VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum-590014

PROJECT PHASE2 REPORT

On

"COVID-19 PRECAUTION TECHNIQUE BY REAL TIME FACE MASK DETECTOR USING MACHINE LEARNING"

Submitted to

Visvesvaraya Technological University

In the partial fulfillment of requirements for the award of degree

BACHELOR OF ENGINEERING
IN
INFORMATION SCIENCE AND ENGINEERING

BY

VIJAYA S RAO	**4CB17IS058**
KANNIKA V PRABHU	**4CB17IS020**
KISHAN KUMAR SHETTY	**4CB17IS060**
MONTEIRO AVELINO ANIL	**4CB17IS027**

Under the Guidance of

Ms. GEETHALAXMI

Assistant Professor, Dept of IS&E

Canara Engineering College
Department of Information Science and Engineering
Benjanapadavu -574219

2020-21

ABSTRACT

This project is focused on face mask detection ,we will discuss our two-phase COVID-19 face mask detector, detailing how our computer vision/deep learning pipeline will be implemented. From there, we'll review the dataset we'll be using to train our custom face mask detector. Will then show you how to implement a Python script to train a face mask detector on our dataset using Keras and TensorFlow. We'll use this Python script to train a face mask detector and review the results. Given the trained COVID-19 face mask detector, will proceed to implement two more additional Python scripts used to Detect face masks in real-time video streams.We will wrap up the post by looking at the results of applying our face mask detector.

In order to effectively prevent the spread of COVID19 virus, almost everyone wears a mask during corona virus epidemic. This almost makes conventional facial recognition technology ineffective in many cases, such as community access control, face access control, facial attendance, facial security checks at train stations, etc. Therefore, it is very urgent to improve the recognition performance of the existing face recognition technology on the masked faces. Most current advanced face recognition approaches are designed based on deep learning, which depend on a large number of face samples.

ACKNOWLEDGEMENT

A successful and satisfactory completion of any significant task is the outcome of in valuable aggregate combination of different people in radial direction explicitly and implicitly. We have been lucky to have received a lot of help and support from our lecturers during the making of this project, we would therefore take the opportunity to thank and express gratitude to all those without whom the completion of our project would not be possible.

We are indebted to our **Principal Dr. Ganesh V. Bhat** and management of Canara Engineering College for providing an environment with all facilities that helped us in completing our major project.

We are extremely grateful to **Dr. Sumathi Pawar, Head of Information Science and Engineering Department**, for her valuable suggestion and for providing best facilities for creative work, guidance and encouragement.

We are thankful to our Project Coordinator **Prof. Sadhana B,** Assistant Professor, Department of Information Science and Engineering and Guide **Prof. Geetha Laxmi,** Assistant Professor ,Project Guide, Information Science and Engineering Department, who has been a source of inspiration. She has been enthusiastic in giving her opinions and critical reviews.

We thank all teaching and non-teaching Staff of Department of Information Science and Engineering for their support and help.

Above all we thank our parents without whose blessings; we would not have been able to accomplish our goal. We would like to thank our friends and classmates for their help and wishes for successful completion of this work.

Project Associates

Vijaya S Rao (4CB17IS058)

Kannika V Prabhu (4CB17IS020)

Kishan Kumar Shetty (4CB17IS060)

Monteiro Avelino Anil (4CB17IS027)

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Face Detection has evolved as a very popular problem in Image processing and Computer Vision. Many new algorithms are being devised using convolutional architectures to make the algorithm as accurate as possible. These convolutional architectures have made it possible to extract even the pixel details. We aim to design a binary face classifier which can detect any face present in the frame irrespective of its alignment. We present a method to generate accurate face segmentation masks from any arbitrary size input image. Beginning from the RGB image of any size, the method uses Predefined Training Weights of VGG - 16 Architecture for feature extraction. Training is performed through Fully Convolutional Networks to semantically segment out the faces present in that image. Gradient Descent is used for training while Binomial Cross Entropy is used as a loss function. Further the output image from the FCN is processed to remove the unwanted noise and avoid the false predictions if any and make bounding box around the faces. Furthermore, proposed model has also shown great results in recognizing non-frontal faces. Along with this it is also able to detect multiple facial masks in a single frame. Experiments were performed on Multi Parsing Human Dataset obtaining mean pixel level accuracy of 93.884 % for the segmented face masks.

1.1 PURPOSE

- Reduces the time for monitoring people when compared to manual inspection.
- This can also be used in places where laborers are hard to find.
- The project also helps in order to collect more data set for the future benefit.

1.2 PROBLEM STATEMENT

Main aim of the project here is to predict people wearing masks or not wearing them, given an image or a video. It is an object detection and classification problem with 2 different classes(Mask and Without Mask). Another challenge is to train object detection models capable of identifying the location of masked faces in an image as well as the location of unmasked faces in the image. These detectors should be robust to noise and provide as little room as possible to accommodate false positives for masks due to the potentially dire consequences that they would lead to. Ideally, they should be fast enough to work well for real-world applications, something we hope to focus on in future rounds of the competition.

1.3 SCOPE

Face Mask Detection Platform uses Artificial Network to recognize if a user is not wearing a mask. The app can be connected to any existing or new IP mask detection cameras to detect people without a mask. App users can also add faces and phone numbers to send them an alert in case they are not wearing a mask.

1.4 OBJECTIVE

- To identify the person on image/video stream wearing face mask with the help of computer vision and deep learning algorithm.
- To create a safe and easy method of monitoring people.
- To keep the track of people in an institution in order to keep them safe.
- To ensure effective use of deep neural networks in identifying different types of masks.

1.5 EXPECTED OUTCOMES

- The system must be able to track anyone in an institution not wearing a mask.
- An acknowledgment should be forwarded to the in charge of that person and himself requesting to wear a mask.

CHAPTER 2

LITERATURE SURVEY

I. In Shashi Yadav's "Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID19 Safety Guidelines Adherence" (2020), Wearing face masks and following safe social distancing are two of the enhanced safety protocols need to be followed in public places in order to prevent the spread of the virus. To create safe environment that contributes to public safety, we propose an efficient computer vision based approach focused on the real-time automated monitoring of people to detect both safe social distancing and face masks in public places by implementing the model on raspberry pi4 to monitor activity and detect violations through camera. After detection of breach, the raspberry pi4 sends alert signal to control centre at state police headquarters and also give alarm to public. In this proposed system modern deep learning algorithm have been mixed with geometric techniques for building a robust modal which covers three aspects of detection, tracking, and validation. Thus, the proposed system favors the society by saving time and helps in lowering the spread of corona virus. It can be implemented effectively in current situation when lockdown is eased to inspect persons in public gatherings, shopping malls, etc. Automated inspection reduces manpower to inspect the public and also can be used in any place.

II. In Zhongyuan Wang's "Masked Face Recognition Dataset and Application" (2020), In order to effectively prevent the spread of COVID19 virus, almost everyone wears a mask during coronavirus epidemic. This almost makes conventional facial recognition technology ineffective in many cases, such as community access control, face access control, facial attendance, facial security checks at train stations, etc. Therefore, it is very urgent to improve the recognition performance of the existing face recognition technology on the masked faces. Most current advanced face recognition approaches are designed based on deep learning, which depend on a large number of face samples. However, at present, there are no publicly available masked face recognition datasets. To this end, this work proposes three types of masked face datasets, including Masked Face Detection Dataset (MFDD), Real-world Masked Face Recognition Dataset (RMFRD) and Simulated Masked Face Recognition Dataset (SMFRD). Among them, to the best of our knowledge, RMFRD is currently the world’s largest real-world masked face dataset. These datasets are freely available to industry and academia, based on which various applications on masked faces can be developed.

III. In Amit Chavda's "Multi-Stage CNN Architecture for Face Mask Detection"(2020), Studies have proved that wearing a face mask significantly reduces the risk of viral transmission as well as provides a sense of protection. However, it is not feasible to manually track the implementation of this policy. Technology holds the key here. We introduce a Deep Learning based system that can detect instances where face masks are not used properly. Our system consists of a dual stage Convolutional Neural Network (CNN) architecture capable of detecting masked and unmasked faces and can be integrated with pre-installed CCTV cameras. This will help track safety violations, promote the use of face masks, and ensure a safe working environment.

IV. In Shiming Ge's "Detecting Masked Faces in the Wild with LLE-CNNs", Detecting faces with occlusions is a challenging task due to two main reasons: 1) the absence of large datasets of masked faces, and 2) the absence of facial cues from the masked regions. To address these two issues, this paper first introduces a dataset, denoted as MAFA, with 30, 811 Internet images and 35, 806 masked faces. Faces in the dataset have various orientations and occlusion degrees, while at least one part of each face is occluded by mask. Based on this dataset, we further propose LLE-CNNs for masked face detection, which consist of three major modules. The Proposal module first combines two pre-trained CNNs to extract candidate facial regions from the input image and represent them with high dimensional descriptors. After that, the Embedding module is incorporated to turn such descriptors into a similarity-based descriptor by using locally linear embedding (LLE) algorithm and the dictionaries trained on a large pool of synthesized normal faces, masked faces and non-faces. In this manner, many missing facial cues can be largely recovered and the influences of noisy cues introduced by diversified masks can be greatly alleviated. Finally, the Verification module is incorporated to identify candidate facial regions and refine their positions by jointly performing the classification and regression tasks within a unified CNN. Experimental results on the MAFA dataset show that the proposed approach remarkably outperforms 6 state-of-the-arts by at least 15.6%.

V. In Kaihan Lin's "Face Detection and Segmentation Based on Improved Mask R-CNN"(2020), Deep convolutional neural networks have been successfully applied to face detection recently. Despite making remarkable progress, most of the existing detection methods only localize each face using a bounding box, which cannot segment each face from the background image simultaneously. To overcome this drawback, we present a face detection and segmentation method based on improved Mask R-CNN, named G-Mask, which incorporates face detection and segmentation into one framework aiming to obtain more fine-grained information of face. Specifically, in this proposed method, ResNet-101 is utilized to extract features.

used as the bounding box loss function to improve the detection accuracy. Compared with Faster R-CNN, Mask R-CNN, and Multitask Cascade CNN, the proposed G-Mask method has achieved promising results on FDDB, AFW, and WIDER FACE benchmarks.

VI. In Harihara Santosh Dadi's "Improved Face Recognition Rate Using HOG Features and SVM Classifier", HOG features and SVM classifier based face recognition algorithm is introduced. This proposed algorithm is compared with standard Eigen feature based PCA algorithm. Results show that the proposed algorithm is having an improved face recognition rate of 8.75% on ORL database. The proposed algorithm is also verified on seven other face data sets. Results show that the proposed algorithm outperforms when compared with PCA algorithm for all the datasets.

VII. In Vinitha's "COVID-19 FACEMASK DETECTION WITH DEEP LEARNING AND COMPUTER VISION" (2020), The corona virus epidemic has given rise to an extraordinary degree of worldwide scientific cooperation. Artificial Intelligence (AI) based on Machine learning and Deep Learning can help to fight Covid-19 in many ways. Machine learning allows researchers and clinicians evaluate vast quantities of data to forecast the distribution of COVID-19,to serve as an early warning mechanism for potential pandemics, and to classify vulnerable populations. The provision of healthcare needs funding for emerging technology such as artificial intelligence, IoT, big data and machine learning to tackle and predict new diseases. In order to better understand infection rates and to trace and quickly detect infections, the AI's power is being exploited to address the Covid-19 pandemic. People are forced by laws to wear face masks in public in many countries. These rules and laws were developed as an action to the exponential growth in cases and deaths in many areas. However, the process of monitoring large groups of people is becoming more difficult. The monitoring process involves the detection of anyone who is not wearing a face mask. Here we introduce a mask face detection model that is based on computer vision and deep learning. The proposed model can be integrated with surveillance cameras to impede the COVID-19 transmission by allowing the detection of people who are wearing masks not wearing face masks. The model is integration between deep learning and classical machine learning techniques with opencv, tensor flow and keras. We have used deep transfer leering for feature extractions and combined it with three classical machine learning algorithms. We introduced a comparison between them to find the most suitable algorithm that achieved the highest accuracy and consumed the least time in the process of training and detection.

VIII G. Jignesh Chowdary's "In Face Mask Detection using Transfer Learning of InceptionV3" (2020), The world is facing a huge health crisis due to the rapid transmission of coronavirus (COVID-19). Several guidelines were issued by the World Health Organization (WHO) for protection against the spread of coronavirus. According to WHO, the most effective preventive measure against COVID-19 is wearing a mask in public places and crowded areas. It is very difficult to monitor people manually in these areas. In this paper, a transfer learning model is proposed to automate the process of identifying the people who are not wearing mask. The proposed model is built by fine-tuning the pre-trained state-of-the-art deep learning model, InceptionV3. The proposed model is trained and tested on the Simulated Masked Face Dataset (SMFD). Image augmentation technique is adopted to address the limited availability of data for better training and testing of the model. The model outperformed the other recently proposed approaches by achieving an accuracy of 99.9% during training and 100% during testing.

SUMMARY OF LITERATURE SURVEY:

Sl.No	Title	Methodology/ Algorithm	Advantage	Drawbacks
1	Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID19 Safety Guidelines Adherence	Deep learning algorithm and a computer vision	Logistic regression, Random forest	A significant disadvantage of DBNs is that they do not account for the two-dimensional structure of an input image, which may significantly affect their performance and applicability in computer vision and multimedia analysis problems.

2.	Masked Face Recognition Dataset and Application	Masked Face Detection Dataset (MFDD) Real-world Masked Face Recognition Dataset (RMFRD) Simulated Masked Face Recognition Dataset (SMFRD)	The accuracy of masked face recognition is about 85%	Memory based. This client dataset advantage can also be a disadvantage. Because client datasets reside in RAM, their size is limited by the amount of available RAMSingle user. Client datasets are inherently single-user datasets because they are kept in RAM.
3.	Multi-Stage CNN Architecture for Face Mask Detection	KNN [k-nearest algorithm] Linear Spectral Pairs-vector quantization	The resultant system exhibits high performance and has the capability to detect face masks in images with multiple faces over a wide range of angles.	1)Accuracy depends on the quality of the data. 2)With large data, the prediction stage might be slow. 3)Sensitive to the scale of the data and irrelevant features. 4)Require high memory – need to store all of the training data. 5)Given that it stores all of the training, it can be computationally expensive
4.	Detecting Masked Faces in the Wild	Locally linear embedding(LLE)	Experimental results on the MAFA dataset show that the proposed approach remarkably out-performs 6	One of the drawbacks of

	with LLE-CNNs	algorithm	state-of-the-arts by at least 15.6%	LLE is a lack of generalization when new points are to be added, i.e., in this case, wehave to rerun LLE on pooled (both old and new) data inorder to obtain the embeddings.
5.	Face Detection and Segmentation Based on Improved Mask R-CNN	The input of the nets is Short Time Fourier Transform(STFT) magnitude spectrum, a stack CNN module used as the feature extractor to learn mid and high level features from the spectrograms	The proposed framework is able to detect faces correctly while also precisely segmenting each face in an image.	1.Data requirements leadingto overfitting & underfitting. 2. Parameter-to-memory requirements. 3.Non-expressive learning 4.Non-expressive logics 5. Parameter tuning requirements 6. Computationally expensive
6.	Improved Face Recognition Rate Using HOG	face recognition algorithm.	This proposed algorithm is compared with standard Eigen feature based PCA	HOG shows good performance for human detection .

	Features and SVM Classifier		algorithm. Results show that the proposed algorithm is having an improved face recognition rate of 8.75% on ORL database.	However, it has a disadvantage that is very sensitive to image rotation. Therefore, HOG is not good choice for classification of textures or objects which can often be detected as rotated image.
7.	COVID-19 FACEMASK DETECTION WITH DEEP LEARNING AND COMPUTER VISION	. Decision trees, NB classifiers, linear SVMs, DNNs, and RNNs.	This specified model could be used as a use case for edge analytics. Achieve state of art result on a public face mask dataset.	1)If a categorical variable has a category in the test dataset, which was not observed in training dataset, then the model will assign a 0 (zero) probability and will be unable to make a prediction. 2)Neural networks usually require much more data than traditional machine learning algorithms, as in at least thousands if not millions of

				labelled samples.
8.	In Face Mask Detection using Transfer Learning of Inception V3	Decision tree induction algorithm, automatic chord transcription algorithm. HOG and SVM	The proposed transfer learning model achieved accuracy and specificity of 99.92%,99.9% during training, and 100%, 100% during testing on the SMFD dataset.	Small change in the data can lead to a large change in the structure of theoptimal decision tree. They are often relatively inaccurate.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

3.1 DEFINITIONS, ACRONYMS AND ABBREVIATIONS:

- **LAN** : Local Area Network
- **DFD** : Data Flow Diagram
- **SRS** : Software Requirement Analysis and Specifications.
- **SQL** : Structured Query Language
- **RAM** : Random Access Memory
- **ASP** : Active Server Page
- **HTTP** : HyperText Transfer Protocol

3.2 DESIGN AND IMPLEMENTATION CONSTRAINTS:

- System should allow the user to get the information from trained dataset.
- User has to register &login to use our system.
- Admin of the system is responsible for managing dataset.
- Software development life cycle: Here iterative process model is used in the project.
- It is assumed that valid data is input by the user to get an accurate result.

3.3 EXTERNAL INTERFACE REQUIREMENTS:

It specifies all the interface of the system: to the people, other system, hardware and other system.

3.3.1 USER INTERFACES:

- The user can access the site through a web browser.
- Home page, which has links to other pages.
- He can navigate to the various icons and view the products on the system.
- A start validation is provided to the login form. On successful validation, the permission to use the system is provided.

3.4 PERFORMANCE REQUIREMENTS:

The performance of the overall system should be faster and error free, with built in

- Error checking and correction facilities.
- In order to run this application, we require an Internet with minimum 56 kbps bandwidth.

3.4.1 DESIGN CONSTRAINTS:

- Requires to specify the information for all the mandatory fields
- The application shall have a relational database.
- The application shall be implemented using PyCharm, Django and TensorFlow.
- The application shall display error messages to the user when an error is detected.

3.4.2 HARDWARE REQUIREMENTS

1.Laptop with Intel i5 Processor and 500GB Hard disk.

3.4.3 SOFTWARE REQUIREMENTS

1.Dataset with images of people with mask and without mask.
2.Python3 IDE
3.Deep Learning Modules like Tensorflow & Keras
4.Machine Learning Modules like Numpy, Pandas and Sci-Kit Learn
5.Python Flask.
6.Browser to run Application.

CHAPTER 4

DESIGN

4.1 ABSTRACT DESIGN

4.1.1 ARCHITECTURE DIAGRAM:

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components. There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc. Architecture is a coherent set of concepts for a structure. These concepts are often visualized at four levels of abstraction.

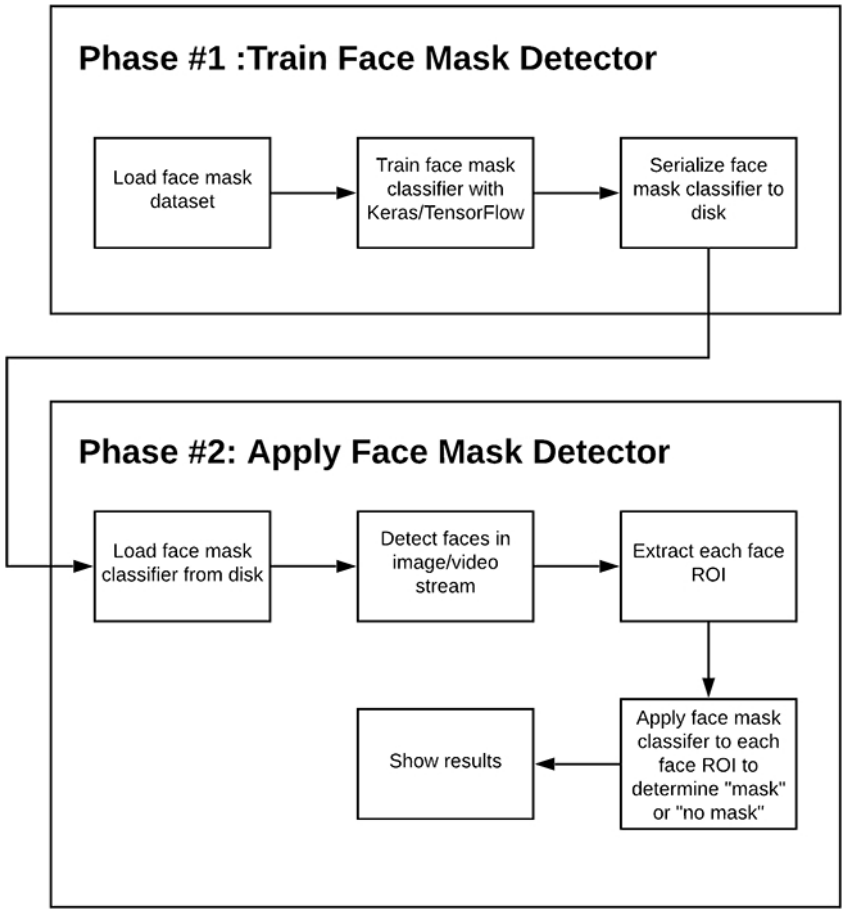

Fig 4.1.1 Architecture diagram

4.1.2 USE CASE:

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State char) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analysed to gather its functionalities, use cases are prepared and actors are identified.

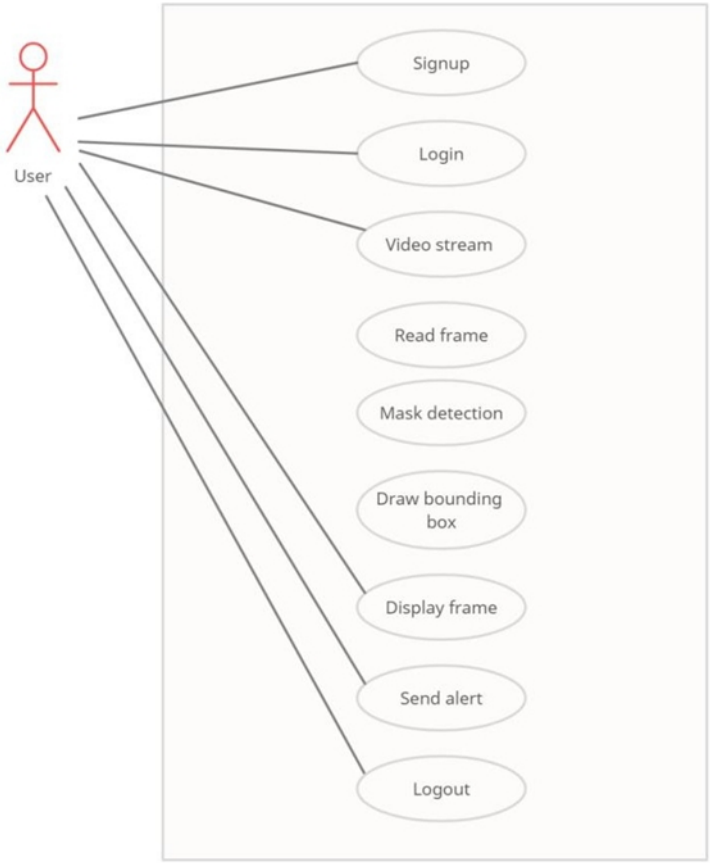

Fig 4.1.2 Use Case diagram

4.2 FUNCTIONAL REQUIREMENTS:

4.2.1 MODULAR DESIGN DIAGRAM

A modular design can be characterized by functional partitioning into discrete scalable and reusable modules, rigorous use of well-defined modular interfaces, and making use of industry standards for interfaces. In this context modularity is at the component level, and has a single dimension, component slot ability. A modular system with this limited modularity is generally known as a platform system that uses modular components.

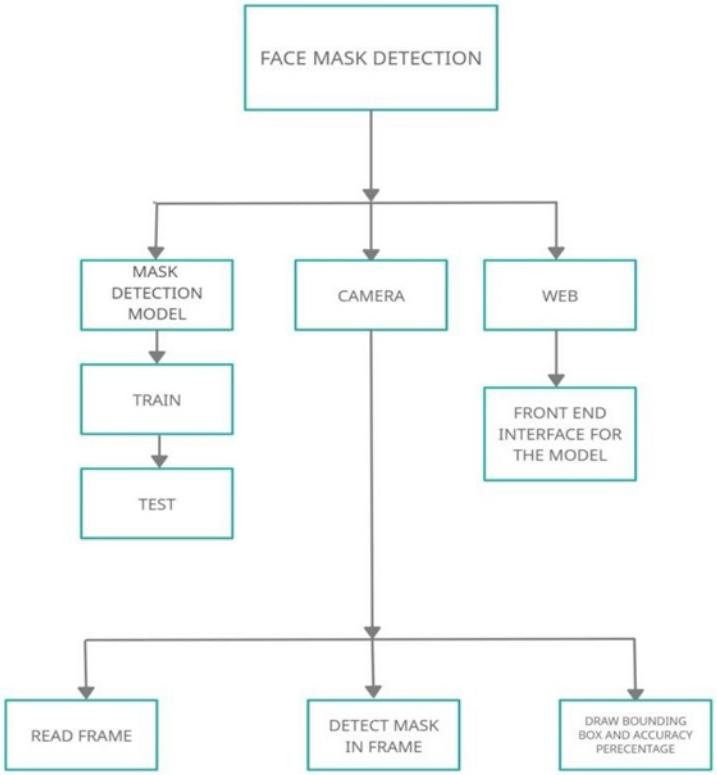

Fig 4.2.1 Modular design

4.2.2 DATA FLOW DIAGRAM:

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing. It is common practice for a designer to draw a context- level DFD first which shows the interaction between the system and outside entities. This context-level DFD is then "exploded" to show more detail of the system being modeled.

A DFD represents flow of data through a system. Data flow diagrams are commonly used during problem analysis. It views a system as a function that transforms the input into desired output. A DFD shows movement of data through the different transformations or processes in the system.

Dataflow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to restock how any system is developed can be determined through a dataflow diagram. The appropriate register saved in database and maintained by appropriate authorities

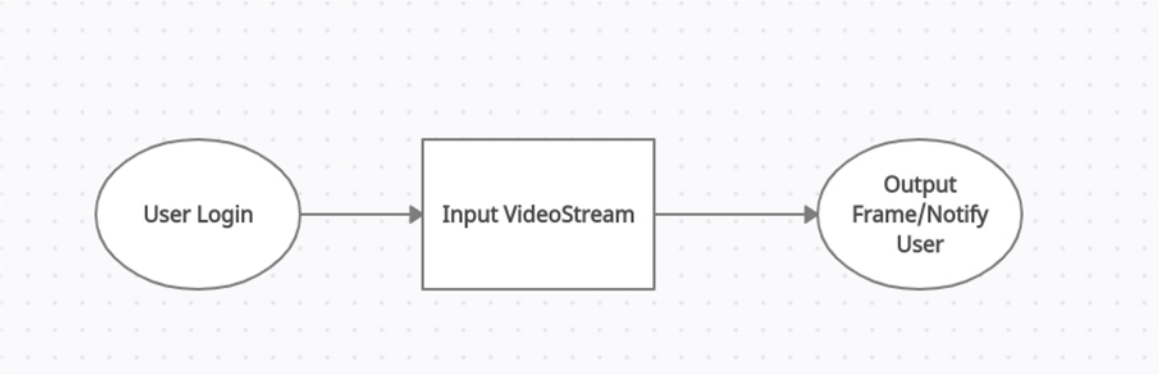

Fig 4.2.2 Level 0 DFD-Frame Extraction of the proposed system

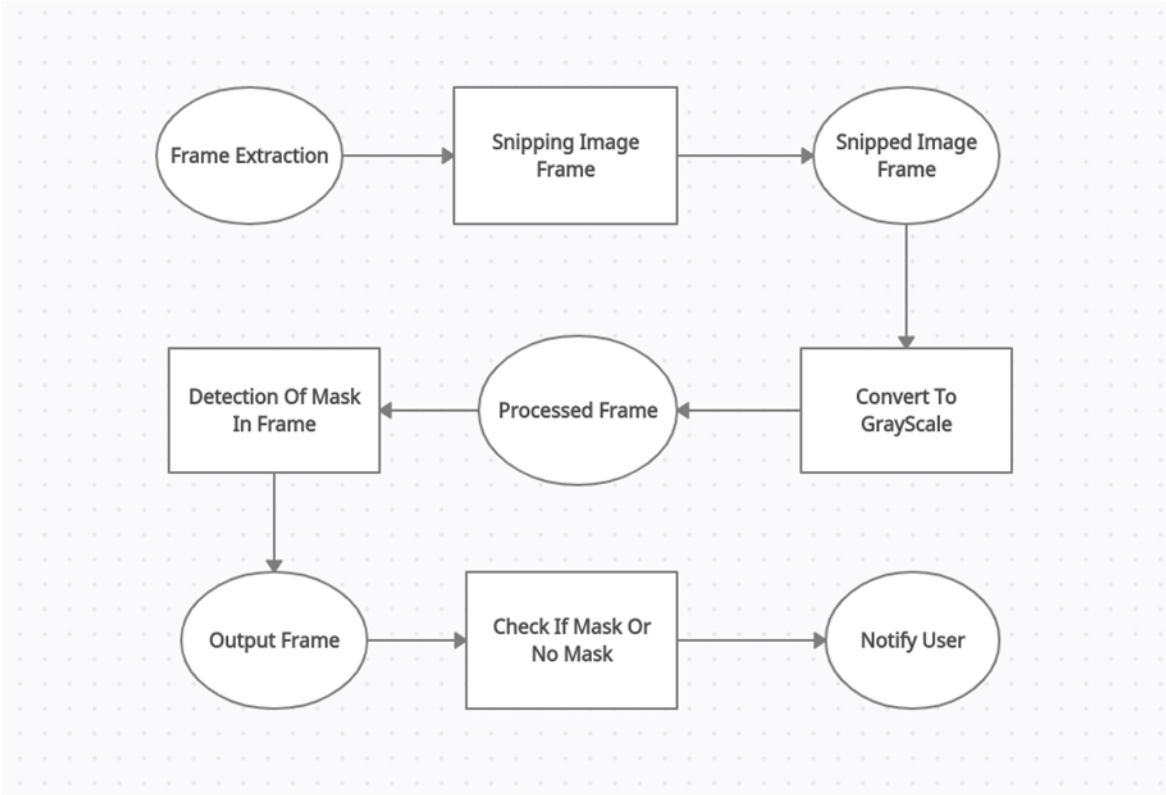

Fig 4.2.2.1 Level 1 DFD-Mask Detection of the proposed system

Phase 1 - Face Detector

A face detector acts as the first stage of our system. A raw RGB image is passed as the input to this stage. The face detector extracts and outputs all the faces detected in the image with their bounding box coordinates. The process of detecting faces accurately is very important for our architecture. Training a highly accurate face detector needs a lot of labeled data, time, and compute resources. For these reasons, we selected a pre-trained model trained on a large dataset for easy generalization and stability in detection.

Phase 2 - Face Mask Classifier

The second stage of our system is a face mask classifier. This stage takes the processed ROI from the Intermediate Processing Block and classifies it as either RMFRD images were biased towards Asian faces. Thus, masked images from the Larxel (Kaggle) were added to the dataset to eliminate this bias. RMFRD contains images for unmasked faces as well. However, as mentioned before, they were heavily biased towards Asian faces. Hence, we decided not to use these images. The Flickr-Faces-HQ (FFHQ) dataset introduced by (Karras et al., 2019) was used for unmasked images. Our dataset also includes images of improperly worn face masks or hands covering the face, which get classified as nonmasked faces.

4.2.2 SEQUENCE DIAGRAM:

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

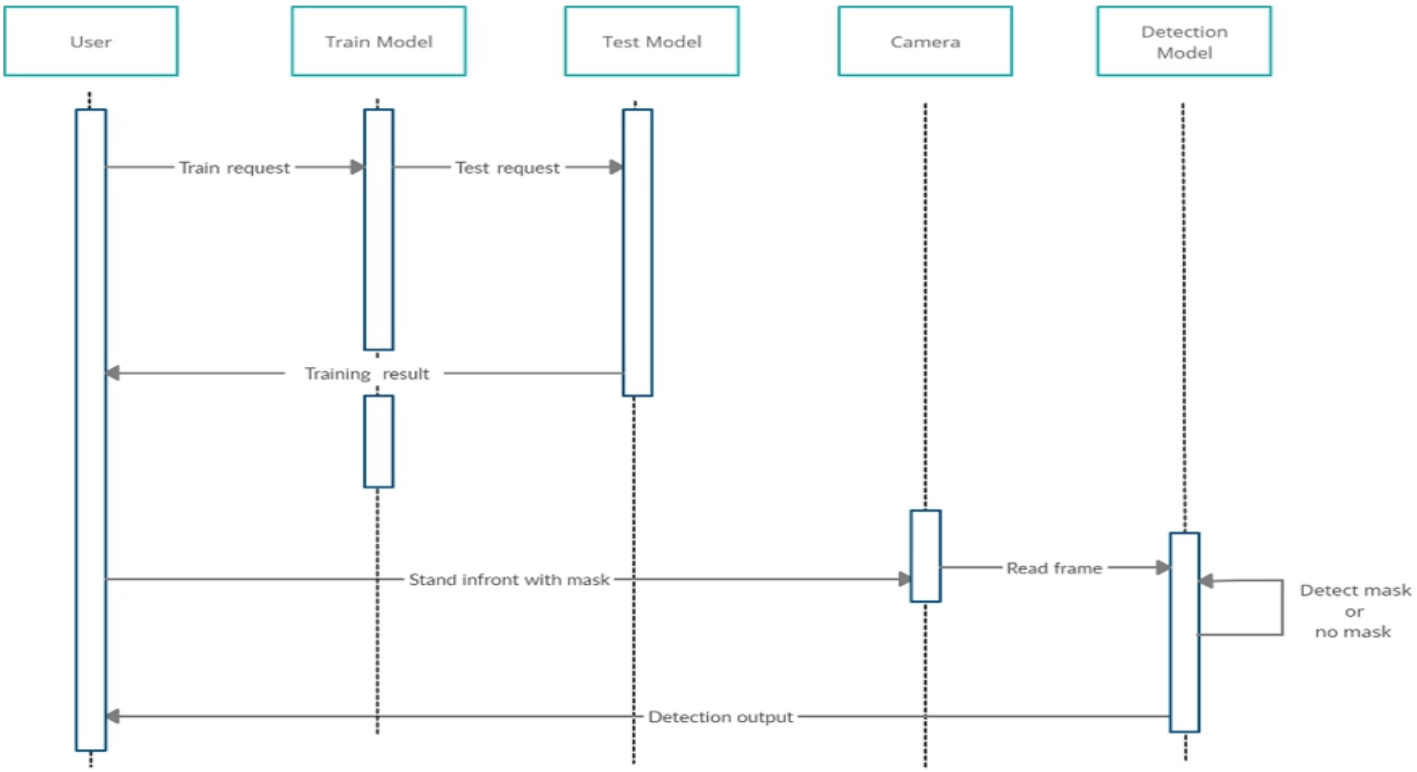

Fig 4.2.3 sequence diagram

4.3CONTROL FLOW DIAGRAM:

4.3.1 COMPLETE CONTROL FLOW DIAGRAM:

It is common practice to draw the context-level data flow diagram first, which shows the interaction between the system and external agents which acts as data source and data sinks. On the context diagram the system's interactions with the outside world are modelled purely in terms of data flows across the system boundary. The context diagram shows the entire systems the single process, and gives no clues as to its internal organization.

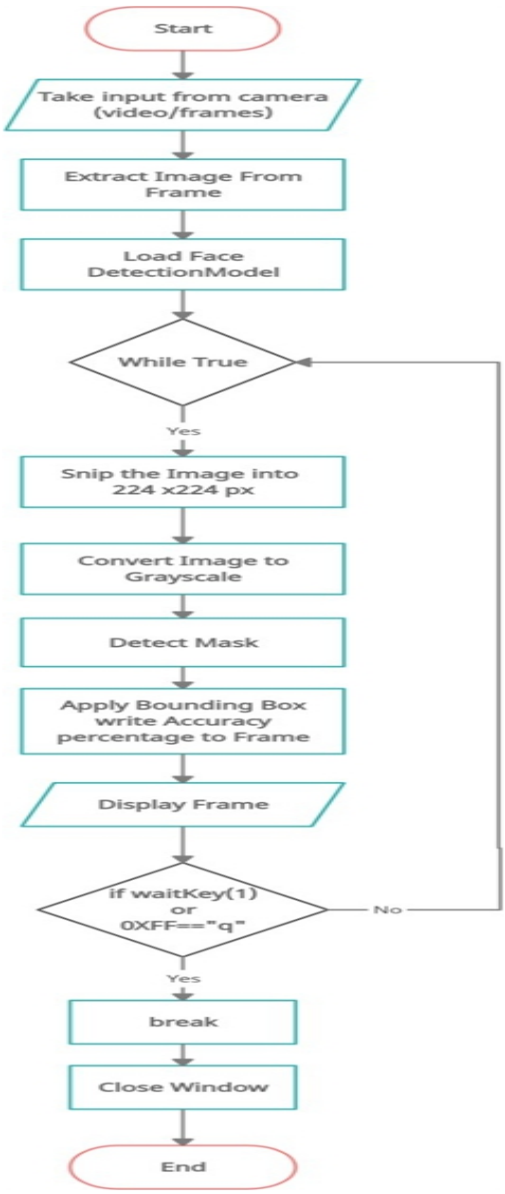

Fig 4.3.1 Control flow diagram

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Algorithm to detect mask video

Step 1: To determine whether the camera is turned on by pressing the button.

Step 2: If a request, such as a POST, is made, the display is turned on.

Step 3: To detect a face, get the frame dimension and use it to build a box.

Step 4: Move the box through the network to acquire a face detection.

Step 5: Create a catalogue of faces and a network to predict face masks.

Step 6: To move through every detection and filter out weak detection, use a for loop.

Step 7: Calculate the area of a box using x and y values.

Step 8: Check that the border box is inside the frame's dimensions.

Step 9: Add the face and boundary boxes to the appropriate lists.

Step 10: Only make predictions if at least one face is found.

Step 11: Loop through frames of a video stream to determine face in the frame are wearing a mask.

Step 12: A loop was discovered over a face mask.

Step 13: On the output frame, show a rectangle boundary box.

Step 14: Display the output frame.

We make use of face mask detection algorithm which simplifies the data set from maser to in masked. The user is shown with a login page which show the credentials such as login and password.The details is stored in database who ever logs in the system. After the login page python script with its later run and connected with the camera. A boundary is drawn against the user face. The dimensions are mention in the code identifies the an object and detect whether a person is wearing a mask or not.

5.2 CODE DETAILS AND CODE EFFECIENCY

Code details

Python code for training ML model

Training the ML model with a pair of datasets with and without mask.

```
# construct the argument parser and parse the arguments
        ap = argparse.ArgumentParser()
        ap.add_argument("-d", "--dataset",type=str,default="dataset",
                help="path to input dataset")
        ap.add_argument("-p", "--plot", type=str, default="plot.png",
                help="path to output loss/accuracy plot")
        ap.add_argument("-m", "--model", type=str,
                default="mask_detector.model",
                help="path to output face mask detector model")
        args = vars(ap.parse_args())

# initialize the initial learning rate, number of epochs to train for,
# and batch size
                INIT_LR = 1e-4
                EPOCHS = 20
                BS = 32

                # grab the list of images in our dataset directory, then initialize
                # the list of data (i.e., images) and class images
                print("[INFO] loading images...")
                imagePaths = list(paths.list_images(args["dataset"]))
                data = []
                labels = []

# loop over the image paths
                for imagePath in imagePaths:
                        # extract the class label from the filename
                        label = imagePath.split(os.path.sep)[-2]

                        # load the input image (224x224) and preprocess it
                        image = load_img(imagePath, target_size=(224, 224))
                        image = img_to_array(image)
                        image = preprocess_input(image)

                        # update the data and labels lists, respectively
                        data.append(image)
                        labels.append(label)
```

```
# convert the data and labels to NumPy arrays
		data = np.array(data, dtype="float32")
		labels = np.array(labels)

# perform one-hot encoding on the labels
		lb = LabelBinarizer()
		labels = lb.fit_transform(labels)
		labels = to_categorical(labels)

# partition the data into training and testing splits using 75% of
# the data for training and the remaining 25% for testing
		(trainX, testX, trainY, testY) = train_test_split(data, labels,
			test_size=0.20, stratify=labels, random_state=42)

# construct the training image generator for data augmentation
		aug = ImageDataGenerator(
			rotation_range=20,
			zoom_range=0.15,
			width_shift_range=0.2,
			height_shift_range=0.2,
			shear_range=0.15,
			horizontal_flip=True,
			fill_mode="nearest")

# load the MobileNetV2 network, ensuring the head FC layer sets are
# left off
		baseModel = MobileNetV2(weights="imagenet", include_top=False,
			input_tensor=Input(shape=(224, 224, 3)))

# construct the head of the model that will be placed on top of the
# the base model
		headModel = baseModel.output
		headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
		headModel = Flatten(name="flatten")(headModel)
		headModel = Dense(128, activation="relu")(headModel)
		headModel = Dropout(0.5)(headModel)
		headModel = Dense(2, activation="softmax")(headModel)

# place the head FC model on top of the base model (this will become
# the actual model we will train)
		model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
		for layer in baseModel.layers:
			layer.trainable = False
```

```
# compile our model
		print("[INFO] compiling model...")
		opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
		model.compile(loss="binary_crossentropy", optimizer=opt,
			metrics=["accuracy"])

# train the head of the network
		print("[INFO] training head...")
		H = model.fit(
			aug.flow(trainX, trainY, batch_size=BS),
			steps_per_epoch=len(trainX) // BS,
			validation_data=(testX, testY),
			validation_steps=len(testX) // BS,
			epochs=EPOCHS)

# make predictions on the testing set
		print("[INFO] evaluating network...")
		predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
		predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
		print(classification_report(testY.argmax(axis=1), predIdxs,
			target_names=lb.classes_))

# serialize the model to disk
		print("[INFO] saving mask detector model...")
		model.save(args["model"], save_format="h5")

# plot the training loss and accuracy
N = EPOCHS
		plt.style.use("ggplot")
		plt.figure()
		plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
		plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
		plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
		plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
		plt.title("Training Loss and Accuracy")
		plt.xlabel("Epoch #")
		plt.ylabel("Loss/Accuracy")
		plt.legend(loc="lower left")
		plt.savefig(args["plot"])
```

Python code for detect_mask_video

Here when the code is executed in later run it detects whether the face is masked or unmasked.

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

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

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

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

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

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

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

```
# return a 2-tuple of the face locations and their corresponding
            # locations
            return (locs, preds)
        # construct the argument parser and parse the arguments
        ap = argparse.ArgumentParser()


 ap.add_argument("-f", "--face", type=str,
                    default="face_detector",


   help="path to face detector model directory")
        ap.add_argument("-m", "--model", type=str,
                    default="mask_detector.model",
                    help="path to trained face mask detector model")
        ap.add_argument("-c", "--confidence", type=float, default=0.5,
                    help="minimum probability to filter weak detections")
        args = vars(ap.parse_args())

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

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

        # initialize the video stream and allow the camera sensor to warm up
        print("[INFO] starting video stream...")

       vs = VideoStream(src=0).start()


          time.sleep(2.0)
      while True:
            # grab the frame from the threaded video stream and resize it
            # to have a maximum width of 400 pixels
            frame = vs.read()
            frame = imutils.resize(frame, width=400)
           # detect faces in the frame and determine if they are wearing a
            # face mask or not
            (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

            # loop over the detected face locations and their corresponding
            # locations
            for (box, pred) in zip(locs, preds):
```

```
    # unpack the bounding box and predictions
                (startX, startY, endX, endY) = box
                (mask, withoutMask) = pred

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

        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

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

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

            # show the output frame
            cv2.imshow("Frame", frame)
            key = cv2.waitKey(1) & 0xFF

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

        # do a bit of cleanup
        # cv2.destroyAllWindows()
        # vs.stop()
        return render_template("index.html")

    if request.form['btn_on'] == "Off":
        print("Pressed Off")
        cv2.VideoCapture(0).release()
        cv2.destroyAllWindows()
        #VideoStream(src=0).start().stop()

return render_template("index.html")


        return render_template("index.html")


        if __name__ == '__main__':
            app.run(debug=True)

        # vs = VideoStream(src=0).start()
        # time.sleep(2.0)
```

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

        # detect faces in the frame and determine if they are wearing a
        # face mask or not


   (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

        # loop over the detected face locations and their corresponding
        # locations
        for (box, pred) in zip(locs, preds):
                # unpack the bounding box and predictions


      (startX, startY, endX, endY) = box
                (mask, withoutMask) = pred

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

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

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



# show the output frame
                cv2.imshow("Frame", frame)
                key = cv2.waitKey(1) & 0xFF

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


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

CODE EFFECIENCY

Code efficiency is directly linked with algorithm efficiency and the speed of the runtime execution for software. It is the key element in ensuring high performance. The goal of code efficiency is to reduce resource consumption and completion time as much as possible with minimum risk to the business or to the operating environment.Our code is reliable, easy to use ,faster and also the code supports for face mask detection in real time as well as in recorded videos.

5.1 Testing

Software testing is the process used to help identify the correctness, completeness, security and quality of developed computer software. This includes the process of executing the program or application with the intent of finding errors. Quality is not an absolute it is value to some person. With that in mind testing can never completely establish the correctness of arbitrary comport software; testing furnishes a criticism or comparison that compares the state and behaviour of product against a specification.

Testing forms the first step in determining the errors in a program. Clearly the success of testing in revealingerrors in programs depends critically on the test cases. Because code is the only product that can be executedand whose actual behaviour can be observed, testing is the face where the errors remaining from all the previous phases must be detected.

The program to be tested is executed with the set of test cases and the output of the program for the test cases are evaluated to determine if the programming is performing as expected. Testing forms the first step in determining errors in a program. The success of testing in revealing errors in programs depends critically on the test cases

5.3.1 Testing Objective

- Testing is carried out to ensure that this software is designed according to the needs of the user.
- Testing further enhances the quality of the software and makes sure that the product is error free.
- To guarantee that all the verification and validation is done.
- Ensure that all the verification and validation is done.
- Testing is a process of executing a program with the intent of finding errors.
- A successful test case is one that uncovers an as-yet-undiscovered error.
- A good test case design is one that has a high probability of finding an undiscovered error.

- It will uncover potential errors and bug entry points in the software.
- Data collected as testing is conducted provide a good indication of software reliability and indication
- of software quality as a whole.
- It will demonstrate that software functions appear to be working according to specification that behavioral and performance requirements appear to have been met.

5.3.2 Testing principal:

All tests should be noticeable to customer requirements, test should be planned long before testing begins, the Pareto principal applies to software testing, testing should begin “in the small” and progress toward testing “in the large”, exhaustive testing is not possible and to be most effective, testing should be conducted by an independent third party.

5.3.3 Testing method

System testing is the stage of implementation. This is to check whether the system after works accurately and efficiently before live operation commences. Testing is the vital to the success of the system. The candidate system is subject to variety of testes: online response, volume, stress, recovery, security and usability tests. A series of tests are performed for the proposed system is ready for user acceptance testing.

The various type of testing on the system is:

- Unit testing
- Integrated testing
- Validation testing
- Output testing
- User acceptance testing

Unit testing:Unit testing focuses on verification effort on the smallest unit of software design module. Using the unit test plans. Prepared in the design phase of the system as a guide important control paths are tested to uncover errors within the boundary of the modules. The interfaces of each of the modules under consideration are alsotested. Boundary conditions were checked.

All independents paths were exercised to ensure that all statements in the module executed at least once and all error-handling paths were tested. Each unit was thoroughly tested to check if it might fall in any possible

situation. This testing was carried out during the programming itself. At the end of this testing phase each unit was found to be working satisfactorily as regarded to the expected out from the module.

Integration Testing:

Data can be across an interface one module can have an adverse effect on an other's Sub functions when combined may not produce the desired major function; global data structures can present problems. Integration testing is a symmetric technique for constructing tests to uncover errors associated with the interface. All modules are combined in this testing step. Then the entire program was tested as a whole.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Validation Testing:

At the culmination of integration testing software is completely assemble as a package. Interfacing errors have been uncovered and corrected and fin; series of software test-validation testing begins. Validation testing can be defined in many ways but a simple definition is that validation succeeds when software functions in manner that is reasonably expected by the consumer.

Software validation is achieved through a series of black box tests that demonstrate conformity with requirement after validation test has been conducted one of two conditions exists.

- The function or performance Characteristics confirm to specification that are accepted.
- A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of user by negotiating to establish a method for resolving deficiencies. Thus, the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

Output Testing:

After performing the validation testing the next step is output testing of the proposed system since a system is useful if it does not produce the required output in the specific format required by them tests the output. generator displayed on the system under consideration. Here the output is considered in the two ways- one is the onscreen and the other is printed format. The output formation the screen is found to be correct as the format was designed in the system design phase according to the user needs. As far as hard copies are considered it goes in terms with the user requirement. Hence output testing does not result any correction in the system.

User acceptance testing:

User acceptance of the system is a key factor for success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required.

5.3.4 Testing criteria

Testing for login:

Test case	Input	Test description	Output
1	Password is left blank	Password cannot be blank	Must Enter password
2	Invalid email entered	Valid email must be entered	Must enter proper mail id

Testing for image:

Test case	Input	Test description	Output
1	Image	Image should be in jpg format	Image will be uploaded
2	Image	Invalid format	Appropriate error message

CHAPTER 6

RESULTS

SCREENSHOTS

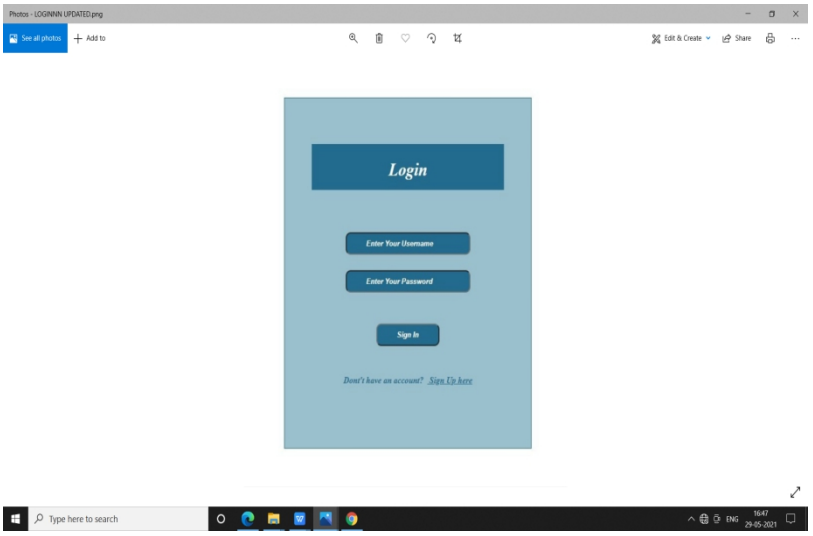

Fig 6.1 login page

We make use of face mask detection algorithm which simplifies the data set from maser to in masked. The User is shown with a login page which show the credentials such as login using email-id and password.

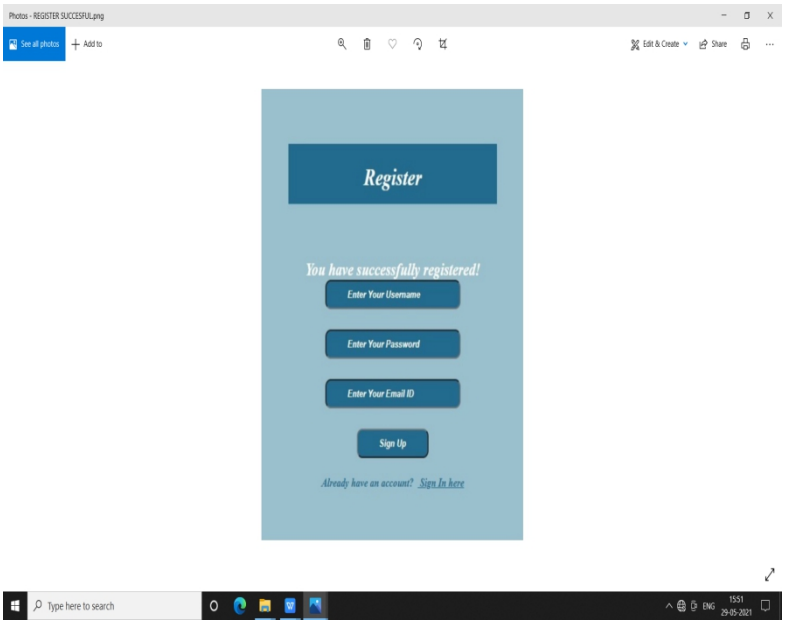

Fig 6.2 Register

From fig 6.2 ,it displays you have successfully registered. If he has already account ,registered then he can sign-in . If he is new user he has to sign up.

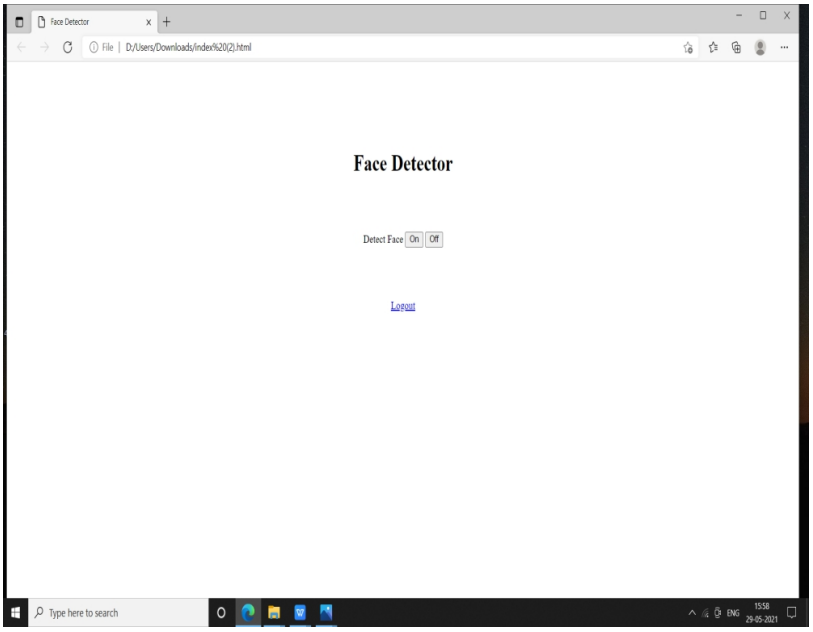

Fig 6.3 Index page

In the Fig 6.3 as shown in fig ,the index page we have the buttons on and off. We need to click on ‘on ‘ button Also we have option of logout ,it logs out from system also Python script with its later run and connected with the camera

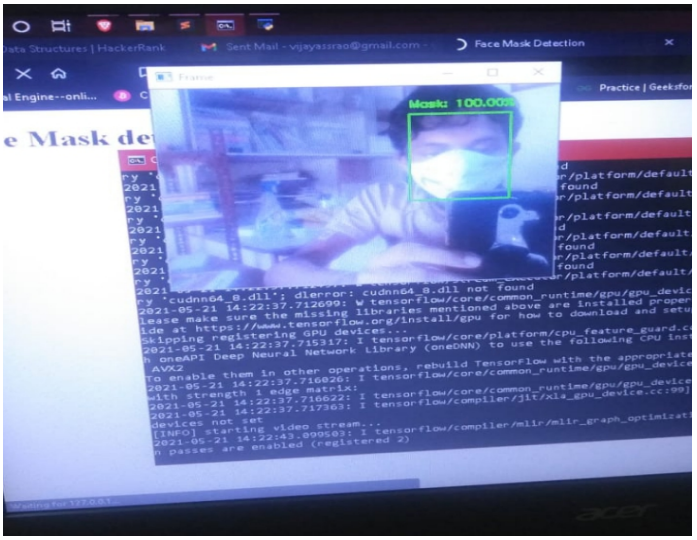

Fig 6.4 Masked face

A boundary is drawn against the user face. The dimensions are mentioned in the code identifies the an object and detect whether a person is wearing a mask or not. As shown in fig 6.4 it is a masked face. So it identifies face with mask.

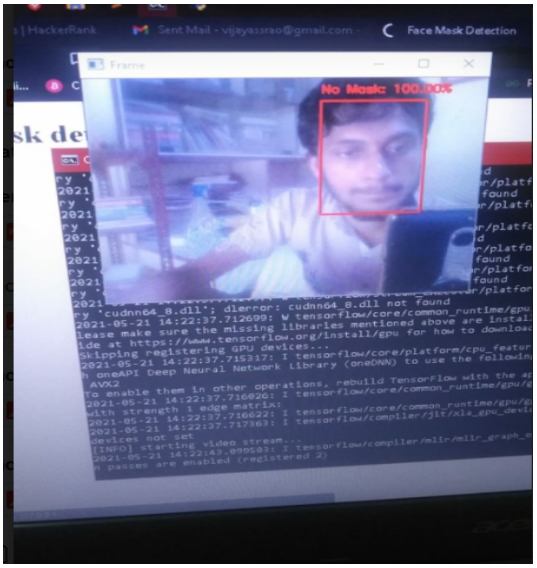

Fig 6.5 Unmasked Face

Fig 6.5 shows it is unmasked face . A boundary is drawn against the user face. The dimensions are mentioned in the code identifies an object and detect whether a person is wearing a mask or not. Now the person has not worn his mask.So it detects no mask.

Fig 6.6 Masked and Unmasked Faces

In the above figure 6.6 it shows people wearing and not wearing mask. The system detects the image whether it is masked or unmasked.

REFERENCES:

[1] Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID19 Safety Guidelines Adherence - https://www.researchgate.net/publication.

[2] Masked Face Recognition Dataset and Application - https://arxiv.org/pdf/2003.09093.pdf

[3] Multi-Stage CNN Architecture for Face Mask Detection - https://www.researchgate.net/publication.

[4] Detecting Masked Faces in the Wild with LLE-CNNs - https://openaccess.thecvf.com.

[5] Face Detection and Segmentation Based on Improved Mask R-CNN https://www.hindawi.com/journals/ddns/2020/9242917/.

[6] Improved Face Recognition Rate Using HOG Features and SVM - https://www.researchgate.net/publication.

[7] COVID-19 Facemask Detection With Deep Learning And Computer Vision.

[8] In Face Mask Detection using Transfer Learning of InceptionV3 - https://arxiv.org/abs/2009.08369