IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 5, May 2022
Impact Factor: 6.252

# Real Time Approach for Face Mask Detection using CNN

**Mehul Shigvan, Mrunal Sakunde, Yash Jadhav**
Student, Department of Computer Science
Sinhgad College of Engineering, Vadgaon, Pune, Maharashtra, India

**Abstract:** *With the horizon still brim-full, effective solutions to contain the COVID-19 pandemic require close attention to reduce severely impacted communal health and worldwide economy. Many approaches are advised by WHO to manage the infection rate and avoid depleting the limited medical resources in the absence of effective antivirals and inadequate medical resources. Wearing a mask is one of the non-pharmaceutical interventions that can be utilised to reduce the principal source of SARS-CoV2 droplets ejected by an infected person. Regardless of debates about medical resources and mask varieties, all countries require public use of nose and mouth coverings. This strategy intends to develop a highly precise and real-time technique that can efficiently detect non- mask faces in public, hence mandating mask usage. For mask detection, the proposed strategy uses a convolutional neural network-based approach.*

**Keywords:** CNN

## I. INTRODUCTION

Neural networks are currently being used to solve difficulties of control. Testing the neuro-control technique on a number of practical problems is one way to assess its reliability. Another option is to directly compare it to existing traditional control approaches to evaluate if it performs effectively and where it needs to be improved. The current study created a neural network-based predictive control strategy for controlling the temperature of the water bath. Internal model control with neural networks is the control approach used.

The COVID-19 corona virus epidemic is causing an increase in the use of face masks in public places around the world [1]. People used to wear masks to protect their health from air pollution before Covid-19. Other people hide their feelings in public to hide their faces, while others are self- conscious about their appearance. COVID19 infected over five million people in 188 countries in less than six months. The virus spreads through close contact and in overcrowded environments. We can use emerging technologies like artificial intelligence, IoT, Big data, and Machine learning to fight and anticipate new diseases. To have a better understanding of how infection rates could be reduced using our approach. Many countries have regulations requiring people to wear face masks in public. These guidelines and laws were created in response to the rapid increase in cases and deaths in several places. In public spaces, however, monitoring big gatherings of individuals is getting more challenging. As a result, we'll automate the face detection procedure. A facemask detection algorithm based on computer vision and deep learning is shown here [2]. The proposed model can be used in conjunction with surveillance cameras to prevent COVID-19 transmission by detecting people who aren't wearing face masks. With OpenCV, TensorFlow, and Keras, the model combines deep learning and traditional machine learning approaches.

## II. LITERATURE SURVEY

### Face Mask Detector

Object detection is accomplished using the Single Shot Detector architecture. Face mask detectors can be used in various places, including shopping malls, airports, and other high-traffic areas, to monitor the public and prevent disease spread by determining who is following basic standards and who is not. The data loading time in Google Colab Notebook is high. It didn't allow access to the webcam, which made evaluating photos and video streams difficult. Deep learning was used to model a facemask detector. We used MobileNetV2 to process a computationally efficient system, making it easier to extract data sets. For greater speed, we adopt CNN architecture.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 5, May 2022
Impact Factor: 6.252

**Face Detection Techniques: A Review**

Since artificial humans do not have the same ability as machines to recognise different faces, automatic face detection systems play a crucial part in face recognition, headpose estimation, and other tasks. Face occlusion and non- uniform illumination are some of the issues. To recognise faces in live video streams, we employ a Neural Network. This system employs tensor flow as well. They employ the Adaboost method in the present system, however in our suggested system, we use the mob net CNN Architecture model. This paper will address each of these issues.

**Multi-Stage CNN Architecture for Face Mask Detection**

This system uses a dual-stage (CNN) architecture that can recognise both masked and unmasked faces and can be used in conjunction with pre-installed CCTV cameras. This will aid in the tracking of safety violations, the promotion of face mask use, and the creation of a safe working environment. Datasets from the public domain were combined with data obtained from the internet. For detection, they only employ pre-trained datasets. Face detection may be done with any camera. Preventing viral transmission will be extremely beneficial to society and humans. Here, we utilise OpenCV to detect live video (python library)

**Real time face mask recognition with alarm system using deep learning**

This method produces accurate and quick results for facemask detection. Real-time face mask recognition on the Raspberry Pi that records the facial image. The architectural elements of VGG-16 are used as the foundation network for face recognition in this system. Deep learning techniques are used to build a classifier that will collect images of people wearing masks and those without masks. For face detection, we propose leveraging CNN's architectural properties as the basis network. It is capable of identifying whether or not a person is wearing a face mask. This research could be effective in combating the spread of the Covid 19 virus..

## III. PROPOSED SYSTEM

The block diagram of the proposed face mask detection system is as shown in Fig.2.1

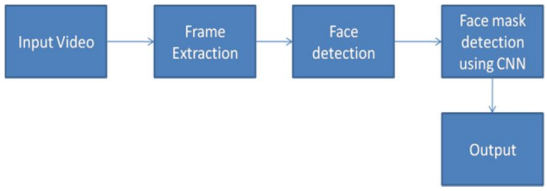



**Figure 1:** Block diagram of face mask detection system

### 3.1 Face Mask Dataset

A mask and no-mask dataset must be created for the proposed system. The system's database is self-created. The photos of the faces are saved. The dataset is divided into two sections: training and testing. 70% of the data is used for training purposes, whereas 25% is used for testing purposes.

### 3.2 Image Pre-processing

To avoid affecting the classification, the image is first resized to the same size. The image is downsized to 256 X 256 in this system.

### 3.3 Face Detection

It's the first stage in detecting human faces by determining the ROI (Region of Interest). Tells whether or not there is a human face in a given image. If there is, locate and measure each human face in the photograph. Face Recognition with OpenCV: It comes with built-in face recognizers and pre- trained Haar classifiers for detecting faces, eyes, and other objects. Haar Classifier is used in this project. Haar Cascade is a machine learning technique in which the cascade function is learned on a large number of positive (faced) and negative (non-faced) images (images without faces). After loading the appropriate XML classifier (haar cascade frontal face default.xml), load the greyscale stream of input photos (or video). It returns the

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 5, May 2022
Impact Factor: 6.252

positions of detected faces with coordianates as Rect if faces are discovered (x,y,w,h). The ROI for the detected face is constructed after these locations are obtained.

### 3.4 Classification

CNNs are a type of Neural Network that has shown to be particularly effective in image recognition and classification. CNNs are a sort of feed-forward neural network with a large number of layers. CNNs are made up of filters, kernels, or neurons with programmable weights, parameters, and biases. Each filter takes a set of inputs, performs convolution, and optionally adds non-linearity to the mix. Convolutional, pooling, Rectified Linear Unit (ReLU), and Fully Connected layers make up CNN's structure. .

Convolutional Layer (CL): The main building element is the CL. It conducts a Convolutional Network'score building block, doing the most intense computation. Convolution's primary aim is to extract the features from the input image. It produces a function map or activation map in the output images and is then put into the next CL. It is mathematically represented as

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

ReLU Layer: This function increases the data's non-linearity. It's an elementwise non-linear operation in which the negative feature map value is replaced with zero. We'll suppose the neuron input is x and the rectifier is defined as to show how the ReLU works.

$$f(x) = max(0, x) \qquad (3.3)$$

Pooling Layer: The PL reduces the complexity of each activation map, but the most important information remains. The data is divided into a set of non-overlapping rectangles. Each region is down- sampled as maximum using a non-linear technique. MPL (Max Pooling Layers) is a relatively easy procedure that requires no prior knowledge.

Flatten Layer (FL): After completing the previous two operations, we should now have a pooled feature map. As the name implies, we'll flatten our pooled feature map into a column in this phase.

Fully Connected Layer (FCL): The FCL separates the input image into groups based on the training dataset using these attributes. The final sheet in the pooling process is FCL, and it feeds components to a classifier using softmax activation. For the Completely Connected Layer, there is just one performance probability. This is accomplished via Softmax's activation capability.

Vgg16: The VGG-16 model is a CNN that uses human input to recognise images. VGG-16 is a CNN that can recognise large volumes of data. The model obtains high accuracy when categorising large datasets into distinct classifications.

The VGG -16 model architecture consists of a stack of convolution layers that transmit the input image and 11 convolution filters that linearly modify the input channels. The model's final layer is the SoftMax layer. Hidden layers are fitted using ReLu non-linearity. The vgg16 model's architecture is depicted in Figure 2.

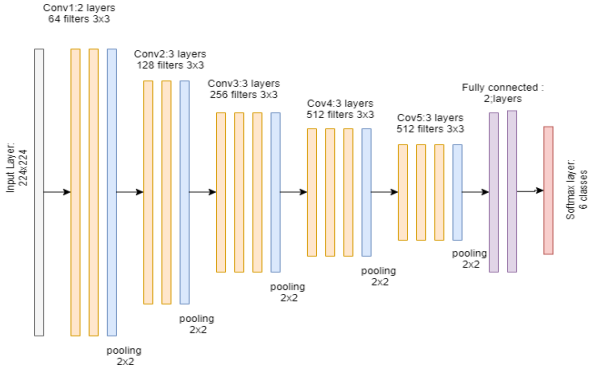



**Figure 2:** VGG-16 model architecture

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 5, May 2022
Impact Factor: 6.252

The precise structure of the VGG-16 network shown in Fig. 7. is as follows:

- The 64 feature kernel filters in the first and second CL each have a 3x3 pixel dimension. The input image's dimensions change to 224x224x64 when it passes through the first and second CL.
- The output is then transmitted to the MPL with a stride of 2. The third and fourth convolutional layers' 124 feature kernel filters have a filter size of 33%. After these two layers, the output is downsized to 56x56x128 using an MPL with stride two.
- In the fifth, sixth, and seventh levels, CLs with kernel sizes of 3x3 are employed. 256 feature maps are used in all three. A stride of 2 MPL follows these layers.
- From the ninth through the thirteenth convolution layers, there are two sets of 3x3 kernel sizes. There are 512 kernel filters in each of these CL sets. These layers are followed by an MPL with a stride of one.
- The fourteenth and fifteenth levels, which follow a 1000-unit SoftMax output layer, are 4096-unit completely connected hidden layers (sixteenth layer).

## IV. RESULTS

In this approach, CNN and Vgg16 algorithms are use to train the system for face mask and no mask recognition. The results of the proposed system are shown below.

### CNN

The training progress of the CNN algorithm is as shown below,

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 223, 223, 256)     3328

 activation (Activation)     (None, 223, 223, 256)     0

 max_pooling2d (MaxPooling2D (None, 111, 111, 256)     0
 )

 conv2d_1 (Conv2D)           (None, 110, 110, 128)     131200

 activation_1 (Activation)   (None, 110, 110, 128)     0

 max_pooling2d_1 (MaxPooling (None, 55, 55, 128)       0
 2D)

 flatten (Flatten)           (None, 387200)            0

 dense (Dense)               (None, 64)                24780864

 activation_2 (Activation)   (None, 64)                0

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 2)                 130

=================================================================
Total params: 24,915,522
Trainable params: 24,915,522
Non-trainable params: 0
_________________________________________________________________
```

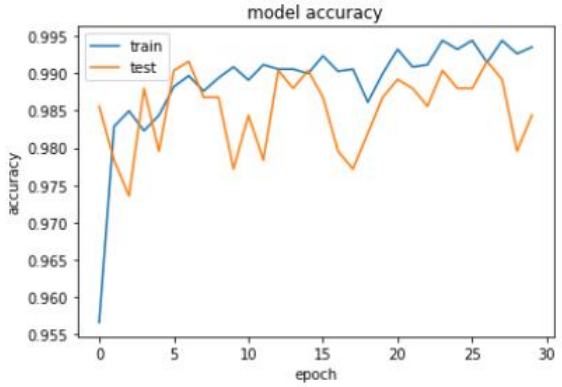
Found 3390 images belonging to 2 classes. Found 841 images belonging to 2 classes. /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:67: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators. Epoch 1/30

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 5, May 2022
Impact Factor: 6.252

```
212/212 [==============================]
    - 671s 3s/step - loss: 0.2184 -
 accuracy: 0.9566 - val_loss: 0.0573 -
         val_accuracy: 0.9856
              Epoch 2/30
212/212 [==============================]
    - 65s 305ms/step - loss: 0.0798 -
 accuracy: 0.9829 - val_loss: 0.1742 -
         val_accuracy: 0.9784
              Epoch 3/30
212/212 [==============================]
    - 64s 301ms/step - loss: 0.0832 -
 accuracy: 0.9850 - val_loss: 0.1129 -
         val_accuracy: 0.9736
              Epoch 4/30
212/212 [==============================]
    - 64s 301ms/step - loss: 0.0783 -
 accuracy: 0.9823 - val_loss: 0.0636 -
         val_accuracy: 0.9880
              Epoch 5/30
212/212 [==============================]
    - 64s 300ms/step - loss: 0.0602 -
 accuracy: 0.9844 - val_loss: 0.1009 -
         val_accuracy: 0.9796
              Epoch 6/30
212/212 [==============================]
    - 64s 300ms/step - loss: 0.0519 -
 accuracy: 0.9882 - val_loss: 0.0723 -
         val_accuracy: 0.9904
              Epoch 7/30
212/212 [==============================]
    - 64s 301ms/step - loss: 0.0539 -
 accuracy: 0.9897 - val_loss: 0.0566 -
         val_accuracy: 0.9916
              Epoch 8/30
212/212 [==============================]
    - 63s 299ms/step - loss: 0.0662 -
 accuracy: 0.9876 - val_loss: 0.0679 -
         val_accuracy: 0.9868
              Epoch 9/30
212/212 [==============================]
    - 63s 298ms/step - loss: 0.0443 -
 accuracy: 0.9894 - val_loss: 0.0643 -
         val_accuracy: 0.9868
              Epoch 10/30
212/212 [==============================]
    - 63s 298ms/step - loss: 0.0451 -
 accuracy: 0.9909 - val_loss: 0.0962 -
         val_accuracy: 0.9772
              Epoch 11/30
212/212 [==============================]
    - 63s 299ms/step - loss: 0.0465 -
 accuracy: 0.9891 - val_loss: 0.0736 -
         val_accuracy: 0.9844
              Epoch 12/30
212/212 [==============================]
    - 64s 300ms/step - loss: 0.0448 -
 accuracy: 0.9912 - val_loss: 0.0959 -
         val_accuracy: 0.9784
              Epoch 13/30
212/212 [==============================]
    - 63s 297ms/step - loss: 0.0486 -
 accuracy: 0.9906 - val_loss: 0.0441 -
         val_accuracy: 0.9904
              Epoch 14/30
212/212 [==============================]
    - 63s 298ms/step - loss: 0.0345 -
 accuracy: 0.9906 - val_loss: 0.0577 -
         val_accuracy: 0.9880
              Epoch 15/30
212/212 [==============================]
    - 64s 303ms/step - loss: 0.0434 -
 accuracy: 0.9900 - val_loss: 0.0400 -
         val_accuracy: 0.9904
              Epoch 16/30
212/212 [==============================]
    - 65s 305ms/step - loss: 0.0329 -
 accuracy: 0.9923 - val_loss: 0.0672 -
         val_accuracy: 0.9868
              Epoch 17/30
212/212 [==============================]
    - 64s 302ms/step - loss: 0.0441 -
 accuracy: 0.9903 - val_loss: 0.1570 -
         val_accuracy: 0.9796
              Epoch 18/30
212/212 [==============================]
    - 63s 299ms/step - loss: 0.0363 -
 accuracy: 0.9906 - val_loss: 0.1090 -
         val_accuracy: 0.9772
              Epoch 19/30
212/212 [==============================]
    - 64s 300ms/step - loss: 0.0600 -
 accuracy: 0.9861 - val_loss: 0.0534 -
         val_accuracy: 0.9820
              Epoch 20/30
212/212 [==============================]
    - 64s 299ms/step - loss: 0.0488 -
 accuracy: 0.9900 - val_loss: 0.0592 -
         val_accuracy: 0.9868
              Epoch 21/30
212/212 [==============================]
    - 64s 303ms/step - loss: 0.0310 -
 accuracy: 0.9932 - val_loss: 0.0486 -
         val_accuracy: 0.9892
              Epoch 22/30
212/212 [==============================]
    - 64s 301ms/step - loss: 0.0490 -
 accuracy: 0.9909 - val_loss: 0.0461 -
         val_accuracy: 0.9880
              Epoch 23/30
212/212 [==============================]
    - 64s 301ms/step - loss: 0.0322 -
 accuracy: 0.9912 - val_loss: 0.1050 -
         val_accuracy: 0.9856
              Epoch 24/30
212/212 [==============================]
    - 64s 301ms/step - loss: 0.0222 -
 accuracy: 0.9944 - val_loss: 0.0566 -
         val_accuracy: 0.9904
              Epoch 25/30
212/212 [==============================]
    - 65s 304ms/step - loss: 0.0279 -
 accuracy: 0.9932 - val_loss: 0.0523 -
         val_accuracy: 0.9880
              Epoch 26/30
```

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 5, May 2022
Impact Factor: 6.252

```
212/212 [==============================]
   - 65s 308ms/step - loss: 0.0196 -
 accuracy: 0.9944 - val_loss: 0.0485 -
         val_accuracy: 0.9880
              Epoch 27/30
212/212 [==============================]
   - 64s 301ms/step - loss: 0.0291 -
 accuracy: 0.9914 - val_loss: 0.0583 -
         val_accuracy: 0.9916
              Epoch 28/30
212/212 [==============================]
   - 64s 302ms/step - loss: 0.0205 -
 accuracy: 0.9944 - val_loss: 0.0505 -
         val_accuracy: 0.9892
              Epoch 29/30
212/212 [==============================]
   - 64s 301ms/step - loss: 0.0260 -
 accuracy: 0.9926 - val_loss: 0.1347 -
         val_accuracy: 0.9796
              Epoch 30/30
212/212 [==============================]
   - 65s 304ms/step - loss: 0.0301 -
 accuracy: 0.9935 - val_loss: 0.0794 -
         val_accuracy: 0.9844
```




```
2555.243246793747
```

**Fig.5.1.** Training progress graph of CNN algorithm (a) Accuracy (b) Loss

- VGG16

The progress of the Vgg16 algorithm for face mask detection is given below

```
                starting
    Found 3390 images belonging to 2
                classes.
Found 841 images belonging to 2 classes.
          Downloading data from
https://storage.googleapis.com/tensorflo
          w/keras-
applications/vgg16/vgg16_weights_tf_dim_
      ordering_tf_kernels_notop.h5
          58892288/58889256
  [==============================] - 0s
                0us/step
          58900480/58889256
  [==============================] - 0s
                0us/step
           Model: "model"
_________________________________________
 _________________________
  Layer (type)                Output
    Shape              Param #
=========================================
       =========================
  input_1 (InputLayer)        [(None,
        200, 200, 3)]      0

 block1_conv1 (Conv2D)        (None, 200,
        200, 64)       1792

 block1_conv2 (Conv2D)        (None, 200,
        200, 64)       36928

 block1_pool (MaxPooling2D)   (None, 100,
        100, 64)       0

 block2_conv1 (Conv2D)        (None, 100,
        100, 128)      73856

 block2_conv2 (Conv2D)        (None, 100,
        100, 128)      147584

 block2_pool (MaxPooling2D)   (None, 50,
        50, 128)       0

 conv2d_2 (Conv2D)            (None, 49,
        49, 256)       131328

 max_pooling2d_2 (MaxPooling   (None, 24,
        24, 256)       0
              2D)

 conv2d_3 (Conv2D)            (None, 23,
        23, 128)       131200

 max_pooling2d_3 (MaxPooling   (None, 11,
        11, 128)       0
              2D)
```

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 5, May 2022
Impact Factor: 6.252

```
flatten_1 (Flatten)          (None,
15488)                0

dense_2 (Dense)              (None, 64)
                  991296

dropout_1 (Dropout)          (None, 64)
                  0

dense_3 (Dense)              (None, 2)
                  130

=================================================
==========================
Total params: 1,514,114
Trainable params: 1,401,538
Non-trainable params: 112,576

_________________________________________________

/usr/local/lib/python3.7/dist-
packages/keras/optimizer_v2/rmsprop.py:1
30: UserWarning: The `lr` argument is
deprecated, use `learning_rate` instead.
  super(RMSprop, self).__init__(name,
**kwargs)
/usr/local/lib/python3.7/dist-
packages/ipykernel_launcher.py:82:
UserWarning: `Model.fit_generator` is
deprecated and will be removed in a
future version. Please use `Model.fit`,
which supports generators.
Epoch 1/10
106/106 [==============================]
 - ETA: 0s - loss: 5.0909 - acc: 0.8971
Epoch 1: saving model to model_vgg.hdf5
106/106 [==============================]
 - 22s 173ms/step - loss: 5.0909 - acc:
0.8971 - val_loss: 0.1056 - val_acc:
0.9880
Epoch 2/10
106/106 [==============================]
 - ETA: 0s - loss: 0.2774 - acc: 0.9752
Epoch 2: saving model to model_vgg.hdf5
106/106 [==============================]
 - 16s 146ms/step - loss: 0.2774 - acc:
0.9752 - val_loss: 0.0891 - val_acc:
0.9880
Epoch 3/10
106/106 [==============================]
 - ETA: 0s - loss: 0.1358 - acc: 0.9791
Epoch 3: saving model to model_vgg.hdf5
106/106 [==============================]
 - 16s 148ms/step - loss: 0.1358 - acc:
0.9791 - val_loss: 0.1702 - val_acc:
0.9892
Epoch 4/10
106/106 [==============================]
 - ETA: 0s - loss: 0.1427 - acc: 0.9782
Epoch 4: saving model to model_vgg.hdf5
106/106 [==============================]
 - 15s 145ms/step - loss: 0.1427 - acc:
0.9782 - val_loss: 0.1993 - val_acc:
0.9880
Epoch 5/10
106/106 [==============================]
 - ETA: 0s - loss: 0.2125 - acc: 0.9858
Epoch 5: saving model to model_vgg.hdf5
106/106 [==============================]
 - 15s 144ms/step - loss: 0.2125 - acc:
0.9858 - val_loss: 0.1713 - val_acc:
0.9916
Epoch 6/10
106/106 [==============================]
 - ETA: 0s - loss: 0.1043 - acc: 0.9914
Epoch 6: saving model to model_vgg.hdf5
106/106 [==============================]
 - 15s 143ms/step - loss: 0.1043 - acc:
0.9914 - val_loss: 0.1250 - val_acc:
0.9880
Epoch 7/10
106/106 [==============================]
 - ETA: 0s - loss: 0.1081 - acc: 0.9853
Epoch 7: saving model to model_vgg.hdf5
106/106 [==============================]
 - 15s 142ms/step - loss: 0.1081 - acc:
0.9853 - val_loss: 0.1893 - val_acc:
0.9868
Epoch 8/10
106/106 [==============================]
 - ETA: 0s - loss: 0.0971 - acc: 0.9870
Epoch 8: saving model to model_vgg.hdf5
106/106 [==============================]
 - 15s 144ms/step - loss: 0.0971 - acc:
0.9870 - val_loss: 0.5242 - val_acc:
0.9760
Epoch 9/10
106/106 [==============================]
 - ETA: 0s - loss: 0.1039 - acc: 0.9894
Epoch 9: saving model to model_vgg.hdf5
106/106 [==============================]
 - 15s 143ms/step - loss: 0.1039 - acc:
0.9894 - val_loss: 0.1967 - val_acc:
0.9904
Epoch 10/10
106/106 [==============================]
 - ETA: 0s - loss: 0.0712 - acc: 0.9912
Epoch 10: saving model to model_vgg.hdf5
106/106 [==============================]
 - 15s 145ms/step - loss: 0.0712 - acc:
0.9912 - val_loss: 0.2789 - val_acc:
0.9904
dict_keys(['loss', 'acc', 'val_loss',
'val_acc'])
```

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 5, May 2022
Impact Factor: 6.252

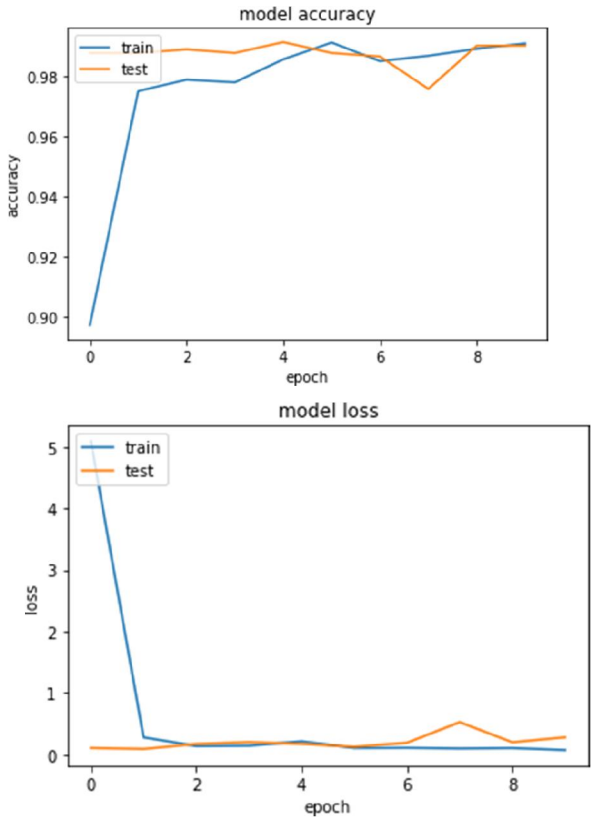



**Figure 5.2:** Training progress graph of Vgg16 algorithm (a) Accuracy (b) Loss

Table 5.1. depicts the comparative analysis of CNN and Vgg16 algorithm.

**Table 5.1. Accuracy of the proposed system**

| Algorithm | Training Accuracy | Validation Accuracy | Training Loss | Validation Loss |
|---|---|---|---|---|
| **CNN** | 0.9912 | 0.0712 | 0.9904 | 0.2789 |
| **Vgg16** | 0.9912 | 0.9904 | 0.0712 | 0.2789 |

From Table 5 it is observed that the results of the Vgg16 algorithm shows the promising results than the CNN algorithm.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 5, May 2022
Impact Factor: 6.252

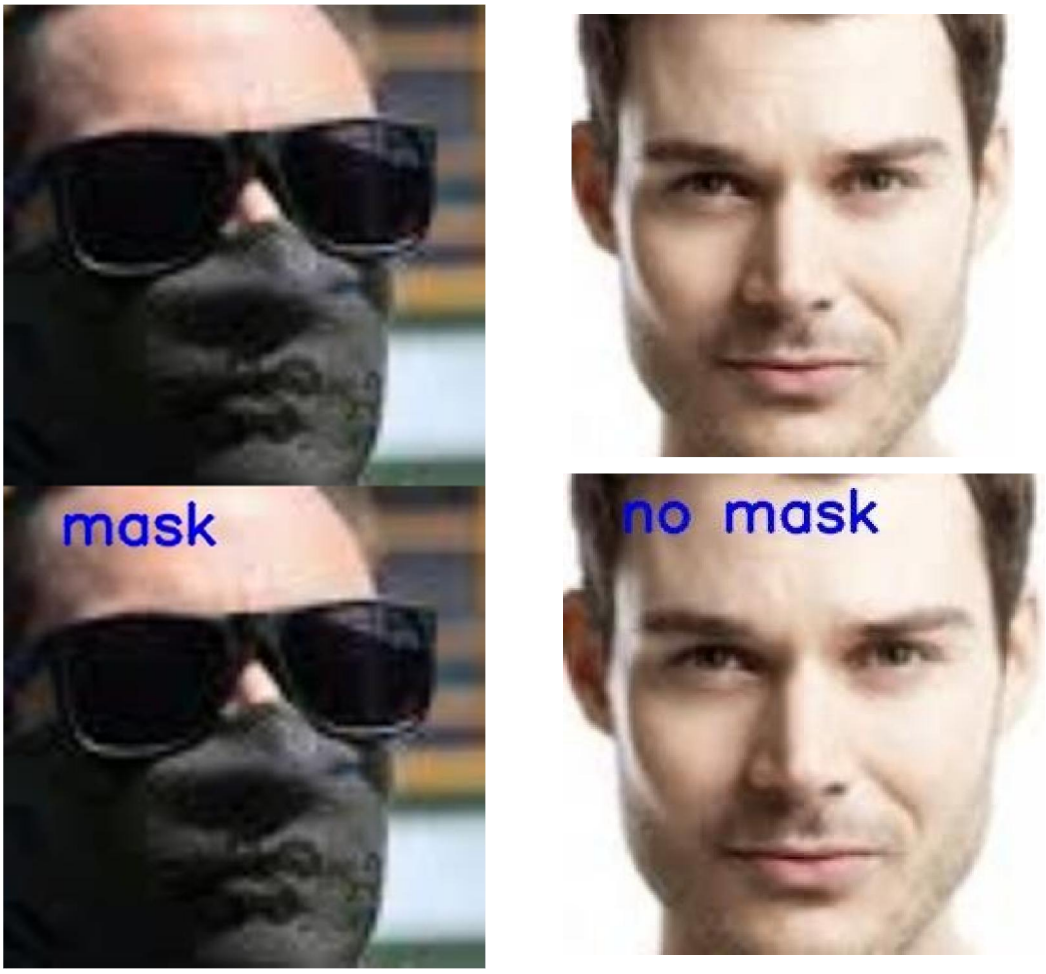



**Fig.5.3**. Qualitative analysis of the system (a) With mask (b) Without mask

From the qualitative analysis of the proposed system, it is observed that the vgg16 algorithm is more accurately classify the mask and no mask images or stored images as well as in real time.

## V. CONCLUSION

COVID-19 is currently a global epidemic, and several governments are fighting it with a variety of processes and technology. A face mask is one of the protection methods among these. The fundamental idea behind our research was to apply the CNN and Vgg16 deep learning classification algorithms to detect (classify) whether a person is wearing a mask or not. The results show that the Vgg16 method, which is based on transfer learning, performs better than the CNN algorithm in reliably classifying face masks and no masks. In addition, the Vgg16 algorithm takes less time to train. It also works best in a real-time setting.

## REFERENCES

**[1].** Why we should all be wearing face masks, https://www.bbc.com/future/article/20200504- coronavirus-what-is-the-best-kind-of-face-mask.

**[2].** Jeremy Howard, Austin Huang, Zhiyuan Li, Zeynep Tufekci, Vladimir Zdimal, Helene-Mari van der Westhuizen ORCID logo, Arne von Delft, Amy Price, Lex Fridman, Lei-Han Tang, Viola Tang, Gregory L. Watson, Christina E. Bax, Reshama Shaikh, Frederik Questier ORCID logo, Danny Hernandez, Larry F. Chu, Christina M. Ramirez, Anne W. Rimoin, " Face Masks Against COVID-19: An Evidence Review"

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 5, May 2022
Impact Factor: 6.252

**[3].** Raza Ali, Saniya Adeel, Akhyar Ahmed Face Mask Detector July 2020

**[4].** A. Kumar, A. Kaur, M. Kumar, "Face detection techniques: A review, "Artificial intelligence review, volume.52, no.2, pp.927- 928, 2019.

**[5].** Amit Chavda, jason Dsouza, Sumeet Badgujar, "Multi- Stage CNN Architecture for Face Mask Detection", September 2020.

**[6].** Sammy v. militante, Nanettev. dionisio "Real time face mask recognition with alarm system using deep learning", 2020 11th IEEE control and system graduate research colloquium.