

**To detect whether a person is wearing a face mask or not,
using Machine Learning algorithms, with the help of
technologies like Keras and OpenCV**

Minor Project I

Submitted by:

**Samarth Sajwan (9918103022)
Rudraksh Bhardwaj (9918103020)
Revaan Mishra (9918103016)**

Under the supervision of:
Dr. Shruti Jaiswal



**Department of CSE/IT
Jaypee Institute of Information Technology University, Noida**

November 2020

ACKNOWLEDGEMENT

I would like to place on record my deep sense of gratitude to Dr Shruti Jaiswal, Assistant Professor, Jaypee Institute of Information Technology, India for her generous guidance, help and useful suggestions.

I express my sincere gratitude to Dr Mukesh Saraswat, Dept. of Computer Science and Engineering, for his stimulating guidance, continuous encouragement and supervision throughout the course of present work.

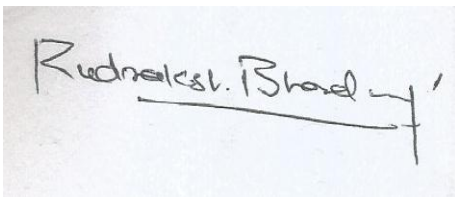
I also wish to extend my thanks to Hare Krishna and other classmates for their insightful comments and constructive suggestions to improve the quality of this project work.

Signature(s) of Students

Samarth Sajwan (9918103022)

A handwritten signature in black ink, appearing to be 'Samarth Sajwan', written on a light-colored background.

Rudraksh Bhardwaj (9918103020)

A handwritten signature in black ink, appearing to be 'Rudraksh Bhardwaj', written on a light-colored background.

Revaan Mishra (9918103016)

A handwritten signature in blue ink, appearing to be 'Revaan Mishra', written on a light-colored background.

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and beliefs, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma from a university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: Jaypee Institute of Information Technology, Noida

Date: 29 November, 2020

Name: Samarth Sajwan

Enrolment No.: 9918103022

Name: Rudraksh Bhardwaj

Enrolment No.: 9918103020

Name: Revaan Mishra

Enrolment No.: 9918103016

CERTIFICATE

This is to certify that the work titled “To detect whether a person is wearing a face mask or not , using machine learning algorithms , with the help of technologies like Keras and OpenCV ” submitted by Samarth Sajwan, Rudraksh Bhardwaj and Revaan Mishra of B.Tech of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of any other degree or diploma.

Digital Signature of Supervisor

Dr. Shruti Jaiswal

Assistant Professor

29 November, 2020

Abstract

The World Health Organization (WHO) reports suggest that the two main routes of transmission of the COVID-19 virus are **respiratory droplets** and **physical contact**.

Respiratory droplets are generated when an infected person coughs or sneezes. Any person in close contact (within 1 m) with someone who has respiratory symptoms (coughing, sneezing) is at risk of being exposed to potentially infective respiratory droplets.

Droplets may also land on surfaces where the virus could remain viable; thus, the immediate environment of an infected individual can serve as a source of transmission (contact transmission). Wearing a medical mask is one of the prevention measures that can limit the spread of certain respiratory viral diseases, including COVID-19.

We wish to apply a Machine Learning algorithm to create a model which will determine the presence of a mask. This project can be used in various places as it is now mandatory to wear masks in public places.

In our project, medical masks are defined as surgical or procedure masks that are flat or pleated (some are shaped like cups); they are affixed to the head with straps. The project will use a set of video streams/images to identify people who are compliant with the government rule of wearing medical masks.

This could help the government to take appropriate action against people who are non-compliant.

Table of Contents

| | Page No. |
|--|--------------|
| <i>Abstract</i> | <i>i</i> |
| <i>Table of Contents</i> | <i>ii</i> |
| <i>List of Figures</i> | <i>iv</i> |
| | |
| Chapter 1: INTRODUCTION | 2 |
| | |
| Chapter 2: BACKGROUND STUDY | 3 |
| | |
| Chapter 3: REQUIREMENT ANALYSIS | |
| 3.1 FUNCTIONAL REQUIREMENTS | 4 |
| 3.2 HARDWARE REQUIREMENTS | 5 |
| 3.3 SOFTWARE REQUIREMENTS | |
| | |
| Chapter 4: DETAILED DESIGN | 6-9 |
| 4.1 FLOCHART | 6 |
| 4.2 PRE-PROCESSING | 7 |
| 4.3 ALGORITHMS | 7 |
| 4.4 MOBILENETV2 | 8 |
| 4.5 TRAINING THE MODEL | 9 |
| 4.6. USING THE MODEL IN A LIVEFEED | 9 |
| | |
| Chapter 5: IMPLEMENTATION | 10-13 |
| 5.1 DATA PREPRCOESSING | 10 |

| | |
|---|--------------|
| 5.2 PLOTTING ACCURACY | 11 |
| 5.3 CREATING LIVE SETUP USING OPENCV | 12 |
| 5.4 LIVE CAMERA IMPLEMENTATION | 13 |
| Chapter 6: EXPERIMENTAL RESULTS AND ANALYSIS | 14-15 |
| 6.1 RESULTS WHILE WEARING MASKS | 14 |
| 6.2 RESULTS WITHOUT WEARING MASKS | 14-15 |
| 6.3 ACCURACY ANALYSIS | 15 |
| Chapter 7: CONCLUSION AND FURTHER SCOPE | 16-18 |
| 7.1 GANTT CHART | 16 |
| 7.2 PLOTTING ACCURACY | 17 |
| 7.3 RESULTS WITH/WITHOUT MASKS | 17-18 |
| 7.4 FUTURE SCOPE | 18 |
| Chapter 8: REFERENCES | 19-20 |

List of Figures

Fig.1. RMFD dataset images samples.

Fig.2. Flowchart for the work done

Fig.3. Layer Structure in MobileNetV2

Fig.4. Learning Process of The Model

Fig.5. Using SciKit to Split the Dataset

Fig.6. Splitting the Dataset to 20% Training Set and 80% Test Set

Fig.7. Using Matplotlib for obtaining results

Fig.8. Processing of Results

Fig.9. Plotting the accuracy Graphs obtained

Fig.10. Face Detection Model

Fig.11. Loading the Mask Model

Fig.12. Implementing the Model in the Camera Setup

Fig.13. Loading of the Face Mask Detector

Fig.14. The live camera setup

Fig.15. Test while wearing Face Masks

Fig.16. Test while wearing Face Masks

Fig.17. Test without wearing Face Masks

Fig.18. Test without wearing Face Masks

Fig.19. Values at the Last Epoch

1.Introduction

The COVID-19 pandemic has reshaped life as we know it. Many of us are staying home, avoiding people on the street and changing daily habits, like going to school or work, in ways we never imagined.

While we are changing old behaviors, there are new routines we need to adopt. First and foremost is the habit of wearing a mask or face covering whenever we are in a public space.

The government of India has also made face masks mandatory for people, while not wearing one is an offence punishable by up to six months in prison.

The importance of masks can be summed up by the following points

- **Masks and face coverings can prevent the wearer from transmitting the COVID-19 virus to others and may provide some protection to the wearer.** Multiple studies have shown that face coverings can contain droplets expelled from the wearer, which are responsible for the majority of transmission of the virus.
- Many people with COVID-19 are unaware they are carrying the virus. It is estimated that 40% of persons with COVID-19 are asymptomatic but potentially able to transmit the virus to others.
- Universal mask use can significantly reduce virus transmission in the community by preventing anyone, including those who are unwittingly carrying the virus, from transmitting it to others. Disease modeling suggests masks worn by significant portions of the population, coupled with other measures, could result in substantial reductions in case numbers and deaths.

Hence, we have chosen this topic in order to help this cause, and detecting whether someone is wearing a mask or not can be helpful to many organizations in determining whether this law is being followed or not.

We wish to apply a Machine Learning algorithm to create a model which will determine the presence of a mask. This project can be used in various places as it is now mandatory to wear masks in public places.

This algorithm can also be used in a live camera, so that it can directly detect the people who are not following this law via CCTV cameras.

2. Background Study

Generally, most of the publication focus is on face construction and identity recognition when wearing face masks. In this research our focus is on recognizing the people who are not wearing face masks to help in decreasing the transmission and spreading of the COVID-19. Researchers and scientists have proved that wearing face masks help in minimizing the spreading rate of COVID-19.

In [1], the authors developed a new facemask-wearing condition identification method. They were able to classify three categories of facemask-wearing conditions. The categories are correct facemask-wearing, incorrect facemask-wearing, and no facemask-wearing. The proposed method has achieved 98.70% accuracy in the face detection phase. Sabbir et al [2], have applied the Principal Component Analysis (PCA) on masked and unmasked face recognition to recognize the person. They found that the accuracy of face recognition using the PCA is extremely affected by wearing masks. The recognition accuracy drops to less than 70% when the recognized face is masked. Also, PCA was used in [3]. The authors proposed a method that is used for removing glasses from a human frontal facial image. The removed part was reconstructed using recursive error compensation using PCA reconstruction.

In [4], the authors used the YOLOv3 algorithm for face detection. YOLOv3 uses Darknet-53 as the backbone. The proposed method achieved 93.9% accuracy. It was trained on CelebA and WIDER FACE dataset including more than 600,000 images. The testing was the Fddb dataset. Nizam et al [5] proposed a novel GAN-based network that can automatically remove masks covering the face area and regenerate the image by building the missing hole. The output of the proposed model is a complete face image that looks natural and realistic.

In [6], the authors presented a system for detecting the presence or absence of a compulsory medical mask in the operating room. The overall objective is to minimize the false positive face detections as possible without missing mask detections in order to trigger alarms only for medical staff who do not wear a surgical mask. The proposed system achieved 95% accuracy.

Muhammad et al [7] presented an interactive method called MRGAN. The method depends on getting the microphone area from the user and using the Generative Adversarial Network to rebuild this area. Shaik et al [8] used deep learning real-time face emotion classification and recognition. They used VGG-16 to classify seven facial expressions. The proposed model was trained on the KDEF dataset and achieved 88% accuracy.

3.Requirement Analysis

3.1. Functional Requirements

- Our goal is to detect whether a person is wearing a face mask or not using technologies like **Keras** for data manipulation and **OpenCV** to implement our model into a live feed.
- MobileNetV2 for training the model
- We will try to use the following ML Algorithms to obtain a model with the most accurate results for face mask detection:

-KNN

-SVR

-Decision Tree Regression

Dataset Characteristics:

This research conducted its experiments on three original datasets. The first dataset is Real-World Masked Face Dataset (RMFD). The author of RMFD created one of the biggest masked face datasets used in this research. The RMFD dataset consists of 5000 masked faces and 90,000 unmasked faces. Fig. 1 illustrates samples of faces with and without masks. In this research, 5000 images for faces with masks and without masks have been used with a total of 10,000 images to balance the dataset. The RMFD dataset used for the training, validation, and testing phases.



Fig.1. RMFD dataset images samples.

3.2. Hardware Requirements

- Processors: Intel® Core™ i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2 cores, 2 threads per core), 8 GB of DRAM Intel® Xeon® processor E5-2698 v3 at 2.30 GHz (2 sockets, 16 cores each, 1 thread per core), 64 GB of DRAM Intel® Xeon Phi™ processor 7210 at 1.30 GHz (1 socket, 64 cores, 4 threads per core), 32 GB of DRAM, 16 GB of MCDRAM (flat mode enabled)
- Disk space: 2 to 3 GB
- A suitable webcam for the face mask detection.

3.3. Software Requirements

- Operating systems: Windows* 7 or later, macOS, and Linux
- Python* versions: 2.7.X, 3.6.
- Jupyter Notebook
- Python Modules: NumPy, Keras (TensorFlow), OpenCV

4.DETAILED DESIGN

4.1. FLOWCHART

The flowchart represents the flow of work we had decided, which will be implemented as shown in Figure 2.

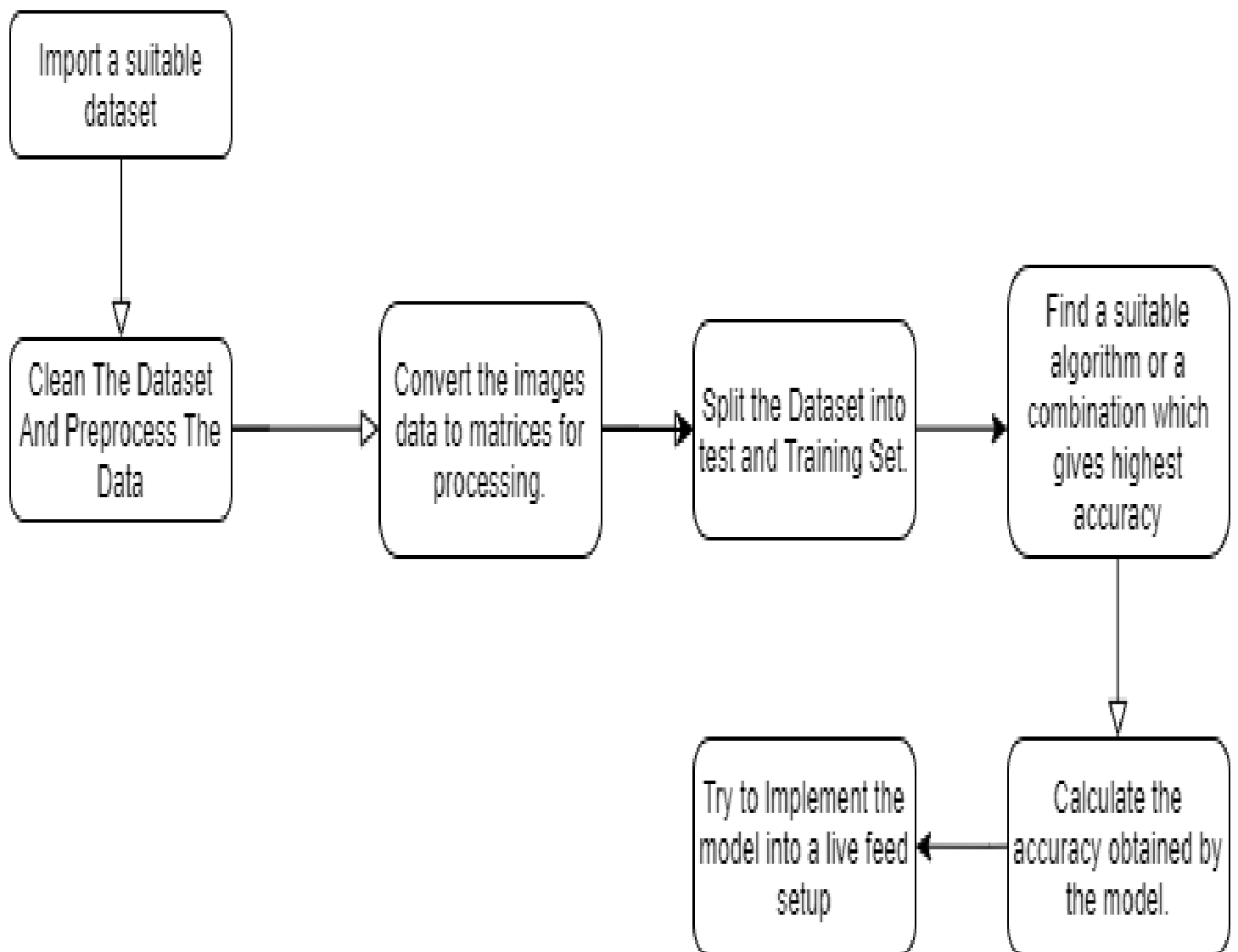


Fig.2. Flowchart for the work done

4.2. Pre- Processing the Dataset

First, we have to import a suitable dataset to work on, cleaning the data, and pre-process it so that it can be worked upon by the machine learning algorithms.

We can pre-process the data by using the following steps:

- Load an input image from disk
- Detect faces in the image
- Apply our face mask detector to classify the face as either with_mask or without_mask.

4.3. The methods and algorithms of Machine Learning we have decided to apply on our dataset are:

1. Support vector Machine

One of the most popular and spectacular supervised learning techniques with related learning algorithms for treatment classification and regression tasks in patterns is SVM. SVM is a classification machine learning algorithm based on hinge function as shown in Fig 2., where z is a label from 0 to 1, $w \cdot I - b$ is the output, w and b are coefficients of linear classification, and I is an input vector. The loss function to be minimized can be implemented below:

$$h_j = \max(0, 1 - z_j(w \cdot I_j - b))$$

$$loss = \frac{1}{n} \sum_{i=1}^n \max(0, h_i)$$

2. Decision tree

The decision tree is the classification model of computation based on entropy function and information gain. Entropy computes the amount of uncertainty in data as shown in Fig.3. Where D is current data, and q is a binary label from 0 to 1, and $p(x)$ is the proportion of q label. To measure the difference of entropy from data, we calculate information gain (I) as illustrated below:

$$E(D) = \sum_{i=1}^m -p(q_i) \cdot \log(p(q_i))$$

$$I = E(D) - \sum_{v \in D} p(v) E(v)$$

4.4 MobileNetV2

MobileNetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers as shown in Figure 3. The intermediate expansion layer uses lightweight depth wise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters as shown in Figure 4 , followed by 19 residual bottleneck layers.

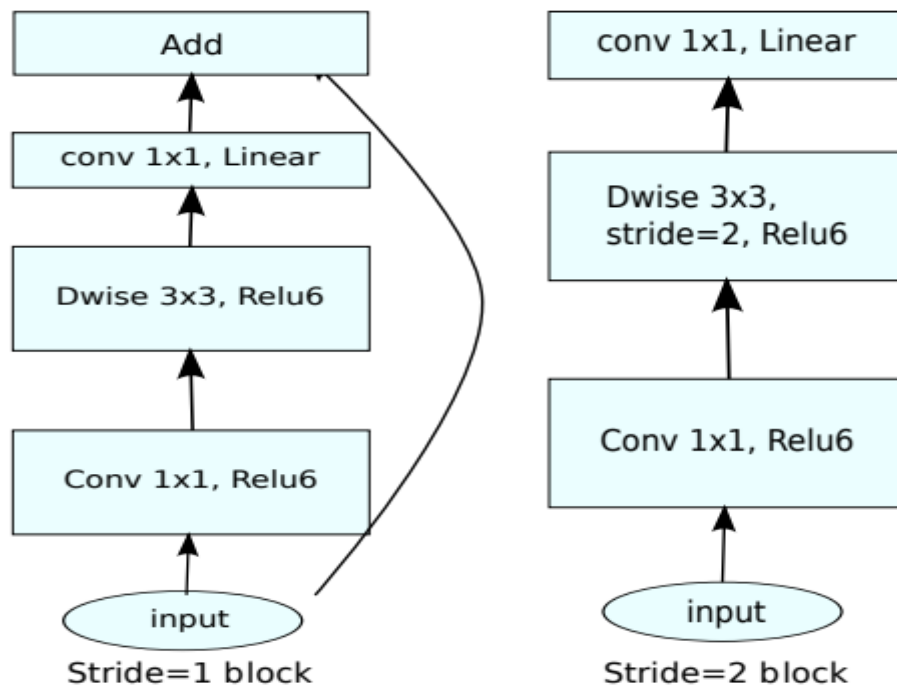


Fig.3. Layer Structure in MobileNetV2

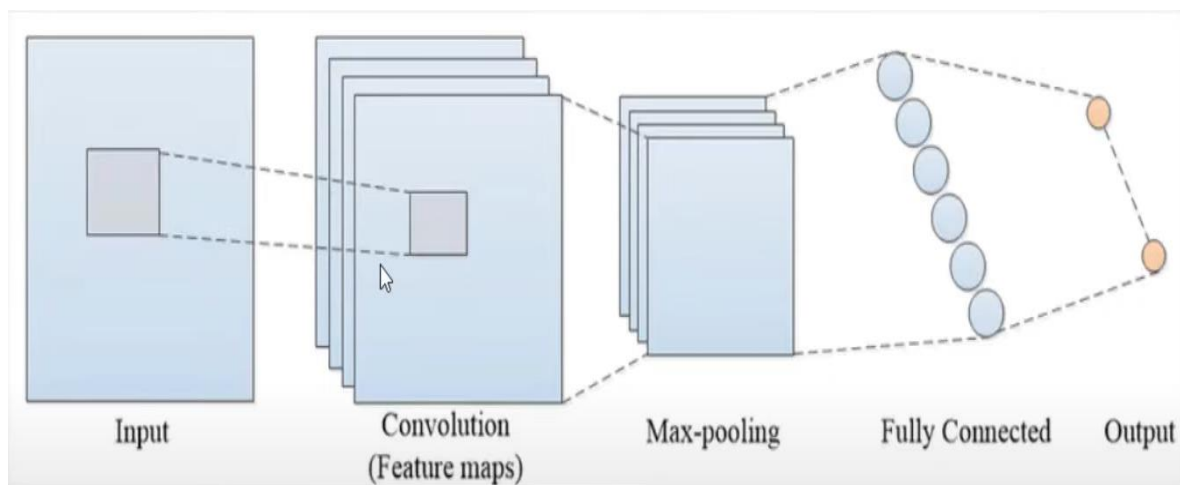


Fig.4. Learning Process of The Model

4.5. Training the Model using Algorithms

- We have to train the model on the training set based on the suitable algorithms selected, and plot the graph of the results obtained to determine the accuracy of the model obtained.
- We will try to use an algorithm like Decision Tree Classification, Support Vector Machine, KNN, or something close to it which gives us the best accuracy while working our test set.

4.6. Implementing the Model into A Live Camera Feed using OpenCV

- Implementing the model that we have created into a real time camera setup, so that it can detect the presence of a face mask through a live feed.
- Implement this model into a live camera setup so that the model can detect whether someone is wearing a mask or not through the live feed of a camera, and can instantly report any abnormalities.
- This is a very useful feature, as it can help the authorities directly check that whether someone is wearing a mask or not.

5.IMPLEMENTATION

1)We use the train_test_split Function of the SciKit Library to split our data set into two halves the Training Set and the Test Set.

```
57
58 (trainX, testX, trainY, testY) = train_test_split(data, labels,
59     test_size=0.20, stratify=labels, random_state=42)
60
```

Fig .5. Using SciKit to Split the Dataset

2)We have kept 20% of our data for the training of our model and the remaining 80% is used for the testing.

```
test_size=0.20,
```

Fig.6. Splitting the Dataset to 20% Training Set and 80% Test Set

3)Using our current model we have tried to plot the results using the Matplotlib library of Python.

```
124
125 # plot the training loss and accuracy
126 N = EPOCHS
127 plt.style.use("ggplot")
128 plt.figure()
129 plt.plot(np.arange(0, N), H.history["loss"], Label="train_loss")
130 plt.plot(np.arange(0, N), H.history["val_loss"], Label="val_loss")
131 plt.plot(np.arange(0, N), H.history["accuracy"], Label="train_acc")
132 plt.plot(np.arange(0, N), H.history["val_accuracy"], Label="val_acc")
133 plt.title("Training Loss and Accuracy")
134 plt.xlabel("Epoch #")
135 plt.ylabel("Loss/Accuracy")
136 plt.legend(loc="lower left")
137 plt.savefig("plot.png")
```

Fig.7. Using Matplotlib for obtaining results

5)After running this Py Code above:

```

2020-07-06 20:07:48.769567: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 39 of 96
2020-07-06 20:07:59.232221: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 46 of 96
2020-07-06 20:08:10.127723: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 51 of 96
2020-07-06 20:08:18.886399: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 57 of 96
2020-07-06 20:08:30.286220: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 63 of 96
2020-07-06 20:08:39.371208: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 68 of 96
2020-07-06 20:08:50.142614: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 73 of 96
2020-07-06 20:08:58.645802: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 78 of 96
2020-07-06 20:09:09.391125: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 86 of 96
2020-07-06 20:09:19.076816: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 94 of 96
2020-07-06 20:09:21.431713: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:193] Shuffle buffer filled.
95/95 [=====] - 579s 6s/step - loss: 0.0774 - accuracy: 0.9703 - val_loss: 0.3531 - val_accuracy: 0.8859
Epoch 20/20
2020-07-06 20:16:18.689748: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 9 of 96
2020-07-06 20:16:28.288819: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 15 of 96
2020-07-06 20:16:37.407941: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 21 of 96
2020-07-06 20:16:49.335176: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 26 of 96
2020-07-06 20:16:59.649703: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 29 of 96
2020-07-06 20:17:10.204371: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 33 of 96
2020-07-06 20:17:18.282668: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 36 of 96
2020-07-06 20:17:27.780390: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 40 of 96
2020-07-06 20:17:37.750808: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 48 of 96
2020-07-06 20:17:47.769769: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 57 of 96
2020-07-06 20:17:57.540658: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 64 of 96
2020-07-06 20:18:07.461209: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 71 of 96
2020-07-06 20:18:17.928219: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 77 of 96
2020-07-06 20:18:27.921219: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 85 of 96
2020-07-06 20:18:39.155468: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:143] Filling up shuffle buffer (this may take a while): 92 of 96
2020-07-06 20:18:47.690327: I tensorflow/core/kernels/data/shuffle_dataset_op.cc:193] Shuffle buffer filled.
95/95 [=====] - 578s 6s/step - loss: 0.0866 - accuracy: 0.9667 - val_loss: 0.2567 - val_accuracy: 0.9117
[INFO] evaluating network...
      precision    recall  f1-score   support

 with_mask         1.00      0.83      0.90       383
without_mask        0.85      1.00      0.92       384

   accuracy
macro avg        0.92      0.91      0.91       767
weighted avg     0.92      0.91      0.91       767

[INFO] saving mask detector model...

```

Fig.8. Processing of Results

6)We get these results:



Fig.9. Plotting the accuracy Graphs obtained

5.3 For Video Detection of Face Mask

- 1) Using a Pre-existing Face Detection model, we detect the face of the subject

```
prototxtPath = r"face_detector\deploy.prototxt"  
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"  
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
```

Fig.10. Face Detection Model

- 2) Load the created Model.

```
maskNet = load_model("mask_detector.model")
```

Fig.11.Loading The Mask Model

- 3) Loop in the Detected face area and calculate the mask probability

```
for (box, pred) in zip(locs, preds):  
    (startX, startY, endX, endY) = box  
    (mask, withoutMask) = pred  
  
    label = "Mask" if mask > withoutMask else "No Mask"  
    color = (0, 255, 0) if label == "Mask" else (0, 0, 255)  
  
    label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)  
  
    cv2.putText(frame, label, (startX, startY - 10),  
                cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)  
    cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
```

Fig.12.Implementing the Model in the Camera Setup

5.3 Live Camera Implementation

- 4) The video stream starts which detects the presence of face masks.

```
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.  
2020-11-30 01:56:19.890725: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x2acc5a40a20 initialized for  
platform Host (this does not guarantee that XLA will be used). Devices:  
2020-11-30 01:56:19.902141: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device (0): Host, Default  
Version  
2020-11-30 01:56:19.909354: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1257] Device interconnect StreamExecutor  
with strength 1 edge matrix:  
2020-11-30 01:56:19.920058: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1263]  
[INFO] starting video stream...
```

Fig.13. Loading of the Face Mask Detector



Fig.14. The live camera setup

6. EXPERIMENTAL RESULTS AND ANALYSIS

- The model works as we expected and provides accurate prediction of whether or not a person is wearing a face mask.
- These are some of the results that the model gives while wearing a mask.

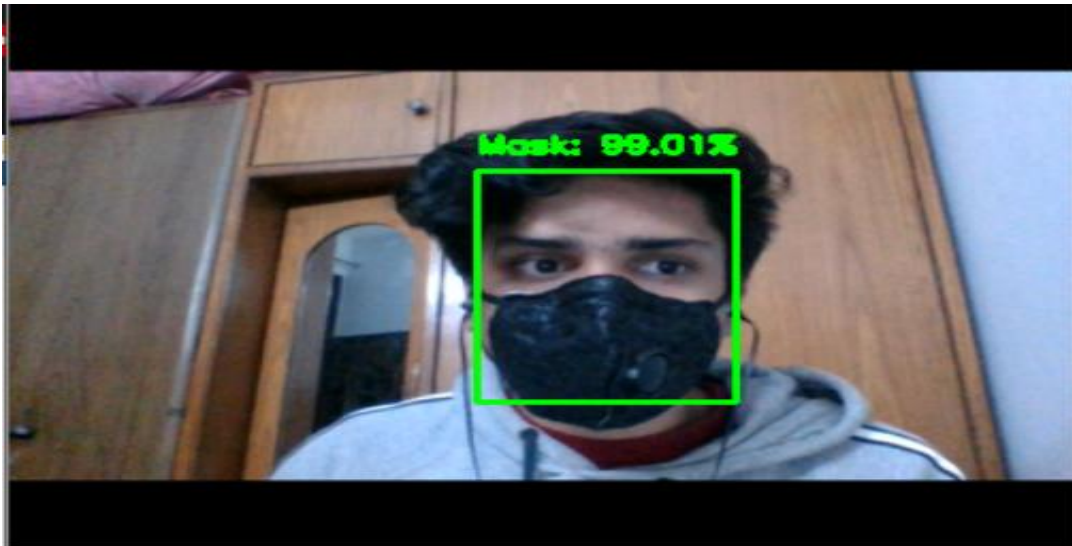


Fig.15. Test while wearing Face Masks



Fig.16. Test while wearing Face Mask

- These are some of the results that model gives while not wearing a mask.

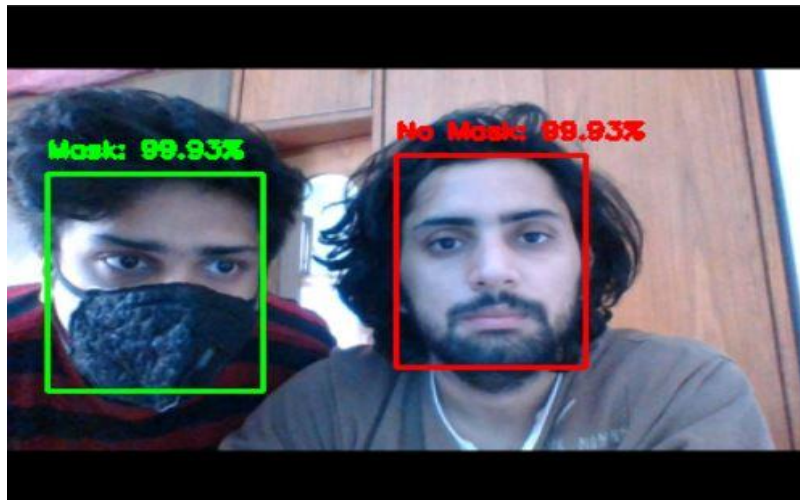


Fig.17: Test without wearing Face Masks



Fig.18: Test without wearing Face Masks

- The model works as expected, and gives correct predictions of whether or not people are wearing a mask. The accuracy and the loss of the model was shown in Fig.6. The value at the 20th epoch was:

```
loss: 0.0866 - accuracy: 0.9667 - val_loss: 0.2567 - val_accuracy: 0.9117
```

Fig.19: Values at the Last Epoch

7. CONCLUSION AND FURTHER SCOPE

- We decided the workflow of our project by the Gantt Chart shown below:

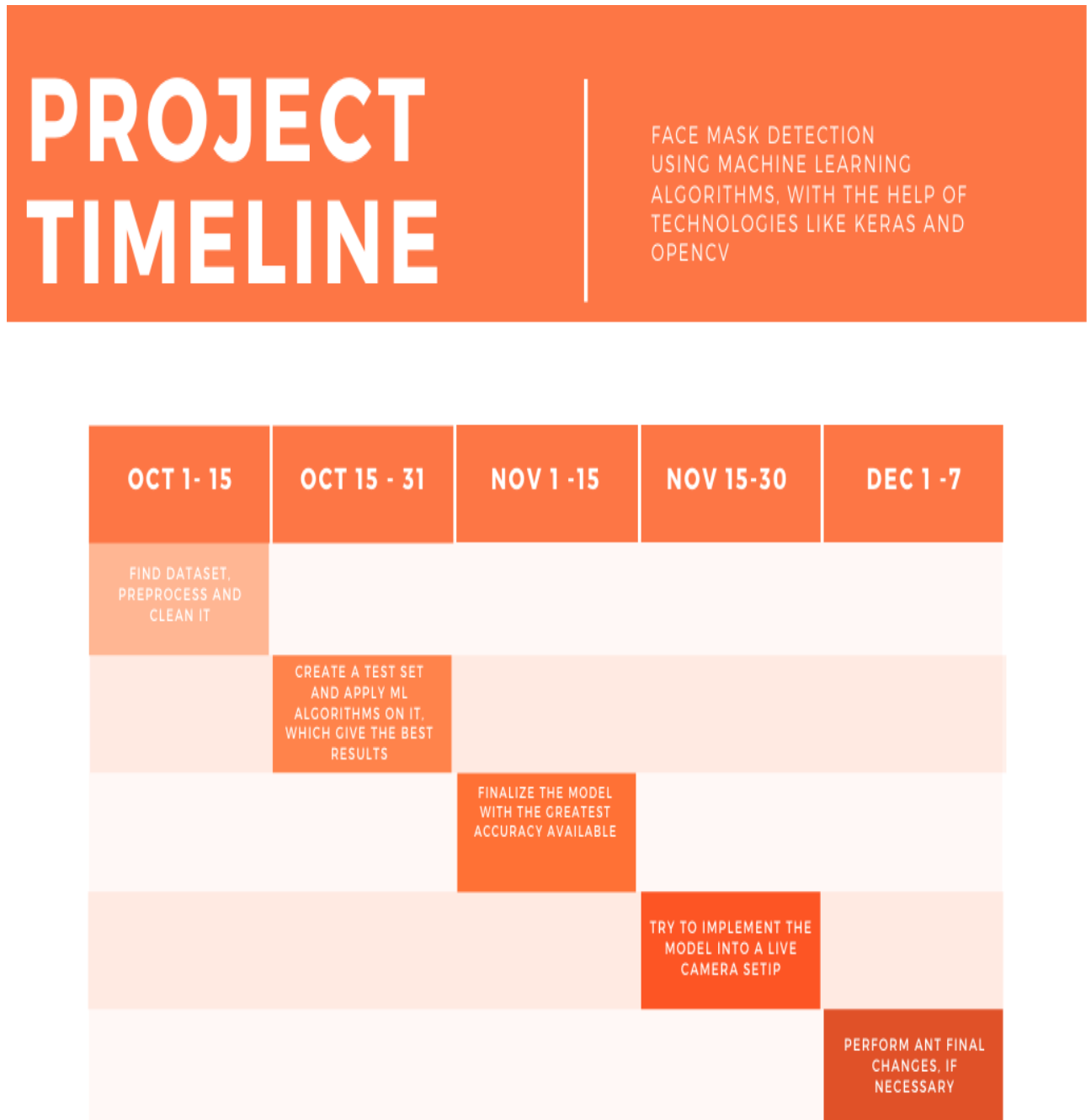


Fig.20. Gantt Chart for Work Done

- The model which we created were suitable and gave us the following results as shown in Figure 6:



Fig.6.

- The live camera gives accurate predictions of the subjects as shown below in Fig.11 and 12:

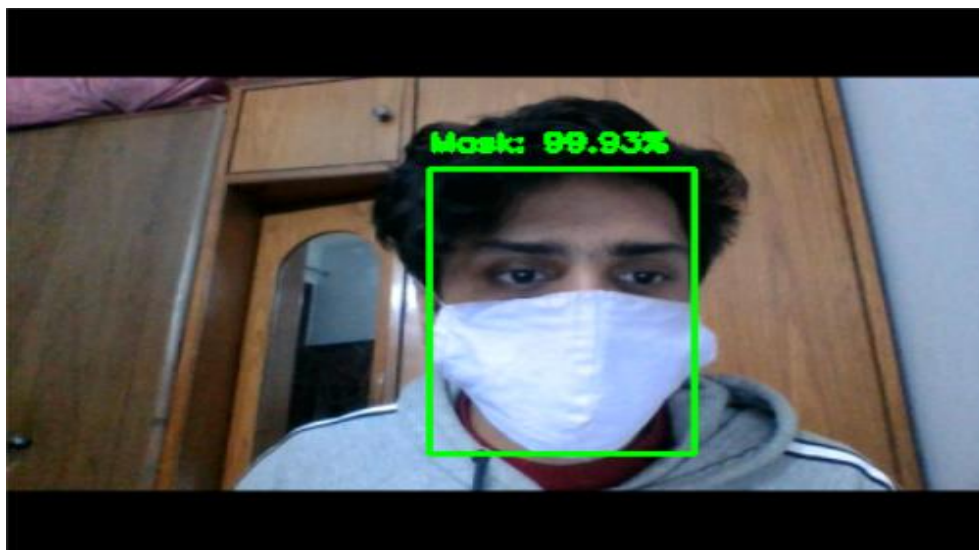


Fig.11

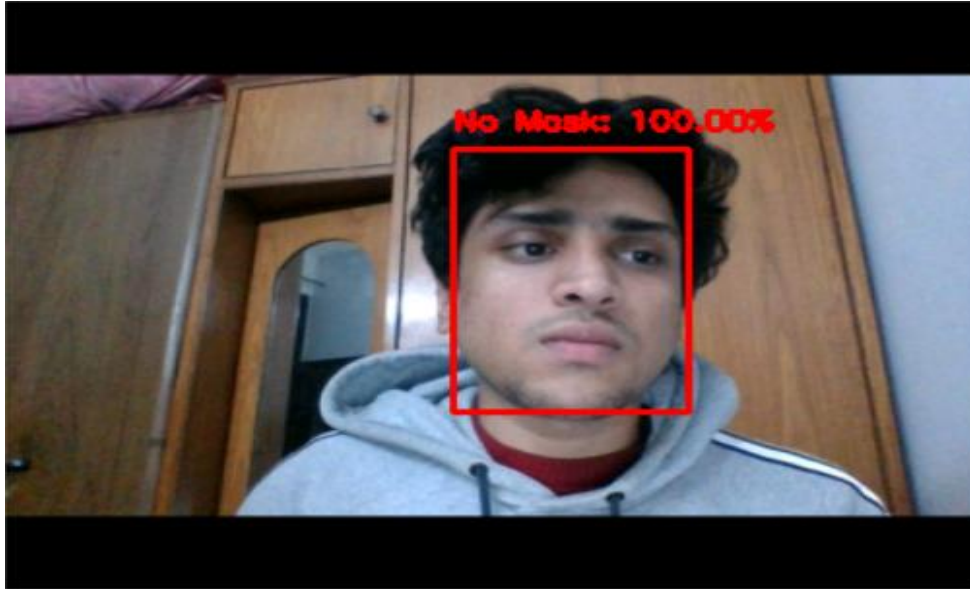


Fig.12.

7.4. FUTURE SCOPE

- We will try to improve on these results in the following ways:
 1. Increasing the Training Data, for better accuracy of the model.
 2. Our Dataset **with_mask** was artificially generated, hence if we obtain a dataset containing pictures of real people wearing real masks our results can be significantly improved.
 3. The model doesn't predict that well at greater distances, since MobileNet is a lightweight network. So, it could be further optimized for better results at a greater distance between the subject and the camera.
- Since we have used MobileNet, which is a very lightweight model, and hence this model can be embedded in an Android or iOS app, or a simple CCTV camera which can be used for face mask detection.
- These CCTV cameras can be used for detection in this pandemic, for catching people who are breaking the law by not wearing face masks.

8.REFERENCES

[1] QIN, B., & LI, D. (2020). Identifying Facemask-wearing Condition Using Image Super-Resolution with Classification Network to Prevent COVID-19.

<https://www.researchsquare.com/article/rs-28668/latest.pdf>

[2] Ejaz, M. S., Islam, M. R., Sifatullah, M., & Sarker, A. (2019, May). Implementation of Principal Component Analysis on Masked and Non-masked Face Recognition. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)* (pp. 1-5). IEEE.

<https://ieeexplore.ieee.org/abstract/document/8934543/>

[3] Park, J. S., Oh, Y. H., Ahn, S. C., & Lee, S. W. (2005). Glasses removal from facial image using recursive error compensation. *IEEE transactions on pattern analysis and machine intelligence*, 27(5), 805-811.

<https://ieeexplore.ieee.org/abstract/document/1407883/>

[4] Li, C., Wang, R., Li, J., & Fei, L. (2020). Face Detection Based on YOLOv3. In *Recent Trends in Intelligent Computing, Communication and Devices* (pp. 277-284). Springer, Singapore.

https://link.springer.com/chapter/10.1007/978-981-13-9406-5_34

[5] Din, N. U., Javed, K., Bae, S., & Yi, J. (2020). A novel GAN-based network for unmasking of masked face. *IEEE Access*, 8, 44276-44287.

<https://ieeexplore.ieee.org/abstract/document/9019697/>

[6] Nieto-Rodríguez, A., Mucientes, M., & Brea, V. M. (2015, June). System for medical mask detection in the operating room through facial attributes. In *Iberian Conference on Pattern Recognition and Image Analysis* (pp. 138-145). Springer, Cham.

https://link.springer.com/chapter/10.1007/978-3-319-19390-8_16

[7] Khan, M. K. J., Ud Din, N., Bae, S., & Yi, J. (2019). Interactive removal of microphone object in facial images. *Electronics*, 8(10), 1115.

<https://www.mdpi.com/2079-9292/8/10/1115>

[8] Wang, Z., Wang, G., Huang, B., Xiong, Z., Hong, Q., Wu, H., ... & Chen, H. (2020). Masked face recognition dataset and application. *arXiv preprint arXiv:2003.09093*.

<https://arxiv.org/abs/2003.09093>