Report On

# Face Mask Detector Using CNN

Submitted in partial fulfillment of the requirements of the Course Project for Advance Artificial Intelligence in Semester VIII of Fourth Year Artificial Intelligence & Data Science Engineering


by

Ojasi Prabhu (Roll No.40 )

Parth Raut (Roll No.41 )

Aditya Shinde (Roll No.49)

Yash Biranje (Roll No.6)


**Under the Guidance**

**Dr.Tatwadarshi P.N**



**University of Mumbai**

**Vidyavardhini's College of Engineering & Technology**

**Department of Artificial Intelligence and Data Science**



**(A.Y. 2024-25)**

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence & Data Science

# CERTIFICATE

This is to certify that the project entitled "Face Mask Detector Using CNN" is a bonafide work of Ojasi Prabhu (Roll No.40 ), Parth Raut (Roll No.41 ),Aditya Shinde (Roll No.49 ),Yash Biranje(Roll No.6)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester VIII of Fourth Year Artificial Intelligence and Data Science engineering.

_____

Guide

# Abstract

This project introduces a Face Mask Detection System using deep learning techniques to enhance public safety during contagious outbreaks. The system employs a pre-trained convolutional neural network (CNN) to accurately identify faces with or without masks. Extensive training data encompassing diverse demographics, lighting conditions, and poses ensures robust performance. The system provides real-time feedback through an intuitive interface, facilitating integration with various devices. Designed for seamless deployment in public spaces, the system assists in enforcing mask mandates. It alerts authorities or individuals when mask non-compliance is detected. The technology's versatility and effectiveness are validated through rigorous experiments. With applications in airports, hospitals, and crowded environments, the system plays a pivotal role in collective protection during pandemics and similar health crises. Its implementation represents a significant stride toward safeguarding public health and safety.

# Table of Contents

# Chapter 1

## 1.1 Problem Statement:

In the post-COVID era, ensuring public health and safety remains paramount. Adherence to mask-wearing mandates is critical in preventing potential outbreaks. However, manual monitoring of mask compliance in crowded spaces is resource-intensive and poses risks to personnel. Existing solutions often lack accuracy, speed, or seamless integration with surveillance systems. Moreover, they may not be optimized for diverse post-pandemic environments. The need for an automated system capable of swiftly and accurately identifying individuals wearing or not wearing masks is pressing. This project aims to develop a state-of-the-art Face Mask Detection System, leveraging advanced deep learning techniques. The system must achieve high accuracy and provide real-time alerts for non-compliance. It should be adaptable to various settings and easily integrate with existing hardware configurations. By automating mask compliance monitoring, this system will alleviate the burden on human resources and ensure a safer environment for all. It represents a crucial step towards safeguarding public health in the aftermath of the COVID-19 pandemic.

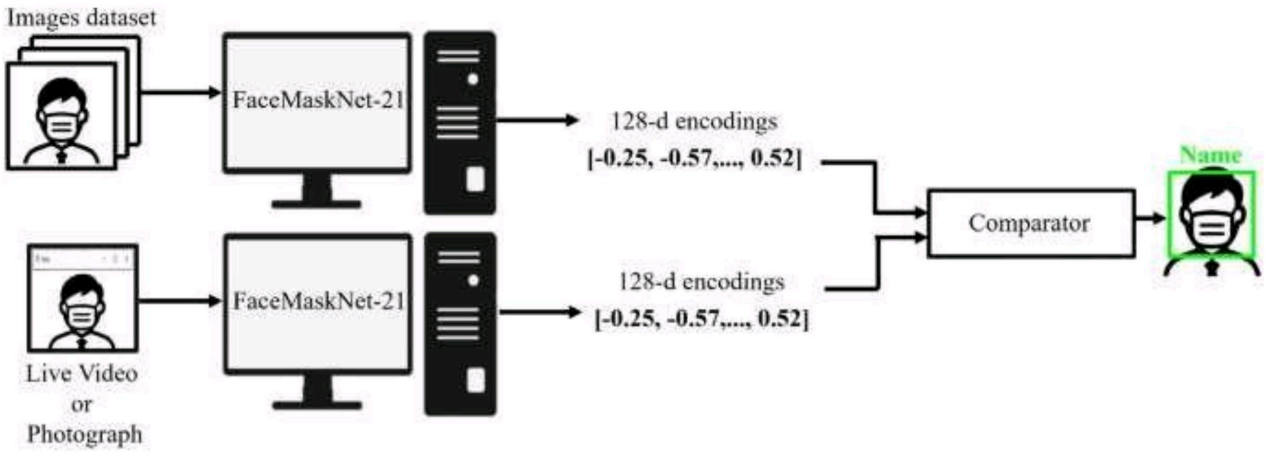
# Chapter 2

## 2.1 Description and Working:

The Post-COVID Face Mask Compliance Monitoring System is an innovative solution designed to ensure public safety in the aftermath of the pandemic. It leverages cutting-edge deep learning technology to accurately detect and classify individuals based on their mask-wearing status in real-time. This system addresses the challenge of enforcing mask mandates in crowded public spaces, providing an automated and efficient alternative to manual monitoring.

The system employs a pre-trained convolutional neural network (CNN), a type of deep learning model, which is adept at understanding visual patterns. It processes live or recorded video feeds from surveillance cameras. As frames of the video stream are captured, the CNN analyzes each frame to identify human faces. Once a face is detected, the system focuses on the region of interest, i.e., the face.

Using the learned features, the CNN then determines whether the individual is wearing a mask or not. This classification decision is made with high accuracy, owing to the extensive training data and advanced neural network architecture. In real-time, the system updates its assessment with each new frame, ensuring swift and accurate identification.



## 2.2 Software & Hardware used:

Software:

- Visual Studio Code
- Python 3.11
- Windows 10 OS
- Google Colab

Hardware:

- 64 bit Operating System
- 6gb RAM
- Intel i5 processor

# Chapter #3

## 3.1 Code

```
!pip install kaggle
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: kaggle in /usr/local/lib/python3.8/dist-packages (1.5.12)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.8/dist-packages (from kaggle) (1.15.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from kaggle) (4.64.1)
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from kaggle) (2.25.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.8/dist-packages (from kaggle) (1.24.3)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.8/dist-packages (from kaggle) (8.0.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.8/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.8/dist-packages (from kaggle) (2022.12.7)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.8/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->kaggle) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->kaggle) (4.0.0)
```

```
# configuring the path of Kaggle.json file
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

rting Face Mask Dataset

```
# API to fetch the dataset from Kaggle
!kaggle datasets download -d omkargurav/face-mask-dataset
```

```
Downloading face-mask-dataset.zip to /content
100% 163M/163M [00:09<00:00, 22.1MB/s]
```

```
# extracting the compessed Dataset
from zipfile import ZipFile
dataset = '/content/face-mask-dataset.zip'

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

```
The dataset is extracted
```

```
[ ] !ls
```

```
data  face-mask-dataset.zip  kaggle.json  sample_data
```

**Importing the Dependencies**

```
[ ] 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])
```

```
['with_mask_193.jpg', 'with_mask_754.jpg', 'with_mask_486.jpg', 'with_mask_2756.jpg', 'with_mask_1328.jpg']
['with_mask_2590.jpg', 'with_mask_1545.jpg', 'with_mask_3357.jpg', 'with_mask_1143.jpg', 'with_mask_2196.jpg']
```

```
[ ] without_mask_files = os.listdir('/content/data/without_mask')
    print(without_mask_files[0:5])
    print(without_mask_files[-5:])
```

```
['without_mask_1871.jpg', 'without_mask_1012.jpg', 'without_mask_2600.jpg', 'without_mask_1623.jpg', 'without_mask_1116.jpg']
['without_mask_2925.jpg', 'without_mask_3559.jpg', 'without_mask_38.jpg', 'without_mask_1333.jpg', 'without_mask_1137.jpg']
```

```
[ ] print('Number of with mask images:', len(with_mask_files))
    print('Number of without mask images:', len(without_mask_files))
```

```
Number of with mask images: 3725
Number of without mask images: 3828
```

**Creating Labels for the two class of Images**

with mask --> 1

without mask --> 0

```
[ ] # create the labels

    with_mask_labels = [1]*3725
```

```
without_mask_labels = [0]*3828
```

```
print(with_mask_labels[0:5])

print(without_mask_labels[0:5])
```

```
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

```
print(len(with_mask_labels))
print(len(without_mask_labels))
```

```
3725
3828
```

```
labels = with_mask_labels + without_mask_labels

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

```
7553
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

**Displaying the Images**

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

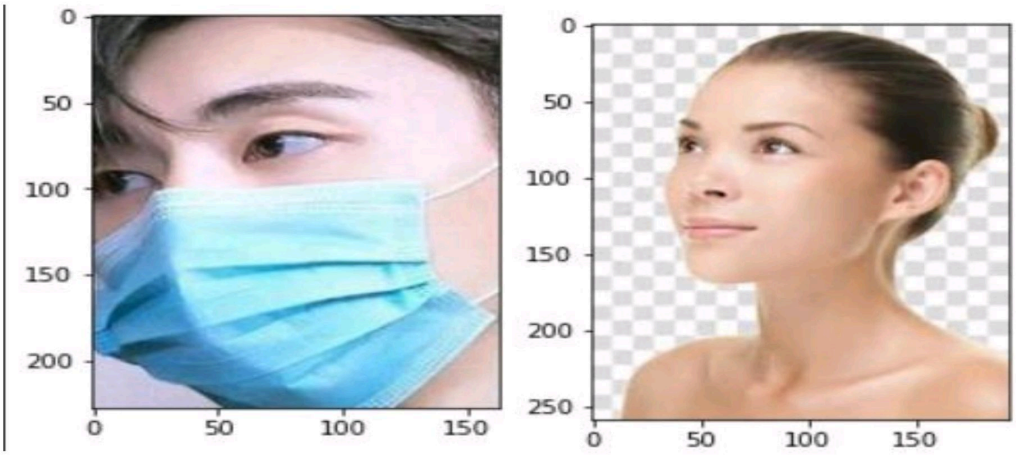



**Image Processing**

1. Resize the Images
2. Convert the images to numpy arrays

```
# convert images to numpy arrays+

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)
```

```
/usr/local/lib/python3.8/dist-packages/PIL/Image.py:959: UserWarning: Palette images with Transparency expressed in bytes should be converted to RGBA images
  warnings.warn(
```

```
type(data)
```

```
list
```

```
len(data)
```

```
7553
```

```
[ ] type(data[0])
```

```
numpy.ndarray
```

```
[ ] data[0].shape
```

```
(128, 128, 3)
```

```
[ ] # converting image list and label list to numpy arrays

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

```
[ ] type(X)
```

```
numpy.ndarray
```

```
[ ] type(Y)
```

```
numpy.ndarray
```

```
[ ] print(X.shape)
    print(Y.shape)
```

```
(7553, 128, 128, 3)
(7553,)
```

```
[ ] print(Y)
```

```
[1 1 1 ... 0 0 0]
```

**Train Test Split**

```
[ ] 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)
```

```
(7553, 128, 128, 3) (6042, 128, 128, 3) (1511, 128, 128, 3)
```

```
[ ] # scaling the data

    X_train_scaled = X_train/255
```

```
[ ] X_test_scaled = X_test/255
```

```
X_train[0]
```

```
array([[[109, 107, 118],
        [114, 113, 121],
        [109, 107, 116],
        ...,
        [ 90,  97, 107],
        [ 90,  94, 105],
        [ 93,  97, 108]],

       [[110, 108, 119],
        [111, 108, 117],
        [110, 105, 114],
        ...,
        [ 86,  93, 103],
        [ 88,  92, 103],
        [ 89,  93, 104]],

       [[112, 107, 118],
        [113, 109, 118],
        [123, 117, 125],
        ...,
        [ 89,  95, 105],
        [ 91,  95, 106],
        [ 87,  91, 102]],

       ...,

       [[ 46,  66,  91],
        [ 45,  65,  90],
        [ 47,  67,  92],
        ...,
        [177, 143, 123],
        [176, 144, 123],
        [177, 145, 124]],

       [[ 49,  69,  93],
        [ 47,  67,  91],
        [ 46,  66,  90],
        ...,
        [179, 146, 126],
        [178, 146, 125],
        [177, 146, 125]],

       [[ 43,  63,  87],
        [ 43,  63,  87],
        [ 44,  64,  88],
        ...,
```

**Building a Convolutional Neural Networks (CNN)**

```
[ ] 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'))
```

```
[ ] # compile the neural network
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['acc'])
```

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

```
Epoch 1/5
170/170 [==============================] - 15s 24ms/step - loss: 0.4886 - acc: 0.7848 - val_loss: 0.3200 - val_acc: 0.8711
Epoch 2/5
170/170 [==============================] - 3s 17ms/step - loss: 0.2937 - acc: 0.8847 - val_loss: 0.2501 - val_acc: 0.9008
Epoch 3/5
170/170 [==============================] - 3s 17ms/step - loss: 0.2523 - acc: 0.9016 - val_loss: 0.2516 - val_acc: 0.8992
Epoch 4/5
170/170 [==============================] - 3s 19ms/step - loss: 0.1970 - acc: 0.9270 - val_loss: 0.2292 - val_acc: 0.9256
Epoch 5/5
170/170 [==============================] - 3s 17ms/step - loss: 0.1810 - acc: 0.9308 - val_loss: 0.2427 - val_acc: 0.9074
```

**Model Evaluation**

```
[ ] loss, accuracy = model.evaluate(X_test_scaled, Y_test)
    print('Test Accuracy =', accuracy)
```

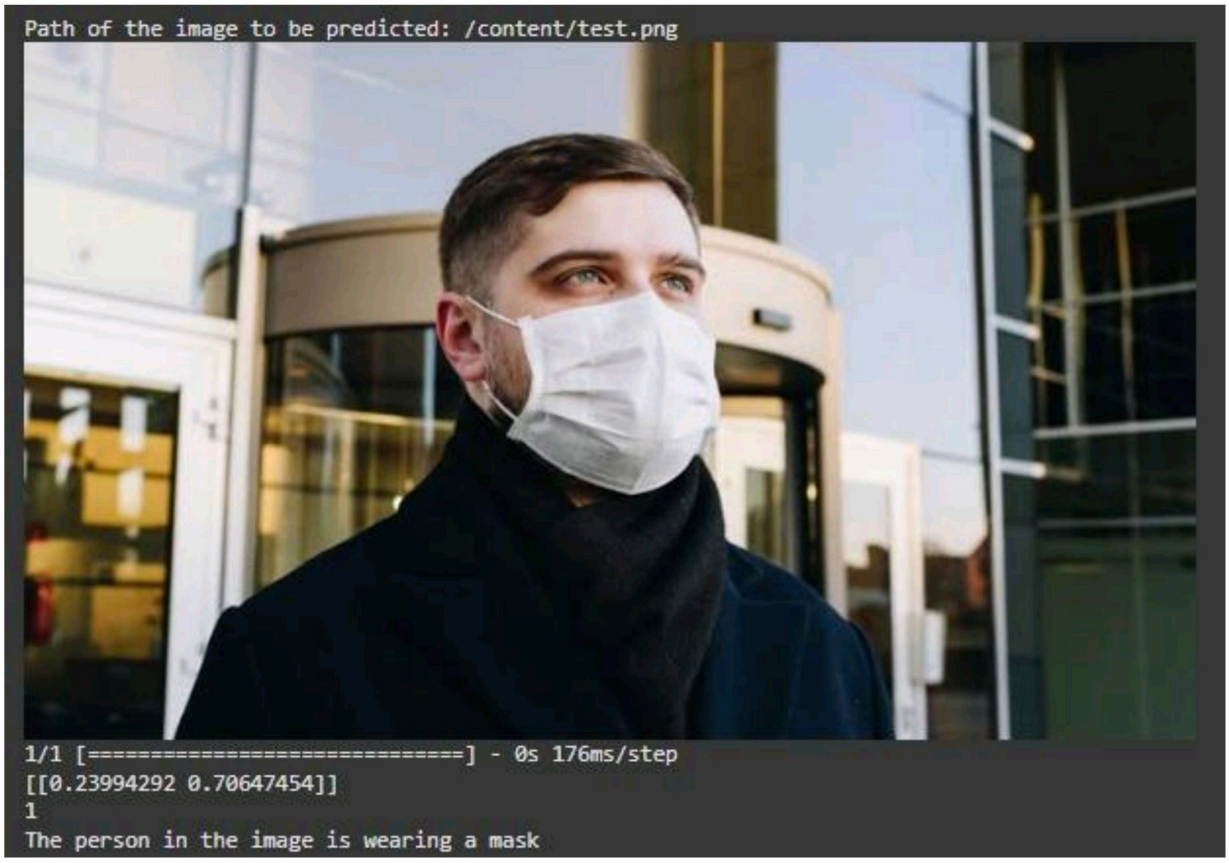
```
48/48 [==============================] - 1s 11ms/step - loss: 0.2065 - acc: 0.9219
Test Accuracy = 0.9219059944152832
```

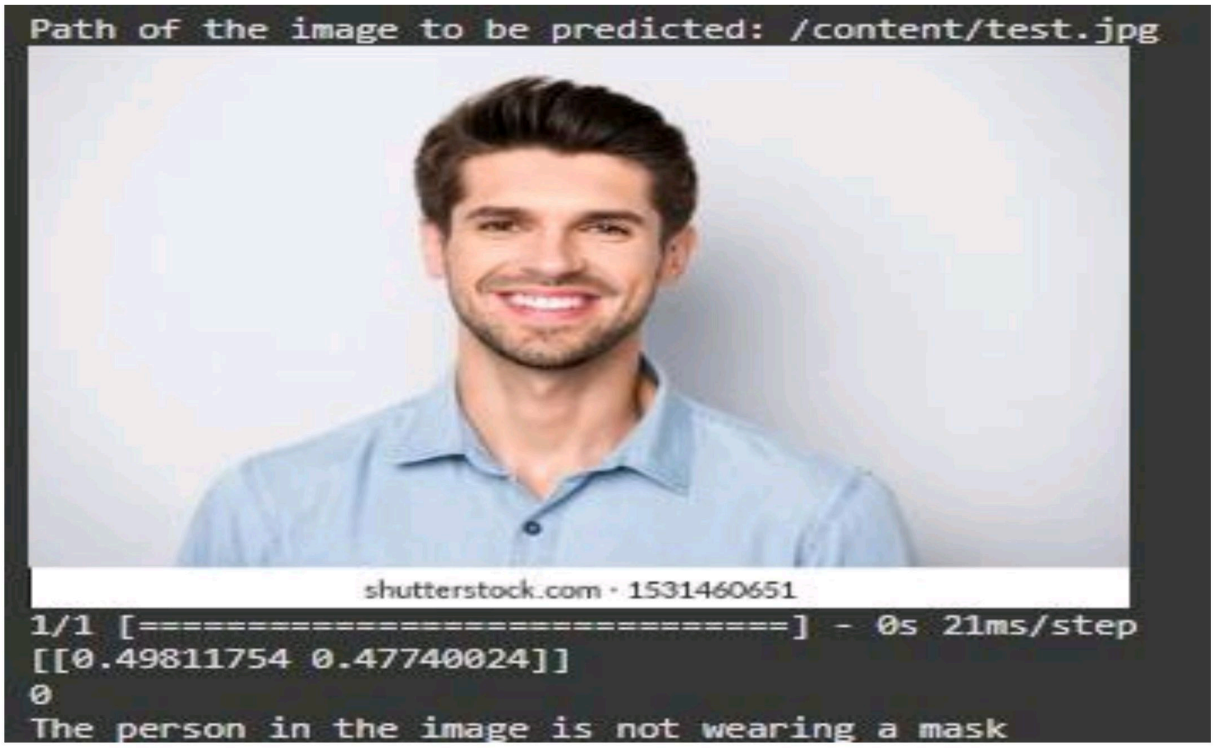
```
    h = history

    # 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()
```

### 3.2 Results


Path of the image to be predicted: /content/test.png
1/1 [==============================] - 0s 176ms/step
[[0.23994292 0.70647454]]
1
The person in the image is wearing a mask


Path of the image to be predicted: /content/test.jpg
shutterstock.com · 1531460651
1/1 [==============================] - 0s 21ms/step
[[0.49811754 0.47740024]]
0
The person in the image is not wearing a mask

## 3.3 CONCLUSION AND FUTURE SCOPE:

The Post-COVID Face Mask Compliance Monitoring System represents a critical advancement in public safety measures, particularly in the wake of the COVID-19 pandemic. By harnessing deep learning technology, it provides an efficient and accurate means of enforcing mask mandates in crowded spaces. The system's ability to swiftly identify individuals wearing or not wearing masks reduces the burden on human resources and minimizes potential health risks. Through extensive training and advanced neural network architecture, it achieves high levels of accuracy in real-time assessments.

**Future Scope:**

The project lays a strong foundation for further enhancements and applications in the field of public health and safety. Future avenues for development include:

1. **Multi-Modal Sensing:** Integration of additional sensors like thermal cameras for temperature screening, enhancing the system's ability to identify potentially symptomatic individuals.
2. **Mask Quality Assessment:** Incorporating features to evaluate the quality and effectiveness of masks worn by individuals.
3. **Social Distancing Monitoring:** Expanding the system's capabilities to include monitoring and alerting for social distancing violations.
4. **Data Privacy and Security:** Strengthening measures to ensure the privacy and security of data collected by the system, adhering to relevant regulations and standards.
5. **Geographical Expansion:** Adapting the system to various geographic regions and diverse demographic profiles, considering cultural differences in mask-wearing norms.
6. **Real-Time Analytics and Reporting:** Incorporating features for generating reports and analytics on mask compliance trends, aiding in policymaking and enforcement strategies.

# Chapter 4

# REFERENCES

[1] Toshan Meenpal, Ashutosh Balakrishnan, Amit Verma," Facial Mask Detection using Semantic Segmentation", 2019 4th International Conference on Computing, pp. 1-6, DOI:10.1109/CCCS.2019.8888092

[2] Jiang, X.; Gao, T.; Zhu, Z.; Zhao, Y. "Real-Time Face Mask Detection Method Based on YOLOv3", Electronics 2021, pp. 1-17, https://doi.org/10.3390/electronics10070837

[3] Mingjie Jiang, Xinqi Fan, Hong Yan," Retina face mask: A face mask detector" June 9, 2020, pp. 1-9, arXiv:2005.03950v2 [cs.CV]

[4] Safa Teboulbi, Seifeddine Messaoud, Mohamed Ali Hajjaji," Real-Time Implementation of AIBased Face Mask Detection and Social Distancing Measuring System for COVID-19 Prevention",27 September 2017, Volume 2021, Article ID 8340779, pp. 1-21,10.1155

[5] G. Jignesh Chowdary, Narinder Singh Punn, Sanjay Kumar Sonbhadra, Sonali Agarwal," Face Mask Detection using Transfer Learning of InceptionV3",20 October 2020, pp. 1-11, arXiv :2009.08369v2 [cs.CV]