<u>UNIVERSITY OF MUMBAI</u>

A DISSERTATION REPORT ON

"<u>Face Mask Detection</u>"

SUBMITTED IN PARTIAL FULFILMENT FOR

THE REQUIREMENTS OF THE DEGREE

BACHELOR OF ENGINEERING

IN

COMPUTER ENGINEERING

GROUP MEMEBERS

PALANDE VAIBHAV JAYENDRA (40)

PANCHAL PRASAD KALAPPA (41)

PARAB MANDAR DINESH (42)

UNDER THE GUIDANCE OF

<u>PROF. PRAJAKTA PISE</u>

DEPARTMENT OF COMPUTER ENGINEERING

G.V. ACHARYA INSTITUTE OF ENGINEERING AND TECHNOLOGY

UNIVERSITY OF MUMBAI

2020-2021

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Palande Vaibhav Jayendra (40) ____________

Panchal Prasad Kalappa (41) ______________

Parab Mandar Dinesh (42) ________________

Date:

Place: SHELU

PROJECT REPORT APPROVAL FOR T. E.

This project report entitled ***“Face Mask Detection using Python”*** by **“Palande Vaibhav Jayendra” (40), “Panchal Prasad Kalappa” (41), “Parab Mandar Dinesh” (42)** is approved for the degree of **“Bachelor of Computer Engineering”.**

Examiners

1.____________________

2. ____________________

Date:

Place: SHELU

<u>CERTIFICATE</u>

This is to certify that the project entitled **"<u>Face Mask Detection using Python</u>"** is a bonafide work of **"Palande Vaibhav Jayendra" (40), "Panchal Prasad Kalappa" (41), "Parab Mandar Dinesh" (42)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Undergraduate"** in **"Bachelor of Computer Engineering".**

__________________	__________________
Prof. Prajakta Pise	Prof. Roshan Bauskar
Project Guide	Head of Department
Department of Computer Engineering	Computer Engineering
G.V.A.I.E.T	G.V.A.I.E.T

____

Dr. Prashant Sonare

Principal

G.V.A.I.E.T

ACKNOWLEDGEMENT

I take the opportunity to thank all of those who have generously helped me to give a proper shape to my work and complete my **T.E** project synopsis successfully. A successful project's completion involves fruitful combination of many people, some directly involved and some indirectly, by providing support and encouragement. So with gratitude I acknowledge all those whose guidance and encouragement served a beacon of light and crowned our efforts with success.

I am thankful to Principal **Dr. Prashant Sonare**, for the constant support and encouragement during the project. It's also a great pleasure to express my deepest gratitude to all faculty members of my department for their cooperation.

I consider it a privilege and honor to express my sincere gratitude and respect to my project guide **Prof. Prajakta Pise**, Department of Computer Engineering for his valuable guidance throughout the tenure of this project seminar.

I would again like to thank **Prof. Roshan Bauskar**, HOD, Department of Computer Engineering who shared his opinions and experiences through which I received the required information which was crucial for this project.

Group Members

Palande Vaibhav Jayendra (40)

Panchal Prasad Kalappa (41)

Parab Mandar Dinesh (42)

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Overview……………………………………………………..

1.2 Objective……………………………………….

2 RELATED WORK

2.1 Related Work……………………………………………

2.2 Existing System…………………………………………………..

3 DATASET

3.1 Introduction………………………………………………..

3.2 Face Mask Detection dataset…………………………………………..

3.3 Incorporated Packages………………………………………

4 PROJECT REQUIREMENTS & EVALUATION

4.1 Hardware Requirements…………………………………….

4.2 Software Requirements………………………………………

4.3 Python Libraries…………………………………………..

4.4 Evaluation…………………………………….

5 WORKING

5.1 Phases of Face Mask Detection………………………………………………

5.2 Facial Landmarks………………………………………….

5.3 Region of Interest (ROI)………………………………………….

5.4 Data Processing………………..………………………..

5.5 Data Visualization……………….………………………..

5.6 Training of Model……………….………………………..

5.7 Project Structure……………….………………………..

5.8 Face Mask Detection with OpenCV……………….………………………..

6 RESULT & ANALYSIS

6.1 Analysis…………………………………………………

7 CONCLUSION & FUTURE SCOPE

7.1 Conclusion……………………………………………….

7.2 Future Scope……………………………………………..

8 REFERENCES

8.1 References ……………………………………………….

LIST OF FIGURES

3.1 Face Mask Detection fig………………………………………

3.2 Phases of Face mask detection ……………………………..

3.3 ROI …………………………………

3.4 Facial Landmarks……………………………………………

5.1.1 With Mask……………………………………………

5.1.2 Without Mask……………………………………………….

5.2 Project Structure…………………………………………………..

5.3 Analysis & Accuracy………………………………………………

LIST OF ABBREVATIONS

ABBREVATIONS	DESCRIPTION
ROI	Region of Interest
WHO	World Health Organization
ML	Machine Learning
CNN	Convolutional Neural Network
SDLC	Software Development Life Cycle

ABSTRACT

Coronavirus (COVID-19) pandemic has rapidly affected our day to-day life disrupting the world trade and movements. Wearing a protective face mask has become a new normal. In the near future, many public service providers will ask the customers to wear masks correctly to avail of their services. Therefore, face mask detection has become a crucial task to help global society. This paper presents a simplified approach to achieve this purpose using some basic Machine Learning packages like TensorFlow, Keras, and OpenCV. The proposed method detects the face from the image correctly and then identifies if it has a mask on it or not. As a surveillance task performer, it can also detect a face along with a mask in motion. The method attains accuracy up to 95.77% and 94.58% respectively on two different datasets. We explore optimized values of parameters using the Sequential Convolutional Neural Network model to detect the presence of masks correctly without causing over-fitting.

CHAPTER 1

INTRODUCTION

INTRODUCTION

Overview

According to the World Health Organization (WHO)'s official Situation Report – 205, coronavirus disease 2019 (COVID-19) has globally infected over 20 million people causing over 0.7 million deaths. Individuals with COVID-19 have had a wide scope of symptoms reported – going from mellow manifestations to serious illness. Respiratory problems like shortness of breath or difficulty in breathing is one of them. Elder people having lung disease can possess serious complications from COVID-19 illness as appear to be at higher risk. Some common human coronaviruses that infect public around the world are 229E, HKU1, OC43, and NL63. Before debilitating individuals, viruses like 2019-nCoV, SARS-CoV, and MERS-CoV infect animals and evolve to human coronaviruses. Persons having respiratory problems can expose anyone (who is in close contact with them) to infective beads. Surroundings of a tainted individual can cause contact transmission as droplets carrying virus may withal arrive on his adjacent surfaces.

To curb certain respiratory viral ailments, including COVID-19, wearing a clinical mask is very necessary. The public should be aware of whether to put on the mask for source control or aversion of COVID-19. Potential points of interest of the utilization of masks lie in reducing vulnerability of risk from a noxious individual during the "Pre-symptomatic" period and stigmatization of discrete persons putting on masks to restraint the spread of virus. WHO stresses on prioritizing medical masks and respirators for health care assistants. Therefore, face mask detection has become a crucial task in present global society.

Face mask detection involves in detecting the location of the face and then determining whether it has a mask on it or not. The issue is proximately cognate to general object detection to detect the classes of objects. Face identification categorically deals with distinguishing a specific group of entities i.e. face. It has numerous applications, such as autonomous driving, education, surveillance, and so on [5]. This paper presents a simplified approach to serve the above purpose using the basic Machine Learning (ML) packages such as TensorFlow, Keras, and OpenCV.

The rest of the paper is organized as follows: Section II explores related work associated with face mask detection. Section III discusses the nature of the used dataset. Section IV presents the details of the packages incorporated to build the proposed model. Section V gives an overview of our method. Experimental results and analysis are reported in section VI. Section VII concludes and draws the line towards future works.

Objective

- A model is created using datasets with_mask and without_mask.
- Our project is to prepare a data set which would detect whether a person on the camera is wearing a Face Mask or not.

CHAPTER 2

RELATED WORK

Related Work

In face detection method, a face is detected from an image that has several attributes in it. According to, research into face detection requires expression recognition, face tracking, and pose estimation. Given a solitary image, the challenge is to identify the face from the picture. Face detection is a difficult errand because the faces change in size, shape, color, etc. and they are not immutable. It becomes a laborious job for opaque image impeded by some other thing not confronting camera, and so forth.

Authors in think occlusive face detection comes with two major challenges:

1. Unavailability of sizably voluminous datasets containing both masked and unmasked faces, and
2. Exclusion of facial expression in the covered area.

Utilizing the locally linear embedding (LLE) algorithm and the dictionaries trained on an immensely colossal pool of masked faces, synthesized mundane faces, several mislaid expressions can be recuperated and the ascendancy of facial cues can be mitigated to great extent. According to the work reported in, convolutional neural network (CNNs) in computer vision comes with a strict constraint regarding the size of the input image. The prevalent practice reconfigures the images before fitting them into the network to surmount the inhibition. Here the main challenge of the task is to detect the face from the image correctly and then identify if it has a mask on it or not. In order to perform surveillance tasks, the proposed method should also detect a face along with a mask in motion.

Existing System

In the Previous versions of Face Mask Detection System which has certain limitations are as follows:

- The existing system didn’t analyze the relationship of training set size versus accuracy/error.
- The existing system do not offer their own extensive code as open-source.
- The existing system do not offer the dataset-independence of the trained models.

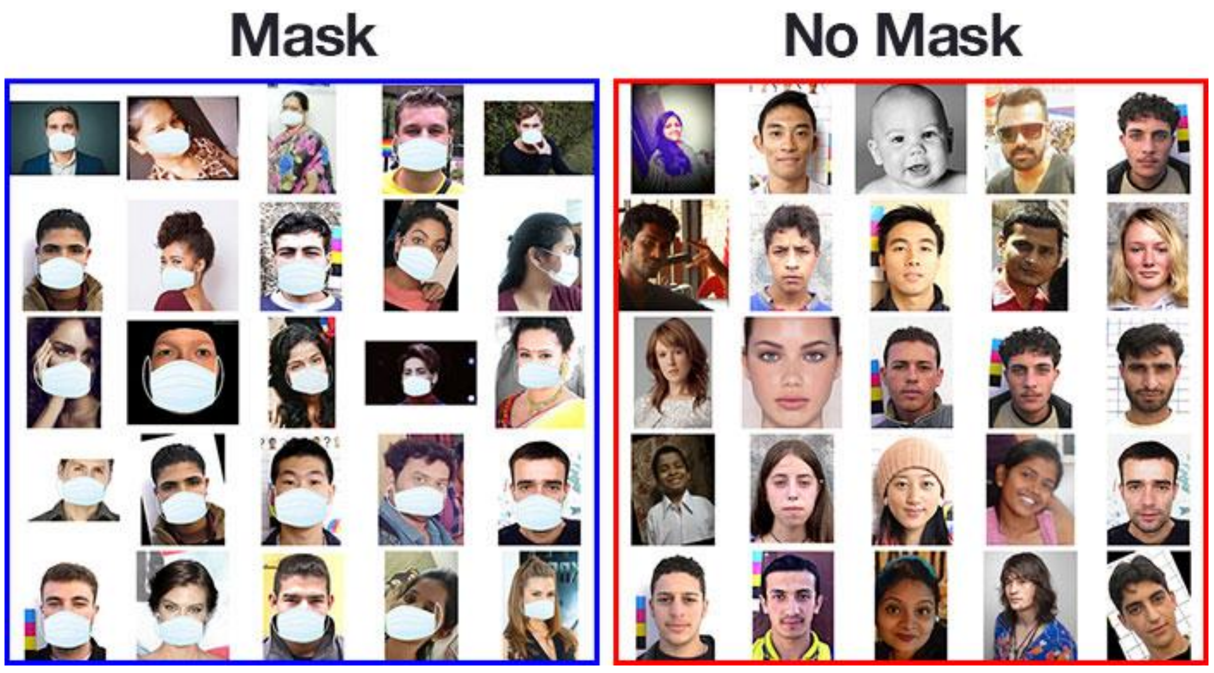
CHAPTER 3

DATASET

DATASET

A face mask detection dataset consists of "with mask" and "without mask" images. We will use the dataset to build a COVID-19 face mask detector with computer vision and deep learning using Python, OpenCV, and TensorFlow/Keras. We collected our data from the Kaggle open source.

Our COVID-19 face mask detection dataset



This dataset consists of **3,833 images** belonging to two classes:

- ❖ **With_mask: 1,915**
- ❖ **Without_mask: 1,918**

INCORPORATED PACKAGES

A. TensorFlow

TensorFlow, an interface for expressing machine learning algorithms, is utilized for implementing ML systems into fabrication over a bunch of 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. In the proposed model, the whole Sequential CNN architecture (consists of several layers) uses TensorFlow at backend. It is also used to reshape the data (image) in the data processing

B. 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. Along with the conversion of the class vector to the binary class matrix in data processing, it helps to compile the overall model.

C. OpenCV

OpenCV (Open Source Computer Vision Library), 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.

CHAPTER 4

PROJECT REQUIREMENTS

PROJECT REQUIREMENTS

HARDWARE REQUIREMENTS

- Processor: 2 GHz frequency or above
- RAM: Minimum of 2GB
- HDD: 500 GB or above

SOFTWARE REQUIREMENTS

- OS: Windows 7,8 or 10
- Mac OSX 10.8, 10.9, 10.10 or 10.11
- Tool: Jupyter notebook
- Programming language: Python 3.5 +
- IDE: IDLE

PYTHON LIBRARIES

- Tensorflow>=1.15.2
- Keras==2.3.1
- Imutils==0.5.3
- Numpy==1.18.2
- Opencv-python==4.2.0.*
- Matplotlib==3.2.1
- Scipy==1.4.1

EVALUATION

(a) Preliminary analysis: The objective of phase 1 is to conduct a preliminary analysis, propose alternative solutions, describe costs and benefits and submit a preliminary plan with recommendations.
(b) Systems analysis, requirements definition: Defines project goals into defined functions and operation of the intended application. Analyze end-user information needs.

(c) Systems design: Describes desired features and operations in detail, including screen layouts, business rules, process diagrams, pseudo code and other documentation.

(d) Integration and testing: Brings all the pieces together into a special testing environment, then checks for errors, bugs and interoperability.

(e) Acceptance, installation, deployment: The final stage of initial development, where the software is put into production and runs actual business.

(f) Maintenance: During the maintenance stage of the SDLC, the system is assessed to ensure it does not become obsolete. This is also where changes are made to initial software. It involves continuous evaluation of the system.

4.3 Feasibility Study

In simple terms, a feasibility study involves taking a judgment call on whether a project is doable. The two criteria to judge feasibility are cost required and value to be delivered. A well – designed study should offer a historical background of the business or project, a description of the product or service, accounting statements, details of operations and

Management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, such studies precede technical development and project implementation.

A feasibility study evaluates the project’s potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions.

4.3.1 Technical Feasibility

We concern here with specifying Equipment and software that will satisfy the user requirement. It will run with minimum system requirements and with minimum system resources acquired during run. It will need a web server, to which it gets from the internet, at run time. Expandability will be maintained in the new system. New modules can be added .Later on, the application, if required in the future.

4.3.2 Economic Feasibility

The procedure is to determine the benefit and savings that are expected from the project and compare them with the cost. As internet is the cheapest way of communication, we can perform communication using web. The cost is just the cost of using the internet based on the channel allocation. So, the project will be economically feasible.

CHAPTER 5

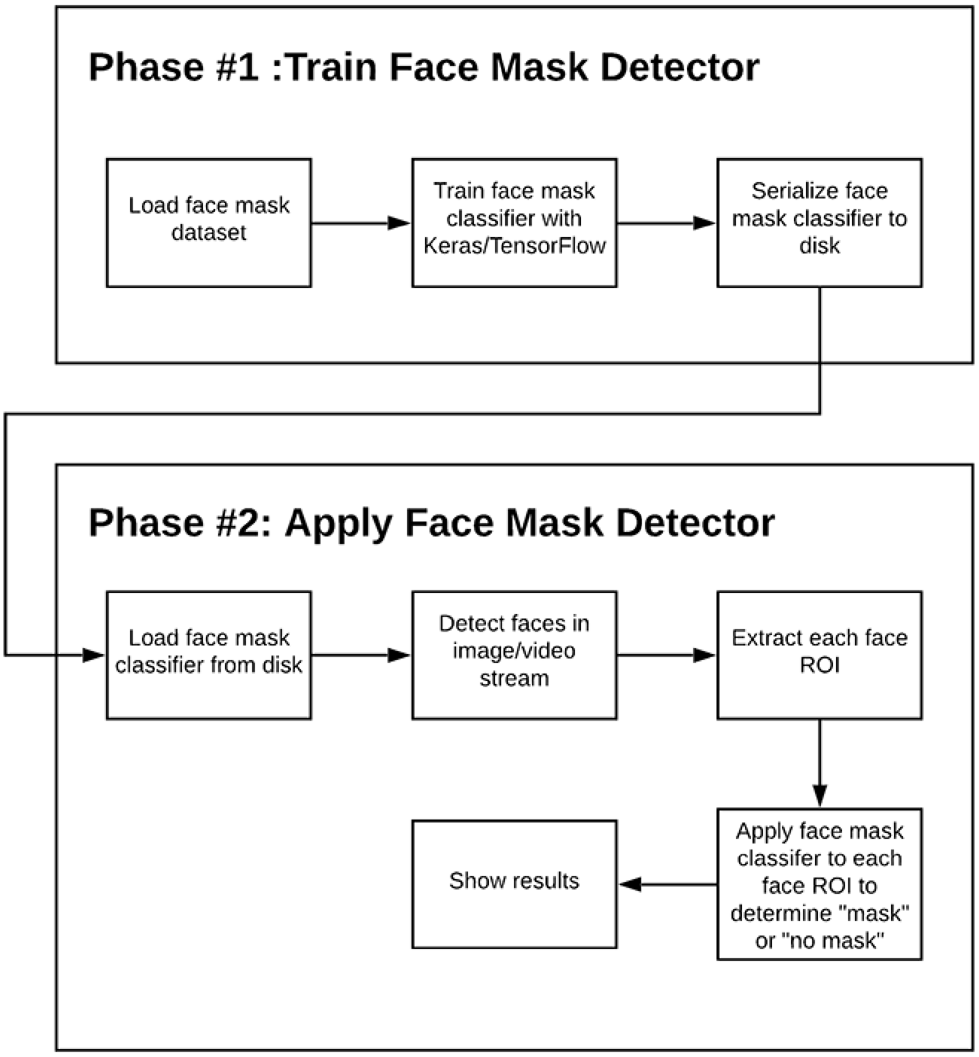
WORKING

WORKING

COVID-19: Face Mask Detector

- Detect face masks most accurately
- Detect face masks in *real-time video streams*

<u>Two-phase COVID-19 face mask detector</u>


Phase #1 :Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifer to each face ROI to determine "mask" or "no mask"
Show results

Phases and individual steps for building a COVID-19 face mask detector with computer vision and deep learning using Python, OpenCV, and TensorFlow/Keras.

Phases Working

In order to train a custom face mask detector, we need to break our project into two distinct phases, each with its own respective sub-steps (as shown by **Figure** above):

1. **Training:** Here we'll focus on loading our face mask detection dataset from disk, training a model (using Keras/TensorFlow) on this dataset, and then serializing the face mask detector to disk.

2. **Deployment:** Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as

   with_mask
   or
   without_mask

We'll review each of these phases and associated subsets in detail in the remainder of this tutorial, but in the meantime, let's take a look at the dataset we'll be using to train our COVID-19 face mask detector.

Our goal is to train a custom model to detect whether a person *is* or *is not* wearing a mask.

Facial Landmarks

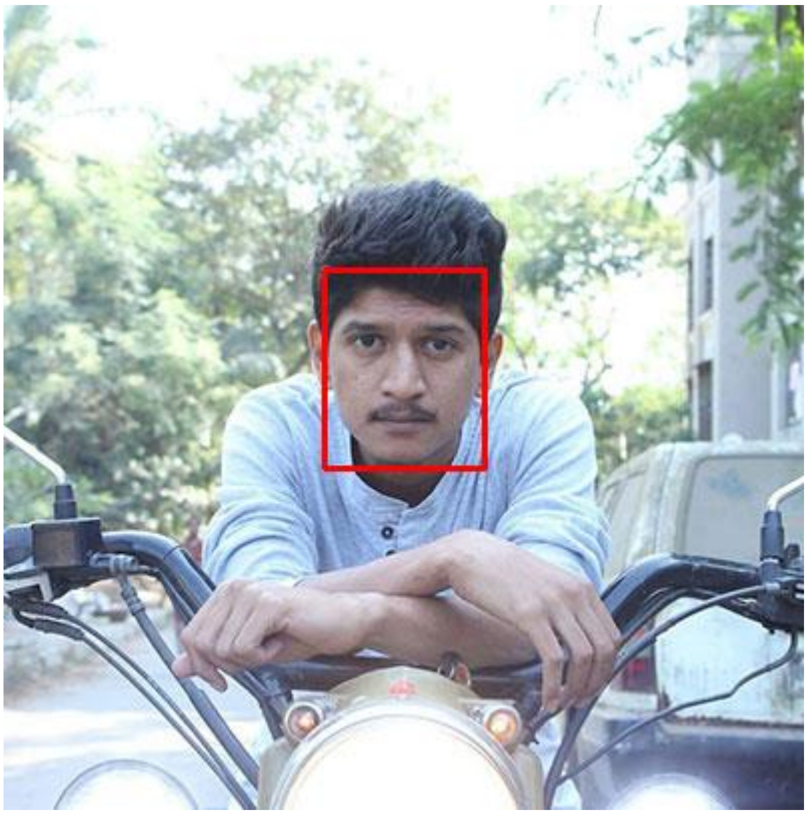
Facial landmarks allow us to automatically infer the location of facial structures, including:

- Eyes
- Eyebrows
- Nose
- Mouth
- Jawline

To use facial landmarks to build a dataset of faces wearing face masks, we need to first start with an image of a person *not* wearing a face mask:

To build a COVID-19/Coronavirus pandemic face mask dataset, we'll first start with a photograph of someone not wearing a face.

From there, we apply face detection to compute the bounding box location of the face in the image:

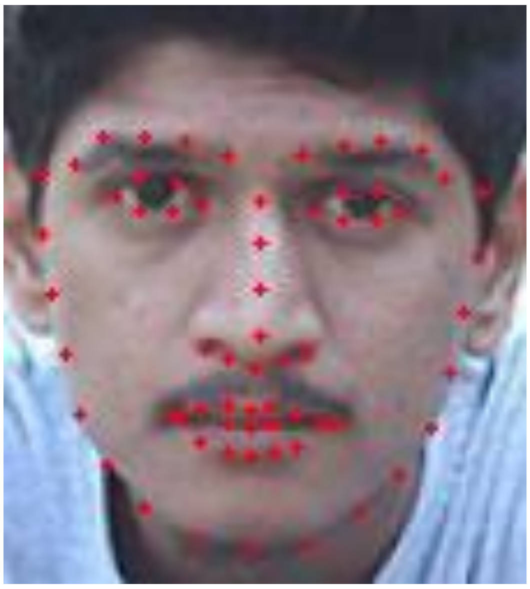


The next step is to apply face detection. Here we've used a deep learning method to perform face detection with OpenCV.

Once we know *where* in the image the face is, we can extract the face Region of Interest (ROI):

The next step is to extract the face ROI with OpenCV and NumPy slicing.

And from there, we apply facial landmarks, allowing us to localize the eyes, nose, mouth, etc.:



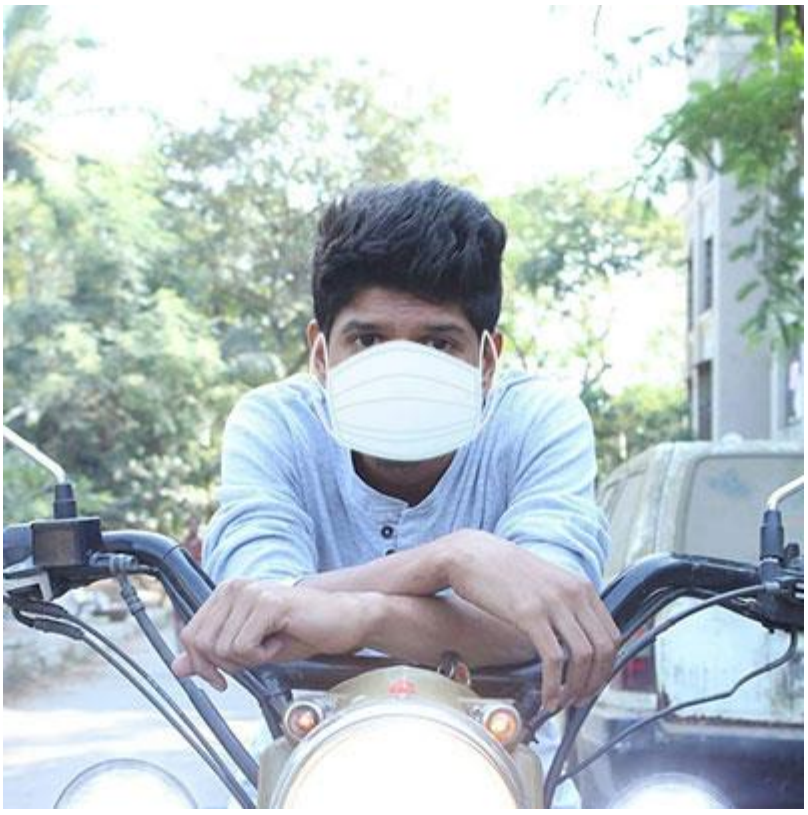
Then, we **detect facial landmarks** using dlib so that we know where to place a mask on the face

Next, we need an image of a mask (with a transparent background) such as the one below:

An example of a COVID-19/Coronavirus face mask/shield. This face mask will be overlaid on the original face ROI automatically since we know the face landmark locations.

This mask will be *automatically* applied to the face by using the facial landmarks (namely the points along the chin and nose) to compute *where* the mask will be placed.

The mask is then resized and rotated, placing it on the face:



In this figure, the face mask is placed on the person’s face in the original frame. It is difficult to tell at a glance that the COVID-19 mask has been applied with computer vision by way of OpenCV and dlib face landmarks.

We can then repeat this process for all of our input images, thereby creating our artificial face mask dataset:

An artificial set of COVID-19 face mask images is shown. This set will be part of our "with mask" / "without mask" dataset for COVID-19 face mask detection with computer vision and deep learning using Python, OpenCV, and TensorFlow/Keras.

If you use a set of images to create an artificial dataset of people wearing masks, **you** *cannot* **"re-use" the images** *without masks* in your training set — you still need to gather non-face mask images that were *not* used in the artificial generation process!

Data Processing

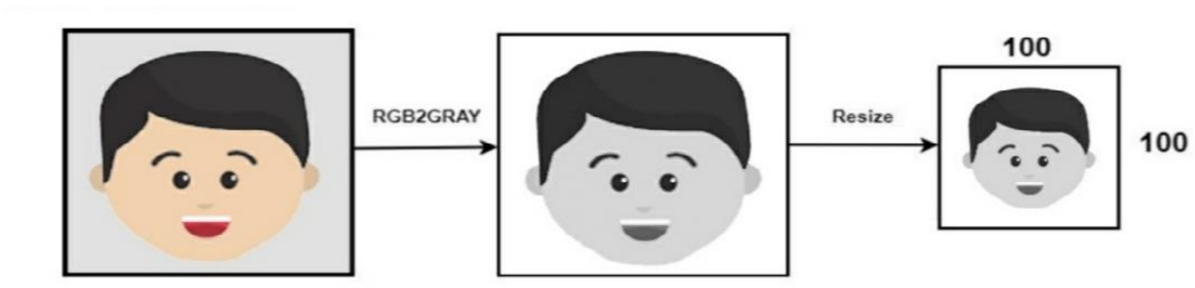
Data preprocessing involves conversion of data from a given format to much more user friendly, desired and meaningful format. It can be in any form like tables, images, videos, graphs, etc. These organized information fit in with an information model or composition and captures relationship between different entities. The proposed method deals with image and video data using Numpy and OpenCV.

Data Visualization

Data visualization is the process of transforming abstract data to meaningful representations using knowledge communication and insight discovery through encodings. It is helpful to study a particular pattern in the dataset. The total number of images in the dataset is visualized in both categories – ‘with mask’ and ‘without mask’.
The statement categories=os.listdir(data path) categorizes the list of directories in the specified data path. The variable categories now looks like: [‘with mask’, ‘without mask’]
Then to find the number of labels, we need to distinguish those categories using labels=[i for i in range(len(categories))]. It sets the labels as: [0, 1]
Now, each category is mapped to its respective label using label dict=dict(zip(categories,labels)) which at first returns an iterator of tuples in the form of zip object where the items in each passed iterator is paired together consequently. The mapped variable label dict looks like: f‘with mask’: 0, ‘without mask’: 1g

Conversion of RGB image to Gray image

Modern descriptor-based image recognition systems regularly work on grayscale images, without elaborating the method used to convert from color-to-grayscale. This is because the color to- grayscale method is of little consequence when using robust descriptors. Introducing nonessential information could increase the size of training data required to achieve good performance. As grayscale rationalizes the algorithm and diminishes the computational requisites, it is utilized for extracting descriptors instead of working on color images instantaneously.



We use the function cv2.cvtColor(input image, flag) for changing the color space. Here flag determines the type of conversion. In this case, the flag cv2.COLOR BGR2GRAY is used for gray conversion.

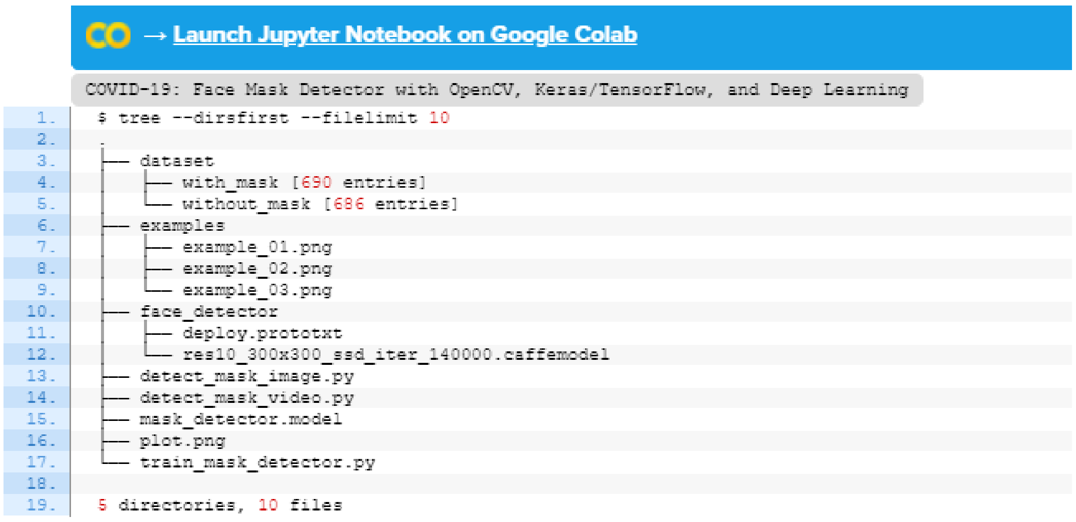
Training of Model

CNN has become ascendant in miscellaneous computer vision tasks. The current method makes use of Sequential CNN. The First Convolution layer is followed by Rectified Linear Unit (ReLU) and MaxPooling layers. The Convolution layer learns from 200 filters. Kernel size is set to 3 x 3 which specifies the height and width of the 2D convolution window. As the model

should be aware of the shape of the input expected, the first layer in the model needs to be provided
with information about input shape. Following layers can perform instinctive shape reckoning [13]. In this case, input shape is specified as data.shape[1:] which returns the dimensions of the data array from index 1. Default padding is "valid" where the spatial dimensions are sanctioned to truncate and the input volume is non-zero padded. The activation parameter to the Conv2D class is set as "relu".

It represents an approximately linear function that possesses all the assets of linear models that can easily be optimized with gradient-descent methods. Considering the performance and generalization in deep learning, it is better compared to other activation functions. Max Pooling is used to reduce the spatial dimensions of the output volume. Pool size is set to 3 x 3 and the resulting output has a shape (number of rows or columns) of: shape of output = (input shape - pool size + 1) / strides), where strides has default value (1,1) [15]. As shown in fig, 4, the second Convolution layer has 100 filters and Kernel size is set to 3 x 3. It is followed by ReLu and MaxPooling layers. To insert the data into CNN, the long vector of input is passed through a Flatten layer which transforms matrix of features into a vector that can be fed into a fully connected neural network classifier.

PROJECT STRUCTURE



COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, and Deep Learning

```
$ tree --dirsfirst --filelimit 10
.
├── dataset
│   ├── with_mask [690 entries]
│   └── without_mask [686 entries]
├── examples
│   ├── example_01.png
│   ├── example_02.png
│   └── example_03.png
├── face_detector
│   ├── deploy.prototxt
│   └── res10_300x300_ssd_iter_140000.caffemodel
├── detect_mask_image.py
├── detect_mask_video.py
├── mask_detector.model
├── plot.png
└── train_mask_detector.py

5 directories, 10 files
```

The dataset/ directory contains the data described in the *"Our COVID-19 face mask detection dataset"* section.

Three image examples/ are provided so that you can test the static image face mask detector.

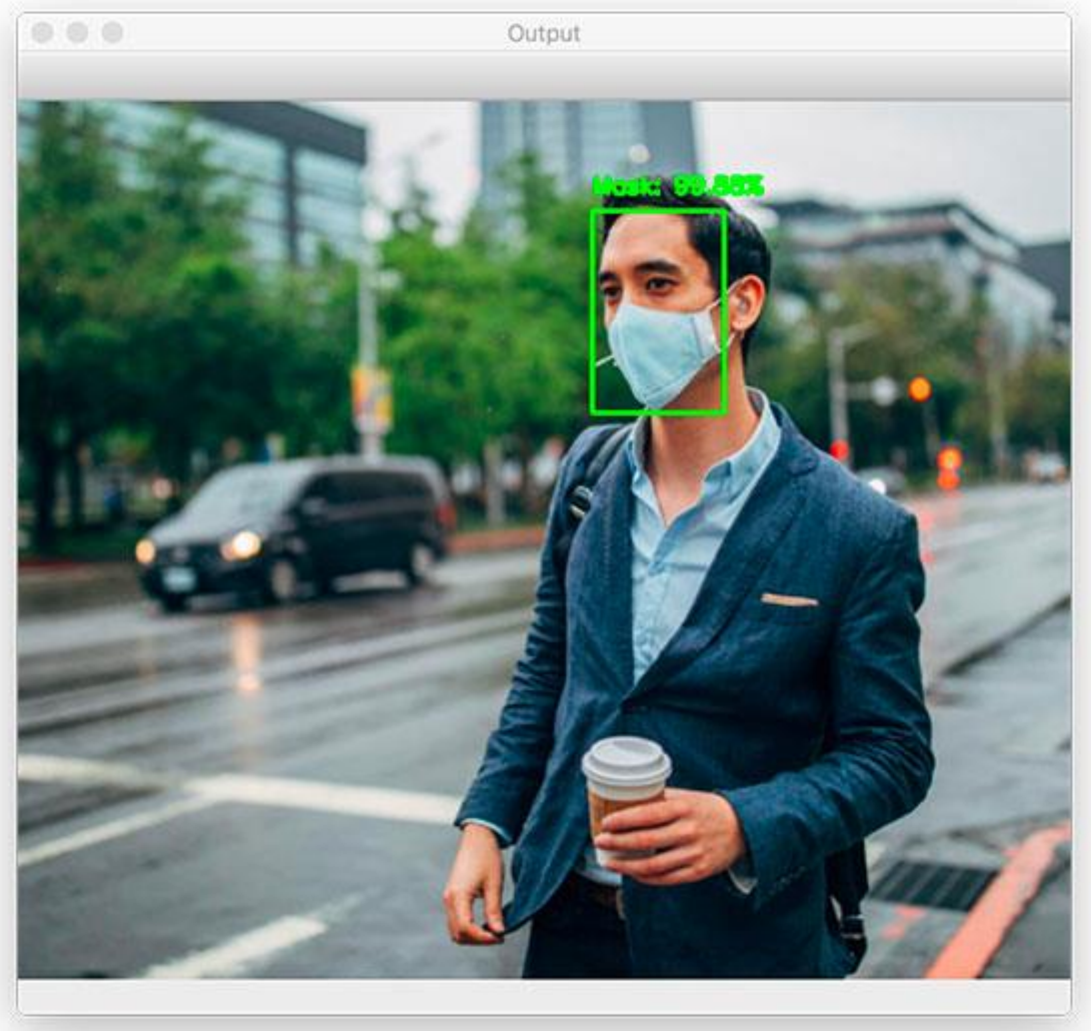
We'll be reviewing three Python scripts in this PROJECT:

- **train_mask_detector.py**: Accepts our input dataset and fine-tunes MobileNetV2 upon it to create our mask_detector.model. A training history plot.png containing accuracy/loss curves is also produced

- **detect_mask_image.py:** Performs face mask detection in static images.

- **detect_mask_video.py**: Using your webcam, this script applies face mask detection to every frame in the stream

COVID-19 face mask detection with OpenCV

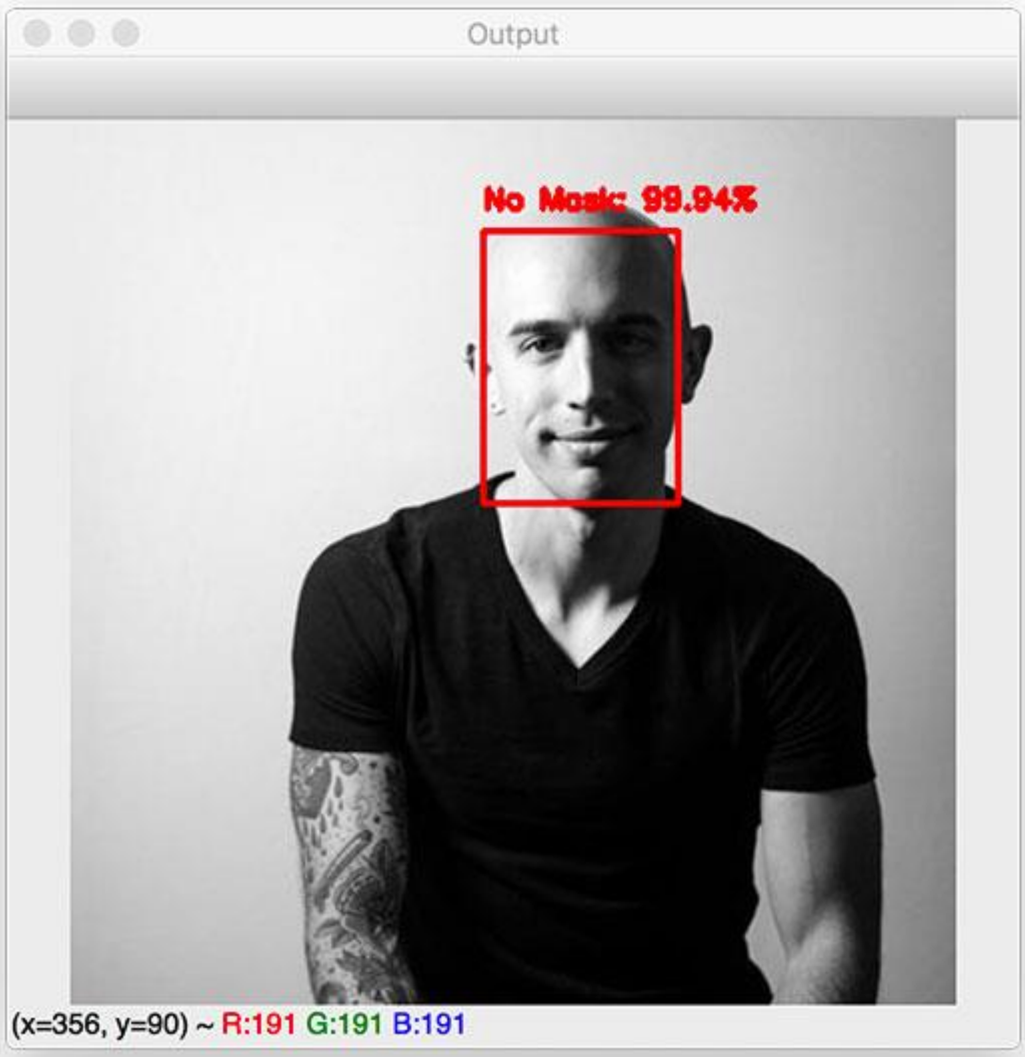


Is this man wearing a COVID-19/Coronavirus face mask in public? Yes, he is and our computer vision and deep learning method using Python, OpenCV, and TensorFlow/Keras has made it possible to detect the presence of the mask automatically

As you can see, our face mask detector correctly labeled this image as **Mask**

.

Let’s try another image, this one of a person *not* wearing a face mask:



Uh oh. I’m not wearing a COVID-19 face mask in this picture. Using Python, OpenCV, and TensorFlow/Keras, our system has correctly detected “No Mask” for my face.

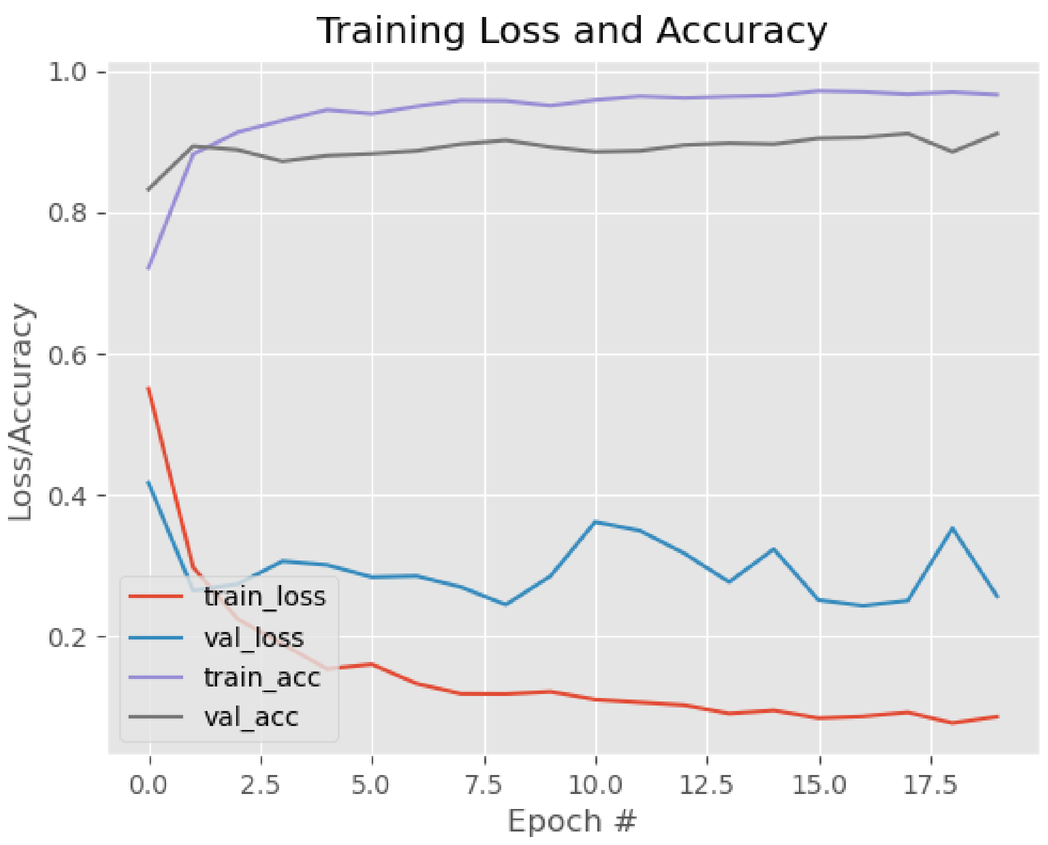
Our face mask detector has correctly predicted
No Mask

.

CHAPTER 6

RESULT & ANALYSIS

RESULT ANALYSIS

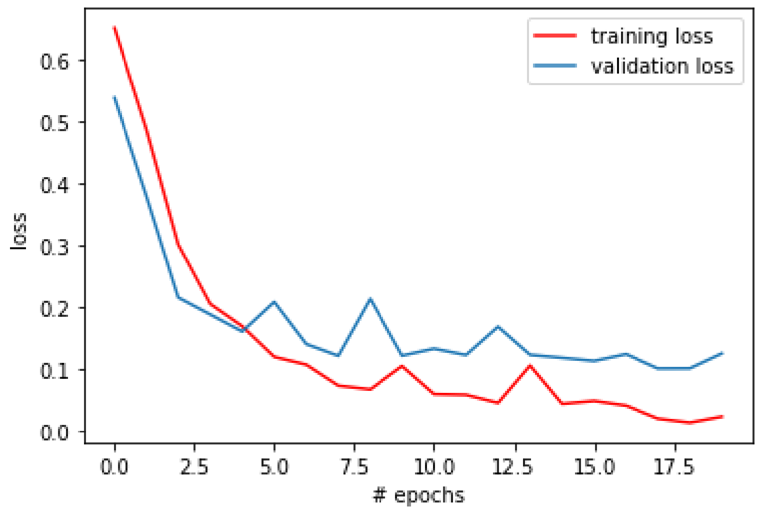


As you can see, we are obtaining **~99% accuracy** on our test set.

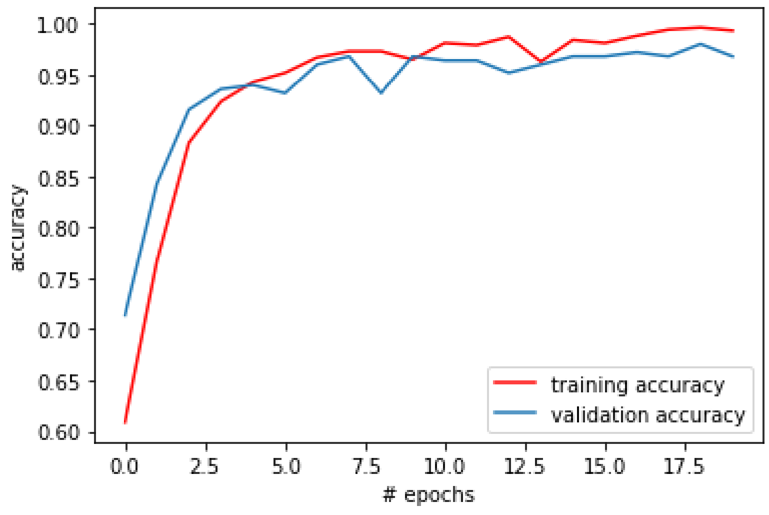
The model is trained, validated and tested upon two datasets. Corresponding to dataset 1, the method attains accuracy up to 95.77% depicts how this optimized accuracy mitigates the cost of error. Dataset 2 is more versatile than dataset 1 as it has multiple faces in the frame and different types of masks having different colors as well. Therefore, the model attains an accuracy of 99% on dataset 2 as shown in Fig. depicts the contrast between training and validation loss corresponding to dataset 2. 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.

Number of neurons has the optimized value of 64 which is not too high. 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 main portion (face) of the image to detect the existence of mask correctly without causing over-fitting.

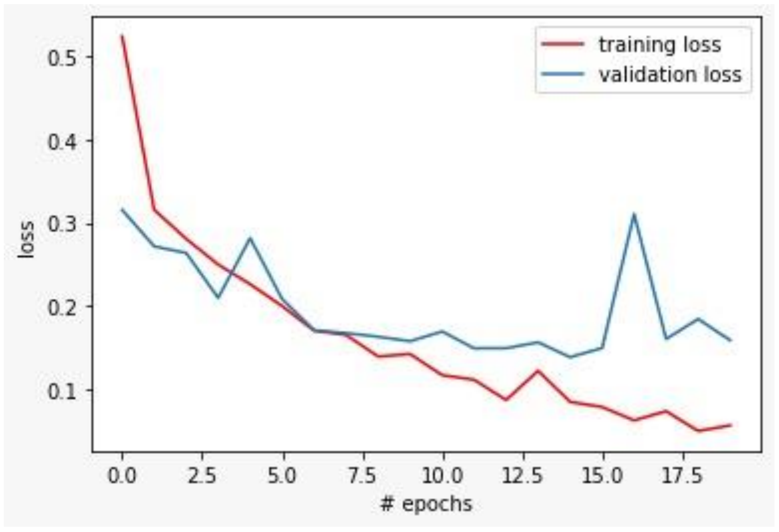


Epochs vs loss corresponding to dataset 1

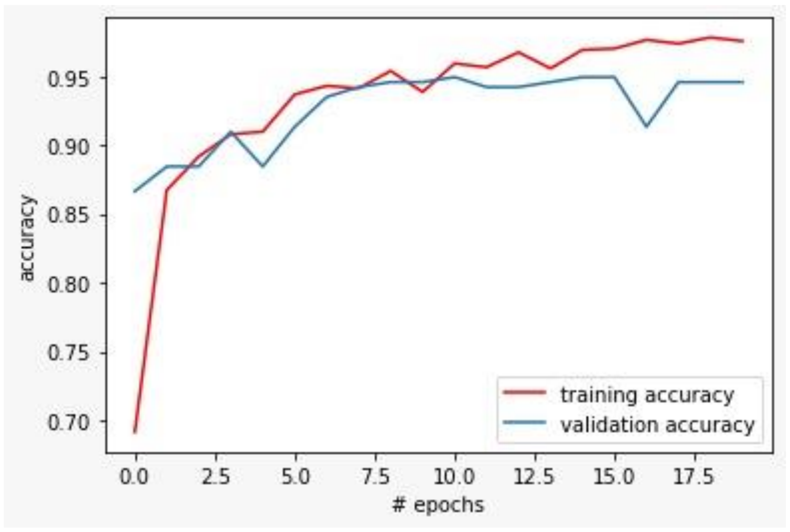


Epoch’s vs accuracy corresponding to dataset 1

The system can efficiently detect partially occluded faces either with a mask or hair or hand. It considers the occlusion degree of four regions – nose, mouth, chin and eye to differentiate 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.



Epoch’s vs loss corresponding to dataset 2



Epoch’s vs accuracy corresponding to dataset 2

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

CHAPTER 7

CONCLUSION & FUTURE SCOPE

CONCLUSION

To mitigate the spread of COVID-19 pandemic, measures must be taken. We created a model using the datasets from Kaggle and we developed a Face Mask Detector that detects whether a person on the camera is wearing a Face Mask or not. We trained our Face Mask Detector model using Keras/Tensor flow. From this we can conclude that, our model has the accuracy of 99%. It can be used for a variety of applications. Wearing a mask may be obligatory in the near future, considering the Covid-19 crisis. Many public service providers will ask the customers to wear masks correctly to avail of their services. The deployed model will contribute immensely to the public health care system. In future it can be extended to detect if a person is wearing the mask properly or not. The model can be further improved to detect if the mask is virus prone or not i.e. the type of the mask is surgical, N95 or not.

FUTURE SCOPE

- In many countries, wearing a mask is compulsory so people have to wear a mask in public places, stores, schools etc. They also like to take measures impressions on digital displays.

- We can work on our software to use in IP cameras and CCTV cameras to detect people without a mask.

- Software operators can also get a notification message or image in case someone is not wearing a mask.

- We can also work to implement an alarm system so that someone without a mask enters in area it will beep. It can be connected to entrance gates.

CHAPTER 8

REFERENCES

REFERENCES

1) W.H.O., “Coronavirus disease 2019 (covid-19): situation report, 205”.
    a. 2020

2) “Coronavirus Disease 2019 (COVID-19) – Symptoms”,
    a. Centers for Disease Control and Prevention, 2020. [Online].
    b. Available: https://www.cdc.gov/coronavirus/2019-ncov/symptomstesting/
    c. symptoms.html. 2020.

3) “Coronavirus — Human Coronavirus Types — CDC”, Cdc.gov, 2020.
    a. [Online]. Available: https://www.cdc.gov/coronavirus/types.html. 2020.

4) "Face Mask Detection", 2020, [online] Available: Kaggle.com

5) "Advice on the use of masks in the context of COVID-19: interim guidance", 2020.

6) M. Jiang, X. Fan and H. Yan, "RetinaMask: A Face Mask detector", 2020, [online] Available: arXiv.org.

7) "Changing Colorspaces — Opencv-Python Tutorials 1 Documentation", 2020, [online] Available: Opencv-python-tutroals.readthedocs.io.

8) "Keras documentation: About Keras", 2020, [online] Available : Keras.io.

9) B. QIN, D. LI, identifying facemask-wearing condition using image super-resolution with classification network to prevent covid-19 (2020).

10) W.H.O., “Advice on the use of masks in the context of COVID-19: interim guidance”, 2020.

11) M. Jiang, X. Fan and H. Yan, “RetinaMask: A Face Mask detector”,
    a. arXiv.org, 2020. [Online]. Available: https://arxiv.org/abs/2005.03950.
    b. 2020.

12) B. Suvarna mukhi and M. Seshashayee, "Big Data Concepts and
    a. Techniques in Data Processing", International Journal of Computer
    b. Sciences and Engineering, vol. 6, no. 10, pp. 712-714, 2018. Available:
    c. 10.26438/ijcse/v6i10.712714.

13) F. Hohman, M. Kahng, R. Pienta and D. H. Chau, "Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers," in
    a. IEEE Transactions on Visualization and Computer Graphics, vol. 25,
    b. No. 8, pp. 2674-2693, 1 Aug. 2019, doi: 10.1109/TVCG.2018.2843369.

14) C. Kanan and G. Cottrell, "Color-to-Grayscale: Does the Method
    a. Matter in Image Recognition?", PLoS ONE, vol. 7, no. 1, p. e29740,
    b. 2012. Available: 10.1371/journal.pone.0029740.

15) Opencv-python-tutroals.readthedocs.io. 2020. Changing Colorspaces
    a. Opencv-Python Tutorials 1 Documentation. [online] Available
    b. at:https://opencv-python-tutroals.readthedocs.io/en/latest/py tutorials/
    c. py imgproc/py colorspaces/py colorspaces.html. 2020.

16) M. Hashemi, "Enlarging smaller images before inputting into convolutional
    a. neural network: zero-padding vs. interpolation", Journal of Big
    b. Data, vol. 6, no. 1, 2019. Available: 10.1186/s40537-019-0263-7. 2020.

17) S. Ghosh, N. Das and M. Nasipuri, "Reshaping inputs for convolutional
    a. neural network: Some common and uncommon methods",
    b. Pattern Recognition, vol. 93, pp. 79-94, 2019. Available: 10.1016/j.patcog.2019.04.009.

18) R. Yamashita, M. Nishio, R. Do and K. Togashi, "Convolutional neural
    a. networks: an overview and application in radiology", Insights into
    b. Imaging, vol. 9, no. 4, pp. 611-629, 2018. Available: 10.1007/s13244-
    c. 018-0639-9.

19) "Guide to the Sequential model - Keras Documentation", Faroit.com,
    a. 2020. [Online]. Available:
    b. https://faroit.com/kerasdocs/1.0.1/gettingstarted/sequential-model-guide/. 2020.

20) Nwankpa, C., Ijomah, W., Gachagan, A. and Marshall, S., 2020.
    a. Activation Functions: Comparison of Trends in Practice And
    b. Research for Deep Learning. [Online] arXiv.org. Available at:
    c. https://arxiv.org/abs/1811.03378. 2020.

21) K. Team, "Keras documentation: MaxPooling2D layer", Keras.io,
    a. 2020. [Online]. Available: https://keras.io/api/layers/pooling layers/ max pooling2d/. 2020.