Detection of Improperly Worn Face Masks using Deep Learning – A Preventive Measure Against the Spread of COVID-19

Anubha Bhaik, Vaishnavi Singh, Ekta Gandotra*, Deepak Gupta

Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat, Solan, HP (India)

Received 10 November 2020 | Accepted 24 June 2021 | Published 9 September 2021

Abstract

Coronavirus disease 2019 has had a pressing impact on people all around the world. Ceasing the spread of this infectious disease is the urgent need of the hour. A vital method of protection against the virus is wearing masks in public areas. Not merely wearing masks but wearing masks properly can ensure that the respiratory droplets do not get transmitted to other people. In this paper, we have proposed a deep learning-based model, which can be used to detect people who are not wearing their face masks properly. A convolutional neural network model based on the concept of transfer learning is trained on a self-made dataset of images and implemented with light-weighted neural network called MobileNetV2 for mobile architectures. OpenCV is used with Caffe framework to detect faces in an input frame which are further forwarded to our trained convolutional neural network for classification. The method has been implemented on various input images and classification results have been obtained for the same. The experimental results show that the proposed model achieves a testing accuracy and training accuracy of 93.58% and 92.27% respectively. Optimal results with high confidence scores and correct classification have also been achieved when the proposed model was tested on individual input images.

Keywords

COVID-19, Mask Classification, MobileNetV2, OpenCV, Transfer Learning.

DOI: 10.9781/ijimai.2021.09.003

I. Introduction

THE coronavirus disease 2019 (COVID-19) is an infectious disease that can result in mild to severe illnesses in people infected by it. It is transmitted mainly through respiratory droplets of saliva or discharge from the nose when a person infected with coronavirus coughs or sneezes. Thus, it is essential to practice a proper respiratory protocol such as covering our face while sneezing or coughing [1]. The mucous membranes of the face should be covered properly with protective equipment to protect oneself and others from the continual transfer of the disease.

Face masks are being used by people all over the world now. In many countries, it is now compulsory to wear a face mask when stepping out of home. However, many people do not wear face masks properly. They fidget with their masks and pull them under their noses or completely off their faces to rest under their chins without realizing that improperly wearing a mask leads to an increased risk of contamination. Wearing a face mask limits the spread of the virus from someone who knows or does not know they have an infection or not. It also reminds others to continue the practice of physical distancing [2]. Moreover, the presence of asymptomatic spreaders of the COVID-19 virus means that wearing a face mask should be a part of lessening the cases of COVID-19 [3]. Thus, the masks must be worn properly covering the mouth and nose appropriately to prevent respiratory droplets from spreading. A recent study [4] has pointed out that surgical face masks can prevent the transmission of coronavirus and influenza viruses from symptomatic people. Another study [5] states that the reproductive rate of COVID-19 is higher as compared to the SARS coronavirus, and thus it is essential to wear masks properly as a measure to keep public health in mind. Countries have been exiting the lockdown lately to reduce the effect of the pandemic on the economy, but the coronavirus persists as an inevitable danger in most of the countries. Since the outbreak of this disease is not only the concern of a single country but of the entire world. The stringent measures implemented by the government have been effective in combating the spread of COVID-19 disease [6]. For instance, China has been using mass surveillance to monitor people and track the spread of coronavirus. Other nations are also deploying technologies like video camera footage, credit card information, and location tracking as they race against the outbreak. Surveillance of activities of people can be effective as we can monitor whether people are properly taking protective measures, and by not letting them enter a public place if they are careless about protection. A recent study [7] has established that face masks have been effective in the containment of COVID-19 in South Korea. This study further states that in addition to maintaining social distancing and sanitizing hands, properly wearing appropriate masks has been efficacious in lessening severe cases in South Korea. Another study [8] focuses on how the universal

* Corresponding author.
E-mail address: ekta.gandotra@gmail.com

use of covering the face by even a simple cloth mask, if not a surgical mask, can help in acting as a preventive measure.

With this study, we would contribute to public healthcare by detection of people not wearing their masks properly at places where the chances of getting infected are high. People present at places where there are low chances of enforcing social distancing, can be checked for wearing their masks properly, especially in severely affected zones of countries. We intend to use mobile devices to check whether someone is wearing their masks properly or not. The main contributions of the paper are:

1. A dataset containing images of people wearing masks properly and improperly.
2. A model for the detection and classification of faces wearing masks properly and improperly.
3. Experiments to evaluate the performance of the proposed model on a dataset using various evaluation metrics.

This paper is organized as follows: Section II discusses the background and the work related to our study. Section III describes the detailed methodology used for the proposed model. Further, Section IV presents the analysis and visualization of the experimental results followed by the conclusion in Section V.

II. Related Work

This section discusses the related research work behind the proposed model for detecting improperly worn face masks.

A. Convolutional Neural Networks

Convolutional neural network (CNN) is a class of deep learning models that largely deals with the analysis of visually descriptive data. CNNs can extract important features from visual data without human supervision with the help of various layers. Different layers perform different kinds of transformations on the data. CNNs treat data as spatial and can simplify the complexity of images to be better understood and processed by the machine and hence are widely used for pattern recognition. A classic CNN is composed of multiple layers namely convolutional layers and pooling layers which are used for the extraction of important features from input data. It also has some layers in the end which take the output from the two mentioned layers and help in classifying the data into labels. CNNs have wide applications like face detection and recognition, classification of malware applications [9], classification of X-ray images [10], etc. Image classification is one of the most popular applications of CNN. Sultana et al. [11] have done a study where they explained different architectures of CNN used for image classification. Shinet et al. [12] explored and evaluated different CNN architectures and discussed when and why transfer learning from pre-trained ImageNet CNN models can be valuable. Demir et al. [13] extracted distinctive face features using CNN and used the Softmax classifier to classify faces in the fully connected layer of CNN. In [14], the inception network has been proposed for allowing the network to learn the best combination of kernels, leading to an effective image classification method as well. Wang et al. [15] proposed the residual attenuation network for image classification achieving 0.6% top-1 accuracy improvement as compared to ResNet-200. They focused on the depth of the network and used the attention mask mechanism to take image classification to a new level.

B. Object Detection

Object detection is a computer vision approach used to identify the objects present in images or videos. Many deep learning-based frameworks for object detection have been proposed in the literature, covering various aspects of its applications in the real world like facial recognition, face detection, detection of obstacles for self-driving cars, and more. A review on some object detection architectures has been carried out by Zhao et al. [16]. Other fast techniques like You Only Look Once (YOLO) [17] have also been proposed for object detection.

C. Face Detection

Face detection is a popular application of object detection that is being widely used today. A comprehensive survey of various techniques for facial detection in digital images is due to Kumar et al. [18]. Sivaram et al. [19] proposed a technique that uses recurrent neural networks (RNN) and deep neural networks (DNN) to take in the shape of the face for accurate facial detection. A camera-based PCA facial recognition system has been built by Khan et al. [20] using programming on technologies like OpenCV, Haar Cascade, and Python.

Face detection or recognition systems have been in demand for several security-based applications too, like surveillance or tracking of suspected people, access management, etc. Zhang et al. proposed a framework for serving better surveillance functionality for ATMs. The technique included tackling severely occluded faces by fusing the features of faces like skin color and facial structure; it achieved 98.56% accuracy on detection of face occlusion [23].

D. Face Mask Detection

Face mask detection has also been explored by researchers to tackle the situation of COVID-19 for ensuring if people are wearing masks or not. In [21], Meenpal et al. have studied facial mask detection using semantic segmentation. They have proposed a binary face classifier that can detect any face in the frame. Their method uses pre-trained weights of VGG-16 architecture for feature extraction and the experimental results give a mean pixel-level accuracy of 93.88% for the segmented face masks. Besides, Jiang et al. [22] have proposed a face mask detector that is able to detect face masks. They have tried to distinguish between people wearing masks and people merely covering their mouths with their hands.

This pandemic certainly demands the need for proper mask detection for the security of the health of citizens. However, in the aforementioned studies, the authors have not considered whether a person is wearing a face mask properly. The algorithms proposed in these studies only detect if the person is wearing a face mask or not. In this paper, we have proposed a CNN-based model that addresses this task of checking if the person is wearing the mask properly.

III. Proposed Methodology

This section describes the proposed methodology for the detection of improperly worn masks. Fig. 1 shows the complete workflow of the used methodology.

A dataset consisting of images of people wearing masks both properly and improperly is created and used in this work. These images were collected from the local, existing datasets, and the Internet. The images were pre-processed to enhance their generalization during the training of the CNN. Further, CNN is trained on the created dataset for classification purpose. For visualization of results, OpenCV and Caffe framework are used. The model creates a bounding box around a person's face in the input image classifying whether the mask worn is proper or improper. Finally, the experimental results are analyzed and visualized. The various steps of the proposed methodology are further elaborated in the following sections.

A. Data Acquisition

The images of people wearing proper face masks are collected from the images present in existing datasets [24] and various other Internet sources. Since these datasets had fewer images of improperly

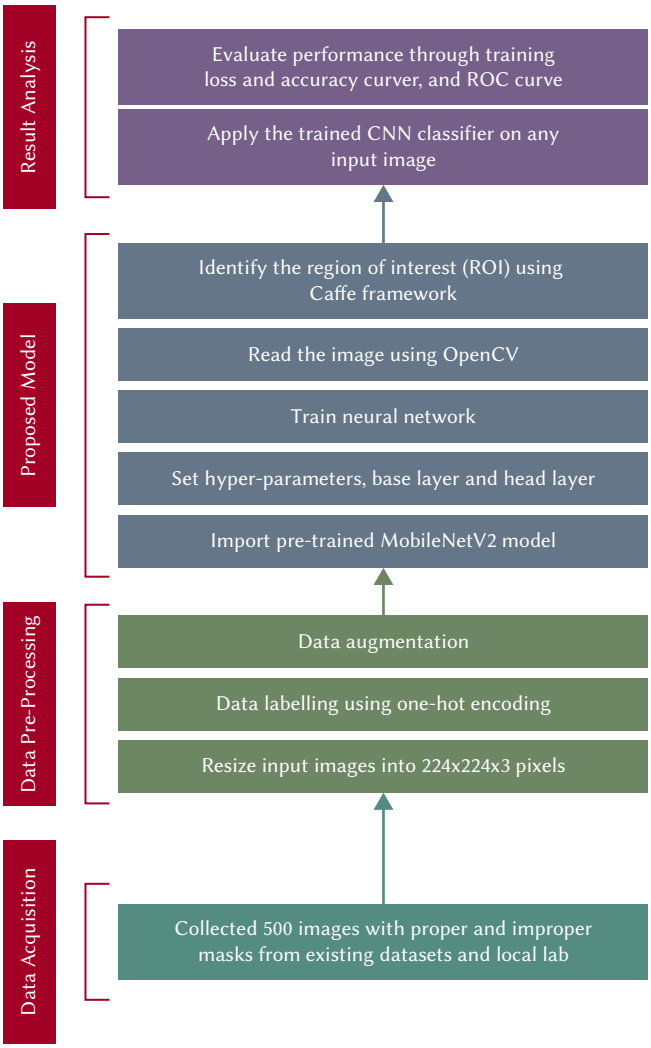

Fig. 1. Workflow of the methodology used.

worn masks, we collected such images from the Internet and local lab. Finally, our dataset consists of 500 images, equally distributed among properly and improperly worn mask categories. Figs. 2 and 3 show the sample images from the dataset of people with proper and improper masks, respectively.

Fig. 2. Sample images of the dataset belonging to the class – proper mask.

Fig. 3. Sample images of the dataset belonging to class – improper mask.

B. Data Pre-processing

The dataset consists of images of different sizes, and thus, these images are converted into a uniform size of 224 × 224 pixels. After the application of RGB reshaping, a 224 × 224 × 3 image is given as input to the proposed model. The class labels are one-hot encoded. These pre-processed images and encoded labels are added to separate lists, one for the pre-processed images and the other for the class labels. Furthermore, data augmentation strategies like random rotation, shift, shear, zoom and flip, are applied on the images for increasing the generalization of data which help in improving the performance of the model.

C. Proposed Model

This section describes the proposed model. The neural network for classification is built and trained after setting the various hyper-parameters and hidden layers. Thereafter, the trained classifier is applied to input image for classification into 'proper' and 'improper'.

1. Training the Neural Network

The problem for our proposed model is to learn the interpretation of various features in images and classify them accordingly. CNNs help in leveraging the spatial information in images. Fig.4 presents a basic architecture of CNN which consists of the input images, the layers of the network, and the corresponding output.

We have split our dataset into training set and testing set in such a way that 80% data is used for the training purpose and 20% data is used for the testing purpose. For achieving the optimum results on our dataset, we use an aspect of deep learning called transfer learning. Transfer learning is the act of transferring the knowledge previously gained by one model on a specific task to a new similar task that will benefit from some or all the layers of the previously built model. To aid the use of our model on mobile devices, we used MobileNetV2 as the base model. It has less computation cost and is an efficient mobile-oriented model for transfer learning [25]. Additionally, it is an effective feature extractor used for object detection that improves the performance of our detection. The pre-trained weights for the ImageNet [26] dataset have been used as the backbone.

Unlike the typical convolution, MobileNetV2 utilizes an advanced version of convolutional operation called the depth-wise separable convolution which leads to a lesser number of computations and transformations on the images than the conventional convolution. It gets applied to images in two parts [27]. The first part is the depth-wise convolution used to perform the filtering stage and the second part is pointwise convolution for the combining stage. It is a light-

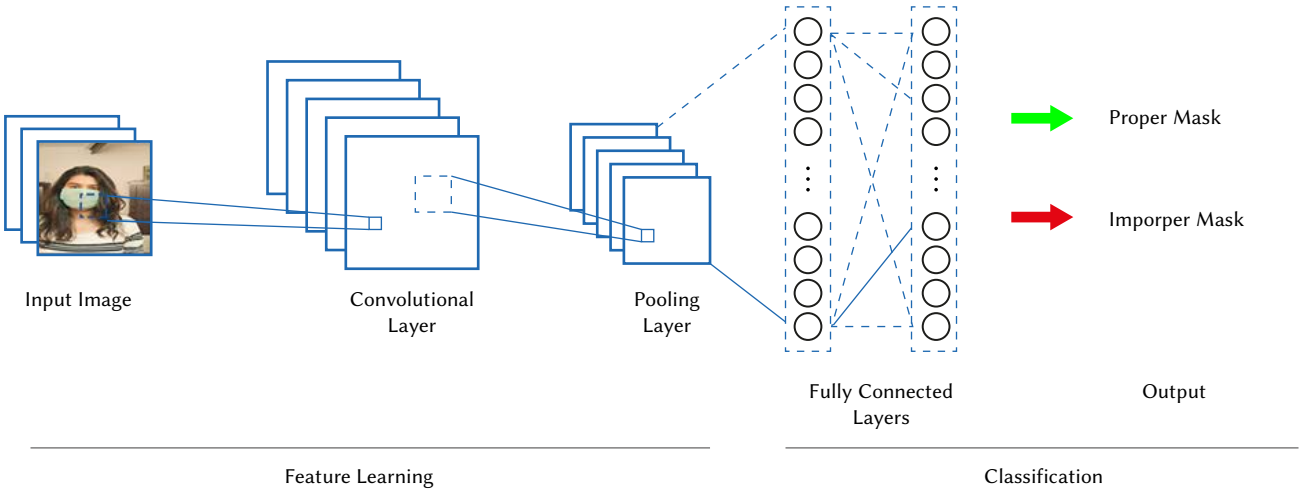

Fig. 4. Elementary CNN architecture for image classification.

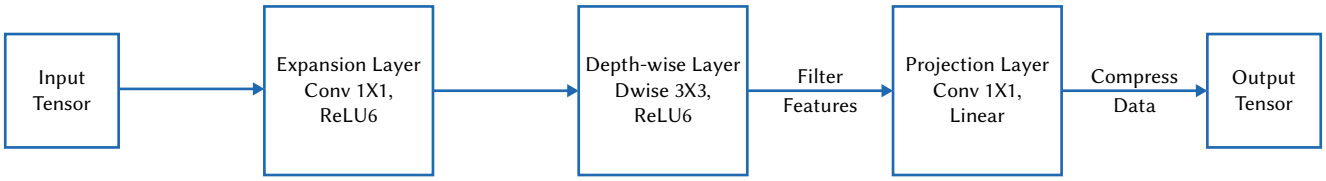

Fig. 5. Basic architecture of a block in MobileNetV2.

weight model with low-latency which provides comparable accuracy to other heavy and complicated models. Since the size of our dataset is small and its affinity to ImageNet is not that high, we fine-tune the top layers of MobileNetV2 for our work. The head layers of the pre-trained MobileNetV2 CNN architecture are unfrozen while importing it and are replaced with new custom layers. The weights for the input to this pre-trained MobileNetV2 architecture are set by default. All base layers (we call these as BaseModel) below the head of the MobileNetV2 architecture are frozen to prevent their weights from getting updated during backpropagation. The first layer in the network is a fully convolutional layer with input size $224 \times 224 \times 3$.

The fundamental building block of MobileNetV2 architecture is a bottleneck depth-separable convolution with residuals, so the input layer is followed by residual bottleneck layers [27]. Each block of the model consists of 3 convolutional layers as shown in Fig.5. First is a 2D convolutional layer (expansion layer) which performs 1×1 convolution for expansion of the number of channels in the data, then batch normalization and ReLU6 non-linearity are applied, which limits the maximum value of activation to 6. The default expansion factor is taken as 6 in the expansion layers. The second is a depth-wise convolution layer, again with batch normalization and ReLU6 non-linearity. Some of the convolutional layers have a stride of 2 for achieving spatial down-sampling since there are no conventional pooling layers, other layers are kept at a stride of 1. The third layer known as a pointwise convolutional layer (projection layer) performs linear convolution to reduce the dimensionality of input (also known as a bottleneck layer) and again accompanied by batch normalization [27]. The first block of the model is different as it comprises 3×3 convolution with 32 channels rather than the default 1×1 convolution which happens in the expansion layer.

The last five custom layers (we call these as HeadModel) which produce output for the model include the average pooling 2D layer with pool size 7×7, reducing the dimensionality by acquiring average values from each region of the image. This layer precedes a flatten layer that reshapes the pooled feature map to a single column vector. The simple feature vector is now put into a dense layer of 128 units of size accompanied by ReLU activation by using (1).

$$f(y) = \{y \; for \; y \geq 0, 0 \; for \; y < 0\} \tag{1}$$

A dropout layer is applied on this dense layer to prevent the model from overfitting, with a threshold value of 0.5. Then a final dense layer is applied with Softmax non-linear activation by using (2) to provide two output values, i.e., probabilities of the image belonging to the proper and improper mask groups, respectively.

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i^J e^{x_i}} \tag{2}$$

where x is a vector of the inputs in the form of images to the output layer and i indexes the output unit such that $i - 1, 2, 3, \ldots f$. The detailed summary of our proposed model consisting of the MobileNetV2 model (BaseModel) and the custom layers (HeadModel) is provided in Appendix A.

We use Adam optimizer for the optimization of the CNN and binary cross-entropy as loss function as shown in (3). This loss function is used in a binary classification problem.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i . log(p(y_i)) + (1 - y) . log(1 - p(y_i)) \tag{3}$$

Here, y is the label (which has been one-hot encoded) and p(y) is the predicted probability of one of the labels.

The initial learning rate is set to 0.0001. A learning rate decay schedule (represented by (4)) is created which helps in increasing the model accuracy and descend into areas of lower loss. Table I shows hyperparameters used in our CNN Model.

$$\alpha = \frac{1}{1 + decay * iteration} * \alpha_0 \tag{4}$$

Where α and α_0 represent the learning rate and initial learning rate respectively, and

$decay = \alpha_0 / total \; number \; of \; epochs$

TABLE I. Hyperparameters Used in Our CNN Model

Parameter Name	Value
Kernel Size	3 x 3
Activation Function	ReLU6 (BaseModel), ReLU (HeadModel)
Average Pooling	7 x 7 (HeadModel)
Optimizer	Adam
Loss Function	Binary cross entropy
Dropout	0.5
Epochs	50
Batch Size	32
Initial Learning Rate	0.0001
Fully Connected (Activation Function)	Softmax

2. Detection of Region of Interest (ROI)

After training the CNN, we have used a deep learning framework, Caffe, along with an open-source computer vision library (OpenCV) for face detection using static input images. For extracting the region of interest (ROI) in the image, the DNN module of OpenCV is used with Caffe. The network model which is stored in Caffe framework format (with the learned network) and a file containing the text description of the network architecture is read using OpenCV DNN module. The file with the Caffe framework format has been provided by the OpenCV for face detection [28], [29] and contains the weights for the actual layers. The Caffe model is based on the Single Shot MultiBox Detector (SSD) framework which uses ResNet as a base network for facial recognition [30].

The trained model is used on various static input images to detect whether the person is wearing the mask properly over his nose. An input image is first uploaded and pre-processed using OpenCV DNN module. The spatial dimensions of the input image are extracted and converted into a 4D Binary Large OBject (BLOB) which is further used to perform functions like scaling, mean subtraction, and resizing on the input image. The scale factor is set to 1.0. After normalizing the input image to create a BLOB, it is passed through the DNN to obtain face detections. The detections obtained are further checked for the probability or confidence which is used to classify the input image as proper or improper. The threshold confidence (or probability) is kept at 0.5 to filter all the weak detections. Further, OpenCV is used to extract the region of interest (ROI) of the face which helps in displaying the bounding box. The extracted face ROI is converted from BGR to RGB ordering of channels and the image size is set to 224 × 224 pixels to pass it through the trained model. Finally, this pre-processed input image is passed through the trained model to determine if the mask is worn correctly or not. This can finally be visualized by a bounding box labelled with the class score in the image. The class score is the probability that the image contains a face with a proper or an improper mask.

3. Algorithm

The proposed algorithm (as shown in Algorithm I) is based on transfer learning and facial detection using OpenCV. θ is the initial learning rate which is used to update weights during the training phase. It has a value that is often in the range of 0.0 to 1.0. β denotes batch size which refers to the number of training images that are used in a single iteration. ϕ represents epochs which define the total number of times the model will iterate over the training images. The values of these three hyper-parameters (θ, β and ϕ) can be tuned based on Steps 3-5 in a hit and trial way to achieve better accuracy results. The updated output weights (ω) are stored in a .h5 file which are used further to get predictions on other random input images. δ gives the number of face detections that are obtained on our static input image. α is used to filter out all the weak face detections in our input image and is known as the confidence (or probability) for detection of facial features.

The algorithm advances by using the architecture of MobileNetV2 as the base layers (refer to Step 4) and the addition of custom head layers (refer to Step5). A for loop is applied in the range of the total number of defined epochs which updates the weights of only the custom layers through forward and back propagation (refer to Steps 8-10). The input static image is uploaded, and features extracted for facial detection (refer to Steps 11-13). Another for loop is applied to detect the region of interest in the input image and plot a bounding box with the indicative predicted probability as a label (refer to Steps 13-16). Finally, the trained model is applied to the input image only if facial features are detected (refer to Step 17).

Algorithm I. Algorithm for Detection of People Wearing Proper and Improper Face Masks

Input: Images of size h×w×d; where h⟶height of image,w⟶width of image,d⟶number of channels in the RGB image.

$\theta \longrightarrow$ initial learning rate

$\beta \longrightarrow$ number of samples of images trained in one iteration

$\phi \longrightarrow$ number of epochs

$\delta \longrightarrow$ number of face detections on input image

$\alpha \longrightarrow$ numeric constant to filter out weak detections

Output: classified output image with probability of prediction

Begin:

1. Split the input images randomly into training set ($\sigma 1$) and testing set ($\sigma 2$) using 80% data for the training set and the remaining 20% for the test set.
2. Construct the image generator for augmentation of images.
3. Initialize the CNN parameters θ, $\beta$and ϕ.
4. Determine the base layers of CNN architecture, i.e., MobileNetV2.
5. Set the head layers, $CNN_{averagepooling2D}$, $CNN_{flatten}$, CNN_{dense}, $CNN_{dropout}$
6. Set the last layer (at the end of the fully connected layers) that contains labels for classification.
7. Train the CNN to compute ω.

for every ϕ:

8. Select a mini batch of size β from $\sigma 1$.
9. Forward propagation and compute loss (binary cross-entropy) via θ.
10. Back propagation only on the head layers, update ω with Adam optimizer.

end

11. Upload static input image and convert it into a 4-D Binary Large OBject (BLOB).
12. Initialize α (confidence).
13. Set the Caffe framework for obtaining face detections.
14. Pass BLOB image through the network to obtain face detections on the input image.

for every δ :

15. Extract probability (confidence) associated with the detection.
16. Filter weak detections by comparison with α
17. Extract face ROI in the input image, add bounding box and labels on face.

end

18. Apply the trained model to the image only if a face is detected.

End.

IV. Evaluation

This section describes the evaluation parameters and the outcomes of the proposed solution. The experimental results are further analyzed and visualized using the performance evaluation metrics.

A. Performance Parameters

The description of various evaluation parameters is given as follows:

- **Confusion Matrix**: A confusion matrix is an n-way matrix where the n is the number of classes for the classification purpose. It is based on four important parameters: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). The following parameters are calculated using the confusion matrix:

  Accuracy: It is calculated by using (5).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{5}$$

  Precision: It is another evaluation metric that tells how many predictions are actually correct out of all the correct predictions. It is given by (6).

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

  Recall: defined as the number of positive predictions made by the model out of the total actual positive classes. It is calculated by (7).

$$Recall = \frac{TP}{TP+FN} \tag{7}$$

  F1-score: It is calculated by using precision and recall as shown in (8).

$$F1-score = \frac{2}{\frac{1}{Precision}+\frac{1}{Recall}} \tag{8}$$

  Sensitivity: It measures how correctly we have detected the positive classes. It is calculated by using (9).

$$Sensitivity = \frac{TP}{TP+FN} \tag{9}$$

  Specificity: It measures how exact or accurate is the assignment to the positive class. It is calculated by using (10).

$$Specificity = \frac{TN}{FP+TN} \tag{10}$$

- **Learning Curves**: A learning curve is a plot for the graphical visualization of the model performance while it is training on the dataset. It is a plot of the accuracy/loss versus the number of epochs. It shows how the accuracy/loss of the model changes during its training phase with the increase in the number of epochs.
- **Receiver Operating Characteristics**: The ROC represents a plot between the true positive rate (TPR) and false negative rate (FNR), and is a trade-off between the specificity and sensitivity [31]. AUC uses the ROC curve and is calculated by using the trapezoidal method, i.e., dividing the area into a number of sections with equal width. Here, the trapezoid (T) refers to integration of points (a, b) from a functional form which is divided into n equal pieces. The addition of the area of each section of the trapezium formed when the upper end is replaced by a chord and the sum of these approximations provides the final AUC value. The trapezoidal formula is indicated as an integral of the function $\int_a^b f(x)\,dx$, and the points of integration (a, b) are labelled as $\{x_0, x_1, ..., x_n\}$; where $\{x_0 = a, x_n = b, x_r = x_0 + r(b - a)/n\}$ as given in [32].

B. Evaluation Results

In our evaluation, we noted the model performance in terms of accuracy, precision, recall, F-score, sensitivity, specificity, and area under curve (AUC) of receiver operating characteristics (ROC). The training history of the model is also plotted for a credible analysis of the loss and accuracy of the training set and the validation set. Confusion matrix is calculated on the test set as shown in Table II.

TABLE II. Confusion Matrix Obtained on Test Data

		Predicted Classes	
		Proper	Improper
Actual Classes	Proper	TP = 47	FN = 3
	Improper	FP = 4	TN = 55

By calculating the overall accuracy of the proposed model by using the confusion matrix on the testing data, we have attained an overall accuracy of 93.58%. We have achieved a precision and recall of 92.15% and 94% respectively. Further, we have achieved F-1 score of 93.10. A sensitivity of 94% has been calculated. This means that out of 50 total people who are wearing the masks properly, 47 have been detected wearing the mask properly from the test set.

We have achieved a specificity of 93.22%. This implies that out of 59 people who are wearing the masks improperly, we are able to correctly predict the person wearing an improper mask with an error rate of only 6.78%. These evaluation parameters are calculated from the confusion matrix as indicated in Table III.

TABLE III. Summary Table of Computed Metrics Using the Confusion Matrix

	Evaluation Metric	Value (in %)
1.	Accuracy	93.58
2.	Precision	92.15
3.	Recall	94.00
4.	F-1 score	93.10
5.	Sensitivity	94.00
6.	Specificity	93.22

The model has been trained for 50 epochs, with an initial learning rate of 0.0001. After the proposed model is trained on the training set data, we observed the training accuracy as 92.27% and the validation accuracy is 93.58% as shown in Fig. 6. The validation set data is used to provide an unbiased evaluation and tune the hyper-parameters of the model, while the model is fit on the training set data. It further helps in determining the error rate in the model by holding out a subset of the data from the fitting process and evaluation of the loss of the model at the end of each epoch. The high training accuracy can be considered as a good measure to assess our classification model. Fig. 7 depicts learning curves which gives the values of training and validation loss as 0.1693 and 0.1595, respectively.

Fig. 6. Learning curves for evaluation of Accuracy.

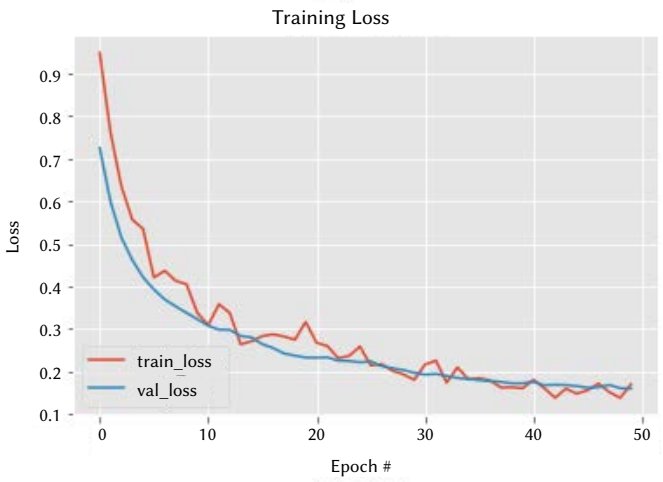

Fig. 7. Learning curves for evaluation of Loss.

Table IV depicts the relevant results obtained from the training curves to determine and compare the loss and accuracy of the training and validation data, respectively. Fig. 8 shows the plotted AUC of ROC curve of the proposed model which is above the threshold level and is calculated as 0.9361.

TABLE IV. Inference of Accuracy and Loss From Obtained Curves and Their Training Vs Validation Comparison

	Accuracy/Loss	Value (in %)
1.	Training Accuracy	92.27
2.	Validation Accuracy	93.58
3.	Training Loss	0.1693
4.	Validation Loss	0.1595

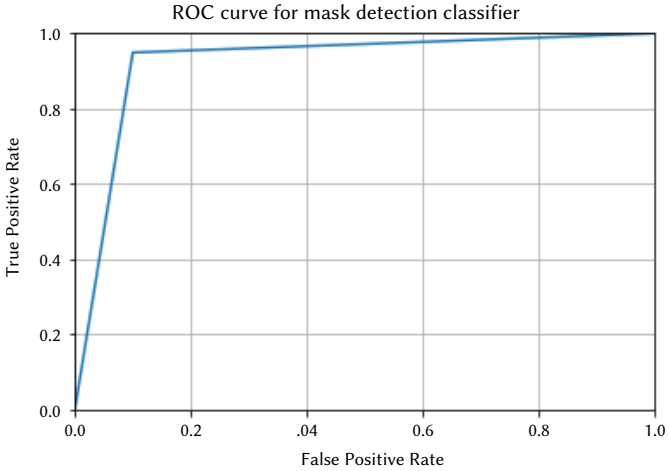

Fig. 8. ROC curve for test data.

The images that were not used in training are provided as input to the proposed model to predict whether they are wearing the mask properly. Fig. 9 depicts that the proposed model is able to categorize faces into two classes: proper and improper, with high confidence scores. Clockwise from top left, confidence scores are 99.06% (Improper), 96.23% (Proper), 59.90% (Improper), 74.18% (Proper), and 99.81% (Proper).

We have not compared our results to any existing work since these do not focus on whether the person is wearing a face mask properly or not.

Fig. 9. A sample of results showing successful classification achieved by the model on input images.

V. Conclusion

The COVID-19 disease is the greatest challenge that the world has faced since World War II. To prevent its rapid spread, face masks must be worn properly by people over their noses. People at crowded places, hospitals, offices, and working spaces can be checked for improperly worn masks to ensure safety. Application of the proposed model can serve as a preventive measure in the COVID- 19 crisis and benefit in safekeeping the health of society. The government can also leverage the model to mobile devices or any device with low computational power for the detection of improperly worn face masks at public places.

Some research work had been done in detecting masks worn and not worn. However, our model specifically focused on classifying the mask worn by a person into two classes: proper and improper. This will be much significant in the various stages of unlocking all over the world as it will contribute to public safety and healthcare. The architecture of the model consists of the light weighted MobileNetV2 neural network as the backbone which overcomes computational issues as it can be used efficiently on devices with low computational power. Transfer learning has been adopted to use weights that have been used for a similar task like face detection and already trained on a very large dataset. Furthermore, OpenCV with the Caffe framework has been used to detect facial features on experimental input images and used on the pre-trained model with our dataset, to produce classification results with indicative results, such as labels and a bounding box. We are able to attain a testing accuracy of 93.58% and an AUC measure of 0.936.

Appendix

The appendix summarizes the complete structure of the proposed model. It includes information about the layers and their order in the model, the output shape of each layer and the information about the parameters (weights). The number of parameters in each layer and the total number of parameters in the model are obtained by using the model summary. A total of 2,422,210 parameters (164,226 trainable parameters and 2,257,984 non-trainable parameters) is present in our model.

Layer(type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)	0
Conv1_pad (ZeroPadding2D)	(None, 225, 225, 3)	0
Conv1 (Conv2D)	(None, 112, 112, 32)	864
bn_Conv1 (BatchNormalization)	(None, 112, 112, 32)	128
Conv1_relu (ReLU)	(None, 112, 112, 32)	0
expanded_conv_depthwise (DepthwiseConvolution)	(None, 112, 112, 32)	288
expanded_conv_depthwise_BN (BatchNormalization)	(None, 112, 112, 32)	128
expanded_conv_depthwise_relu (ReLU)	(None, 112, 112, 32)	0
expanded_conv_project (Conv2D)	(None, 112, 112, 16)	512
expanded_conv_project_BN (BatchNormalization)	(None, 112, 112, 16)	64
block_1_expand (Conv2D)	(None, 112, 112, 96)	1536
block_1_expand_BN (BatchNormalization)	(None, 112, 112, 96)	384
block_1_expand_relu (ReLU)	(None, 112, 112, 96)	0
block_1_pad (ZeroPadding2D)	(None, 113, 113, 96)	0
block_1_depthwise (DepthwiseConvolution)	(None, 56, 56, 96)	864
block_1_depthwise_BN (BatchNorm	(None, 56, 56, 96)	384
block_1_depthwise_relu (ReLU)	(None, 56, 56, 96)	0
block_1_project (Conv2D)	(None, 56, 56, 24)	2304
block_1_project_BN (BatchNormalization)	(None, 56, 56, 24)	96
block_2_expand (Conv2D)	(None, 56, 56, 144)	3456
block_2_expand_BN (BatchNormalization)	(None, 56, 56, 144)	576
block_2_expand_relu (ReLU)	(None, 56, 56, 144)	0
block_2_depthwise (DepthwiseConvolution)	(None, 56, 56, 144)	3456
block_2_depthwise_BN (BatchNormalization)	(None, 56, 56, 144)	96
block_2_depthwise_relu (ReLU)	(None, 56, 56, 144)	0
block_2_project (Conv2D)	(None, 56, 56, 24)	3456
block_2_project_BN (BatchNormalization)	(None, 56, 56, 24)	576
block_2_add (Add)	(None, 56, 56, 24)	0
block_3_expand	(ne, 56, 56, 144)	0
block_3_expand_BN (BatchNormalization)	(None, 56, 56, 144)	1296
block_3_expand_relu (ReLU)	(None, 56, 56, 144)	576
block_3_pad (ZeroPadding2D)	(None, 57, 57, 144)	0
block_3_depthwise (DepthwiseConvolution)	(None, 28, 28, 144)	4608
block_3_depthwise_BN (BatchNormalization)	(None, 28, 28, 144)	128
block_3_depthwise_relu (ReLU)	(None, 28, 28, 144)	6144
block_3_project (Conv2D)	(None, 28, 28, 32)	768
block_3_project_BN (BatchNormalization)	(None, 28, 28, 32)	0
block_4_expand (Conv2D)	(None, 28, 28, 192)	1728
block_4_expand_BN (BatchNormalization)	(None, 28, 28, 192)	768
block_4_expand_relu (ReLU)	(None, 28, 28, 192)	0
block_4_depthwise (DepthwiseConvolution)	(None, 28, 28, 192)	6144
block_4_depthwise_BN (BatchNormalization)	(None, 28, 28, 192)	768
block_4_depthwise_relu (ReLU)	(None, 28, 28, 192)	0
block_4_project (Conv2D)	(None, 28, 28, 32)	1728
block_4_project_BN (BatchNormalization)	(None, 28, 28, 32)	768
block_4_add (Add)	(None, 28, 28, 32)	0
block_5_expand (Conv2D)	(None, 28, 28, 192)	6144
block_5_expand_BN (BatchNormalization)	(None, 28, 28, 192)	128
block_5_expand_relu (ReLU)	(None, 28, 28, 192)	0
block_5_depthwise (DepthwiseConvolution)	(None, 28, 28, 192)	6144
block_5_depthwise_BN (BatchNormalization)	(None, 28, 28, 192)	128

Layer(type)	Output Shape	Param #
block_5_depthwise_relu (ReLU)	(None, 28, 28, 192)	0
block_5_project (Conv2D)	(None, 28, 28, 32)	6144
block_5_project_BN (BatchNormalization)	(None, 28, 28, 32)	768
block_5_add (Add)	(None, 28, 28, 32)	0
block_6_expand (Conv2D)	(None, 28, 28, 192)	0
block_6_expand_BN (BatchNormalization)	(None, 28, 28, 192)	1728
block_6_expand_relu (ReLU)	(None, 28, 28, 192)	768
block_6_pad (ZeroPadding2D)	(None, 29, 29, 192)	0
block_6_depthwise (DepthwiseConvolution)	(None, 14, 14, 192)	12288
block_6_depthwise_BN (BatchNormalization)	(None, 14, 14, 192)	256
block_6_depthwise_relu (ReLU)	(None, 14, 14, 192)	24576
block_6_project (Conv2D)	(None, 14, 14, 64)	1536
block_6_project_BN (BatchNormalization)	(None, 14, 14, 64)	0
block_7_expand (Conv2D)	(None, 14, 14, 384)	24576
block_7_expand_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_7_expand_relu (ReLU)	(None, 14, 14, 384)	0
block_7_depthwise (DepthwiseConvolution)	(None, 14, 14, 384)	3456
block_7_depthwise_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_7_depthwise_relu (ReLU)	(None, 14, 14, 384)	0
block_7_project (Conv2D)	(None, 14, 14, 64)	24576
block_7_project_BN (BatchNormalization)	(None, 14, 14, 64)	256
block_7_add (Add)	(None, 14, 14, 64)	0
block_8_expand (Conv2D)	(None, 14, 14, 384)	24536
block_8_expand_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_8_expand_relu (ReLU)	(None, 14, 14, 384)	0
block_8_depthwise (DepthwiseConvolution)	(None, 14, 14, 384)	3456
block_8_depthwise_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_8_depthwise_relu (ReLU)	(None, 14, 14, 384)	0
block_8_project (Conv2D)	(None, 14, 14, 64)	24576
block_8_project_BN (BatchNormalization)	(None, 14, 14, 64)	256
block_8_add (Add)	(None, 14, 14, 64)	0
block_9_expand (Conv2D)	(None, 14, 14, 384)	24576
block_9_expand_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_9_expand_relu (ReLU)	(None, 14, 14, 384)	0
block_9_depthwise (DepthwiseConvolution)	(None, 14, 14, 384)	3456
block_9_depthwise_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_9_depthwise_relu (ReLU)	(None, 14, 14, 384)	0
block_9_project (Conv2D)	(None, 14, 14, 64)	24576
block_9_project_BN (BatchNormalization)	(None, 14, 14, 64)	256
block_9_add (Add)	(None, 14, 14, 64)	0
block_10_expand (Conv2D)	(None, 14, 14, 384)	24576
block_10_expand_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_10_expand_relu (ReLU)	(None, 14, 14, 384)	0
block_5_expand_BN (BatchNormalization)	(None, 28, 28, 192)	128
block_5_expand_relu (ReLU)	(None, 28, 28, 192)	0
block_5_depthwise (DepthwiseConvolution)	(None, 28, 28, 192)	6144
block_5_depthwise_BN (BatchNormalization)	(None, 28, 28, 192)	128
block_5_depthwise_relu (ReLU)	(None, 28, 28, 192)	0
block_5_project (Conv2D)	(None, 28, 28, 32)	6144
block_5_project_BN (BatchNormalization)	(None, 28, 28, 32)	768
block_5_add (Add)	(None, 28, 28, 32)	0

Layer(type)	Output Shape	Param #
block_6_expand (Conv2D)	(None, 28, 28, 192)	0
block_6_expand_BN (BatchNormalization)	(None, 28, 28, 192)	1728
block_6_expand_relu (ReLU)	(None, 28, 28, 192)	768
block_6_pad (ZeroPadding2D)	(None, 29, 29, 192)	0
block_6_depthwise (DepthwiseConvolution)	(None, 14, 14, 192)	12288
block_6_depthwise_BN (BatchNormalization)	(None, 14, 14, 192)	256
block_6_depthwise_relu (ReLU)	(None, 14, 14, 192)	24576
block_6_project (Conv2D)	(None, 14, 14, 64)	1536
block_6_project_BN (BatchNormalization)	(None, 14, 14, 64)	0
block_7_expand (Conv2D)	(None, 14, 14, 384)	24576
block_7_expand_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_7_expand_relu (ReLU)	(None, 14, 14, 384)	0
block_7_depthwise (DepthwiseConvolution)	(None, 14, 14, 384)	3456
block_7_depthwise_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_7_depthwise_relu (ReLU)	(None, 14, 14, 384)	0
block_7_project (Conv2D)	(None, 14, 14, 64)	24576
block_7_project_BN (BatchNormalization)	(None, 14, 14, 64)	256
block_7_add (Add)	(None, 14, 14, 64)	0
block_8_expand (Conv2D)	(None, 14, 14, 384)	24536
block_8_expand_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_8_expand_relu (ReLU)	(None, 14, 14, 384)	0
block_8_depthwise (DepthwiseConvolution)	(None, 14, 14, 384)	3456
block_8_depthwise_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_8_depthwise_relu (ReLU)	(None, 14, 14, 384)	0
block_8_project (Conv2D)	(None, 14, 14, 64)	24576
block_8_project_BN (BatchNormalization)	(None, 14, 14, 64)	256
block_8_add (Add)	(None, 14, 14, 64)	0
block_9_expand (Conv2D)	(None, 14, 14, 384)	24576
block_9_expand_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_9_expand_relu (ReLU)	(None, 14, 14, 384)	0
block_9_depthwise (DepthwiseConvolution)	(None, 14, 14, 384)	3456
block_9_depthwise_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_9_depthwise_relu (ReLU)	(None, 14, 14, 384)	0
block_9_project (Conv2D)	(None, 14, 14, 64)	24576
block_9_project_BN (BatchNormalization)	(None, 14, 14, 64)	256
block_9_add (Add)	(None, 14, 14, 64)	0
block_10_expand (Conv2D)	(None, 14, 14, 384)	24576
block_10_expand_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_10_expand_relu (ReLU)	(None, 14, 14, 384)	0
block_15_project (Conv2D)	(None, 7, 7, 160)	153600
block_15_project_BN (BatchNormalization)	(None, 7, 7, 160)	640
block_15_add (Add)	(None, 7, 7, 160)	0
block_16_expand (Conv2D)	(None, 7, 7, 960)	153600
block_16_expand_BN (BatchNormalization)	(None, 7, 7, 960)	3840
block_16_expand_relu (ReLU)	(None, 7, 7, 960)	0
block_16_depthwise (DepthwiseConvolution)	(None, 7, 7, 960)	8640
block_16_depthwise_BN (BatchNormalization)	(None, 7, 7, 960)	3840
block_16_depthwise_relu (ReLU)	(None, 7, 7, 960)	0
block_16_project (Conv2D)	(None, 7, 7, 320)	307200
block_16_project_BN(BatchNormalization)	(None, 7, 7, 320)	1280
Conv_1 (Conv2D)	(None, 7, 7, 1280)	409600

Layer(type)	Output Shape	Param #
Conv_1_bn (BatchNormalization)	(None, 7, 7, 1280)	5120
out_relu (ReLU)	(None, 7, 7, 1280)	0
average_pooling2d_2 (AveragePooling)	(None, 1, 1, 1280)	0
flatten (Flatten)	(None, 1280)	0
dense_4 (Dense)	(None, 128)	163968
dropout_2 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 2)	258

Total parameters: 2,422,210
Trainable parameters: 164,226
Non-trainable parameters: 2,257,984

References

[1] V. C.-Ch. Cheng *et al.*, "The role of community-wide wearing of face mask for control of coronavirus disease 2019 (COVID-19) epidemic due to SARS-CoV-2," *Journal of Infection*, vol. 81, no. 1, pp. 107–114, 2020, doi: 10.1016/j.jinf.2020.04.024.

[2] A. N. Desai and D. M. Aronoff, "Masks and Coronavirus Disease 2019 (COVID-19)," *JAMA*, vol. 323, no. 20, p. 2103, May 2020, doi: 10.1001/jama.2020.6437.

[3] C. Kenyon, "The prominence of asymptomatic superspreaders in transmission mean universal face masking should be part of COVID-19 de-escalation strategies," *International Journal of Infectious Diseases*, vol. 97, pp. 21–22, 2020.

[4] N. H. L. Leung *et al.*, "Respiratory virus shedding in exhaled breath and efficacy of face masks," *Nature Medicine*, vol. 26, no. 5, pp. 676–680, 2020, doi: 10.1038/s41591-020-0843-2.

[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*, vol. 27, no. 2, Feb. 2020, doi: 10.1093/jtm/taaa021.

[6] M. P. Fang, Yaqing, Yiting Nie, "Transmission dynamics of the COVID-19 outbreak and effectiveness of government interventions: A data-driven analysis," *Journal of medical virology*, vol. 92, no. 6, pp. 645–659, 2020, doi: 10.1002/jmv.25750.

[7] E. S. K. Lim, Soo, Ho Il Yoon, Kyoung-Ho Song and H. Bin Kim, "Face Masks and Containment of Coronavirus Disease 2019 (COVID-19): Experience from South Korea," *The Journal of Hospital Infection*, 2020, doi: 10.1016/j.jhin.2020.06.017.

[8] S. Esposito, N. Principi, C. C. Leung, and G. B. Migliori, "Universal use of face masks for success against COVID-19: evidence and implications for prevention policies," *European Respiratory Journal*, p. 2001260, Jan. 2020, doi: 10.1183/13993003.01260-2020.

[9] M. Dhalaria and E. Gandotra, "Convolutional Neural Network for Classification of Android Applications Represented as Grayscale Images," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 12S, pp. 2278–3075, 2019.

[10] H. Panwar, P. K. Gupta, M. K. Siddiqui, R. Morales-Menendez, and V. Singh, "Application of deep learning for fast detection of COVID-19 in X-Rays using nCOVnet," *Chaos, Solitons & Fractals*, vol. 138, p. 109944, 2020, doi: 10.1016/j.chaos.2020.109944.

[11] F. Sultana, A. Sufian, and P. Dutta, "Advancements in Image Classification using Convolutional Neural Network," in *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 2018, pp. 122–129, doi: 10.1109/ICRCICN.2018.8718718.

[12] H. Shin *et al.*, "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016, doi: 10.1109/TMI.2016.2528162

[13] M. Coşkun, A. Uçar, Ö. Yildirim, and Y. Demir, "Face recognition based on convolutional neural network," in *2017 International Conference on Modern Electrical and Energy Systems (MEES)*, 2017, pp. 376–379, doi: 10.1109/MEES.2017.8248937.

[14] Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[15] Wang *et al.*, "Residual attention network for image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3156–3164.

[16] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object Detection With Deep Learning: A Review," *IEEE Transaction on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019, doi: 10.1109/TNNLS.2018.2876865.

[17] Redmon, Joseph, S. Divvala, R. Girshik, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[18] A. Kumar, A. Kaur, and M. Kumar, "Face detection techniques: a review," *Artificial Intelligence Review, Springer*, vol. 52, no. 2, pp. 927–948, 2019, doi: 10.1007/s10462-018-9650-2.

[19] A. S. M. and V. M. M. Sivaram, V. Porkodi, "Detection of Accurate Facial Detection using Hybrid Deep Convolutional Recurrent Neural Network," *ICTACT Journal of Soft Computing*, vol. 9, no. 2, 2019, doi: 10.21917/ijsc.2019.0256.

[20] M. Khan, S. Chakraborty, R. Astya, and S. Khepra, "Face Detection and Recognition Using OpenCV," in *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2019, pp. 116–119, doi: 10.1109/ICCCIS48478.2019.8974493.

[21] T. Meenpal, A. Balakrishnan, and A. Verma, "Facial Mask Detection using Semantic Segmentation," in *2019 4th International Conference on Computing, Communications and Security (ICCCS)*, 2019, pp. 1–5, doi: 10.1109/CCCS.2019.8888092.

[22] M. Jiang, X. Fan, and H. Yan, "RetinaMask: A Face Mask detector." 2020, [Online]. Available: http://arxiv.org/abs/2005.03950.

[23] T. Zhang, J. Li, W. Jia, J. Sun, and H. Yang, "Fast and robust occluded face detection in ATM surveillance," *Pattern Recognition Letters*, vol. 107, pp. 33–40, 2018, doi: 10.1016/j.patrec.2017.09.011.

[24] H. Baojin, "Real-World Masked Face Dataset, RMFD." Real-World Masked Face Dataset, RMFD.

[25] "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, [Online]. Available: https://arxiv.org/abs/1704.04861.

[26] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[28] "How to train a Face Detector," https://github.com/opencv/opencv/blob/4.0.0-beta/samples/dnn/face_detector/how_to_train_face_detector.txt.

[29] "deploy.prototxt," https://github.com/opencv/opencv/blob/4.0.0-beta/samples/dnn/face_detector/deploy.prototxt.

[30] Y. Jia *et al.*, "Caffe: Convolutional Architecture for Fast Feature Embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*, 2014, pp. 675–678, doi: 10.1145/2647868.2654889.

[31] D. L. Streiner and J. Cairney, "What's under the ROC? An Introduction to Receiver Operating Characteristics Curves," *The Canadian Journal of Psychiatry*, vol. 52, no. 2, pp. 121–128, Feb. 2007, doi: 10.1177/070674370705200210.

[32] Yeh and Shi-Tao, "Using trapezoidal rule for the area under a curve calculation," *Proceedings of the 27th Annual SAS User Group International* 2002.

Anubha Bhaik

Anubha Bhaik has recently completed her B.Tech in Computer Science & Engineering from Jaypee University of Information Technology, India. She will be pursuing her Master's in Computer Science at the University of Florida, USA. Her interests lie in deep learning, computer vision and data science. She is an experienced team lead with a demonstrated record of contributing to the research industry.

Vaishnavi Singh

Vaishnavi Singh has recently completed her B.Tech in Computer Science & Engineering from Jaypee University of Information Technology, India. She enjoys figuring out the different building blocks of the technical world and rearranging them to discover new possibilities. She is passinate about exploring and applying various fields of computer science to develop adept solutions for the real-world problems. Her interests lie in deep learning, computer vision, software engineering and database systems. She has a significant history of contributions to the research industry, whereas majority of her work is related to solving the challenges of COVID-19.

Ekta Gandotra

Ekta Gandotra is currently working as Assistant Professor in the Department of Computer Science & Engineering at Jaypee University of Information Technology, Waknaghat (India). She has completed her Ph.D. in Computer Science and Engineering from PEC University of Technology, Chandigarh (India). She has around 12 years of teaching and research experience. Her research areas include network & cybersecurity, malware threat profiling, cyber threat intelligence, machine/deep learning, and big data analytics.

Deepak Gupta

Deepak Gupta is working as Assistant Professor in the Department of Computer Science & Engineering at Jaypee University of Information Technology, Waknaghat (India). He has completed his Ph.D. in Computer Science & Engineering from Thapar Institute of Engineering and Technology (Deemed to be University), Patiala (India). Prior to his foray into academia, he worked in IT industry for a decade performing different roles in software product development and program management. In all, he has more than 20 years of rich experience in IT industry and academics. His research interests include big data analytics, machine/deep learning, cybersecurity, and programming languages.