasses, hats, or hands covering the face can reduce accuracy.

3.3 Proposed System

1. Data Collection:

Dataset used: Face Mask Detection Dataset from Kaggle

Classes:

- with_mask
- without_mask

Number of samples:

- With mask: 3725 images
- Without mask: 3828 images

2. Feature Selection:

1. Images are resized to a uniform shape (commonly 128x128 or similar).
2. Images are converted to numpy arrays.
3. Normalization is applied (pixel values scaled between 0 and 1).
4. Labels:
    1 for images with mask
    0 for images without mask

3. Splitting Dataset:

- Used train_test_split() from sklearn to create training and testing sets.

4. Model Architecture

- Model built using **TensorFlow/Keras**
- Layers used:
    - Convolutional layers (Conv2D)
    - Pooling layers (MaxPooling2D)
    - Flattening
    - Dense layers with activation functions
    - Final output layer with **sigmoid activation** (binary classification)

5. Model Training

- Optimizer: Adam
- Loss function: Binary Crossentropy
- Metrics: Accuracy
- Epochs: typically set to 10–20 for demonstration

DESIGN

4. DESIGN

4.1 Introduction:

System design is the structured process of defining the architecture, components, interfaces, and data flow within a system to meet specified functional and non-functional requirements. It forms a blueprint that ensures efficient implementation and performance of the system. In the context of facemask detection, the system involves data preprocessing, model training, evaluation, and real-time prediction functionalities using deep learning frameworks and supporting libraries. The goal is to create a modular, scalable, and robust detection system capable of accurate classification between masked and unmasked faces.

1. Pandas

Pandas is a powerful open-source data manipulation and analysis library for Python. It provides efficient data structures like Series and DataFrame, which are essential for handling structured data. In this project, Pandas is used to:

- Load and explore dataset metadata (e.g., image labels, paths)
- Perform preliminary dataset statistics
- Handle label encoding
- Support preprocessing pipelines before training

Pandas enhances readability and performance by offering easy-to-use data handling functions, making it a crucial part of the preprocessing stage.

2. NumPy

NumPy (Numerical Python) is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices along with a wide collection of mathematical functions to operate on them. In this project, NumPy is used to:

- Convert images into array format
- Normalize pixel values to a range of 0–1

- Support batch processing and tensor transformation for model input

It ensures efficient memory and computational performance, especially during training.

3. Matplotlib

Matplotlib is a data visualization library used for plotting graphs and visualizing model metrics. In this facemask detection project, it is used to:

- Plot training and validation accuracy/loss curves
- Visualize predictions and classification outcomes
- Display sample images from the dataset

These visualizations help in better understanding model performance and debugging errors.

4. Scikit-learn

Scikit-learn is a versatile machine learning library built on top of NumPy, SciPy, and Matplotlib. It is used in this project for:

- Splitting the dataset into training and testing sets (train_test_split)
- Computing model performance metrics like accuracy score and confusion matrix
- Supporting data preprocessing and model evaluation steps

Scikit-learn plays an important role in preparing data and validating the model.

5. TensorFlow/Keras

TensorFlow, with its high-level API Keras, is used to build and train the deep learning model for facemask detection. Key roles include:

- Constructing the CNN architecture with layers such as Conv2D, MaxPooling2D, Flatten, Dense, and Output
- Compiling the model with loss functions (binary crossentropy), optimizers (Adam), and performance metrics (accuracy)

- Training and validating the model over a set number of epochs

TensorFlow provides GPU acceleration and scalability for efficient training and deployment.

6. Algorithm Used

The facemask detection project combines traditional computer vision techniques and deep learning. The key algorithmic components include:

- Convolutional Neural Networks (CNNs) for image classification
- Binary classification using sigmoid activation
- Image preprocessing via resizing, normalization, and label encoding
- Training with loss function optimization (Binary Crossentropy + Adam Optimizer)

Each module contributes to a pipeline that ensures the system accurately classifies whether a person is wearing a mask or not.

4.2 Flowchart of the Model:

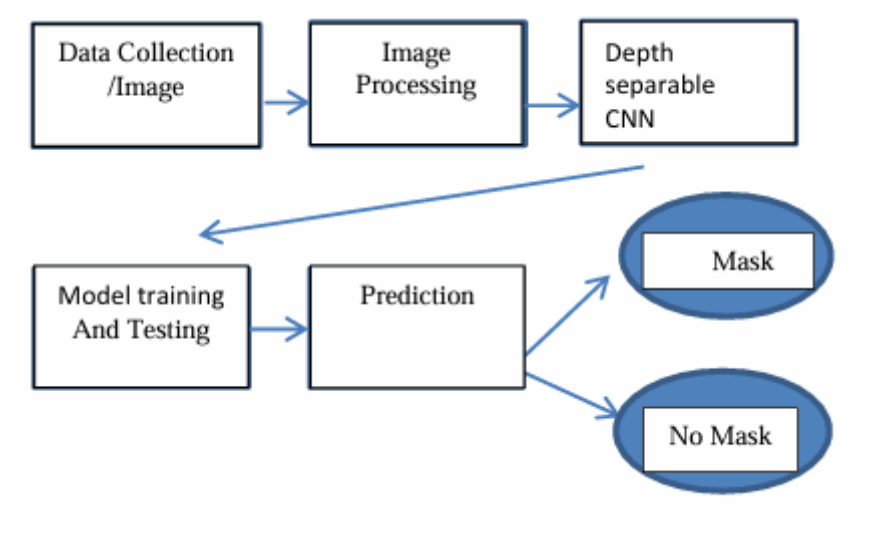

Figure 1: Flow chart

4.5 UML Diagrams:

Sequence Diagram:

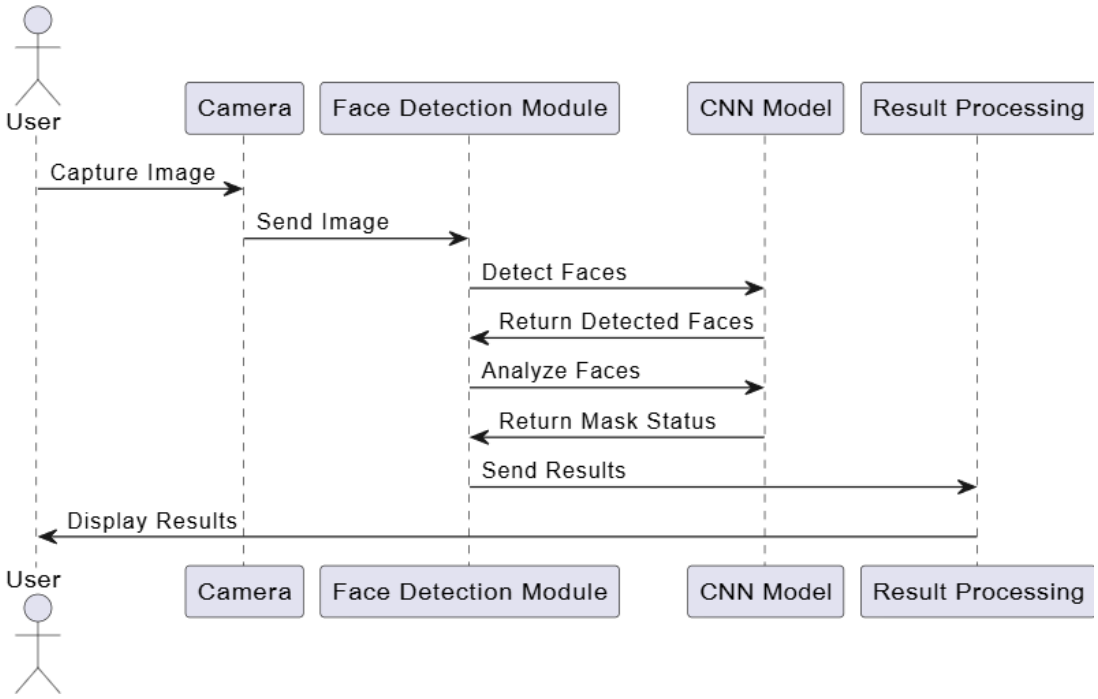

Figure 2: Sequence Diagram

Use Case Diagram:

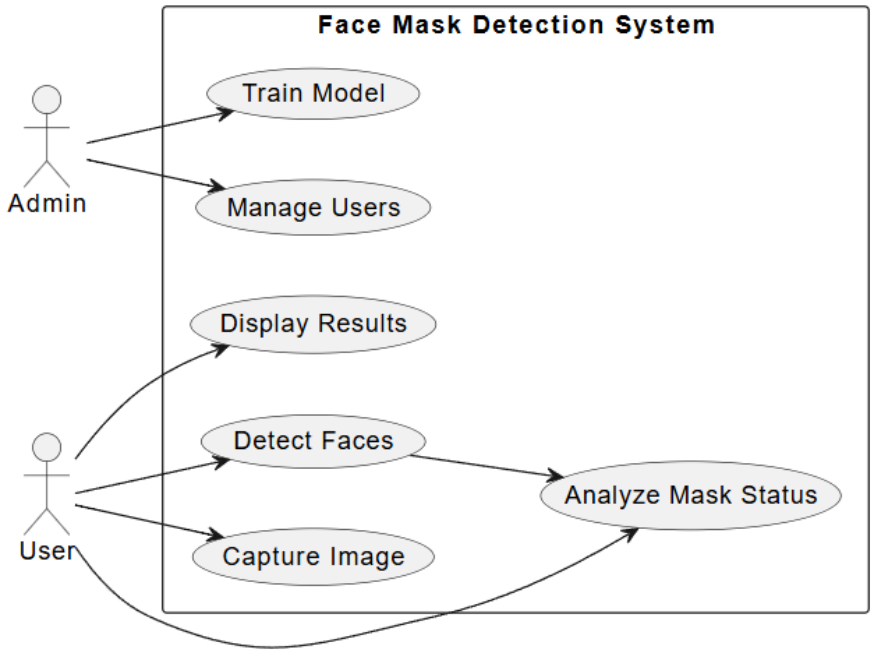

Figure 3: Use Case Diagram

Class Diagram:

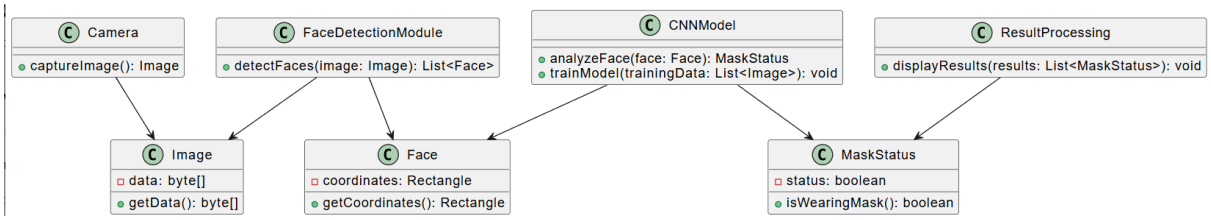

Figure 4: Class Diagr

Structure Diagram:

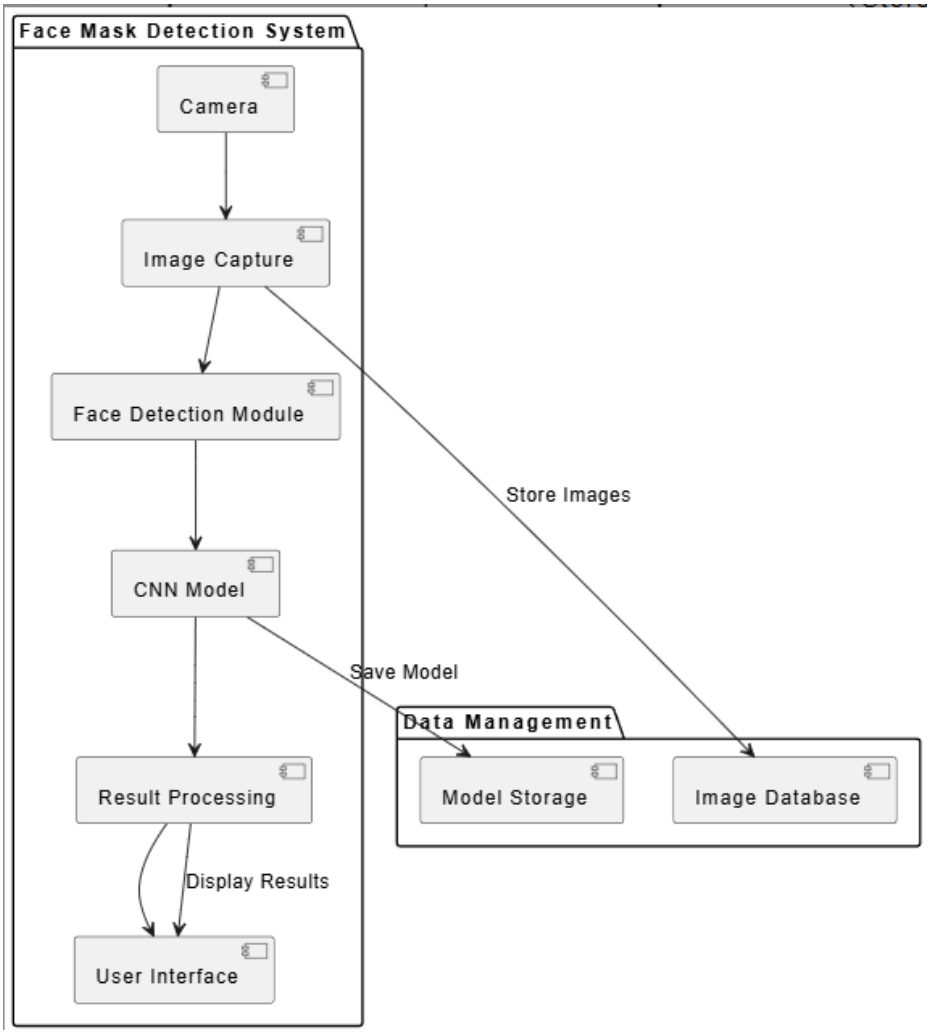

Figure 5: Structure Diagram

Activity Diagram:

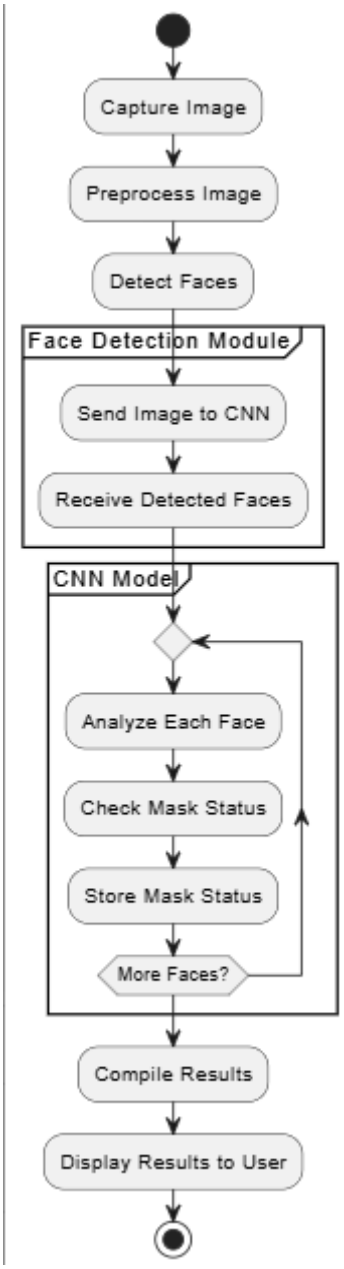

Figure 6: Activity Diagram

IMPLEMENTATION AND RESULT

5. IMPLEMENTATION AND RESULT

5.1 Introduction

This chapter covers the practical implementation of the face mask detection system. It outlines the dataset used, walks through the source code, provides detailed explanations, and includes visual outputs to demonstrate the effectiveness of the developed model.

Data Collection

For this project, the Face Mask Detection Dataset was sourced from Kaggle. This dataset is curated to help train models in recognizing whether individuals are wearing masks or not. It contains real-world images categorized into two distinct classes:

- With Mask: 3,725 images
- Without Mask: 3,828 images

The dataset is relatively balanced, which helps improve the model's generalization and reduces bias during training.

Data Preprocessing

Before feeding the data into the model, several preprocessing steps were applied:

- Resizing: All images were resized to a uniform shape (typically 128×128 pixels) to ensure consistency in input dimensions.
- Conversion: Images were converted into NumPy arrays for computational processing.
- Normalization: Pixel values were scaled between 0 and 1 by dividing by 255, improving training efficiency and convergence.
- Label Encoding: Each image was labeled as:

  1 for with mask

  0 for without mask

These steps help clean and structure the data, making it suitable for training deep learning

models.

Splitting Dataset

To evaluate the model effectively, the dataset was divided into two parts using train_test_split() from the sklearn library:

- Training Set: Used to train the model.
- Testing Set: Used to evaluate the model's performance on unseen data.

This split ensures that the model is validated on data it hasn't seen during training, reducing overfitting and improving reliability.

5.2 About Dataset

The project utilizes the Face Mask Detection Dataset, which consists of two categories:

WithMask:3,725images

WithoutMask:3,828images

The images are collected in real-world settings, providing a good basis for model training and evaluation. The dataset is balanced and suitable for binary classification tasks.

5.3 Source Code

The implementation is carried out using Python and libraries such as:

- TensorFlow / Keras – Model building and training
- OpenCV – Image processing
- sklearn – Dataset splitting and evaluation
- Matplotlib-Visualization

  The code includes steps for dataset extraction, preprocessing, model design, training, and performance visualization.

```
!pip install kaggle
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

```
!chmod 600 ~/.kaggle/kaggle.json

from zipfile import ZipFile
dataset = '/content/face-mask-dataset.zip'

with ZipFile(dataset,'r') as zip:
  zip.extractall()
  print('The dataset is extracted')

import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
from google.colab.patches import cv2_imshow
from PIL import Image
from sklearn.model_selection import train_test_split

with_mask_files = os.listdir('/content/data/with_mask')
print(with_mask_files[0:5])
print(with_mask_files[-5:])
without_mask_files = os.listdir('/content/data/without_mask')
print(without_mask_files[0:5])
print(without_mask_files[-5:])
print('Number of with mask images:', len(with_mask_files))
print('Number of without mask images:', len(without_mask_files))
with_mask_labels = [1]*3725
without_mask_labels = [0]*3828
print(with_mask_labels[0:5])
print(without_mask_labels[0:5])
print(len(with_mask_labels))
print(len(without_mask_labels))
labels = with_mask_labels + without_mask_labels
```

```
print(len(labels))
print(labels[0:5])
print(labels[-5:])

img = mpimg.imread('/content/data/with_mask/with_mask_3478.jpg')
imgplot = plt.imshow(img)
plt.show()

img = mpimg.imread('/content/data/without_mask/without_mask_2314.jpg')
imgplot = plt.imshow(img)
plt.show()
with_mask_path = '/content/data/with_mask/'

data = []

for img_file in with_mask_files:

  image = Image.open(with_mask_path + img_file)
  image = image.resize((128,128))
  image = image.convert('RGB')
  image = np.array(image)
  data.append(image)
without_mask_path = '/content/data/without_mask/'

for img_file in without_mask_files:
  image = Image.open(without_mask_path + img_file)
  image = image.resize((128,128))
  image = image.convert('RGB')
  image = np.array(image)
  data.append(image)

X = np.array(data)
Y = np.array(labels)
```

```
type(X)
type(Y)
print(X.shape)
print(Y.shape)
print(Y)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
print(X.shape, X_train.shape, X_test.shape)
X_train_scaled = X_train/255

X_test_scaled = X_test/255
X_train[0]
X_train_scaled[0]

import tensorflow as tf
from tensorflow import keras
num_of_classes = 2

model = keras.Sequential()

model.add(keras.layers.Conv2D(32, kernel_size=(3,3), activation='relu',
input_shape=(128,128,3)))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))

model.add(keras.layers.Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(128, activation='relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dropout(0.5))
```

```
model.add(keras.layers.Dense(num_of_classes, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['acc'])
history = model.fit(X_train_scaled, Y_train, validation_split=0.1, epochs=5)
loss, accuracy = model.evaluate(X_test_scaled, Y_test)
print('Test Accuracy =', accuracy)

h = history

# plot the loss value
plt.plot(h.history['loss'], label='train loss')
plt.plot(h.history['val_loss'], label='validation loss')
plt.legend()
plt.show()

# plot the accuracy value
plt.plot(h.history['acc'], label='train accuracy')
plt.plot(h.history['val_acc'], label='validation accuracy')
plt.legend()
plt.show()
input_image_path = input('Path of the image to be predicted: ')

input_image = cv2.imread(input_image_path)

cv2_imshow(input_image)

input_image_resized = cv2.resize(input_image, (128,128))

input_image_scaled = input_image_resized/255

input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])
```

```
input_prediction = model.predict(input_image_reshaped)

print(input_prediction)

input_pred_label = np.argmax(input_prediction)

print(input_pred_label)

if input_pred_label == 1:

  print('The person in the image is not wearing a mask')

else:
  print('The person in the image is wearing a mask')
```

5.4 Code Explanation

Step 1: Dataset Preparation

Images are loaded and resized. Labels are assigned (1 for mask, 0 for no mask), and the data is split into training and testing sets.

Step 2: Model Construction

A CNN is built with layers: Conv2D, MaxPooling2D, Flatten, Dense, and Dropout. Sigmoid activation is used in the final layer for binary classification.

Step 3: Compilation & Training

Model compiled with binary_crossentropy loss and Adam optimizer. It is trained over multiple epochs, with validation split.

Step 4: Evaluation & Plotting

Accuracy and loss are plotted. Model predictions are tested with new data and visualized.

5.5 Output Screens

```
# displaying with mask image
img = mpimg.imread('/content/data/with_mask/with_mask_1545.jpg')
imgplot = plt.imshow(img)
plt.show()
```

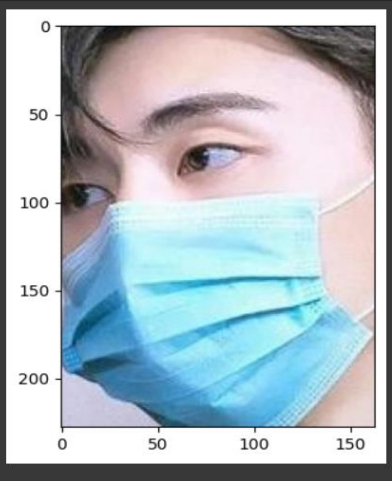

Figure 7: Dataset picture 1

```
# displaying without mask image
img = mpimg.imread('/content/data/without_mask/without_mask_2925.jpg')
imgplot = plt.imshow(img)
plt.show()
```

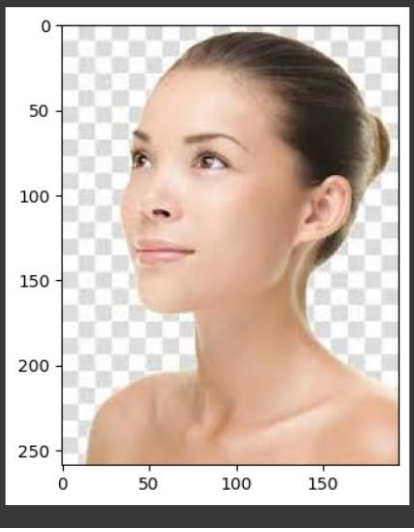

Figure 8: Dataset image 2

```
# training the neural network
history = model.fit(X_train_scaled, Y_train, validation_split=0.1, epochs=5)
```

```
Epoch 1/5
170/170 ━━━━━━━━━━━━━━━━━━━━ 136s 785ms/step - acc: 0.6811 - loss: 0.7606 - val_acc: 0.8314 - val_loss: 0.3713
Epoch 2/5
170/170 ━━━━━━━━━━━━━━━━━━━━ 133s 784ms/step - acc: 0.8731 - loss: 0.3377 - val_acc: 0.8793 - val_loss: 0.2798
Epoch 3/5
170/170 ━━━━━━━━━━━━━━━━━━━━ 140s 770ms/step - acc: 0.8968 - loss: 0.2566 - val_acc: 0.8909 - val_loss: 0.2515
Epoch 4/5
170/170 ━━━━━━━━━━━━━━━━━━━━ 141s 765ms/step - acc: 0.9117 - loss: 0.2201 - val_acc: 0.9124 - val_loss: 0.2080
Epoch 5/5
170/170 ━━━━━━━━━━━━━━━━━━━━ 141s 762ms/step - acc: 0.9299 - loss: 0.1805 - val_acc: 0.9091 - val_loss: 0.2138
```

Figure 9: Model training

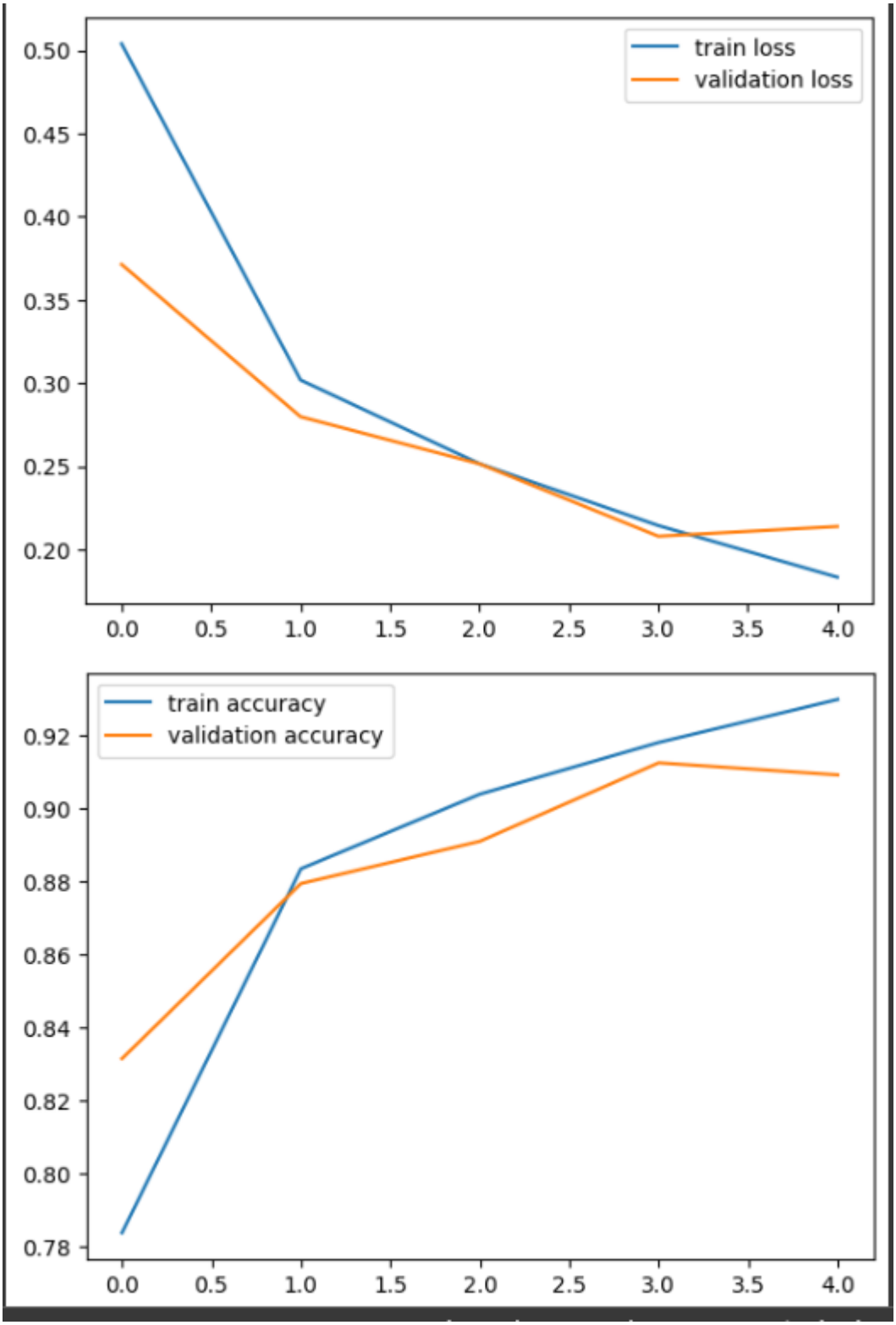

Figure 10:Accuracy and loss

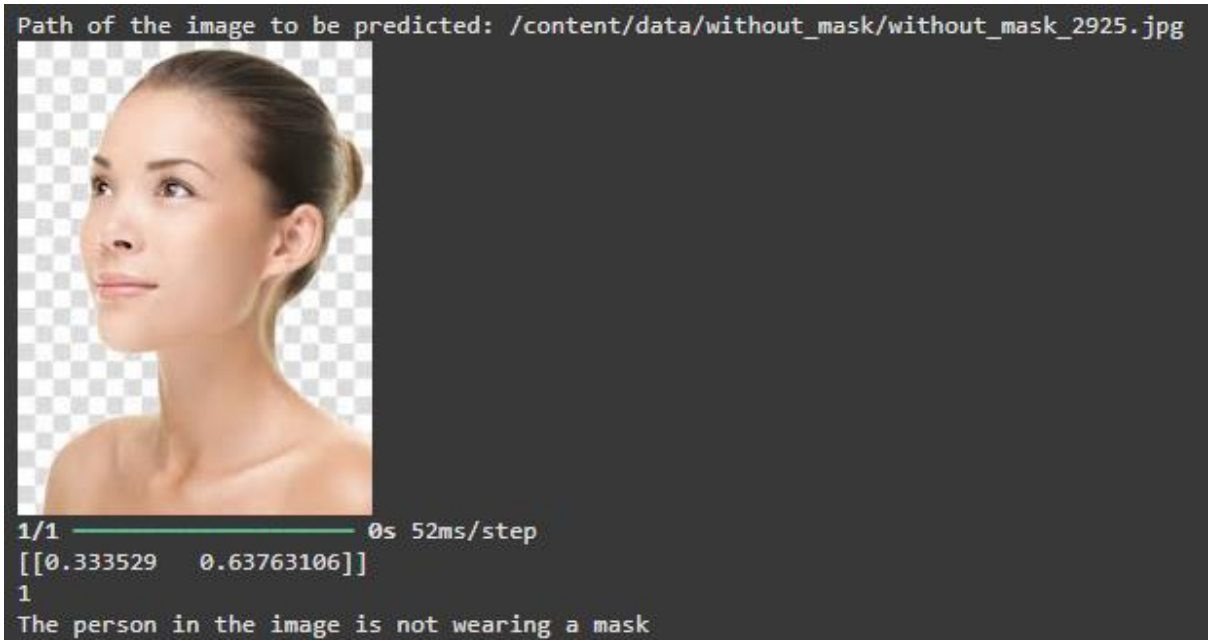

Figure 11: Output for without mask

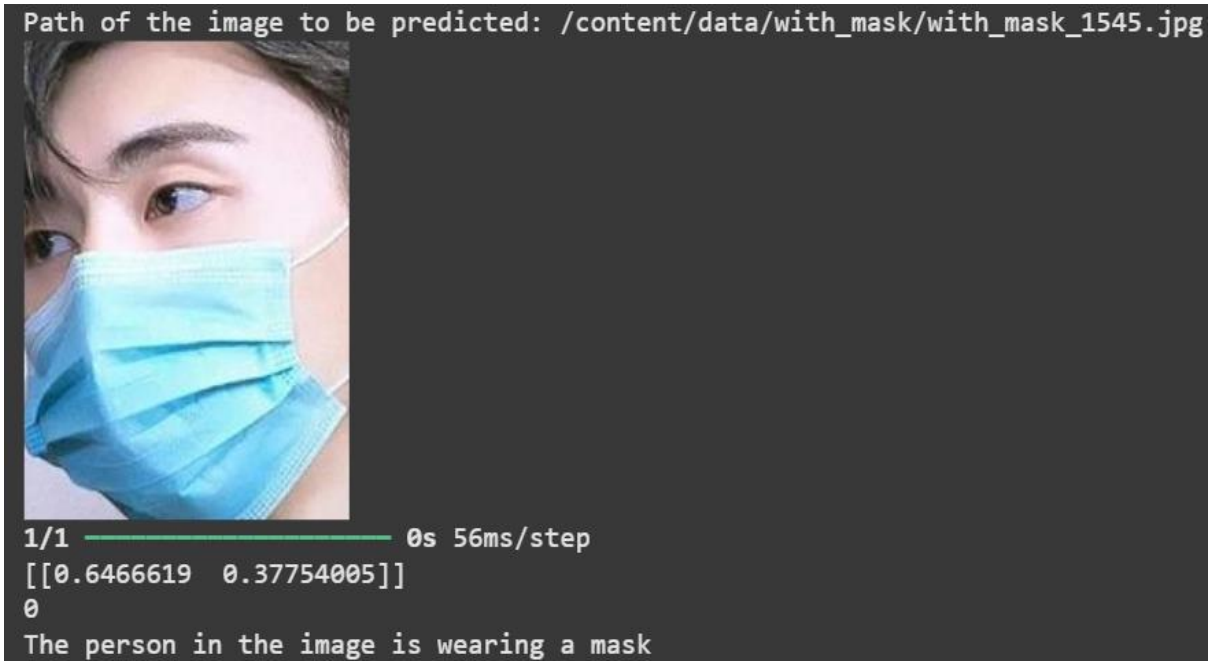

Figure 12: Output for mask

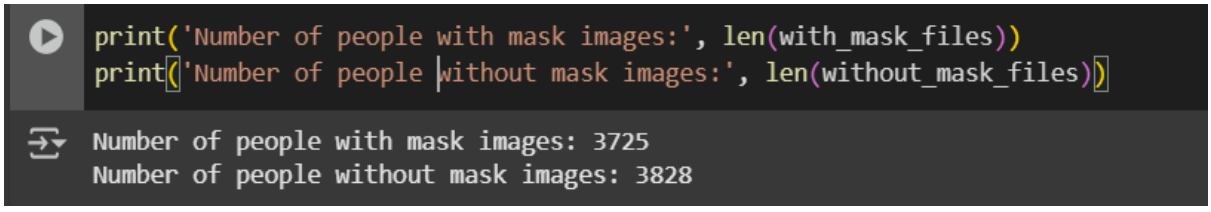

Figure 13: Validation

TESTING AND VALIDATION

6. TESTING AND VALIDATION

6.1 Testing:

Effective testing ensures the reliability, accuracy, and usability of the facemask detection system. The testing process is carried out across several levels to ensure that individual components work correctly and integrate well into the full system.

Unit Testing

- Purpose: To test individual modules and components such as data preprocessing, feature extraction, and the prediction model independently.
- Tools Used: unittest or pytest in Python.
- Focus Areas:
    - Ensure that each function returns expected results.
    - Handle edge cases such as empty input, invalid file types, or corrupted images.
    - Validate proper normalization and label encoding**.**

Integration Testing

- Purpose: To verify the interaction and data flow between system components.
- Focus Areas:
    - Confirm seamless integration between user input (image upload or webcam), preprocessing modules, and the prediction pipeline.
    - Check if feedback loops (if implemented) are correctly captured and used for retraining or monitoring.
    - Verify model loading and response coordination within the Flask web app.

System Testing

- Purpose: To test the entire system as a cohesive unit.

- Focus Areas:
    - Validate the full workflow from image input to prediction output.
    - Confirm adherence to functional requirements.

Performance Testing

- Purpose: To assess how well the system performs under different workloads and usage conditions.
- Focus Areas:
    - Measure average response time for predictions.
    - Evaluate memory usage and inference time for large datasets or real-time webcam streams.
    - Test system stability under concurrent usage.

User Acceptance Testing (UAT)

- Purpose: To evaluate the system's usability and effectiveness from the end-user's perspective.
- Focus Areas:
    - Collect feedback from users (e.g., volunteers or testers) on system functionality, ease of use, and accuracy.
    - Ensure the interface is intuitive and the predictions align with user expectations.
    - Make final refinements before deployment based on feedback.

6.2 Validation

When developing a face mask detection system using machine learning, particularly with Convolutional Neural Networks (CNNs), it's essential to implement various validation techniques to ensure the model's performance and reliability. Here are some common validation techniques:

1. Cross-Validation

K-Fold Cross-Validation: The dataset is divided into K subsets (folds). The model is trained K times, each time using a different fold as the test set and the remaining folds as the training set.

Purpose: Provides a more reliable estimate of model performance by reducing variance associated with a single train-test split.

2. Confusion Matrix

Description: A matrix that summarizes the performance of a classification model by comparing predicted labels to actual labels.

Purpose: Helps in understanding true positives, false positives, true negatives, and false negatives, providing insights into model accuracy.

3. Precision, Recall, and F1-Score

Precision: The ratio of true positive predictions to the total predicted positives.

Recall: The ratio of true positive predictions to the total actual positives.

F1-Score: The harmonic mean of precision and recall, useful for imbalanced datasets.

Purpose: These metrics provide a deeper understanding of model performance beyond accuracy.

4. Data Augmentation

Description: Techniques such as rotation, scaling, and flipping are used to artificially expand the training dataset.

Purpose: Improves model robustness and generalization by exposing it to a wider variety of data.

5. Validation on Real-World Data

Description: Testing the model on images collected in real-world scenarios, outside of the training and validation datasets.

CONCLUSION

AND

FUTURE ENHANCEMENT

7. CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion:.

The facemask detection system developed in this project demonstrates the effective use of deep learning techniques for real-time detection of masked and unmasked faces. Using a Convolutional Neural Network (CNN) model trained on a balanced dataset of masked and unmasked face images, the system successfully classifies images with a high degree of accuracy. The implementation integrates key machine learning concepts such as data preprocessing, feature extraction, and model evaluation, along with tools like TensorFlow, Keras, and OpenCV.

The model has shown promising results in identifying mask usage in various real-world scenarios. This project not only provides a foundation for improving public safety through technology but also highlights the significance of AI in automating health compliance monitoring, particularly in the context of global health crises like COVID-19.

Despite its strong performance, the system has some limitations, such as dependency on lighting conditions, occlusions, and camera angles. These challenges point to opportunities for further development.

7.2 Future Enhancement:

Integration with CCTV and IoT Devices

Connecting the detection system to surveillance cameras for automated real-time monitoring in public areas like malls, airports, and hospitals.

Multi-Class Classification

Extend the model to detect not just mask/no mask, but also improperly worn masks, enabling more detailed classification (e.g., mask under nose, chin mask).

Real-Time Alert System

Add functionality to trigger alerts or notifications when individuals without masks are detected, making the system actionable.

Larger and Diverse Dataset

Enhance model robustness by training with a larger, more diverse dataset featuring various ethnicities, lighting conditions, mask types, and backgrounds.

Mobile and Web Deployment

Deploy the model as a web or mobile application for broader accessibility and practical usage in low-resource settings.

Advanced Models and Transfer Learning

Improve performance by leveraging pre-trained models like MobileNet, ResNet, or EfficientNet for better feature extraction and reduced training time.

REFERENCES

8. REFERENCES

References:

[1] Z. Zhang, B. Bowes The future of artificial intelligence (AI) and machine learning (ML) in landscape design: a case study in coastal Virginia, USA

[2] S.Y. Kung, M.W. Mak Machine learning for multimodality genomic signal processing

[3] D. Duarte, F. Nex, N. Kerle, G. Vosselman

Satellite image classification of building damages using airborne and satellite image samples in a deep learning approach

ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 4(2) (2018)

[4] P. Gupta, N. Saxena, M. Sharma, J. Tripathi

Deep neural network for human face recognition

Int. J. Eng. Manufact., 8 (1) (2018), pp. 63-71, 10.5815/ijem.2018.01.06

[5] K.J. Bhojane, S.S. Thorat

A review of face recognition based car ignition and security system

Int. Res. J. Eng. Technol., 05 (01) (2018), pp. 532-533

Google Scholar

[6] S. Mahmud, J. KimAn Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network