Industrial Oriented Mini Project Report (CS704PC)

on

Real Time Face Mask Detection Using Deep Learning Network

Submitted
in partial fulfilment of the requirements for
the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering
by

Sangishetty Navya Akshitha

(18261A05B0)

Under the guidance of

Ms. K. Shirisha

Assistant Professor

Department of Computer Science and Engineering
Mahatma Gandhi Institute of Technology
(Affiliated to Jawaharlal Nehru Technological University Hyderabad)
Kokapet(V), Gandipet(M), Hyderabad.
Telangana-500 075.
2021 - 2022

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

Gandipet, Hyderabad-500075, Telangana

CERTIFICATE

This is to certify that the Industrial Oriented Mini Project (CS704PC) entitled **Real Time Face Mask Detection Using Deep Learning Network** is being submitted by **Sangishetty Navya Akshitha** bearing **Roll No: 18261A05B0** in partial fulfilment of the requirements for the Award of the **Degree of Bachelor of Technology** in **Computer Science and Engineering** to **Jawaharlal Nehru University Hyderabad** is a record of bonafide work carried out under the supervision of **Ms. K. Shirisha, Asst. Professor, Dept. of CSE**.

The results in this has not been submitted to any other University or Institute for the award of any degree or diploma.

Supervisor	Head of the Department
Ms. K. Shirisha	**Dr. C. R. K. Reddy**
Assistant Professor	Professor

External Examiner

DECLARATION

I here by declare that the work reported in the Industrial Oriented Mini Project(CS704PC) titled **Real Time Face Mask Detection Using Deep Learning Network** is original and bonafide work carried out by me as a part of fulfilment for Bachelor of Technology in Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad, under the guidance of Ms. K. Shirisha , Assistant Professor , Dept. of CSE.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by me and not copied from any other source.

Sangishetty Navya Akshitha

18261A05B0

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the project.

I would like to express my sincere thanks to **Prof. K. Jaya Sankar, Professor & Principal MGIT**, for providing the work facilities in the college.

I am also extremely thankful to **Dr. C.R.K Reddy, Professor & Head of Department, Dept. of CSE**, for providing excellent infrastructure and a conducive atmosphere for completing this project successfully.

I am also extremely thankful to Project Coordinators, **Dr. A. Nagesh ,** Professor, **Dr. V. Subba Ramaiah**, Senior Assistant Professor, **Mr. A. Ratnaraju,** Assistant Professor, Dept. Of CSE, for their valuable suggestions and interest throughout the course of this project.

I would like to express my sincere gratitude and indebtedness to Project Guide, **Ms. K. Shirisha, Assistant Professor, Dept. of CSE** who has supported me throughout the project with patience and expertise.

Finally, I would like to take this opportunity to thank my family for their support throughout the work. I sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

Sangishetty Navya Akshitha

18261A05B0

INDEX

List of Figures

List of Tables

ABSTRACT

COVID-19 pandemic has rapidly affected our day-to-day lives disrupting the world trade and movements. Wearing a protective face mask has become a new normal. Soon, many public service providers will ask the customers to wear masks correctly to avail their services. Therefore, face mask detection has become a crucial task to help global society. The proposed approach uses Deep Learning approach for detection of face mask. The SSDMNV2 approach uses Single Shot Multibox Detector as a face detector and MobilenetV2 architecture as a framework for the classifier, which is very lightweight and can even be used in embedded devices to perform real-time mask detection.

This face mask identifier is least complex in structure and gives quick results and hence can be used in CCTV footage to detect whether a person is wearing a mask perfectly so that he/she does not pose any danger to others. Mass screening is possible and hence can be used in crowded places like railway stations, bus stops, markets, streets, mall entrances, schools, colleges, etc. By monitoring the placement of the face mask on the face, we can make sure that an individual wears it the right way and helps to curb the scope of the virus.

1. INTRODUCTION

1.1 Problem Definition

Building a system that can detect faces in *real-world videos* and identify if the detected faces are wearing masks or not. If you look at the people in videos captured by CCTV cameras, you can see that the faces are small, blurry, and of low resolution. People are not looking straight to the camera, and the face angles vary from time to time. These real-world videos are entirely different from the videos captured by webcam or selfie cameras, making the face mask detection problem much more difficult in practice.

This approach will first explore mask / no mask classification in webcam videos, and next, shift to the mask / no mask classification problem in real-world videos as our final goal. Our reported model can detect faces and classify masked faces from unmasked ones in webcam videos as well as real-world videos where the faces are small and blurry, and people are wearing masks in different shapes and colors.

1.2 Existing System

There are several models available to detect Face mask using deep learning techniques such as VGG-16, Res-Net 50, Inception V3 models. Most of the previous models can detect Face masks. But their accuracy is low and varying with only detecting faces in photos but not in video streams sometimes, their results are inaccurate due to the imbalanced data , take more time to train and consume large amount of resources and computation.

Disadvantages

- Input dimensions decreases by a large margin this causes the decrease in
  Neural network accuracy.
- The performance boost found from mixing images is very difficult to understand.
- Takes large number of resources and (time and computation power) to Train.

1.3 Proposed System

To predict whether a person has worn a mask correctly, the initial stage would be to train the model using a proper dataset. After training the classifier, an accurate face detection model is required to detect faces, so that the SSDMNV2 model can classify whether the person is wearing a mask or not. The approach main aim is to raise the accuracy of mask detection without being too resource heavy. For doing this task, the DNN module was used from OpenCV, which contains a 'Single Shot Multibox Detector' (SSD) object detection model with ResNet-10 as its backbone architecture. This approach helps in detecting faces in real-time, even on embedded devices. The following classifier uses a pre-trained model MobileNetV2 to predict whether the person is wearing a mask or not.

Advantages

- MobileNetV2 models are faster for the same accuracy across the entire latency spectrum.
- Single Shot Multibox Detector is a Deep Learning based technique, and it comes with more robust, more accurate and faster results than regular Haar Cascade Classifier.

1.4 System Requirements

1.4.1 Hardware Requirements

1. System: i3 or more

2. Hard Disk: 100GB

3. RAM: 2GB or more

4. Input Devices: Keyboard, Mouse

1.4.2 Software Requirements

1. Operating System: Windows 7 or more
2. Tool: Visual Studio Code
3. Software: Python 3.7

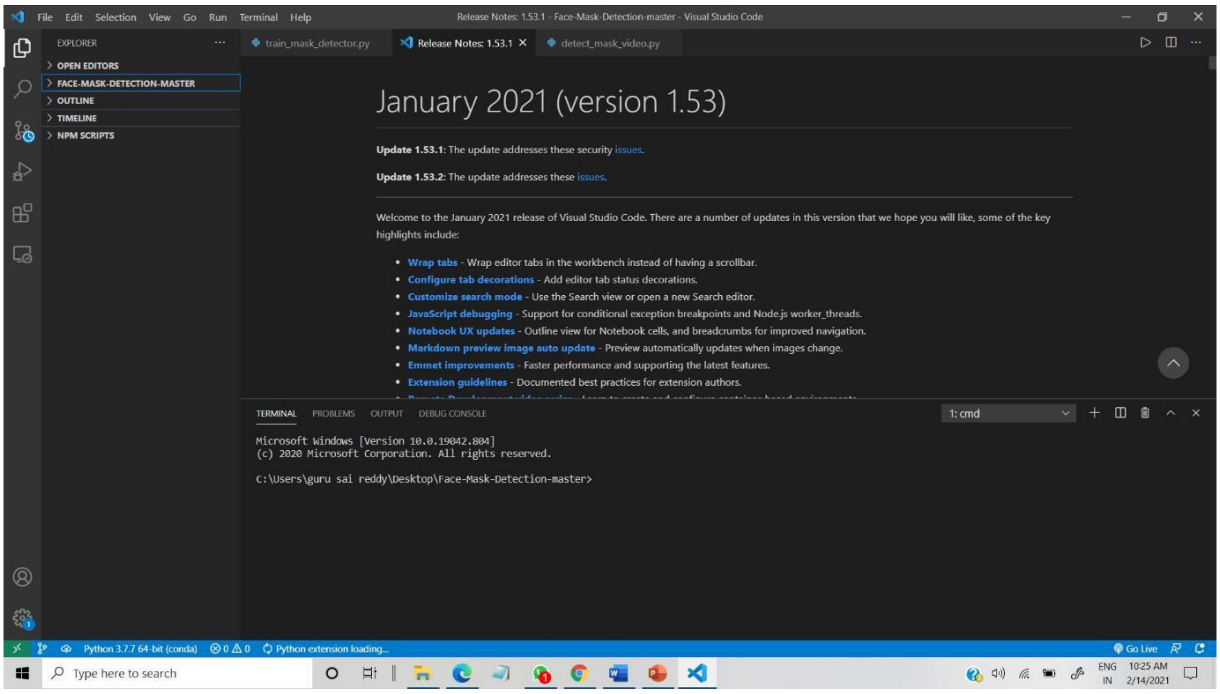
1.4.3 Software Specification

Visual Studio Code

Visual Studio Code is a free source-code editor made by Microsoft for Windows macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. Microsoft has released Visual Studio Code's source code on the Microsoft/ VS code repository of GitHub, under the permissive MIT License, while the releases by Microsoft are freeware.

In the Stack Overflow 2019 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 50.7% of 87,317 respondents reporting that they use it. Visual Studio Code was first announced on April 29, 2015, by Microsoft at the 2015 Build conference. A Preview build was released shortly thereafter.

On November 18, 2015, Visual Studio Code was released under the MIT License, having its source code available on GitHub. Extension support was also announced. On April 14, 2016, Visual Studio Code graduated from the public preview stage and was released to the Web.


File Edit Selection View Go Run Terminal Help
Release Notes: 1.53.1 - Face-Mask-Detection-master - Visual Studio Code
EXPLORER
OPEN EDITORS
FACE-MASK-DETECTION-MASTER
OUTLINE
TIMELINE
NPM SCRIPTS
train_mask_detector.py
Release Notes: 1.53.1
detect_mask_video.py
TERMINAL
PROBLEMS
OUTPUT
DEBUG CONSOLE
1: cmd
Python 3.7.7 64-bit (conda)
Python extension loading...
Go Live
Type here to search

January 2021 (version 1.53)

Update 1.53.1: The update addresses these security issues.

Update 1.53.2: The update addresses these issues.

Welcome to the January 2021 release of Visual Studio Code. There are a number of updates in this version that we hope you will like, some of the key highlights include:

- **Wrap tabs** - Wrap editor tabs in the workbench instead of having a scrollbar.
- **Configure tab decorations** - Add editor tab status decorations.
- **Customize search mode** - Use the Search view or open a new Search editor.
- **JavaScript debugging** - Support for conditional exception breakpoints and Node.js worker_threads.
- **Notebook UX updates** - Outline view for Notebook cells, and breadcrumbs for improved navigation.
- **Markdown preview image auto update** - Preview automatically updates when images change.
- **Emmet improvements** - Faster performance and supporting the latest features.
- **Extension guidelines** - Documented best practices for extension authors.

```
Microsoft Windows [Version 10.0.19042.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\guru sai reddy\Desktop\Face-Mask-Detection-master>
```

Figure 1.1: Visual Studio Code Interface

Figure 1.1 shows the Interface of Visual Studio Code

2. LITERATURE SURVEY

Table 2.1: Literature Survey for Real-Time Face Mask Detection using Deep Learning Network

S.NO	YEAR	TITLE	AUTHOR	METHOD	MERITS	DEMERITS
1.	2021	Tiny CNN Architecture for Medical Face Mask Detection	1. Puranjan Mohan 2. Aditya Jyothi Paul 3. Abhay Chirania	SqueezeNet Model	This model requires less bandwidth to export a new model from the cloud and requires less communication	This model contains relatively small amount of parameters.
2.	2021	Facial Recognition System for people with and without mask in times of COVID-19 Pandemic.	1.Jonathan S Talahua 2.Jorge Buele 3.Jose Varela Aldas	ANN based classifier	Implementation Costs is less in this model.	This model produces low accuracy compared to other methods and is not suitable for larger datasets.
3.	2020	A hybrid deep transfer learning model with machine learning methods for face mask detection.	1.Mohamed Loey 2.Gunasekaran Manogaran	Res-Net 50	Strengthen the Feature propagation by decreasing the Vanishing Gradient problem.	The performance boost found from mixing images is very difficult to understand or explain.

4.	2020	Face Mask Detection using Transfer Learning of Inception V3	1. Narinder Singh 2. G. Jignesh Chowdary	Inception V3	Transfer Learning proved to be effective . It has robust performance and is easily deployable.	Input dimensions decreases by a large margin which causes decrease in neural network's accuracy.
5.	2019	Facial Mask Detection Using Semantic Segmentation	1. Amit Verma 2. Toshan Monpal	VGG-16	This model is capable of decreasing the training time.	This model takes large amount of resources, time, and computation power to train and also it has many weight parameters and modes which are very heavy.

3. METHODOLOGY FOR REAL-TIME FACE MASK DETECTION USING DEEP LEARNING NETWORK

3.1 System Design

The major requirement for implementing this project using python programming language along with Deep learning and with python libraries. The architecture consists of Mobile Net as the backbone, it can be used for high and low computation scenarios. We are using CNN Algorithm in our proposed system.

We have four modules

1. Datasets Collecting: We collect no of data sets with face mask and without masks. we can get high accuracy depends on collecting the number of images.
2. Datasets Extracting: We can extract the features using mobile net v2 of mask and no mask sets.
3. Models Training: We will train the model using OpenCV, keras (python library).
4. Face Mask Detection: We can detect Pre-processing image and detect via live video. If people wear mask, it will permit them, if not then it will give the buzzer to wear mask to prevent them from virus transmission

3.2 UML Diagrams

Unified Modelling Language is a standard language for writing software blueprints. It can be used to visualize, specify, construct, and document the artifacts of a software-intensive system. Modelling is a proven and well-accepted engineering technique.

3.2.1 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and tells how the user handles a system.The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

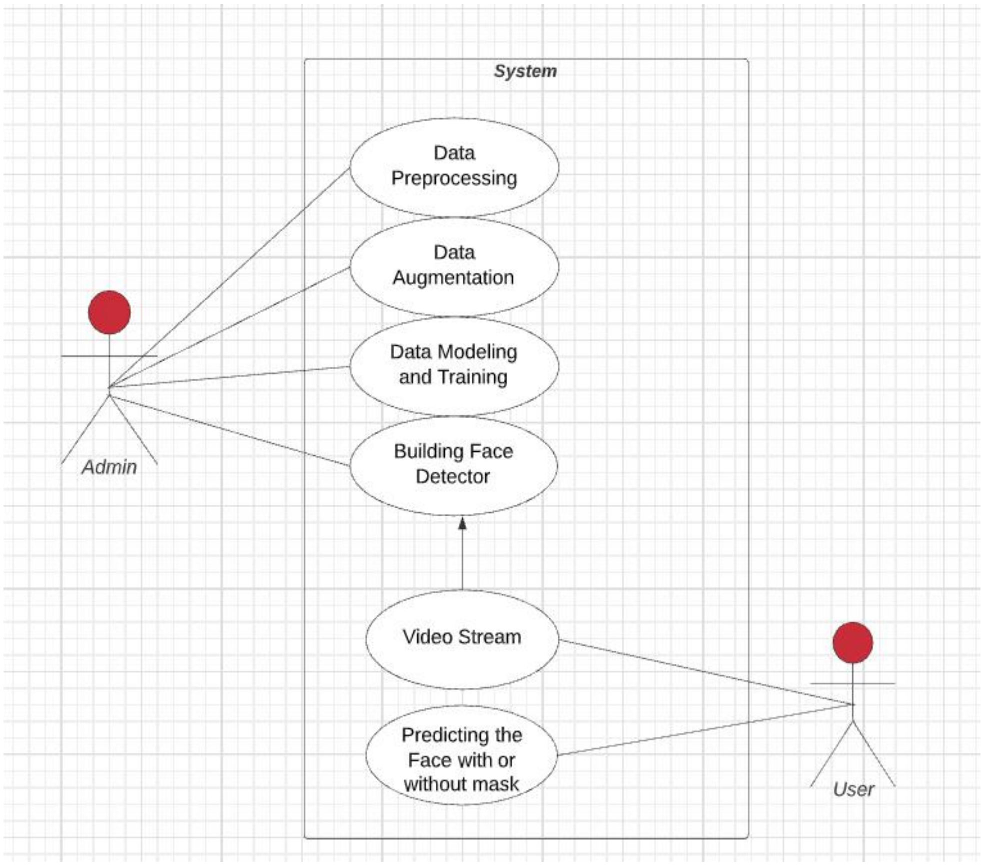


Figure 3.1: Use Case Diagram

The Use Case Diagram of Real-Time Face Mask Detection using Deep Learning network is shown in Figure 3.1. As shown in Figure 3.1 the role of the system administrator is to give input. The data set used should allow operations like cleaning, updating, adding, and deleting. The other actor is the user and system who deploys a model based on the maximum accuracy for an application. The administrator divides the data set into training and testing phases and modelling is performed iteratively to get the best model for deployment.

3.2.2 Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. It is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent.The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

Activity diagram is another important behavioral diagram in UML diagram to describe dynamic aspects of the systems. Activity diagram is essentially an advanced version of the flow chart that is modeling the flow from one activity to another. It is used to show the steps of each method and show the activity of the project. It can be used to represent the path of execution of a project. As shown in figure 3.2 we load he Data, then Pre-processing is done followed by Augmentation, splitting, and training the model using MobilenetV2 and Predicting the model with mask and without mask.

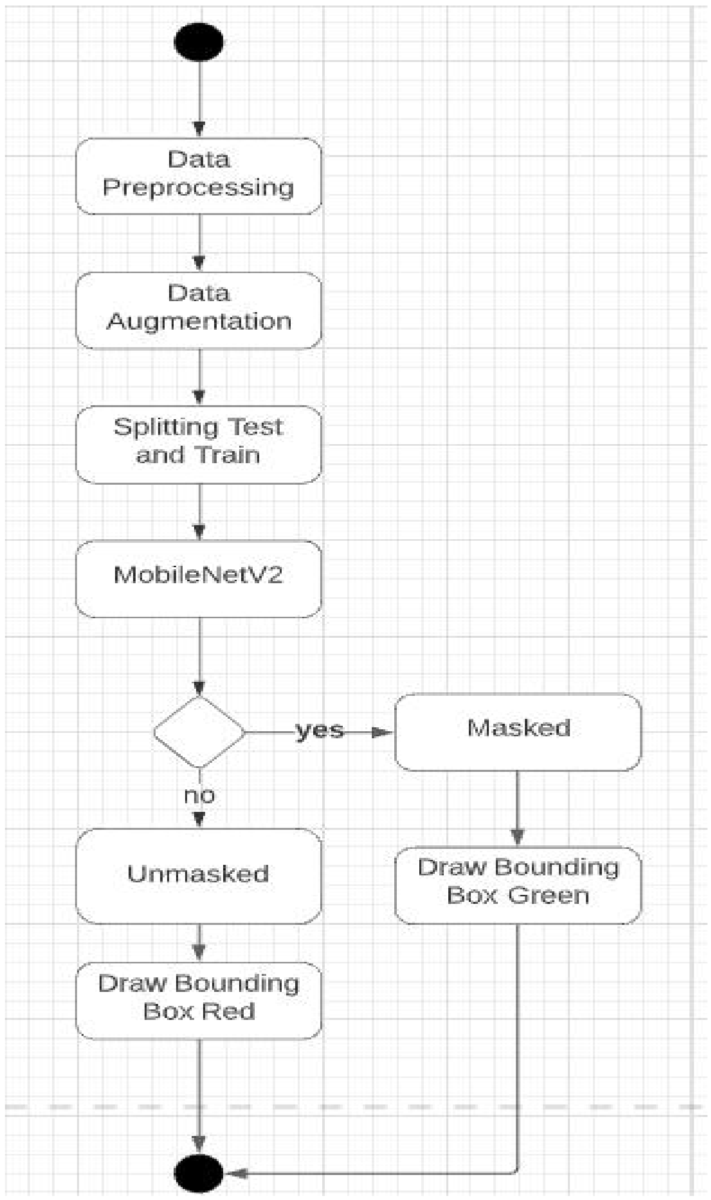


Figure 3.2: Activity Diagram

Activity Diagram of Real-Time Face Mask Detection using Deep Learning Network is shown in Figure 3.2.

3.2.3 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

It shows a collection of classes, interfaces, associations, collaborations, and constraints and is also known as a structural diagram.The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.Class diagram is also considered as the foundation for component and deployment diagrams. Class diagrams are not only used to visualize the static view of the system but they are also used to construct the executable code for forward and reverse engineering of any system.In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

Here, in the class diagram shown below each class contains a set of attributes and functions, there are three main classes to implement this algorithm in which user class and admin class would provide the access to the respective person.

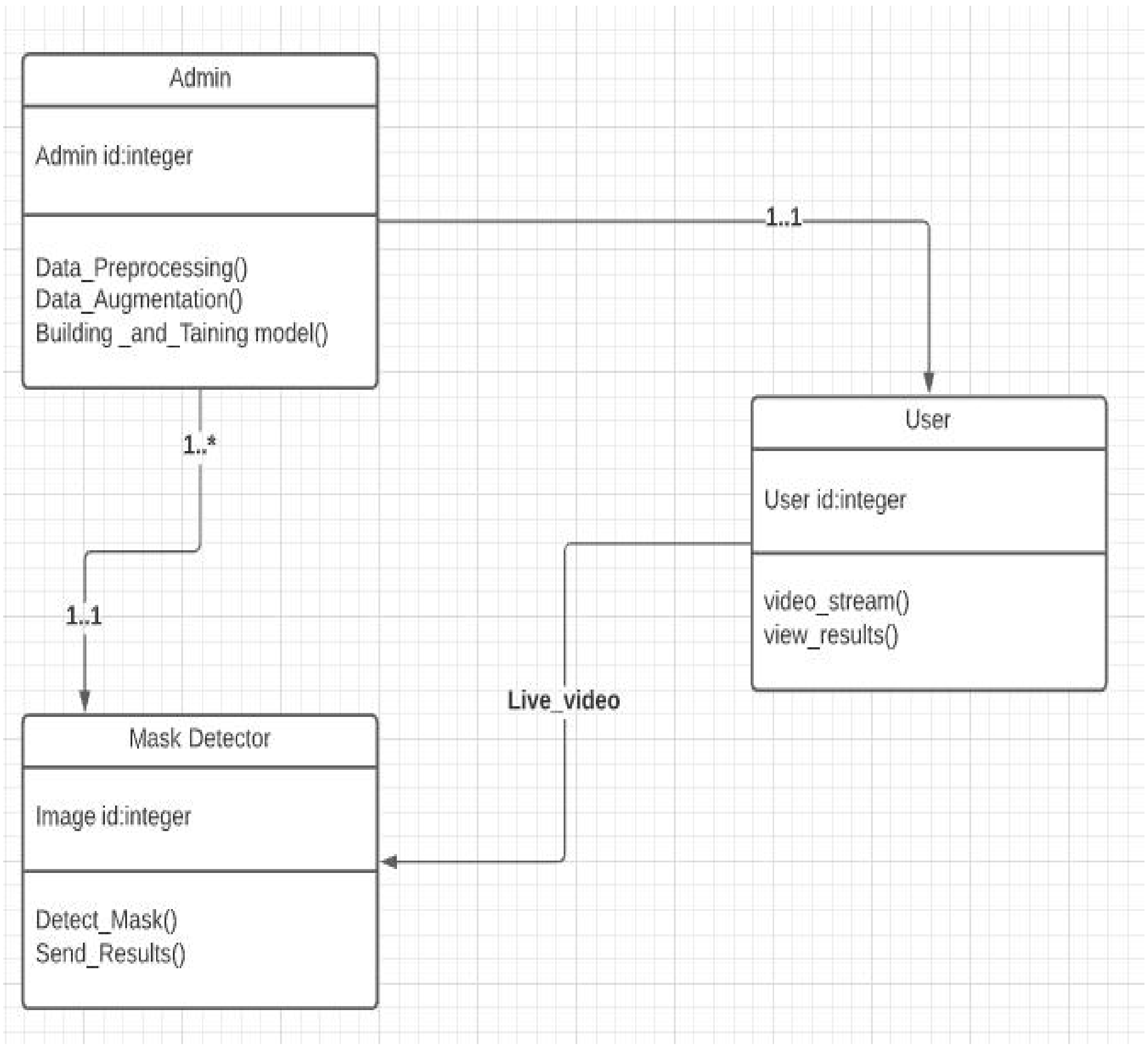


Figure 3.3: Class Diagram

In Figure 3.3, Class Diagram for Real-Time Face Mask Detection using Deep Learning Network is shown. As Shown in Figure 3.3 Each class contains a set of attributes and functions, there are Three main classes to implement this algorithm. Out of which User class and Admin class would provide the access to the respective person.

3.3 System Architecture

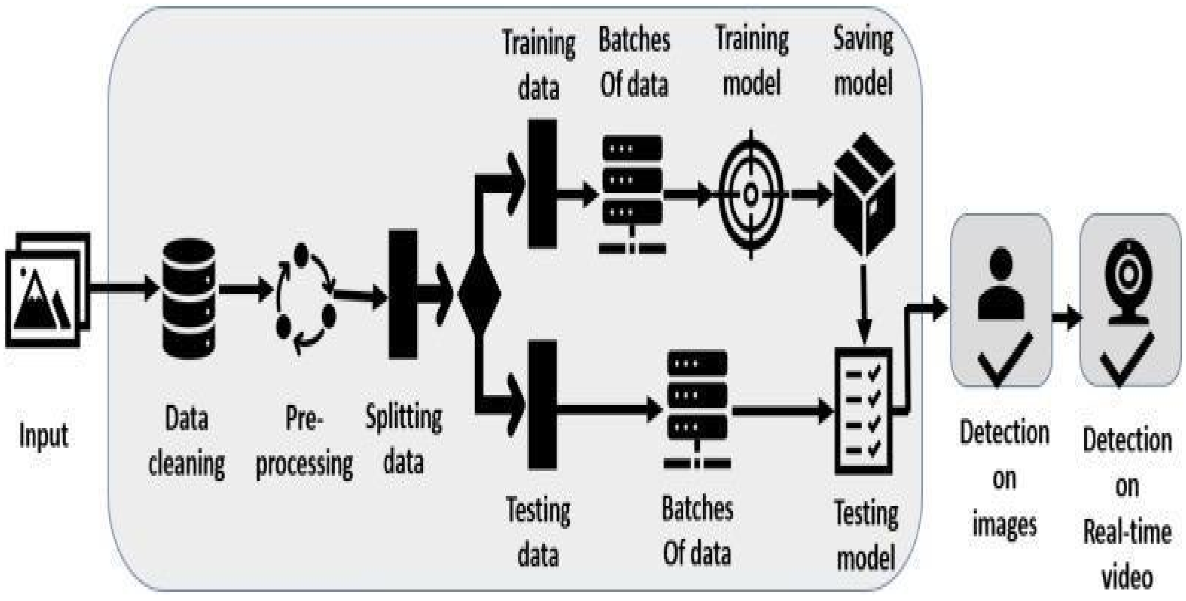


Figure 3.4: System Architecture

The above Figure 3.4 System Architecture which is referred from the research paper [5] describes the steps in constructing a model for Face Mask detection.

URL: https://www.semanticscholar.org/paper/SSDMNV2%3A-A-real-time-DNN-based-face-mask detection-Nagrath-Jain/d4c1991d8633b9a08ec8089ed0d06ccd23570491#references

As shown in above figure, first data is retrieved from datasets. The data has to be prepared for training. So, the data is undergone into pre-processing step. Now, the model is constructed with deep learning algorithms. Now, the model is trained with dataset. The model is validated and tuned to improve the classification accuracy of model. Finally, the model is tested with test dataset to know whether the model is detecting the mask correctly or not.

Data pre-processing is a data mining technique which is used to transform the raw data in a useful and efficient format.It means the process of identifying the incorrect, incomplete, inaccurate, irrelevant or missing part of the data and then modifying, replacing or deleting them according to the necessity. It is considered a foundational element of the basic data science.

Data is the most valuable thing for Analytics and Machine learning. In computing or Business data is needed everywhere. When it comes to the real world data, it is not improbable that data may contain incomplete, inconsistent or missing values. If the data is corrupted then it may hinder the process or provide inaccurate results.

Data transformation is the process of converting data from one format or structure into another format or structure. It is a fundamental aspect of most data integration and data management tasks such as data wrangling, data warehousing, data integration and application integration.

Data reduction is a capacity optimization technique in which data is reduced to its simplest possible form to free up capacity on a storage device. There are many ways to reduce data, but the idea is very simple—squeeze as much data into physical storage as possible to maximize capacity.

For models like these to give reasonable results, we not only need to feed in large quantities of data but also have to ensure the quality of data. Though making sense out of raw data is an art in itself and requires good feature engineering skills and domain knowledge (in special cases), the quality data is of no use until it is properly used. The major problem which ML/DL practitioners face is how to divide the data for training and testing. Though it seems like a simple problem at first, its complexity can be gauged only by diving deep into it. Poor training and testing sets can lead to unpredictable effects on the output of the model. It may lead to overfitting or underfitting of the data and our model may end up giving biased results.

The data has to be prepared for training. So, the data is undergone into pre-processing step. Now, the model is constructed with deep learning algorithms. Now, the model is trained with dataset. The model is validated and tuned to improve the classification accuracy of model. Finally, the model is tested with test dataset to know whether the model is detecting the mask correctly or not.

3.4 Deep Learning Technique

MobileNetV2

MobileNets are a family of neural network architectures released by Google to be used on machines with limited computing power, like mobile devices. They strive to provide state of the art accuracy, while requiring as little memory and computing power as possible. This makes them a very fast family of networks to use for image processing.

MobileNets achieve this performance by reducing dramatically the number of learnable parameters, which also makes them faster and easier to train compared to more traditional networks. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depth-wise convolutions to filter features as a source of non-linearity. The architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.

MobileNet model is a network model using depth wise separable convolution as its basic unit. Its depth wise separable convolution has two layers: depth wise convolution and point convolution [1] . It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depth wise convolutions to filter features as a source of non-linearity.

The typical MobilenetV2 architecture has as many layers listed below, In Pytorch we can use the models library in Torch Vision to create the MobileNetV2 model instead of defining/building our own model. The weights of each layer in the model are predefined based on the ImageNet dataset. The weights indicate the padding, strides, kernel size, input channels and output channels. Based on ImageNet dataset MobileNetV2 outperforms MobileNet and ShuffleNet with comparable model size and computational cost. And also, it will perform well for the smaller dataset.

The model proposed here is designed and modeled using python libraries namely Tensorflow, Keras and OpenCV. The model we used is the MobileNetV2 of convolutional neural network. The method of using MobileNetV2 is called using Transfer Learning. Transfer learning is using some pre-trained model to train your present model and get the prediction which saves time and makes using training the different models easy. We tune the model with the hyper parameters : learning rate, number of epochs and batch size.

Further the different hyper parameters are tried for the model. The hyper parameters tried are learning rate, it is a tuning parameter that is used in optimization models which determines the step size of the model and helps to reduces the loss function. It is a very important hyper parameter as it results in either convergence or overshoots the model. The other hyper parameters used are batch size, epochs etc. The model has used OpenCV to fulfil the purpose of using the video stream for capturing the frames in the video stream.

3.5 Module Description

3.5.1 Dataset

There are only a few datasets available for the detection of face masks. Most of them are either artificially created, which doesn't represent the real world accurately, or the dataset is full of noise and wrong labels.The dataset used in for training the model in a given approach was a combination of various open-source datasets and pictures, which included data from Kaggle's Medical Mask Dataset by Mikolaj Witkowski and Prajna Bhandary dataset available at PyImageSearch.

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

Data were also collected using the dataset provided by the masked face recognition dataset and application. The dataset includes about 1915 masked images and 1918 without mask images.

The model is trained with a dataset of images with two class, with mask and without mask. With this dataset, First we feed the dataset in the model, run the training program, which trains the model on the given dataset.

Then we run the detection program, which turns on the video stream, captures the frames continuously from the video stream with an anchor box using object detection process. This is passed through the MobileNetV2 model layers which classifies the image as with or without mask.The face mask recognition system uses AI technology to detect the person with or without a mask. It can be connected with any surveillance system installed at your premise.

If the person is wearing a mask, a green anchor box is displayed and red if not wearing a mask with the accuracy for the same tagged on the anchor box.

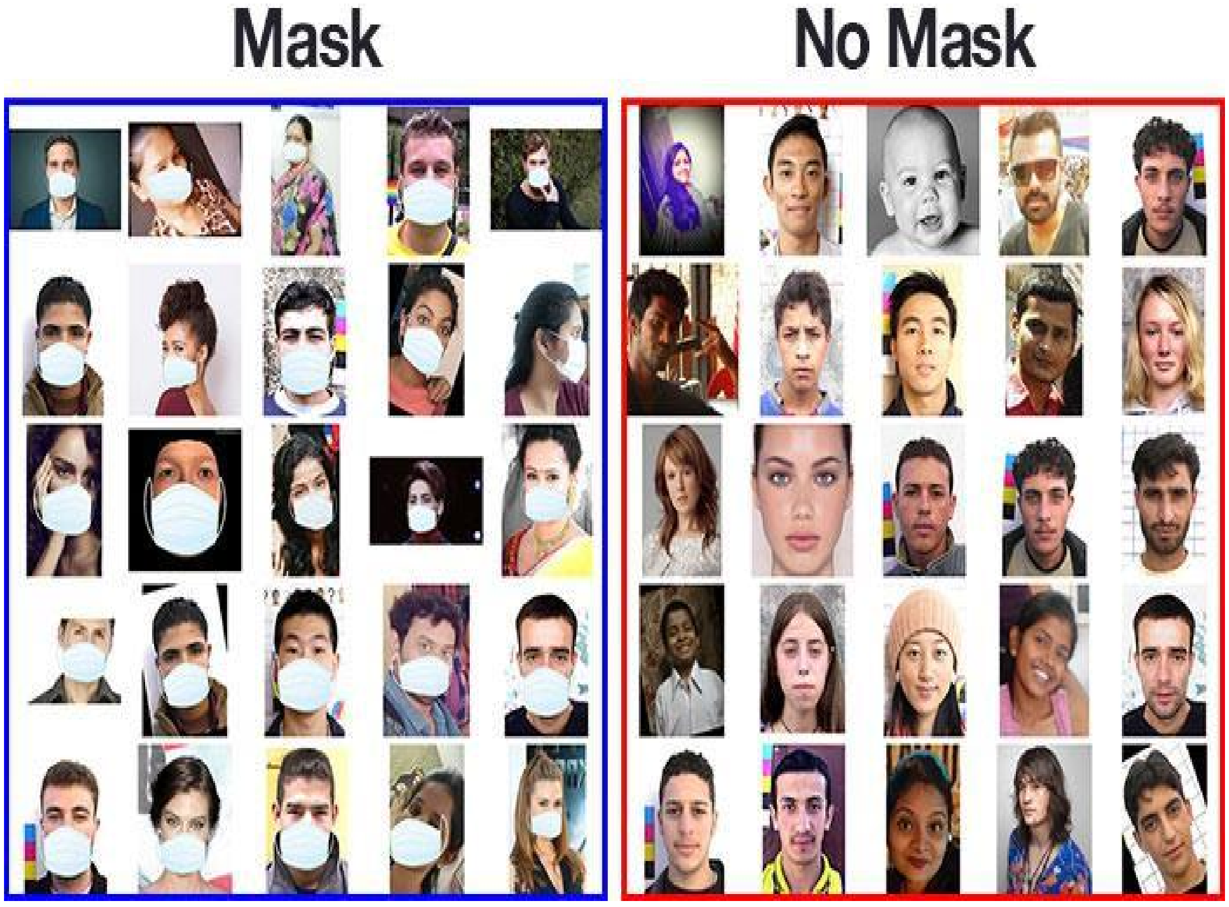


Figure 3.5: Dataset Images

The images of the Dataset for Real-Time Face Mask Detection using Deep Learning Network is shown in Figure 3.5.

3.5.2 Importing Libraries and Incorporated Packages

- **Numpy**

  Numpy is the fundamental package for scientific computing with Python. As it is used to divide the given image from construction site into n dimensional objects such that it can be easily compared with the dataset.

- **Matplotlib**

  Matplotlib is a Python 2D plotting library which is used to plot a given image in a number of small images and it will check with the dataset and it will produce publication quality figures in a variety of hard copy formats and interactive environments across platforms.

- **TensorFlow**

  TensorFlow , an interface for expressing machine learning (ML) algorithms, is utilized for implementing ML systems into various areas of computer science, including sentiment analysis, voice recognition, geographic information extraction, computer vision, text summarization, information retrieval, computational drug discovery and flaw detection to pursue research. The proposed model, the whole Sequential CNN architecture (consists of several layers) uses TensorFlow at backend. It is also used to reshape the data in the data processing.

- **Keras**

  Keras gives fundamental reflections and building units for creation and transportation of ML arrangements with high iteration velocity. It takes full advantage of the scalability and cross-platform capabilities of TensorFlow. The core data structures of Keras are layers and models .

All the layers used in the CNN model are implemented using Keras, the conversion of the class vector to the binary class matrix in data processing, helps to compile the overall model.

- **OpenCV**

OpenCV (Open-Source Computer Vision Library) , is an open source computer vision and ML software library, is utilized to differentiate and recognize faces, recognize objects, group movements in recordings, trace progressive modules, follow eye gesture, track camera actions, expel red eyes from pictures taken utilizing flash, find comparative pictures from an image database, perceive landscape and set up markers to overlay it with increased reality and so forth .The proposed method makes use of these features of OpenCV in resizing and color conversion of data images.

.

3.5.3 Data Preprocessing

The Dataset from Masked face recognition and application contained a lot of noise, and a lot of repetitions were present in the images of this dataset. Since a good dataset dictates the accuracy of the model trained on it, so the data from the above-specified datasets were taken. They were then processed, and also all the repetitions were removed manually. The data cleaning was done manually to remove the corrupt images which were found in the said dataset. Finding these images was a vital part. As it is well known, the corrupt image was a tricky task, but due to valid efforts, we divided the work and cleaned the data set with mask images and without mask images. Cleaning, identifying, and correcting errors in a dataset removes adverse effects from any predictive model.

This explains the procedure of pre-processing the data and then training on data. First, we define a function name sorted alphanumerically to sort the list in lexicographical order. A function pre-processing is defined, which takes the folder to the dataset as input, then loads all the files from the folder and resizes the images according to the SSDMNV2 model. Then the list is sorted using sorted alphanumerically, and then the images are converted into tensors. Then the list is converted to Numpy array for faster calculation. After this, the process of data augmentation is applied to increase the accuracy after training the model.

3.5.4 Data Augmentation

For the training of SSDMNV2 model, an enormous quantity of data is necessary to perform training effectively since due to the non-availability of an adequate amount of data for training the proposed model. The method of data augmentation is used to solve this issue. In this technique, methods like rotation, zooming, shifting, shearing, and flipping the picture are used for generating numerous versions of a similar picture. In the proposed model, image augmentation is used for the data augmentation process. A function image data generation is created for image augmentation, which returns test and train batches of data.

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.Training deep learning neural network models on more data can result in more skillful models, and the augmentation techniques can create variations of the images that can improve the ability of the fit models to generalize what they have learned to new images.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the *ImageDataGenerator* class.

The performance of deep learning neural networks often improves with the amount of data available. Data augmentation is a technique to artificially create new training data from existing training data. This is done by applying domain-specific techniques to examples from the training data that create new and different training examples.Image data augmentation is perhaps the most well-known type of data augmentation and involves creating transformed versions of images in the training dataset that belong to the same class as the original image.

If a dataset is very small, then a version augmented with rotation and mirroring etc. may still not be enough for a given problem. Another solution is the sourcing of entirely new, synthetic images through various techniques, for example the use of generative adversarial networks to create new synthetic images for data augmentation. Additionally, image recognition algorithms show improvement when transferring from images rendered in virtual environments to real-world data.

In addition to our focus on limited datasets, we will also consider the problem of class imbalance and how Data Augmentation can be a useful oversampling solution. Class imbalance describes a dataset with a skewed ratio of majority to minority samples. Many of the existing solutions will show how class-balancing oversampling in image data can be done with Data Augmentation.

Many aspects of Deep Learning and neural network models draw comparisons with human intelligence. For example, a human intelligence anecdote of transfer learning is illustrated in learning music. If two people are trying to learn how to play the guitar, and one already knows how to play the piano, it seems likely that the piano-player will learn to play the guitar faster. Analogous to learning music, a model that can classify ImageNet images will likely perform better on images than a model with random weights.

All of the augmentation methods discussed above are applied to images in the input space. Neural networks are incredibly powerful at mapping high-dimensional inputs into lower-dimensional representations. These networks can map images to binary classes in flattened layers. The sequential processing of neural networks can be manipulated such that the intermediate representations can be separated from the network as a whole. The lower-dimensional representations of image data in fully-connected layers can be extracted and isolated.

3.5.5 Face detection using OPENCV DNN

This model is included in the GitHub repository of OpenCV, starting from the latest version. It is established on the 'Single Shot Multibox Detector’ (SSD) and uses ‘ResNet-10’ architecture as the base-model. The Single Shot Multibox Detector is like YOLO technique which takes only one shot to detect multiple objects present in an image using Multibox. It is significantly faster in speed and high-accuracy object detection algorithm.

OpenCV does not (and does not intend to be) to be a tool for training networks — there are already great frameworks available for that purpose. Since a network (such as a CNN) can be used as a classifier, it makes logical sense that OpenCV has a Deep Learning module that we can leverage easily within the OpenCV ecosystem. OpenCV can’t be used for training deep learning networks. This might sound like a bummer but fret not, for training neural networks you shouldn’t use OpenCV there are other specialized libraries like Tensorflow, PyTorch, etc for that task.

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today’s systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis.

3.5.6 Classification of images using MobileNetV2

MobileNetV2 is a Deep Neural Network that has been deployed for the classification problem. Pre-trained weights of ImageNet were loaded from TensorFlow. Then the base layers are frozen to avoid impairment of already learned features. Then new trainable layers are added, and these layers are trained on the collected dataset so that it can determine the features to classify a face wearing a mask from a face not wearing a mask. Then the model is fine-tuned, and then the weights are saved. Using pre- trained models helps avoid unnecessary computational costs and helps in taking advantage of already biased weights without losing already learned features.

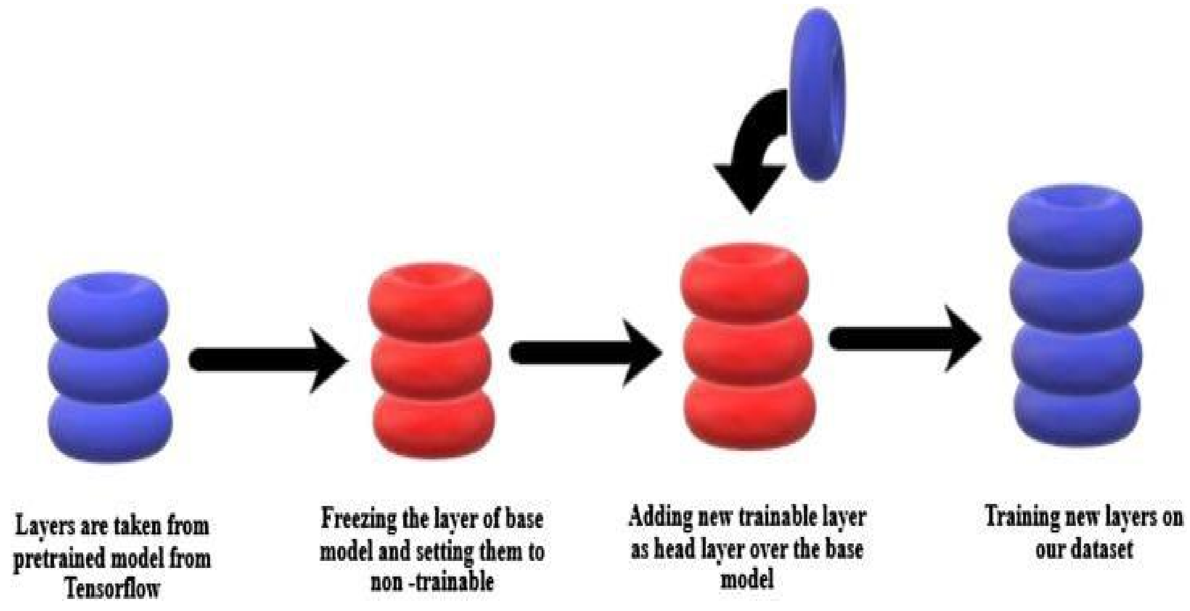

Layers are taken from pretrained model from Tensorflow
Freezing the layer of base model and setting them to non -trainable
Adding new trainable layer as head layer over the base model
Training new layers on our dataset

Figure 3.6: Classification of Images

This plan of action is shown in Figure 3.6 is referred from the URL attached below which is from the reasearch paper[5].

URL: https://www.sciencedirect.com/science/article/pii/S2210670720309070?via%3Dihub

3.5.7 Building blocks of MobileNetV2

MobileNetV2 is a deep learning model based on Convolutional Neural Network, which uses the following layers and functions. The structure of MobileNetV2 has been shown in Figure 3.7. This layer is the fundamental block of the Convolutional Neural Network. The term convolution implies a mathematical combination of two functions to get the third function. It works on a sliding window mechanism, which helps in extracting features from an image. This helps in generation feature maps. The convolution often functional matrices, one being the input image matrix A and the other being convolutional kernel B give us the output C.

Source: https://www.sciencedirect.com/science/article/pii/S2210670720309070?via%3Dihub

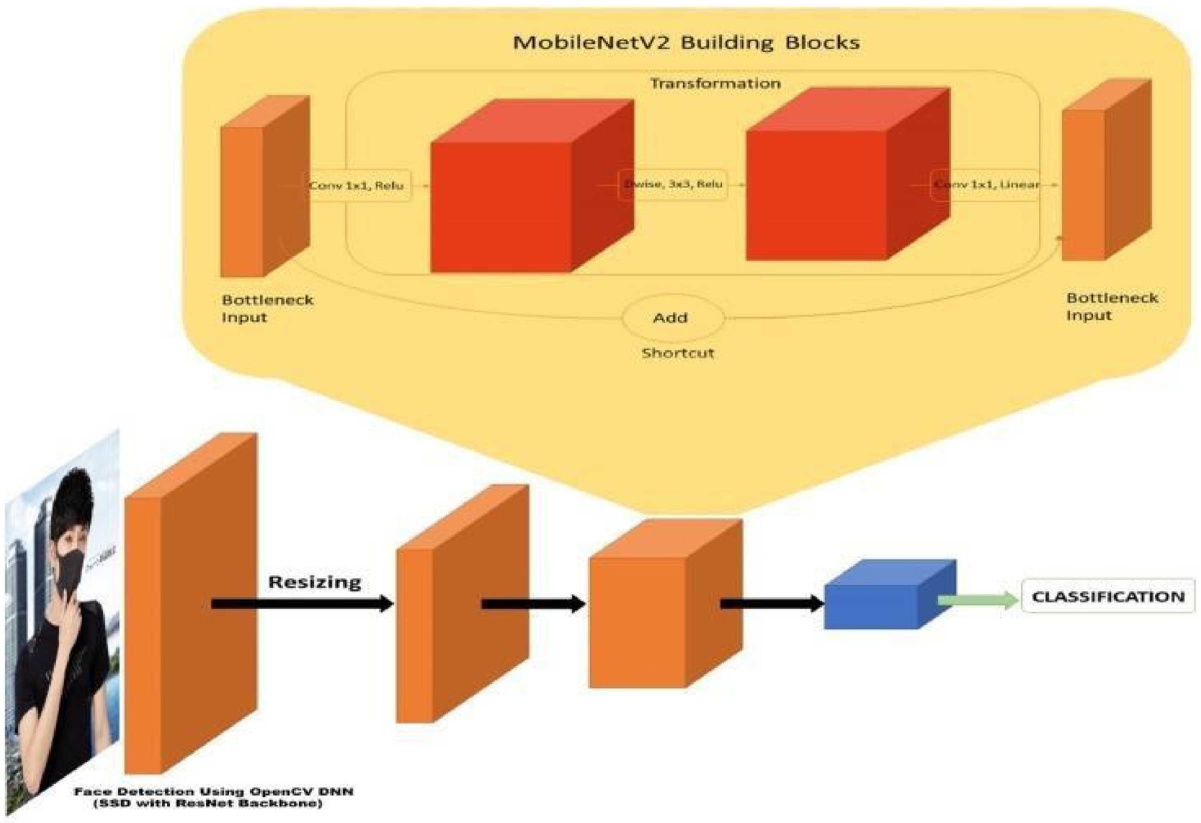


Figure 3.7: MobileNetV2 Architecture

MobileNetV2 Architecture is shown in the above figure.

4. RESULTS AND ANALYSIS

Testing is a very important module in software development to verify, validate and provide quality and service for different components of software. It is used to minimize the risks by efficient use of resources in the development life cycle. This module can be employed at any point of the development process. It is efficient for the testing phase to be implemented at initial level to lower down the risks of defects and failures. Software is tested and implemented in various conditions and environments to examine different aspects of software. There exist various organizations for testing which is based on the type of software developed. This testing phase helps each and every module to work effectively and generate the optimum results.

To test a data first we must train the data set using an algorithm. After training the dataset with algorithms we must test the dataset with same algorithms and check the accuracy of each and algorithm and predicted values of each algorithm. The algorithm which gives high accuracy and best predicted value in testing is the best algorithm which should be used further in testing.

In this project we are training over 2880 images and validated over 720 images we take input size as (224,224,3).

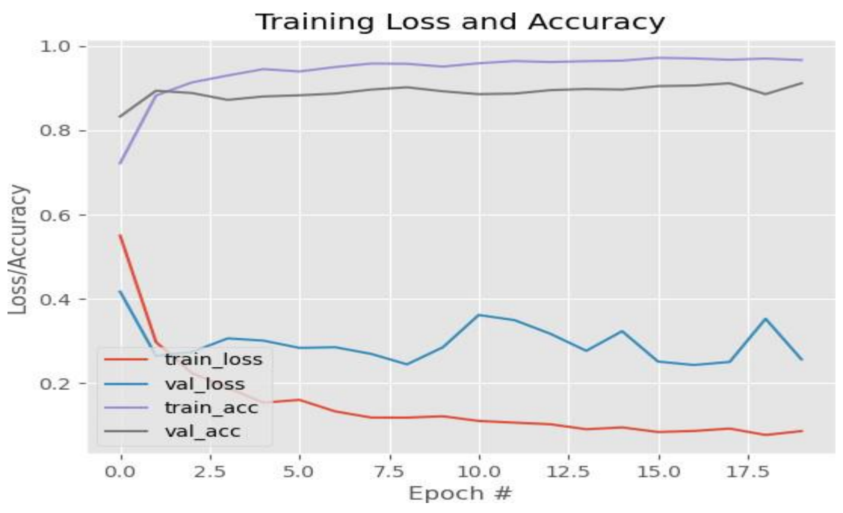


Figure 4.1: Training Accuracy

According to the above results, It shows the plot for training loss, validation loss, training accuracy and validation accuracy for Number of epochs versus loss or accuracy. It is evident from the plot that as the number of epochs increases the training and validation accuracy increases and the training and validation accuracy decreases. And also the validation accuracy is higher than the training accuracy which proves that the model is not suffering through over-fitting.

The above figure 4.1 shows Accuracy for each epoch from 0 to 20. Here we can see that there is a gradual decrease in the Error rate in both training and validation. But we can see there is a huge curve going upwards over the 10 epoch it states that there may be a new class introduced to the network so getting the new thing as an input it is unable to find the result and the error has increased compared to the he previous epochs. But the training goes on the training loss decreases to 0.086 and Accuracy Increases to 96.67%.

One of the main reasons behind achieving this accuracy lies in MaxPooling. It provides rudimentary translation invariance to the internal representation along with the reduction in the number of parameters the model has to learn. This sample-based discretization process down-samples the input representation consisting of image , by reducing its dimensionality. A much higher number of neurons and filters can lead to worse performance. The optimized filter values and pool size help to filter out the face in order to detect the existence of mask correctly without causing over-fitting.

The system can efficiently detect faces that are partially occluded (either with a mask or hair or hand). Based on the occlusion degree of four regions (nose, mouth, chin and eye) it differentiates between annotated mask and face covered by hand. Therefore, a mask covering the face fully including nose and chin will only be treated as “with mask” by the model.

The main challenges faced by the method mainly comprise of varying angles and lack of clarity. The movement of indistinct faces in the video stream makes it more difficult. However, following the trajectories of several frames of the video helps to create a better decision – “with mask” or “without mask”.

Figure below shows the predictions on some images. These are the predictions made on 2 test images by SSDMNV2 model using MobileNetV2. The rectangular green box depicts the correct way of wearing a mask with an accuracy score on the top left as shown in Fig 4.3, while the red rectangular box represents the without mask as shown in Fig 4.2. The model learns from the pattern of train dataset and labels and then makes predictions.

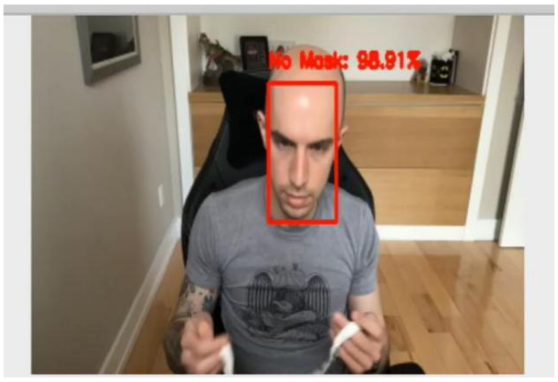


Figure 4.2: Face without Mask

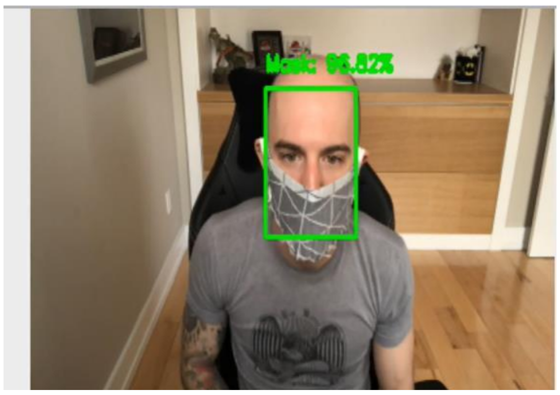


Figure 4.3: Face with Mask

The above figures shows the outputs in which accuracy score is also given.

5. CONCLUSION AND FUTURE WORK

Conclusion

In the proposed face mask detection model , both the training and development of the image dataset, which was divided into categories of people having masks and people not having masks have been done successfully. The technique of OpenCV deep neural networks used in this model generated fruitful results. Classification of images was done accurately using the MobilenetV2 image classifier, which is one of the uniqueness of the proposed approach.

The problem of various wrong predictions has been successfully removed from the model as the dataset used was collected from various other sources and images used in the dataset was cleaned manually to increase the accuracy of the results. Real-world applications are a much more challenging issue for the upcoming future.

Future Work

Finally, this project opens interesting future directions for researchers. Initially, the proposed technique can be integrated into any high-resolution video surveillance devices and not limited to mask detection only. Then, the model can be extended to detect facial landmarks with a facemask for biometric purposes and also classifying the specific person face with mask. This can be helpful in unlocking the phone with mask and also can be used in identifying people in an organization.

Bibliography

[1] Martin Fowler,UML Distilled Third edition A Brief GuideTo the Standard Object Modelling language.2003

[2] Ahmed I., Ahmad M., Rodrigues J.J., Jeon G., Din S. A deep learning-based social distance monitoring framework for COVID-19. *Sustainable Cities and Society.* 2020

[3] Alom M.Z., Taha T.M., Yakopcic C., Westberg S., Sidike P., Nasrin M.S.…Asari V.K. 2018. The history began from alexnet: A comprehensive survey on deep learning approaches. arXiv preprint arXiv:1803.01164.

[4] Anisimov D., Khanova T. Towards lightweight convolutional neural networks for object detection. 2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS); IEEE; 2017. pp. 1–8. August.

[5] P. Nagrath, Rachna Jain, Agam Madan, Rohan Arora, Piyush Kataria, Jude D. Hemanth, A real time DNN-based face mask detection system using single shot multibox detector. *Sustainable Cities and Society.* 2020

[6] Ge S., Li J., Ye Q., Luo Z. Detecting masked faces in the wild with lle-cnns. *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2017:2682–2690.

[7] Ge X.Y., Pu Y., Liao C.H., Huang W.F., Zeng Q., Zhou H.…Chen H.L. Evaluation of the exposure risk of SARS-CoV-2 in different hospital environment. *Sustainable Cities and Society.* 2020;61

[8] Ghiasi G., Fowlkes C.C. Occlusion coherence: Localizing occluded faces with a hierarchical deformable part model. *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2014:2385–2392.

[9] Gulcehre C., Moczulski M., Denil M., Bengio Y. *International conference on machine learning.* 2016. Noisy activation functions; pp. 3059–3068. June.

[10] Hahnloser R.H., Sarpeshkar R., Mahowald M.A., Douglas R.J., Seung H.S. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature.* 2000;405(6789):947–951.

[11] He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016:770–778.

Appendix

## Code

#Code for building MASK DETECTOR MODEL

```
# import the necessary packages

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os


'''initialize the initial learning rate, number of epochs to train for,
 and batch size;initially lr should be less ,so that loss will be calculated properly with better
accuracy'''

INIT_LR = 1e-4
EPOCHS = 20
BS = 32

#Data Preprocessing - converting all the imgs into arrays
#so that with those arrays we can create a deep learning network

DIRECTORY = r"C:\IV\mini\CODE\FMD\dataset"

CATEGORIES = ["with_mask", "without_mask"]
```

```
# 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...")


data = [] #for image arrays
labels = [] #corresponding img with a label

for category in CATEGORIES:

    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):

            img_path = os.path.join(path, img)
            image = load_img(img_path, target_size=(224, 224))
            image = img_to_array(image)
            image = preprocess_input(image)

            data.append(image) #numericals
            labels.append(category) #alphabets/characters


# perform one-hot encoding on the labels
#label binarizer to convert alphabetics to categorical variables(0s&1s)

lb = LabelBinarizer()

labels = lb.fit_transform(labels)
labels = to_categorical(labels)

data = np.array(data, dtype="float32")

labels = np.array(labels)

(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")


#Modelling

'''MobilenetV2 creates 2 models: base and head
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("mask_detector.model", save_format="h5")
```

```
# plot the training loss and accuracy and other metrics

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("plot.png")
```

#Code for building FACE MASK DETECTOR

```
# 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))

	# 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)
```

```
# load our serialized face detector model from disk

prototxtPath = r"face_detector\deploy.prototxt"

weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"

faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)


# load the face mask detector model from disk

maskNet = load_model("mask_detector.model")


# initialize the video stream

print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()


# 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()
```