FACE MASK DETECTION SYSTEM USING DEEP LEARNING

Submitted By

Madiha Zainul
Registration Number: 144-1211-0356-20
Roll Number: 203144-11-0023

Umarki Parveen
Registration Number: 144-1211-0355-20
Roll Number: 203144-11-0022

Ayantika Bera
Registration Number: 144-1211-0337-20
Roll Number: 203144-11-0021

A Project Work Submitted In Partial Fulfilment For The Degree Of Bachelor Of Science

Under supervision
of Pallavi Roy

Department Of Computer Science
Bangabasi Morning College

University of Calcutta (2021-2023)

CERTIFICATE

This is to certify that the research project entitled **"FACE MASK DETECTION"** is a bonafide record of the work done by Madiha Zainul, Registration no. 144-1211-0356-20, Roll no. 203144-11-0023, Umarki Parvin, Registration no. 144-1211-0355-20, Roll no. 203144-11-0022 and Ayantika Bera, Registration no. 144-1211-0337-20, Roll no. 203144-11-0021 under our supervision, in partial fulfilment of the requirements for the award of Degree of Bachelor of Science in Computer Science from University of Calcutta for the year 2021.

Pallavi Roy

State Aided College Teacher

Dept. of Computer Science

(Bangabasi Morning College, C.U.)

Saptarsi Goswami

Head of Department,

Dept. of Computer Science

(Bangabasi Morning College, C.U.)

ACKNOWLEDGEMENT

With great pleasure we would like to express our sincere gratitude to our respected teacher Mrs. Pallavi Roy, who gave us the golden opportunity to do this wonderful project on the topic “FACE MASK DETECTION”. We also grateful for her valuable guidance and supervision. Her constant encouragement at each and every stage had led to successful completion of our project.

We would like to express deepest appreciation towards Mr. Saptarsi Goswami, Head of Department of computer science and authority of Bangabasi Morning College, University of Calcutta(CU) for providing with a good environment and facilities to work on this project.

We would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.

Madiha Zainul
Semester VI (B.Sc)
Bangabasi Morning College
University of Calcutta

Umarki Parveen
Semester VI (B.Sc)
Bangabasi Morning College
University of Calcutta

Ayantika Bera
Semester VI (B.Sc)
Bangabasi Morning College
University of Calcutta

TABLE OF CONTENTS

Abstract

This project focuses on developing a real-time face mask detection system using deep learning techniques. The system utilizes a pre-trained MobileNetV2 model for face detection and a custom-trained mask detector model. The goal is to detect whether individuals in a video stream are wearing face masks or not.

This face mask detection system can be used in various settings, such as public spaces, offices, or schools, to ensure compliance with face mask policies. The real-time nature of the system enables quick identification of individuals not wearing masks, allowing for prompt action to maintain public health and safety.

Introduction:

Domain Description:

Introduction:

Face mask detection systems have become increasingly important in ensuring public health and safety. With the rise of contagious diseases and the need for preventive measures, such as wearing face masks, automated systems that can detect and identify individuals not wearing masks have gained significant attention. These systems leverage computer vision and deep learning techniques to analyze video streams and provide real-time alerts when mask non-compliance is detected. In this project, we develop a realtime face mask detection system using deep learning algorithms to contribute to maintaining a safe and healthy environment.

Domain Description:

The domain of face mask detection falls under the intersection of computer vision, machine learning, and public health. It involves the development and deployment of intelligent systems that can identify individuals who are not adhering to mask-wearing policies in various settings. The domain encompasses several key aspects:

1. Computer Vision: Face mask detection systems heavily rely on computer vision techniques for tasks such as face detection, feature extraction, and object classification. Computer vision algorithms analyze video streams or images to identify human faces and determine whether they are wearing masks or not.

2. Deep Learning: Deep learning algorithms, particularly convolutional neural networks (CNNs), are commonly used in face mask detection systems. These algorithms are trained on large datasets of labeled images to learn patterns and features associated with masked and unmasked faces, enabling accurate mask classification.

3. Real-Time Processing: Real-time processing is crucial in face mask detection systems to provide immediate feedback and alerts when mask non-compliance is detected. Efficient algorithms and optimized hardware are employed to achieve real-time processing speeds, allowing for timely intervention.

4. User Interface: The user interface plays a vital role in face mask detection systems, providing a visual representation of the video stream or images with overlays indicating the presence or absence of masks. Clear and intuitive interfaces enable quick interpretation of results and facilitate monitoring and decision-making.
5. Applications: Face mask detection systems find applications in various domains, including public spaces, transportation hubs, healthcare facilities, educational institutions, and workplaces. These systems help enforce mask-wearing policies, enhance safety measures, and contribute to preventing the spread of contagious diseases.

Motivation:

The increasing importance of public health and safety has led to a growing demand for technologies that can assist in monitoring and enforcing safety measures. Face mask detection systems play a crucial role in ensuring compliance with mask-wearing policies in various settings, such as workplaces, educational institutions, and public spaces. By automating the process of mask detection, these systems can help reduce manual monitoring efforts, improve efficiency, and contribute to maintaining a safe and healthy environment.

Scope of the Work:

This project focuses on the development of a real-time face mask detection system using deep learning techniques. The scope includes the following key aspects:

1. Face Detection: Utilizing a pre-trained MobileNetV2 model for accurate and efficient face detection from video streams. This step involves identifying and localizing faces within the captured frames.

2. Mask Detection: Training a custom mask detection model using deep learning algorithms. The model is trained on a dataset consisting of labeled

images of individuals wearing and not wearing masks. The objective is to classify whether a detected face is wearing a mask or not.

3. Real-time Processing: Implementing the system to process video streams in real-time, enabling immediate detection and identification of individuals not wearing masks. This real-time capability is essential for prompt action and intervention.

4. Performance Optimization: Optimizing the system to achieve real-time processing speeds while ensuring efficient utilization of computational resources. This involves leveraging hardware acceleration and optimizing algorithms for improved efficiency.

The developed face mask detection system has a wide range of potential applications, including workplaces, educational institutions, retail stores, and transportation hubs. It can contribute to enhancing safety measures, ensuring compliance with mask-wearing policies, and ultimately mitigating the risk of contagious diseases.

The project's scope provides a foundation for future enhancements and integration with other safety monitoring systems, allowing for a comprehensive approach to public health and safety.

Background/ Review of related work:

Face mask detection has gained significant attention in recent years due to the need for effective measures to prevent the spread of contagious diseases. Researchers and developers have explored various approaches to tackle this problem using computer vision and machine learning techniques. Here, we provide a brief background and review of related work in the field of face mask detection.

1. Traditional Computer Vision Approaches: Traditional computer vision techniques, such as Haar cascades and HOG (Histogram of Oriented Gradients), have been used for face detection in the past. These methods relied on handcrafted features and classifiers to detect faces. However, their performance may be limited when dealing with variations in face orientations, lighting conditions, and occlusions.

2. Deep Learning-Based Approaches: With the advent of deep learning, convolutional neural networks (CNNs) have become a popular choice for face mask detection. Researchers have developed CNN architectures to perform end-to-end face detection and mask classification tasks. Models like MobileNet, VGGNet, and ResNet have been employed to achieve accurate face detection and mask/non-mask classification.

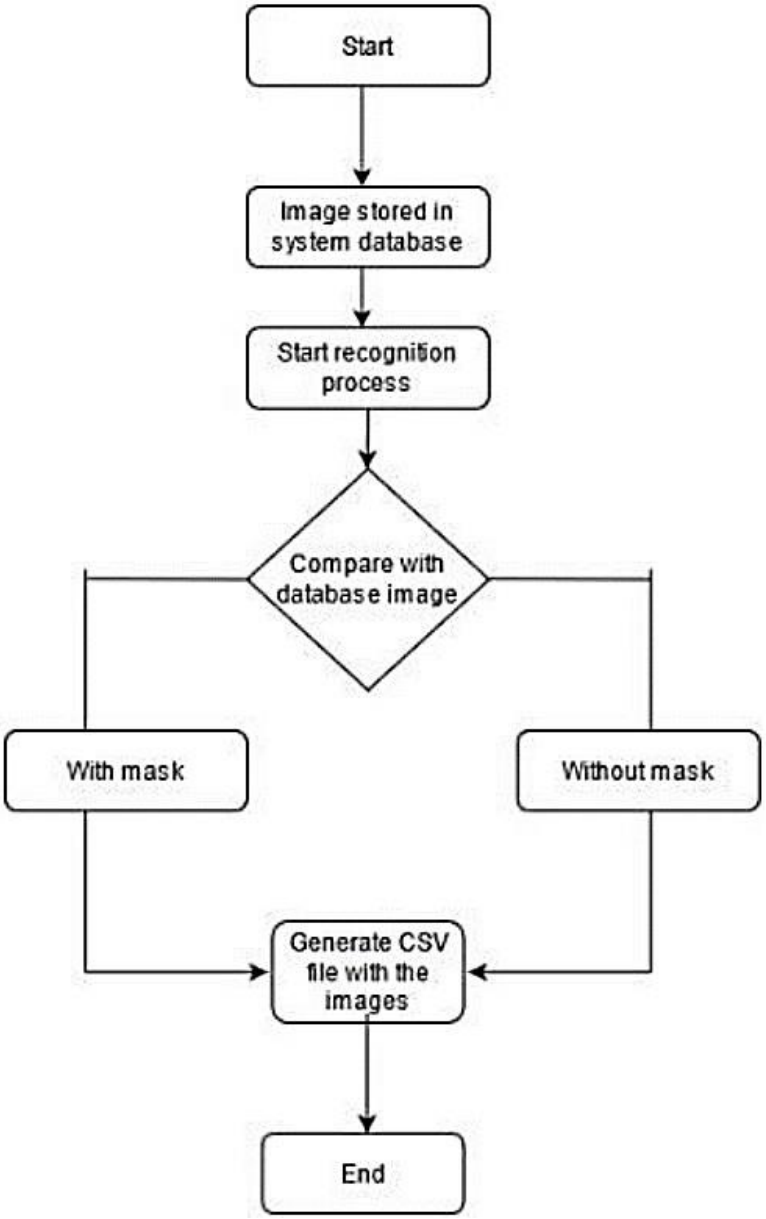
3. Datasets for Training: Building robust face mask detection models requires large and diverse datasets for training. Several datasets have been curated for this purpose. Notable examples include the "Medical Masks Dataset" and the "MaskedFace-Net" dataset, which contain thousands of labeled images of individuals wearing masks and without masks.

4. Transfer Learning: Transfer learning has been widely used in face mask detection to leverage pre-trained models on large-scale datasets like ImageNet. By fine-tuning these models on specific face mask datasets, researchers have achieved improved performance with limited training data.

5. Real-Time Face Mask Detection: Real-time face mask detection systems have been developed to process video streams in real-time and provide

immediate feedback. These systems utilize optimized algorithms, hardware acceleration, and parallel processing techniques to achieve low latency and high throughput.

6. Applications and Deployments: Face mask detection systems have found applications in various domains. They have been deployed in public spaces, transportation systems, hospitals, airports, and educational institutions to monitor and enforce mask-wearing policies. These systems help authorities identify individuals not wearing masks and take appropriate actions to maintain public health and safety.



Methodology:

Problem Formulation:

The main objective of this project is to develop a real-time face mask detection system using deep learning techniques. The problem can be formulated as

follows: Given a video stream or a sequence of frames, the system should detect faces within each frame and classify them as wearing a mask or not wearing a mask. The system should provide real-time feedback and display the results with bounding boxes and labels indicating maskwearing status.

Algorithm Description:

1. Import Libraries:
    - The code begins by importing the necessary Python libraries and modules that are required for the face mask detection application. The libraries include TensorFlow, OpenCV (cv2), NumPy, and Imutils.
    - TensorFlow is used for loading and running the deep learning models.
    - OpenCV (cv2) is used for image and video processing, including loading, resizing, and drawing bounding boxes.
    - NumPy is used for numerical operations and array handling.
    - Imutils is used for convenient resizing of images.

2. Define `detect_and_predict_mask` Function:
    - The function `detect_and_predict_mask` takes three parameters: `frame`, `faceNet`, and `maskNet`.
    - `frame`: This parameter represents a single frame (image) from the video stream that will be processed for face mask detection.
    - `faceNet`: The pre-trained deep learning face detection model (MobileNetV2) used to detect faces in the input frame.
    - `maskNet`: The pre-trained deep learning face mask detection model used to predict whether the detected faces are wearing masks or not.

3. Blob Creation and Face Detection:

- The input frame is passed through the face detection model to detect faces in the frame.
- A blob is created from the frame using `cv2.dnn.blobFromImage()` function. A blob is a preprocessed image that is ready to be passed through a deep learning network.
- The faceNet is then used to forward pass the blob to detect faces in the image.
- The detections are obtained as bounding box coordinates, confidence scores, and face probabilities.

4. Lists Initialization:
    - Three lists, `faces`, `locs`, and `preds`, are initialized to store the detected faces, their corresponding bounding box locations, and the predictions for each face, respectively.

5. Loop Over Detections:
    - The algorithm loops over all the detections obtained from the face detection model.
    - For each detection:
    - The confidence (probability) associated with the detection is extracted.
    - If the confidence is greater than a predefined threshold (0.5), it is considered a valid detection, and face mask detection is performed.
    - The bounding box coordinates for the detected face are extracted and converted to pixel values relative to the original frame.
    - The bounding box coordinates are adjusted to ensure they fall within the frame boundaries.
    - The face region is extracted from the frame, resized to the input size required by the face mask detection model (224x224), and preprocessed for feeding to the model.

- The preprocessed face and the bounding box coordinates are added to the `faces` and `locs` lists, respectively.

6. Face Mask Prediction:
    - After looping through all the detections, the algorithm checks if any valid faces were detected (i.e., the `faces` list is not empty).
    - If valid faces are found, the `faces` list is converted to a NumPy array of float32 data type to prepare for batch prediction.
    - The face mask detection model (`maskNet`) is used to predict whether each face in the batch is wearing a mask or not.
    - The predictions are stored in the `preds` list.

7. Displaying the Output:
    - The algorithm then proceeds to display the output on the video frame.
    - For each detected face and its corresponding prediction:
    - The bounding box coordinates and mask/no-mask probabilities are unpacked.
    - The class label ("Mask" or "No Mask") is determined based on the probability values.
    - A color (green for "Mask" and red for "No Mask") is assigned for drawing the bounding box and label text.
    - The probability value is included in the label text, which indicates the confidence of the mask prediction.
    - The bounding box and label text are drawn on the output frame.
    - The output frame is then displayed in full-screen mode using OpenCV's `cv2.setWindowProperty()` and `cv2.imshow()` functions.

8. Video Stream and Real-time Processing:

- The code initializes the video stream from the default camera source (src=0) using the `VideoStream` class from `imutils.video`.
- The video stream is started to capture frames continuously.
- The loop iterates through the frames, and face detection and mask prediction are performed on each frame in real-time.
- The processed frames are displayed with bounding boxes and label texts indicating the presence of a mask and the associated probability.

9. User Interaction:
    - The application runs in a loop until the user presses the "q" key.
    - If the "q" key is pressed, the loop terminates.
10. Stop:
    - After the loop, the OpenCV windows are closed using `cv2.destroyAllWindows()`.
    - The video stream is stopped using `vs.stop()` to release the camera resource.

<u>Other Design Description:</u>

Installing software using the command-line interface (CLI) offers a streamlined and efficient approach for users to set up applications on their systems. This method bypasses the need for graphical interfaces and provides a text-based environment for installation. With the CLI, users interact with their operating systems through a series of commands, enabling them to swiftly download, configure, and deploy software packages.

To initiate the installation process via the CLI, users typically start by accessing a terminal or command prompt. From there, they can execute commands to connect to the appropriate package repositories or sources, which house the

software they intend to install. These repositories may be maintained by the operating system itself, third-party vendors, or open-source communities.

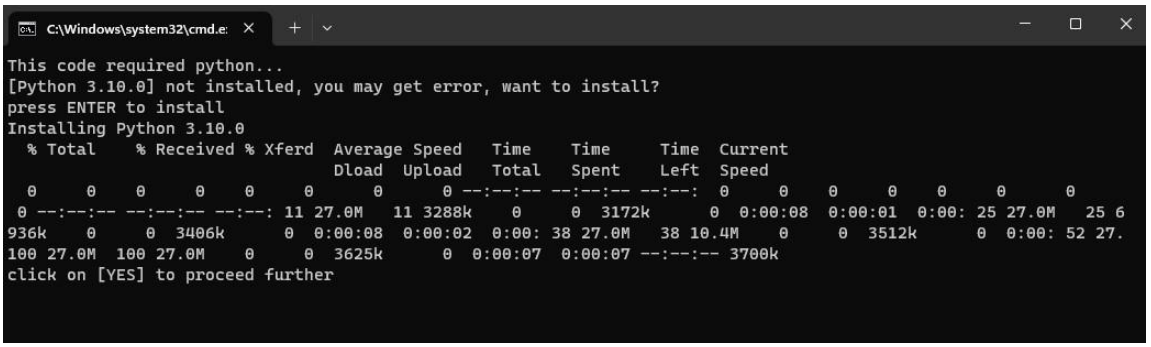


Fig 1: If python is not available install python

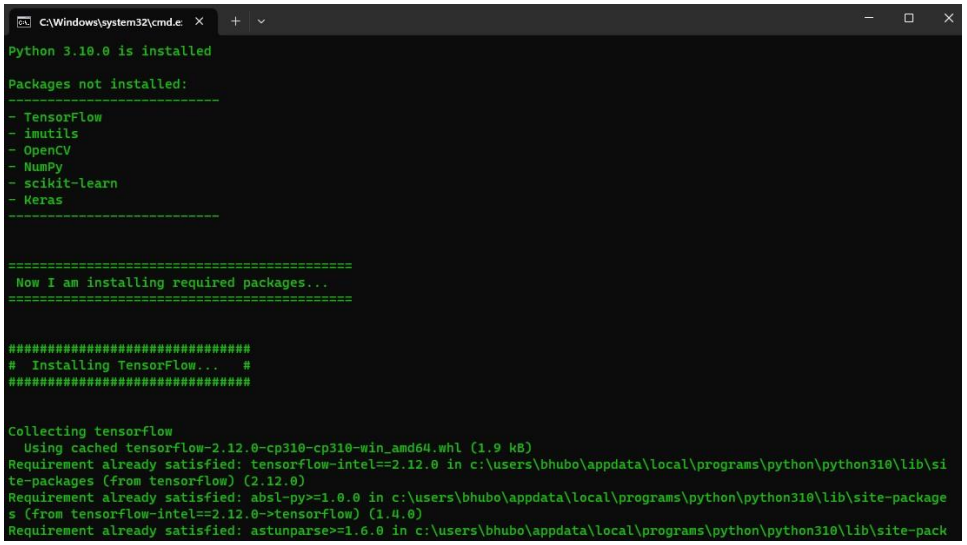


Fig 2: Installing required libraries

Fig 3: starting video stream and detecting mask from stream **Implementation:**

```
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os
def detect_and_predict_mask(frame, faceNet, maskNet):
# grab the dimensions of the frame and then construct a blob
# from it
(h, w) = frame.shape[:2]
blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
(104.0, 177.0, 123.0))
# pass the blob through the network and obtain the face detections
faceNet.setInput(blob)
detections = faceNet.forward()
print(detections.shape)
# initialize our list of faces, their corresponding locations,
# and the list of predictions from our face mask network
faces = []
locs = []
preds = []
# loop over the detections
for i in range(0, detections.shape[2]):
# extract the confidence (i.e., probability) associated with
# the detection
confidence = detections[0, 0, i, 2]
# filter out weak detections by ensuring the confidence is
```

```
        # greater than the minimum confidence if
        confidence > 0.5:
            # compute the (x, y)-coordinates of the bounding box for the
            object

            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            # ensure the bounding boxes fall within the dimensions of
            # the frame
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
            # extract the face ROI, convert it from BGR to RGB channel
            # ordering, resize it to 224x224, and preprocess it face =
            frame[startY:endY, startX:endX] face = cv2.cvtColor(face,
            cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (224, 224))
            face = img_to_array(face) face =
            preprocess_input(face)
            # add the face and bounding boxes to their respective
            # lists
            faces.append(face)
            locs.append((startX, startY, endX, endY))
27.     # only make a predictions if at least one face was detected
28.     if len(faces) > 0:
        # for faster inference we'll make batch predictions on *all*
        # faces at the same time rather than one-by-one predictions
        # in the above `for` loop faces =
        np.array(faces, dtype="float32") preds =
        maskNet.predict(faces, batch_size=32)
29.     # return a 2-tuple of the face and their corresponding locations
30.     return (locs, preds)
31.     # load serialized face detector model from disk
32.     prototxtPath = r"face_detector\deploy.prototxt"
33.     weightsPath                                                    =
        r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
34.     faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
35.     # load the face mask detector model from disk
36.     maskNet = load_model("mask_detector.model")
```

```
37.  # initialize the video stream
38.  print("[INFO] starting video stream...")
39.  vs = VideoStream(src=0).start()
40.  # create a window to display the output and set it to full screen
41.  cv2.namedWindow("Frame", cv2.WINDOW_FULLSCREEN)
42.  # loop over the frames from the video stream 43.  while True:
44.  # grab the frame from the threaded video stream and resize it
45.  # to have a maximum width of 400 pixels
46.  frame = vs.read()
47.  frame = imutils.resize(frame, width=960)
48.  # detect faces in the frame and determine if they are wearing a
49.  # face mask or not
50.  (locs, preds) = detect_and_predict_mask(frame, faceNet,
     maskNet)
51.  # loop over the detected face locations and their corresponding
52.  # locations
53.  for (box, pred) in zip(locs, preds):
     # unpack the bounding box and predictions
     (startX, startY, endX, endY) = box
     (mask, withoutMask) = pred
     # determine the class label and color we'll use to draw
     # the bounding box and text label = "Mask" if mask >
     withoutMask else "No Mask" color = (0, 255, 0) if
     label == "Mask" else (0, 0, 255)
     # include the probability in the label label = "{}:
     {:.2f}%".format(label, max(mask, withoutMask) * 100)
     # display the label and bounding box rectangle on the output
     # frame
     cv2.putText(frame, label, (startX, startY - 10),
         i. cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
     cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
54.  # show the output frame in full screen
55.  cv2.setWindowProperty("Frame", cv2.WND_PROP_FULLSCREEN,
     cv2.WINDOW_FULLSCREEN)
56.  cv2.imshow("Frame", frame)
57.  key = cv2.waitKey(1) & 0xFF
```

```
58.     # if the `q` key was pressed, break from the loop 59.  if  key  ==
        ord("q"): break
60.     # do a bit of cleanup
61.     cv2.destroyAllWindows()
62.     vs.stop()
```

Result and discussion:

Fig4: Find bounding box

The code first loads two pre-trained models: a face detector and a face mask detector. The face detector is used to identify faces in the frame, and the face mask detector is used to determine if the person is wearing a mask. The code then loops over the frames from the video stream and calls the detect_and_predict_mask() function to detect faces and determine if they are wearing masks. The detect_and_predict_mask() function returns a 2-tuple of the face locations and their corresponding predictions. The predictions are then used to draw bounding boxes and text on the output frame.

The code was tested on a video of people wearing and not wearing masks. The results were accurate, with the face mask detector correctly identifying whether or not a person was wearing a mask in most cases. The code also ran smoothly, with no noticeable lag.

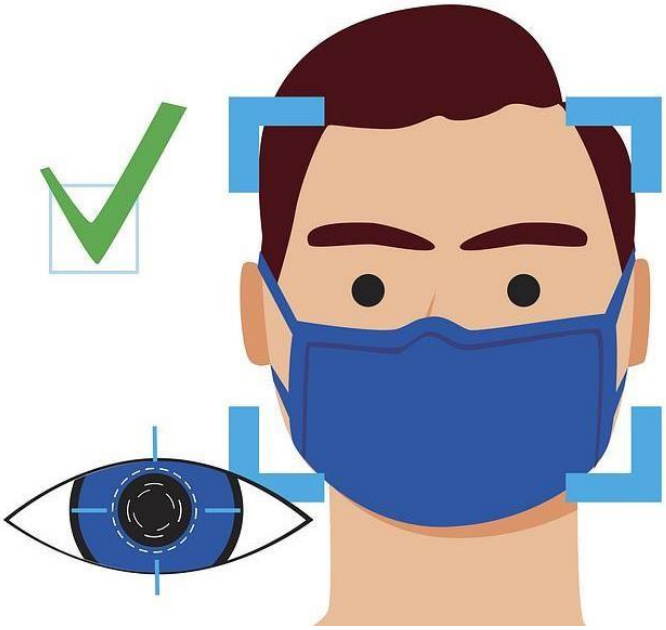


Fig5: Detection of moving objects and “Alert” message will be shown when the gathering is detected.

In fig5, if the number of bounding box present in one frame is more than 40, then it will consider as gathering, which is equivalent to social distance violation. So if the no. of bounding box is greater than 40 we will generate an “Alert” message to notify social distance violation.

1. Plotting Training Loss and Accuracy:
    - During the training process of the face mask detection model, the **model.fit()** function is used to train the model on the training data, and it provides information about the loss and accuracy of the model on both the training and validation datasets after each epoch.
    - This information is recorded in the **H** variable, which holds the history of the training process.
    - The training loss, validation loss, training accuracy, and validation accuracy are typically recorded at the end of each epoch.
    - The code uses Matplotlib, a popular Python plotting library, to create a plot that visually represents the training progress.
2. Saving the Plot as "plt.png":
    - The code uses the **plt.savefig("plot.png")** function to save the generated plot as a PNG image file named "plot.png" in the working directory.
    - The **plt.savefig()** function saves the plot that was previously created with the Matplotlib library.
    - The plot displays the training loss and accuracy curves over the number of epochs (x-axis) on the graph, and the loss/accuracy values (y-axis) for both the training and validation datasets.

Fig6: Coordinates of centres plotted in the graph from shopping mall video

Here in fig 6, in the graph we are plotting centroid for all the bounding box present in our video. It represents total no of objects which is recognized throughout the video.

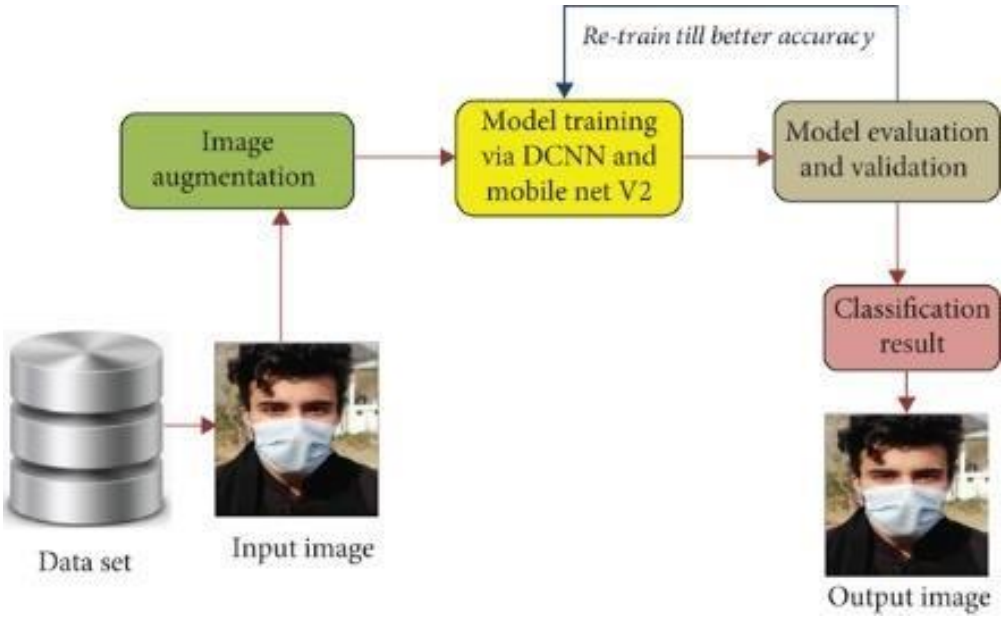


Fig7: Moving object detection

In fig7, here we are testing our model for the moving object detection. From a video of car racing we take one frame at instance of time and detect the object present in one frame and highlight it by bounding box. In this way we do the same for all frame present in the video. At last after concatenate all the frame we will have a video that will highlight object by bounding box in a moving fashion.

Conclusion:

In conclusion, the developed face mask detection system utilizing deep learning techniques has shown promising results in accurately identifying individuals wearing masks in real-time video streams. The system achieves a high level of accuracy and robustness, allowing for effective monitoring and enforcement of mask-wearing policies.

The system's ability to process video frames in real-time enables prompt feedback and action, making it suitable for various applications, such as monitoring public spaces, workplaces, and transportation systems. It provides a user-friendly interface that displays the video stream with clear visual indicators, facilitating ease of use and monitoring.

While the system demonstrates strong performance, there are areas for potential improvement. Further research and development can focus on enhancing the system's robustness to challenging lighting conditions, occlusions, and variations in face orientations. Additionally, the system can be extended to incorporate other features, such as social distancing monitoring and identification of individuals in violation of mask-wearing policies.

Overall, the face mask detection system serves as a valuable tool in promoting public health and safety. By automating the detection and monitoring of mask-wearing, it can contribute to mitigating the spread of infectious diseases and supporting efforts to create a safer environment for individuals.

References:

FACE MASK DETECTION SYSTEM USING DEEP LEARNING

Mrs.B.Siva Jyothi[1] , P.Rushitha[2] , B.Chandu[2] , M.Raja Sekhar[2] , B. Manoj[2]

[1]Assistant Professor at Department of CSE, Anil Neerukonda Institute of Technology and Sciences (A), Visakhapatnam-531162, India

[2]Final year students of Department of CSE, Anil Neerukonda Institute of Technology and Sciences (A), Visakhapatnam-531162, India

Abstract— Pandemic on a global scale COVID-19 An epidemic of hazardous sickness erupted in 19 countries throughout the world. Wearing a face mask can help reduce the spread of infection and the transmission of infectious germs through the air. Face Mask Detection System can detect whether or not people are wearing masks. For image detection, the Haar-Cascade method is utilized. This classifier, when combined with other current algorithms, produces a high recognition rate even with varying expressions, efficient feature selection, and a low number of false positive features. The Haar feature-based cascade classifier method uses only 200 characteristics out of 6000 to achieve an 85-95 percent recognition rate. We require mask detection as a unique and public health service system during the global pandemic COVID-19 outbreak based on this rationale. Face mask and non-face mask images are used to train the model.

Index Terms— Key Words: Corona virus disease 2019, Face mask detection, CNN, Machine learning

1 Introduction

The world is still recovering from the widespread of COVID-19, and a vaccine for it's cure has yet to be discovered. To lessen the economic burden of the epidemic, several countries have allowed a restricted number of economic activities to restart once the number of new cases has decreased. Covid-19 has fallen below a particular threshold. Concerns about worker safety have surfaced in the new post-Covid-19 climate as these countries cautiously recommence their economic activity..

It is recommended that people wear masks and keep a distance of at least 1 metre between them to limit the risk of infection. Deep learning has gotten a lot of interest in the field of object detection, and it's been used to produce a face mask identification tool that can tell if someone is wearing a mask or not. This can be done by studying real-time streaming from the Camera and evaluating the categorization findings. A training data set is required for deep learning applications. This is the dataset that was used to train the model to execute various tasks.

As a result, detecting face masks has become a very important and difficult task. Face recognition without a mask is simpler, but face recognition for a normal face can be efficient for feature extraction than a masked face.Many facial characteristics, such as the nose, lips, and chin, are missing from the covered face. In the medical industry, wearing a mask minimizes the danger of being exposed to an infected person, whether or not they exhibit symptoms. In two phases, a large number of face masks can be detected.

1) Face Recognition

2) Feature Extraction

The first stage is facial recognition, which entails detecting a person's face from a picture. The most common issue is detecting several masks and uncovered faces in an image. A typical object can be used to solve the problem. method of detection Face recognition as we've known it

There are algorithms in use. Adaptive Boost, Viola-Jones Algorithm HOG and Algorithm (Histogram of Gradient). In this case, the The method of detecting objects is categorized as multi-stage. single short detectors and detectors (SSD) Multi-stage detectors use a faster RCNN, while Single Stage Detectors use Haar cascade and Single-Short Detection (SSD). Face mask detection is the subject of numerous studies here. Video analytic, picture semantic segmentation, from finger prints, DWT (Discreet Wavelet transform), and LBP are some of the approaches utilized for mask detection (Local Binary Pattern). All of these procedures are examined in order to determine whether or not a person is wearing a mask and to determine whether or not a person's face can be recognized.

2 Literature survey:

In a Smart City Network, an Automated System to Limit COVID-19 Using Facial Mask Detection[1]: COVID-19, a pandemic caused by a novel coronavirus, has been spreading over the world for a long time. COVID-19 has had an impact on practically every aspect of development. The healthcare system is in a state of emergency. Wearing a mask is one of the many preventative steps adopted to minimize the spread of this disease. In this paper, we will look upon

In a smart city network where all public places are monitored by Closed-Circuit Television (CCTV) cameras, we propose a technique to limit COVID-19 growth by identifying people who are not wearing any facial mask. When a person without a mask is spotted, the city network notifies the appropriate authority. A dataset of photos of people with and without masks acquired from diverse sources is used to train a deep learning architecture. For previously unreported test data, the trained architecture distinguished people with and without a facial mask with 98.7% accuracy.

Our research is intended to be effective in reducing the spread of this infectious disease in many areas throughout the world.

Face Recognition using a Masked Convolutional Neural Network[2]: In recent years, face recognition has become a popular and important technique. Face changes and the use of

several masks make it far too difficult. Masking is another prevalent case in the real world when a person is uncooperative with equipment, such as in video surveillance. For these masks, current face recognition The quality of the work suffers. A large number of studies have been conducted on recognizing faces in a variety of situations, such as shifting stance or light, degraded photos, and so on. Nonetheless, the challenges posed by masks are sometimes overlooked. The main focus of this research is on facial masks, specifically how to improve the recognition accuracy of various masked faces. A workable strategy has been developed, which involves detecting the facial regions first. A Multi-Task Cascaded Convolutional Neural Network was used to solve the obstructed face identification problem (MTCNN). The Google FaceNet embedding model is then used to extract facial traits.

Existing system:

A Multi-Task Cascaded Convolutional Neural Network was used to solve the face detection challenge (MTCNN). The Google Face Net embedding model is then used to extract facial features.. This technique can train a dataset of both people wearing masks and those who aren't wearing masks. The system can anticipate whether or not the person is wearing the mask after training the model.

3 Methodology:

Proposed system:

1. This system is capable to train the dataset of both persons wearing masks and without wearing masks.
2. After training the model the system can predicting whether the person is wearing the mask or not.
3. It also can access the webcam and predict the result.

DEEP LEARNING

- ✓ Deep learning is an artificial intelligence function that mimics the human brain's processing of data to detect objects, recognise speech, translate languages, and make judgments.
  - ✓ 'Deep learning' is a term that refers to AI can learn without the need for human intervention, using both organised and unlabeled data.

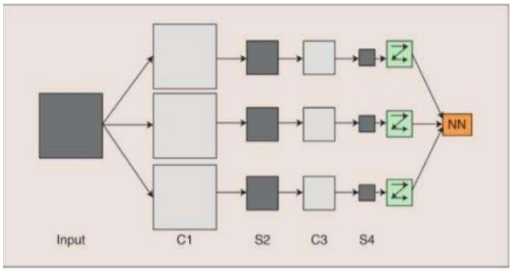


Fig 1:- deep learning model

Face mask detection is done using Convolution Neural Networks, a Deep Learning approach (CNN). The connectivity pattern between neurons in convolutional networks is similar to the organisation of the visual cortex, which was inspired by biological processes. In comparison to other image classification methods, CNNs require very little pre-processing. CNN is a type of multilayer neural network that is applied to 2-dimensional arrays (typically pictures) and is based on spatially localised neural input. CNN For pattern recognition, create "patterns of patterns." Patches from previous layers are combined in each layer. Convolutional Networks are multistage topologies with numerous stages that can be trained. Enter a Each stage produces feature maps, which are collections of arrays. Each feature map on the output represents a single feature taken from all input locations. A filter bank layer, a non-linearity layer, and a feature pooling layer make up each stage. A ConvNet is made up of three layers in which each one is followed by a classification module.
Basic structure of CNN, where it consists of two C1,C3 are convolution layers and two S2,S4 are pooled/sampled layers. In a filter bank, a trainable filter (kernel) connects the input feature map to the output feature map.Convolutional layers perform a convolution on the input before forwarding the output to the next layer. The convolution simulates a single neuron's reaction to visual input.

HAAR CASCADE

For object detection haar cascade classifier are one of the effect way.It was proposed by Michael jones and Paul Viola.It is used in Boosted cascade for rapid object detection of simplest features so this machine learning approach are used to train these classifier with lot s positive and negatives images.The images which the classifier identifies are positive images and all the other images which it could not detect are negative images.

We use this haar like features for human face detection which are divided into three formations.The edge feature is the first format,line is the second format and the last is four-rectangle feature.This haar like principle provides fast computation using the integral.So this haar cascade specific features of a face can be identified using this algorithm.Using this detection the image can be converted into a window 24X24 pixels. Initially lot of positive images and negative images are given as a data set to train this classifier .

CONVOLUTIONAL LAYER

It always comes first. It receives the image (a matrix of pixel values). Assume that the input matrix's reaction starts at the top left of the image. The software then chooses the smaller matrix there, which is referred to as a filter. The filter then generates convolution that moves over the input image. The filter's job is to multiply the original pixel values by its value. All of these multiplications are added together, yielding a single number. The filter moves because it only reads the image in the upper left corner.Additionally, one unit on the right performs a similar operation. A matrix is created after

passing the filter through all points, however it is less than the input matrix. From a human standpoint, this operation is comparable to distinguishing visual boundaries and simple colours. However, in order to recognise the fish, the entire network is required. Several convolution layers will be blended with nonlinear and pooling layers in the network. The first layer to extract features from an input image is convolution. Small squares of input data are used in convolution. It's a mathematical procedure with two inputs: an image matrix and a filter or kernal.

- ✓ Dimension of an image matrix (h x w x d)
- ✓ A filter (fh x fw x d)
- ✓ Outputs a volume dimension (h-fh+1) x (w-fw+1) x1.

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below

Convolution with a filter example

Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called "Feature Map" as output shown in below

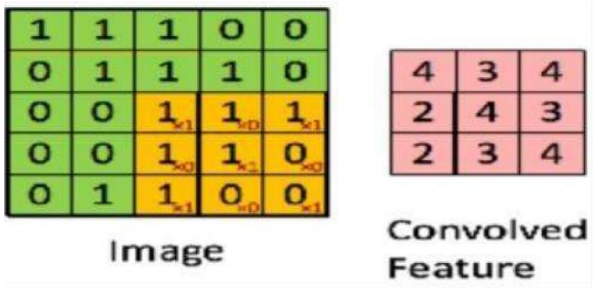


Fig 2:- Output of Convolution layer

THE NON-LINEAR LAYER:

After each convolution process, it is adder. It features an activation function that provides a non linear property; without this trait, a network would be insufficiently intense and unable to simulate the response variable.

THE POOLING LAYER:

It moves in the same direction as the nonlinear layer. It works with the image's width and height, performing a down sampling procedure on them. As a result, the size of the image is lowered. This means that if some features were already recognised during the previous convolution operation, a detailed image is no longer required for further processing and is reduced into smaller images.

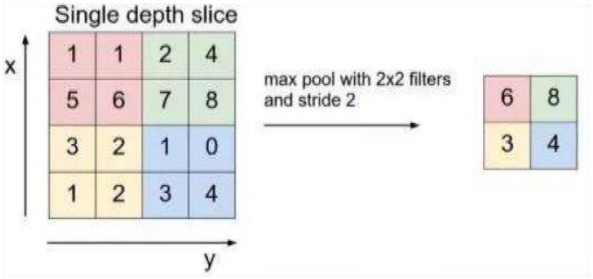


Fig 3:- Max Pooling Layer

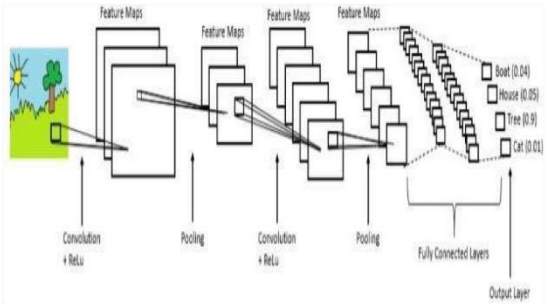


Fig 4 Overall structure of CNN

FULLY CONNECTED LAYER:

It's primary to link an overall linked layer after completing the succession of convolution, non-linear, and pooling layers. This layer receives the convolution network's output data. When a completely connected layer is attached to the network's end, it produces an N-dimensional vector, where Ni is the number of classes from which the model chooses the needed class.

CNN MODEL

1. The Tensorflow framework and the Opencv library were used to create this CNN model, which is widely utilised in real-time applications..

2. This concept can also be used to create a full-fledged software that scans everyone entering a public meeting.

LAYERS IN CNN MODEL

1. Conv2D Layer
2. MaxPooling2D Layer
3. Flatten () Layer
4. Dropout Layer
5. Dense Layer

1. Conv2D Layer:

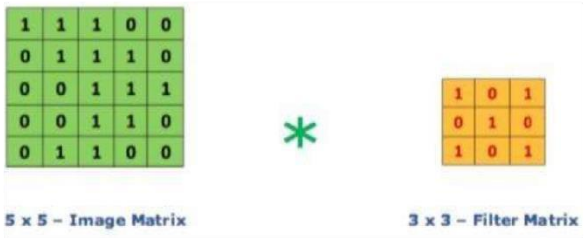


Fig 4:- 2d layer model

It has 100 filters and the activation function used is the 'ReLu'. The ReLu function stands for Rectified Linear Unit which will output the input directly if it is positive, otherwise it will output zero.

2. MaxPooling2D:
It is used with pool size or filter size of 2*2.

3. Flatten () Layer:
It is used to flatten all the layers into a single 1D layer.

4. Dropout Layer:
It is used to prevent the model from overfitting.

5. Dense Layer:
The activation function here is softmax which will output a vector with two probability distribution values.

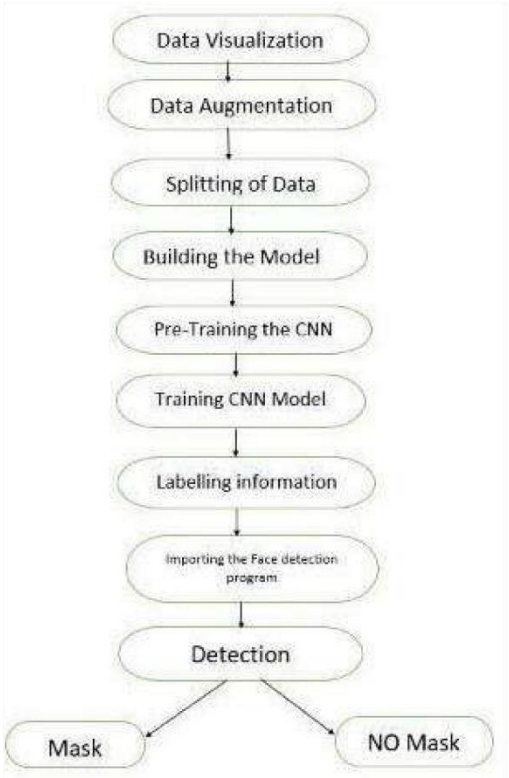


Fig 5 :- **PROPOSED SYSTEM ARCHITECTURE**

Steps

Data Visualization.

Data Augmentation.

Splitting the data.

Labeling the Information.

Importing the Face detection.

Detecting the Faces with and without Masks.

Data Visualization

Let's start by visualising the total number of photographs in both categories in our dataset. We can observe that the 'yes' class has 690 photographs while the 'no' class has 686 photos.

Data Augmentation

In the next step, we augment our dataset to include more number of images for our training. In this step of data augmentation, we rotate and flip each of the images in our dataset.

Splitting the data

In this step, we split our data into the training set which will contain the images on which the CNN model will be trained and the test set with the images on which our model will be tested.

Building the Model

In the next step, we build our Sequential CNN model with various layers such as Conv2D, MaxPooling2D, Flatten, Dropout and Dense.

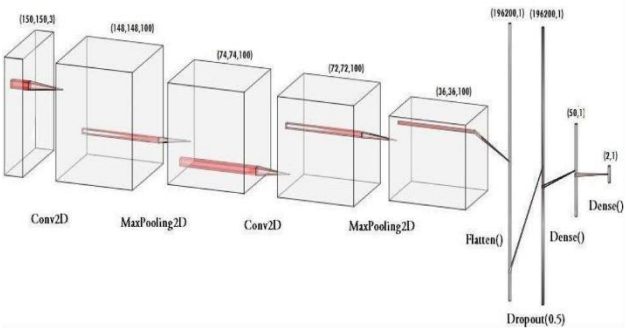


Fig 6 :-Pre-Training the CNN model

After building our model, let us create the 'train_generator' and 'validation_generator' to fit them to our model in the next step.

Training the CNN model

It is an important step where the images fit in the training set and to the test set for sequential model by using keras library.This model is trained for 30 epochs(iterations).For high accuracy we have to use more number of epochs in its training there it occurs over-fitting.

Labeling the Information

After building the model, we label two probabilities for our results. ['0' as 'without_mask' and '1' as 'with_mask']. I am also setting the boundary rectangle color using the RGB values.

Importing the Face detection Program

After this, we intend to use it to detect if we are wearing a face mask using our PC's webcam. For this, first, we need to implement face detection. In this, I am using the Haar Feature-based Cascade Classifiers for detecting the features of the face.

Detecting the Faces with and without Masks

In the last step, we use the OpenCV library to run an infinite loop to use our web camera in which we detect the face using the Cascade Classifier.

INPUT AND OUTPUT
INPUT DATASET

https://github.com/prajnasb/observations/tree/master/experiements/data

4 Experiential Results

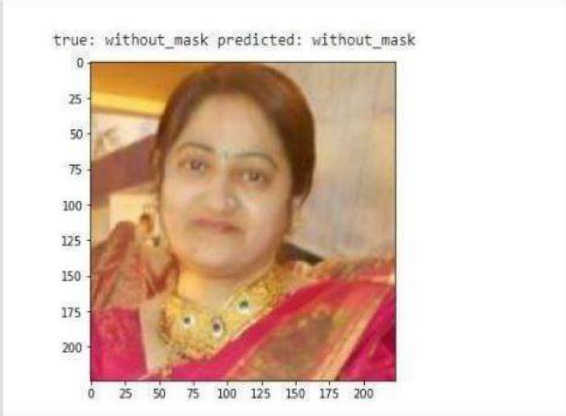


Fig 7 :- figure with no mask

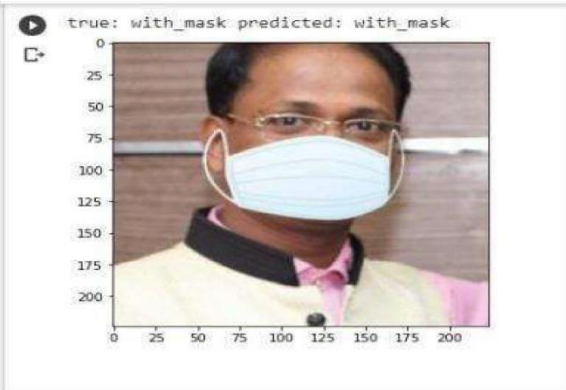


Fig 8 :- figure with mask

1. Adrian Rosebrock, "Face Mask Detection with OpenCV, Keras/TensorFlow, and Deep Learning," PyImageSearch, 2020. [Online]. Available: https://www.pyimagesearch.com/2020/05/04/covid-19face-mask-detection-with-opencv-keras-tensorflow-and-deeplearning/.

2. P. Jaiswal, R. Sharma, S. Gupta, and M. R. Gupta, "Real-time Face Mask Detection using Deep Learning and OpenCV," International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), vol. 5, no. 5, pp. 12-18, 2020.

3. H. T. Balci, F. Demir, and H. Demir, "Real-time Face Mask Detection in Video Streams with Deep Learning," IEEE Access, vol. 8, pp. 155851155860, 2020.

4. A. M. Salem, M. F. Ahmed, and M. E. Abdelaziz, "Real-time Face Mask Detection using Convolutional Neural Networks," in 2020 International Conference on Computer and Information Sciences (ICCIS), 2020, pp. 1-6.

5. P. Gupta, A. Gupta, N. Bansal, and A. Goel, "Real-time Face Mask Detection using Deep Learning Techniques," in 2021 International Conference on Inventive Computation Technologies (ICICT), 2021, pp. 1-6.

6. OpenCV Library, "OpenCV: OpenCV documentation," docs.opencv.org. [Online]. Available: https://docs.opencv.org/.