FACE MASK DETECTION AND ALERT SYSTEM

CREATIVE AND INNOVATIVE PROJECT REPORT

Submitted by

ARUNRAJ M (2018103011)

MUNI RAJ KALYAN CHEMBETI (2018103044)

Abstract

The whole world is suffering with a pandemic called CoronaVirus. The Coronavirus disease 2019 has affected the world seriously. One major protection method for people is to wear masks in public areas. Furthermore, many public service providers require customers to use the service only if they wear masks correctly. Correct facemask wearing is valuable for infectious disease control, but the effectiveness of facemasks has been diminished, mostly due to improper wearingHowever, there are only a few research studies about face mask detection based on image analysis.A face mask detection dataset consists of with mask and without mask images , we are going to use OpenCV to do real-time face detection from a live stream via our webcam. We will use the dataset to build a COVID-19 face mask detector with computer vision using Python, OpenCV, and TensorFlow and Keras. Our goal is to identify whether the person on the image/video stream is wearing a face mask or not with the help of computer vision and deep learning.

1. INTRODUCTION

"Artificial Intelligence (AI) based on Machine learning and Deep Learning can help to fight Covid-19 in many ways." ***International Research Journal of Engineering and Technology (Vinitha.V, Velantina.V August 2020)***

The rapid worldwide spread of Coronavirus Disease 2019 (COVID-19) has resulted in a global pandemic. Correct facemask wearing is valuable for infectious disease control, but the effectiveness of facemasks has been diminished, mostly due to improper wearing. To avoid global tragedy, a practical and straightforward approach to preventing the spread of the virus is urgently desired worldwide. Studies have found that facemask-wearing is valuable in preventing the spread of respiratory viruses. Facemask-wearing is a non-invasive and cheap method to reduce mortality and morbidity from respiratory infections. Since the outbreak of COVID-19, face masks have been routinely used by the general public to reduce exposure to airborne pathogens in many countries.However, the effectiveness of facemasks in containing the spread of airborne diseases in the general public has been diminished, mostly due to improper wearing. Therefore, it is necessary to develop an automatic detection approach for facemask-wearing conditions, which can contribute to personal protection and public epidemic prevention.The proposed model can be integrated with surveillance cameras to impede the COVID-19 transmission by allowing the detection of people who are wearing masks not wearing face masks. The model is integration between deep learning and classical machine learning techniques with opencv and keras.The problem is closely related to general object detection to detect the classes of objects and face detection is to detect a particular class of objects, i.e. face[2][6].

2. RELATED WORK

2.1 Object Detection

Traditional object detection uses a multi-step process. A well-known detector is the Viola-Joins detector, which is able to achieve real-time detection. The algorithm extracts features by Haar feature descriptor with an integral image method, selects useful features, and detects objects through a cascaded detector. Although it utilizes integral image to facilitate the algorithm, it is still very computationally expensive. In for human detection, an effective feature extractor called HOG is proposed, which computes the directions and magnitudes of oriented gradients over image cells.[11]

Rather than using handcrafted features, deep learning based detectors demonstrated excellent performance recently,due to its robustness and high feature extraction capability.

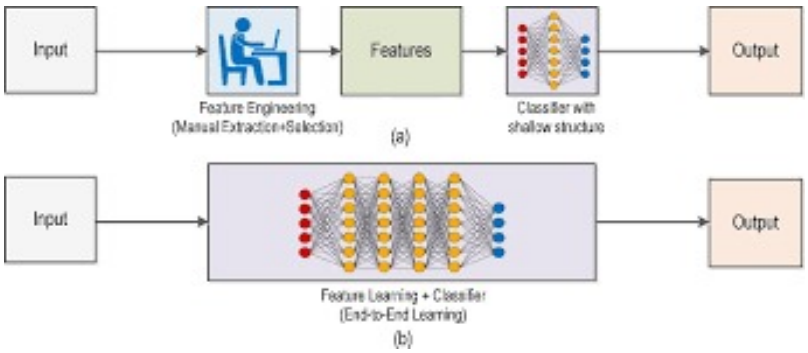

Figure 1 - Traditional Object Detection

2.2 Convolutional Neural Network

CNN plays an important role in computer vision related pattern recognition tasks, because of its superior spatial feature extraction capability and fewer computation costs[9]. CNN uses convolution kernels to convolve with the original images or feature maps to extract higher-level features. However, how to design better convolutional neural network architecture still remains as an opening question.Tuning CNNs for better accuracy has been an area of intensive research over the past several years, and some high-performance CNN architectures (e.g., AlexNet, VGGNet, GoogLeNet, and ResNet) have been introduced. Recently,the tuning of a CNN has progressed in two separate ways: One drawing representational power from deeper or wider architectures by increasing the number of trainable parameters (e.g., Inception-v4,Xception, and DenseNet), while other research has focused on building small and efficient CNNs due to limitations in computational power.As CNNs have become deeper and wider, overfitting problems have been raised, mainly due to the limitations of datasets, which are detrimental to the generalization of networks. To prevent overfitting,one way is to change the architecture of neural networks, for example, by adding dropout layers.Some studies have focused on the hyper-parameters in training options and adding regularization terms. Data augmentations such as random rotation, random cropping, and random reflections have also been widely applied for prevention of the overfitting problem.

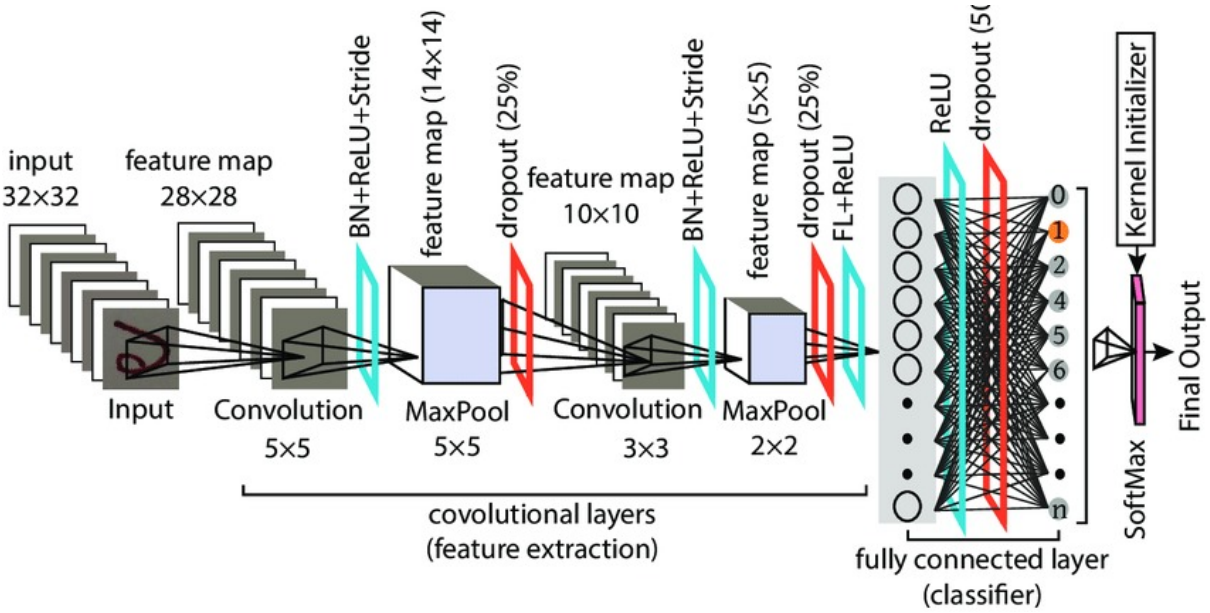

Figure 2 - Sample Convolutional Neural Network

J. Deng,W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248–255.

2.3 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces

are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms.

2.4 Keras

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

2.5 SMTP

Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending email and routing email between mail servers.Python provides smtplib module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.t has methods that support a full repertoire of SMTP and ESMTP operations.

3. SYSTEM DESIGN

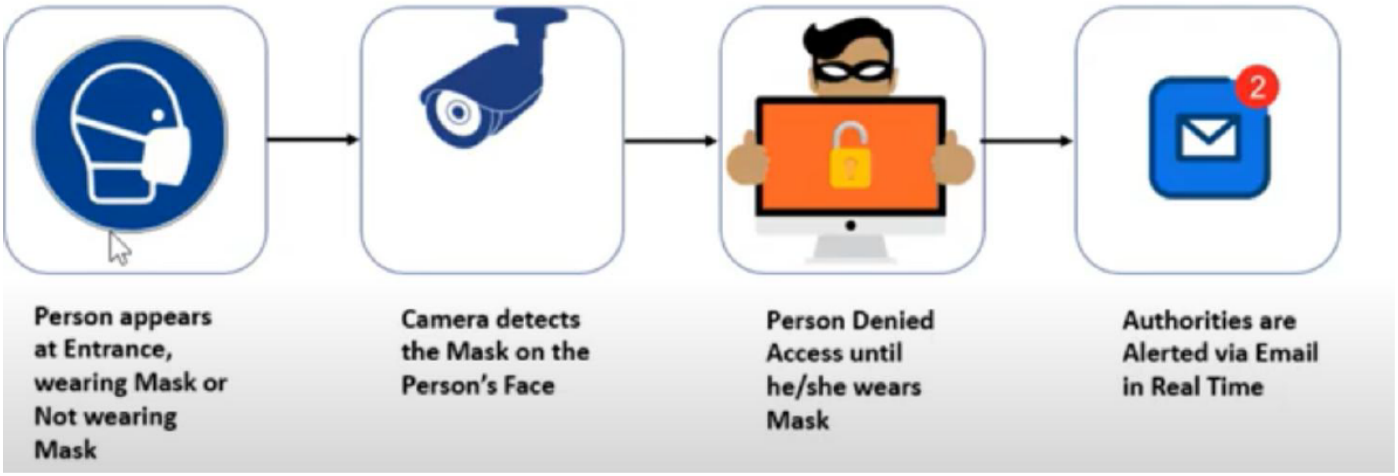

Figure 3 - Project Flow Diagram

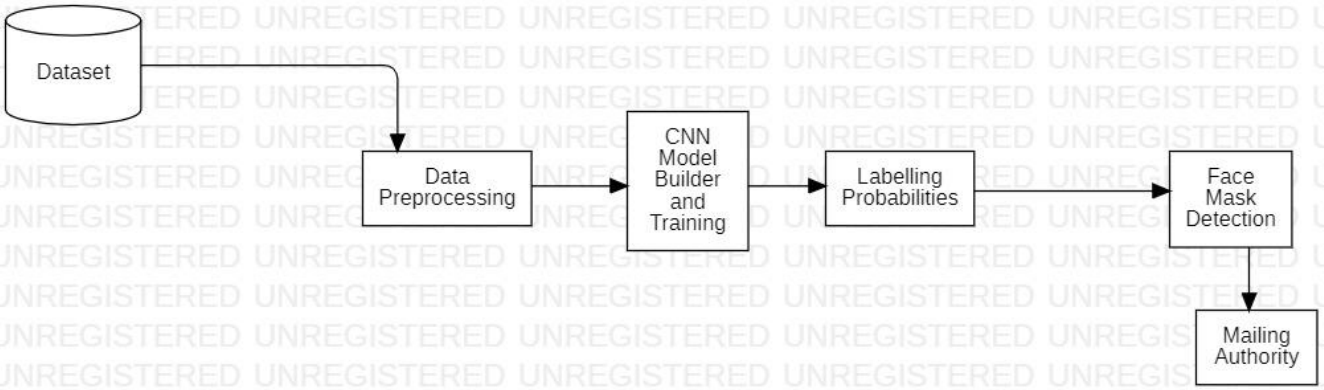

Figure 4 - Block Diagram

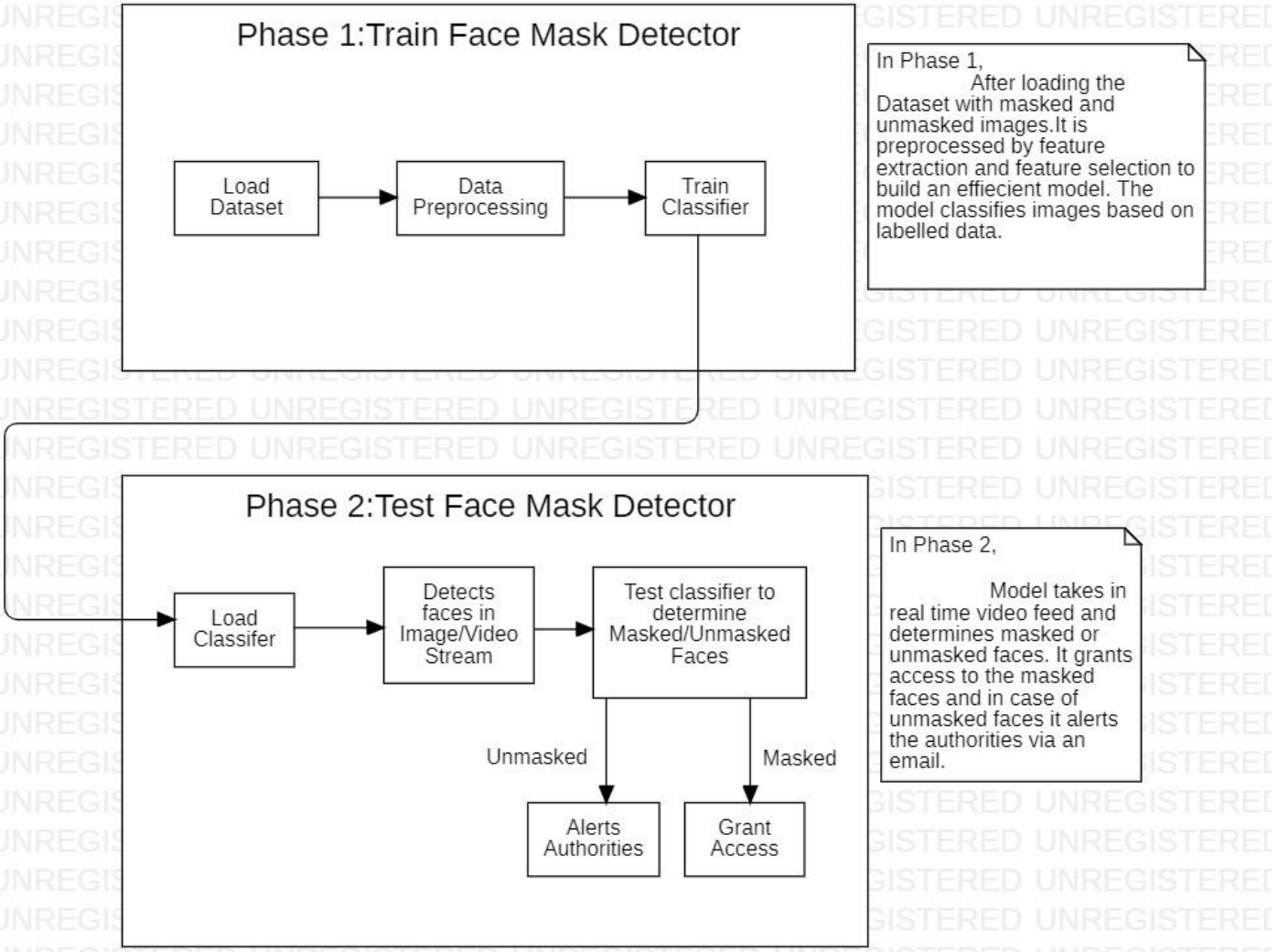

Figure 5 - Detailed Block Diagram

3.1 Dataset

The loaded dataset which consists of masked and unmasked face images is split into a number of masked images and unmasked images.The dataset is split into a training set and testing set.

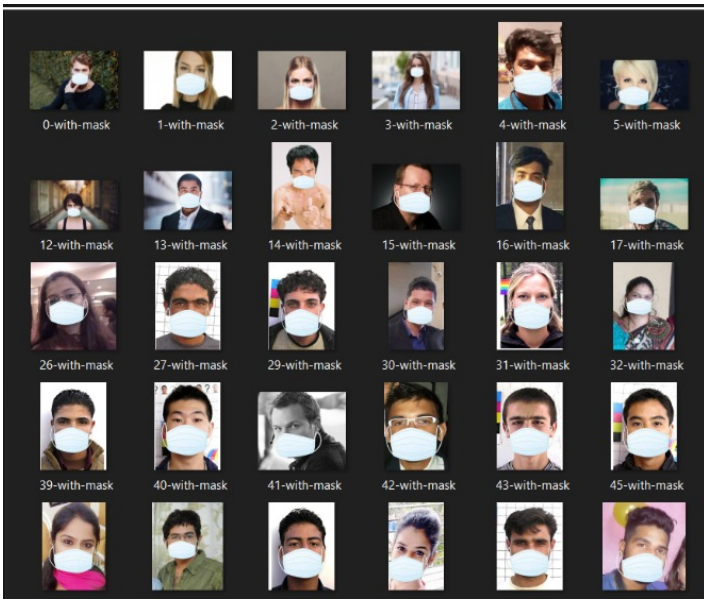

Figure 6 - With Mask Images

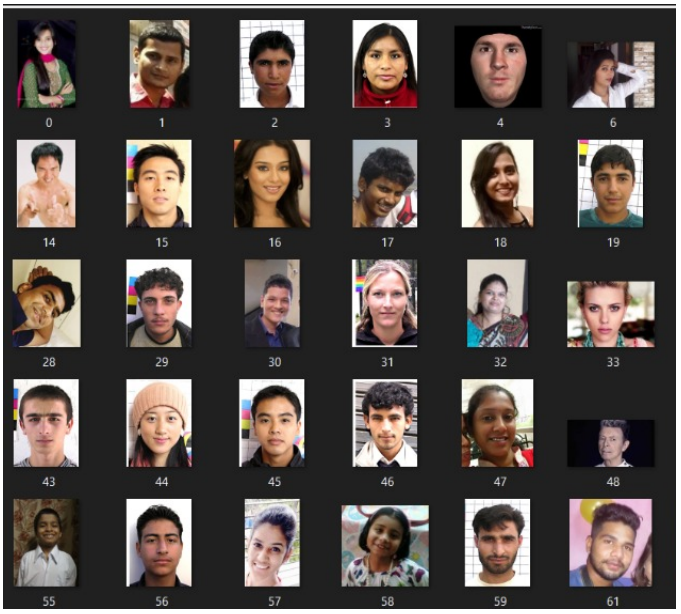

Figure 7 - Without Mask Images

3.2 DATA PREPROCESSING

Data Preprocessing involves

- Resizing Image
- Converting to Grayscale

We want to convert our original image from the BGR color space to gray, we use the code COLOR_BGR2GRAY.

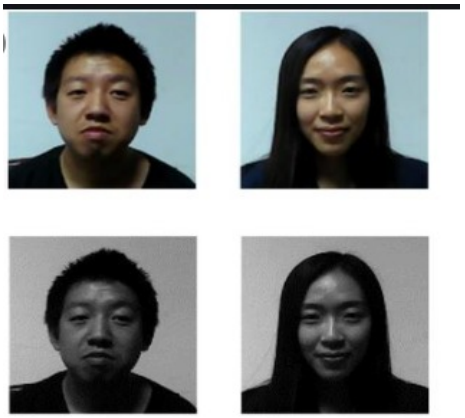

Figure 8 - Grayscale Images

3.3 CNN Model Builder and Training

Use a sequential API for model building and create a CNN with layers like convolutional, RelU, Maxpooling, Dropout.We also use softmax function to output a vector.For training the CNN Model, the images in the training set and the testing set are fitted to the sequential model we built using Keras library.

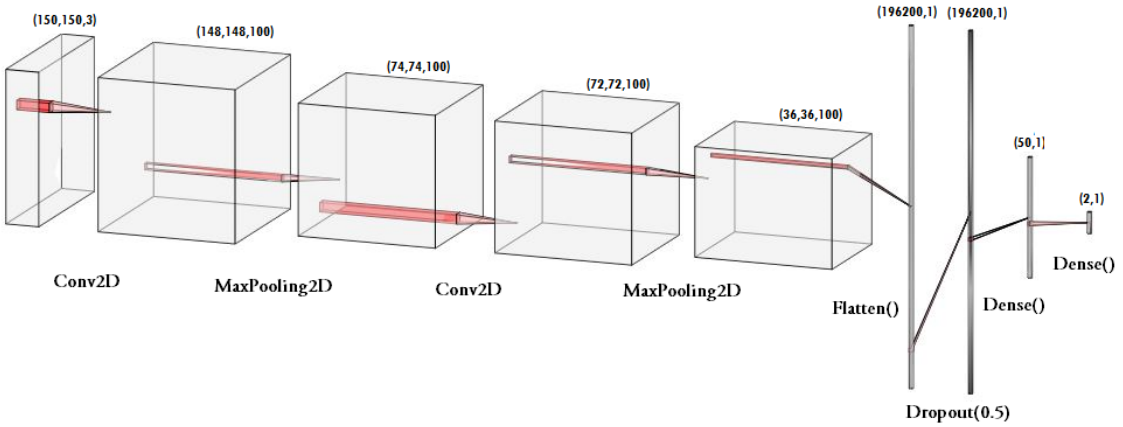

Figure 9 - CNN Model

3.4 Labeling Probabilities

After obtaining the trained CNN model,we label two probabilities for our results i.e RED for unmasked images and GREEN for masked images.We set the boundary rectangle colour using BGR values.

3.5 Mailing Authorities

Incase of an unmasked face is detected the authorities are alerted via an email and the access is denied.

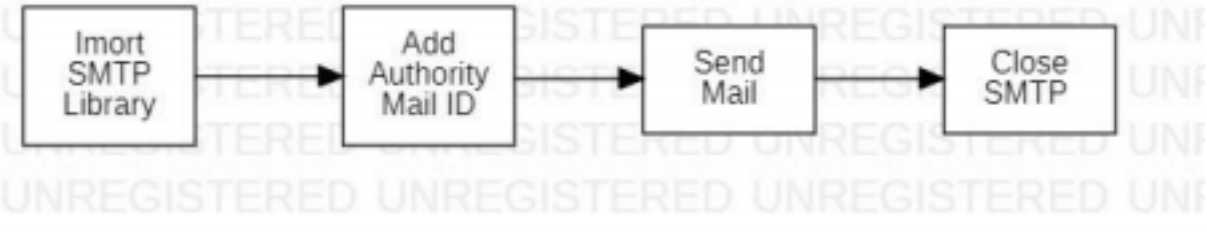

Figure 10 - Block Diagram of Mailing Authorities Module

4. EXPERIMENTAL RESULTS

4.1 Evaluation Parameters

4.1.1 Accuracy

Equation 4.1

$$\text{Accuracy} = \frac{\text{Number of Classified Samples}}{\text{Total number of Samples}}$$

OR

$$\text{Accuracy} = \frac{TP + TN}{TN + TP + FN + FP}$$

4.1.2 Loss Function

Categorical Loss Function

It is a Softmax activation plus a Cross-Entropy loss. In this loss, we will train a CNN to output a probability over the C classes for each image. It is used for multi-class classification.

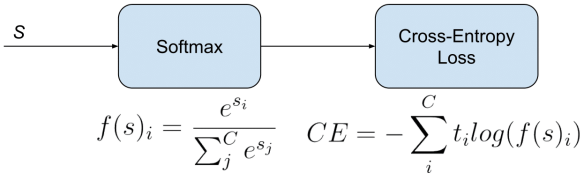

Figure 11 - Softmax Function

4.1.3 Plotting Graph

Number of Epochs v/s Loss Value
Number of Epochs v/s Accuracy

Table 1: Model Summary

layer	Layer shape details
Optimizer	**Adam**
Loss function	**Categorical cross_entrophy**
metrics	**[accuracy]**
Conv2D	**64 filter,3 x 3filter size,Relu activation function**
Max pooling	**2 x 2 kernel size**
Dropout	**50%**
Conv2D	**128 filter,3 x 3filter size,Relu activation function**
Max pooling	**2 x 2 kernel size**
Dropout	**50%**
Dense	**64 neurons,Relu**
output	**Softmax,2 classes**

Inference 1:

In our Model we used Adam optimizer which is relatively better than other optimizers especially for Binary Classification. We measure our model with Accuracy(Training and Validation).Our model consists of Conv2D,Dense and Dropout layers where the unit of dropout layer is 50%.

4.2 Results

Here we build a sequential model using keras library.
Since the number of classes are 2(masked and unmasked) we add 2 conv2D layers. We then flatten the input and then to avoid overfitting we use dropout.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 110, 110, 64)      640
_________________________________________________________________
activation (Activation)      (None, 110, 110, 64)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 55, 55, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 53, 53, 128)       73856
_________________________________________________________________
activation_1 (Activation)    (None, 53, 53, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 26, 26, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 86528)             0
_________________________________________________________________
dropout (Dropout)            (None, 86528)             0
_________________________________________________________________
dense (Dense)                (None, 64)                5537856
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 130
=================================================================
Total params: 5,612,482
Trainable params: 5,612,482
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 12 - Model Summary

Here in order to train the module we use Adam optimizer with a learning rate of 0.001 and the epoch value is set to 50.Here the model is trained and then tested with a validation set.The validation split is 0.25.The model will be trained and validated continuously within given epochs till stable accuracy and loss is attained.

```
Epoch 1/50
14/14 [==============================] - 11s 751ms/step - loss: 0.8894 - accuracy: 0.5352 - val_loss: 0.6968 - val_accuracy: 0.
4694
Epoch 2/50
14/14 [==============================] - 9s 648ms/step - loss: 0.6695 - accuracy: 0.5851 - val_loss: 0.6890 - val_accuracy: 0.5
782
Epoch 3/50
14/14 [==============================] - 11s 785ms/step - loss: 0.6307 - accuracy: 0.6348 - val_loss: 0.6809 - val_accuracy: 0.
5374
Epoch 4/50
14/14 [==============================] - 9s 638ms/step - loss: 0.5126 - accuracy: 0.7248 - val_loss: 0.5660 - val_accuracy: 0.6
939
Epoch 5/50
14/14 [==============================] - 11s 829ms/step - loss: 0.4497 - accuracy: 0.7727 - val_loss: 0.3976 - val_accuracy: 0.
8367
Epoch 6/50
14/14 [==============================] - 9s 640ms/step - loss: 0.3213 - accuracy: 0.8860 - val_loss: 0.4318 - val_accuracy: 0.8
435
Epoch 7/50
14/14 [==============================] - 9s 664ms/step - loss: 0.1907 - accuracy: 0.9471 - val_loss: 0.3228 - val_accuracy: 0.8
980
Epoch 8/50
14/14 [==============================] - 12s 881ms/step - loss: 0.1399 - accuracy: 0.9446 - val_loss: 0.3391 - val_accuracy: 0.
8912
Epoch 9/50
14/14 [==============================] - 9s 656ms/step - loss: 0.1065 - accuracy: 0.9688 - val_loss: 0.3694 - val_accuracy: 0.8
844
Epoch 10/50
14/14 [==============================] - 11s 833ms/step - loss: 0.0793 - accuracy: 0.9756 - val_loss: 0.3495 - val_accuracy: 0.
8776
Epoch 11/50
14/14 [==============================] - 9s 643ms/step - loss: 0.0734 - accuracy: 0.9633 - val_loss: 0.3182 - val_accuracy: 0.9
116
Epoch 12/50
14/14 [==============================] - 12s 844ms/step - loss: 0.0837 - accuracy: 0.9695 - val_loss: 0.4271 - val_accuracy: 0.
8707
```

Figure 13 - Experimental Result after testing Model

Here we plot two graphs between

- Accuracy and number of epochs
- Loss and number of epochs

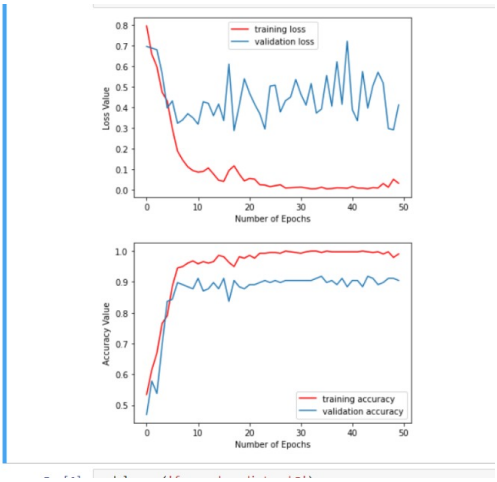

Figure 14 - Plotted Graphs

After training the model we load it and capture live video feed and detect face masks.
Here we label the video feed with either “Mask ON” or “No Mask” based on the video feed that is being captured.

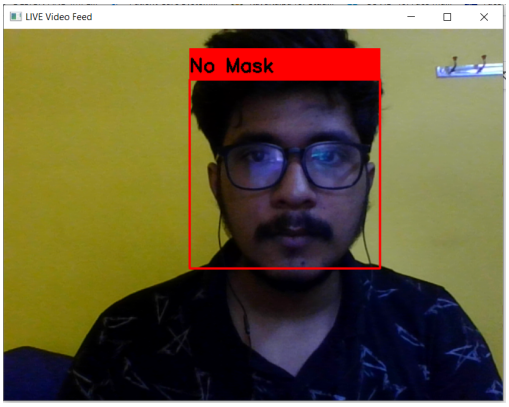

Figure 15 - NO MASK

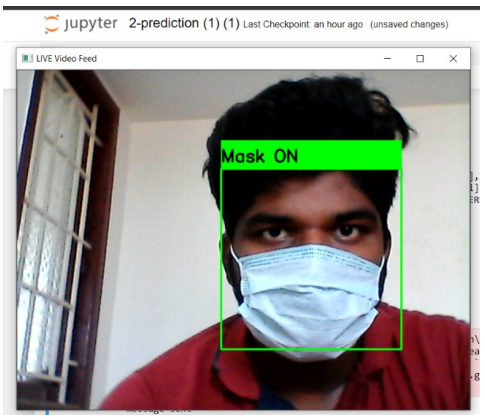

Figure 16 - MASK ON

Here once an image is detected without a masked face it attaches the image along with some message and a mail is sent to the authority.
Here we establish an SMTP server using smtplib.SMTP and encapsulates an SMTP connection and allows us to access its methods.

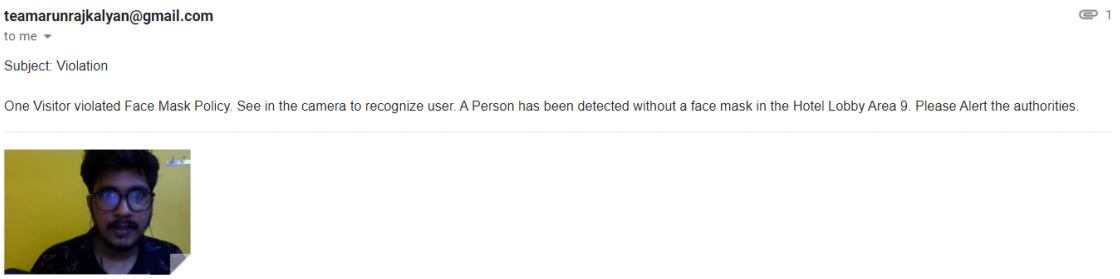

Figure 17 - Alerting Authority via E-mail

5. RESULT ANALYSIS

Accuracy and Loss with different activation function and layers

An activation function in a neural network defines how the weighted sum of the **input** is transformed into an **output** from a node or nodes in a layer of the network.

Table 2 : Comparison between Activation Functions with Loss and Accuracy

S.No	Activation Function	Layers Used	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
1.	Relu	2	99%	90%	0.0195	0.4123
2.	Softmax	2	53%	46%	0.5376	0.6969
3.	Sigmoid	2	54%	47%	0.6909	0.6969
4.	Relu	3	98%	98%	0.0470	0.1788

Inference 2:

As we can see the accuracy of ReLU is way higher than the other two functions we thus prefer ReLU over softmax and sigmoid. Our final model is having Relu as an activation function for better results.

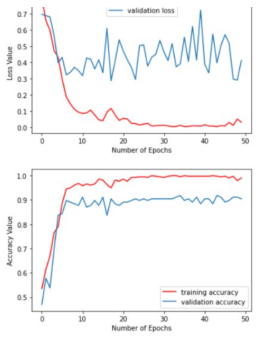

Figure 18 - Plot_Relu

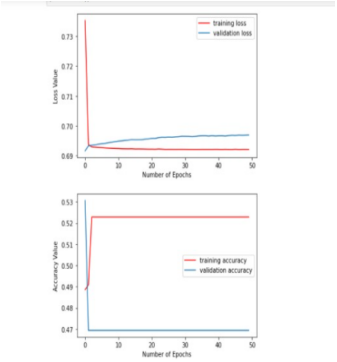

Figure 19 - Plot_ Softmax

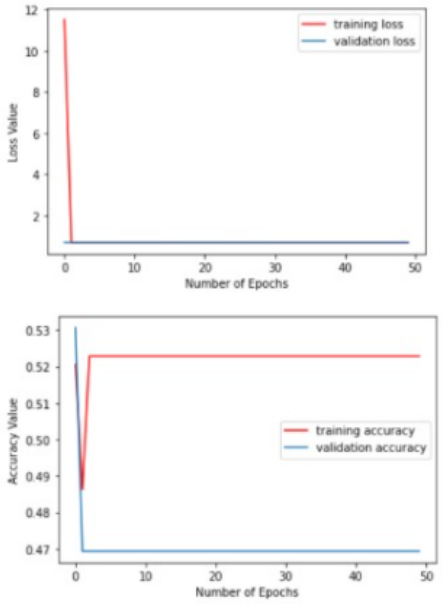

Figure 20 - Plot_Sigmoid

Table 3 : Comparing Models with Epoch Value

S.No	Epoch	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
1.	10	100%	93%	0.0054	0.296
2.	20	100%	95%	0.0016	0.2093
3.	25	98%	97%	0.0190	0.1362
4.	50	99%	98%	0.0470	0.1788

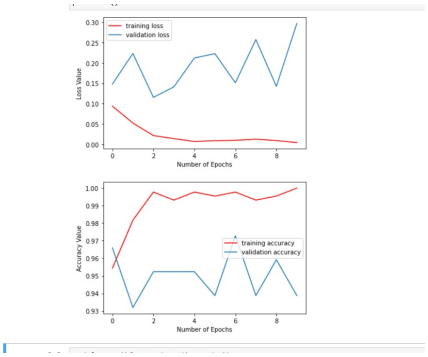

Figure 21 - Plot with Epoch 10

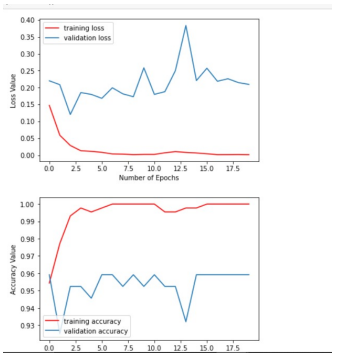

Figure 22 - Plot with Epoch 20

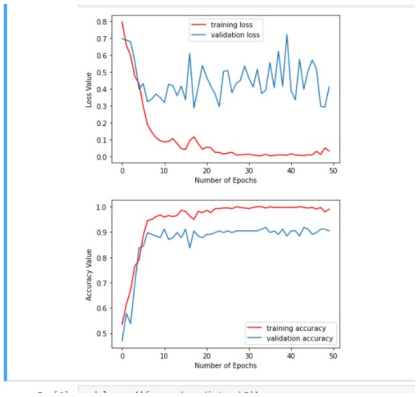

Figure 23 - Plot with Epoch 25

6. CONCLUSION

As the technology is blooming with emerging trends the availability so we have novel face mask detectors which can possibly contribute to public healthcare. We used OpenCV, keras and CNN to detect whether people were wearing face masks or not. The models were tested with images and real-time video streams. Training the model is the first part of this project and testing using a webcam using OpenCV is the second part. By the development of face mask detection we can detect if the person is wearing a face mask and allow their entry would be of great help to the society.

References

1. D. Chiang., “Detect faces and determine whether people are wearing mask,” https://github.com/AIZOOTech/FaceMaskDetection, 2020.
2. A. Kumar, A. Kaur, and M. Kumar, “Face detection techniques: a review,” Artificial Intelligence Review, vol. 52,no. 2, pp. 927–948, 2019.
3. S. Yang, P. Luo, C.-C. Loy, and X. Tang, “Wider face: A face detection benchmark,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 5525–5533.
4. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
5. Y. Liu, A. A. Gayle, A. Wilder-Smith, and J. Rocklöv, “The reproductive number of covid-19 is higher compared to sars coronavirus,”Journal of travel medicine, 2020
6. Z. Zou, Z. Shi, Y. Guo, and J. Ye, “Object detection in 20 years: A survey,” arXiv preprint arXiv:1905.05055,2019.
7. A. Shrivastava, A. Gupta, and R. Girshick, “Training region-based object detectors with online hard example mining,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 761–769.
8. R. Girshick, “Fast r-cnn,” in Proceedings of the IEEE international conference on computer vision, 2015, pp.1440–1448.
9. J. Deng,W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248–255.
10. W. H. Organization et al., “Coronavirus disease 2019 (covid-19): situation report, 96,” 2020.
11. N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05), vol. 1. IEEE, 2005, pp. 886–893.