JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

# Machine learning-based mask detection system

**Merugu sushmitha1, M.Shyamsunder2**
[1] Student of ECE Department, University college of engineering, Osmania University,
[2] Assistant Professor, ECE Department, University college of engineering, Osmania University,

*Abstract:* Covid-19 has recently had a cascade effect on industry, travel, tourism, and hospitality, devastating the world economy. This effort, which relies on computer vision and deep learning, aims to have an influence and significantly tackle the real-world problem of safety measures in addition to the expanding curve of human mortality caused by the epidemic around the world. Through the use of OpenCV, Keras/TensorFlow, and Deep Learning, a Face Mask Detection system is being developed in this work. The system's use of the MobileNetV2 architecture makes it simple to integrate into a variety of embedded devices with low processing capability. Face masks will be picked up in both real-time videos and photos.

## 1. Introduction

The World Health Organization (WHO) acknowledged that the new Corona virus illness (COVID-19) case, which emerged quickly in Wuhan, China, in December 2019, is a hazardous virus that can spread from human to human by droplets and airborne. Regarding preventive, it's imperative to use a face mask when going outside or interacting with others. However, some careless individuals have so many justifications for refusing to use face masks. Additionally, the development of the face mask detector is essential in this scenario [1-3].

This study intends to create a face mask detector that can recognise many types of face masks [4-5]. The face mask detection technique has been developed using OpenCV, Keras, TensorFlow, and Convolutional Neural Network (CNN) [6–8].

The experimental results have been done in The device and real-time application can be deployed in public spaces where wearing a mask is required during this epidemic. According to the experimental findings, this device can accurately identify individuals who are wearing or are not wearing a face mask, even if they are moving to different locations [4–7]. The work's aims are

- To develop a specific deep learning model for determining whether or not someone is wearing a mask.
- To create a unique dataset of with mask photos using the RMFD dataset, a few existing Kaggle datasets, and the Bing Search API..
- Datasets will be split into the "with mask" and "without mask" classes.
- To use Keras and TensorFlow to train the face mask detector on the unique dataset.
- To use the trained model to identify masks in both real-time videos and photos.
- The model should have accuracy of at least 90%..

In the present scenario due to Covid-19, there need for efficient face mask detection applications, which are now in high demand for transportation means, densely populated areas, residential districts, large-scale manufacturers and other enterprises to ensure that the safety guidelines are strictly followed. Also, the absence of large datasets of '*with_mask*' images has made this task more cumbersome and challenging. Therefore, the need of the hour is to generate a huge custom dataset of 'with_mask' images with the help of existing datasets and search APIs followed by developing a face mask detection system

This system is need of the hour as India tries to battle the novel corona virus that hasinfected more than 2 crore and has caused more thean lakhs deaths with the figures still increasing at a rapid face.

## 2. Requirement Specification And Analysis

Functional Requirements of Face Mask Dataset

1. The 'with mask' dataset used by the system must be impartial.
2. The dataset must have over 1500+ images in both 'with_mask' and 'without_mask' classes.
3. The dataset must not re-use the same images in training and testing phases.

Functional Requirements of Face Mask Detector

1. The face mask classifier model must be correctly loaded by the system.
2. Face detection in pictures or video streams is a need for the system.
3. The Region of Interest (ROI) for each face must be able to be extracted by the system.

4.For successful face detection, and hence successful face mask detection, there must be no item in the way of the system and the user's face.
5. The face's final location must fit within the webcam's frame and be closer to the camera.
6. Correctly able to recognise masks in images in the "png," "jpg," "jpeg," and "gif" formats.
7. In every frame of a live video, the system must be able to recognise face masks on human faces.
8. The output of "Mask" or "No Mask" must be displayed with the probability in order to see the findings.

## 3. Fundamental Steps to Perform for Face Mask Detection

The data will be collected from the below mentioned sources:

- Real-World Masked Face Dataset (RMFD).
- Kaggle Datasets.
- Bing Search API.

i. The aim is to collect more than 1800+ images in both “with_mask” and “without_mask” classes.

ii. For the Bing Search API, a Python script will be used to find photos while searching for terms like "covid mask," "facial mask," and "N95 mask."

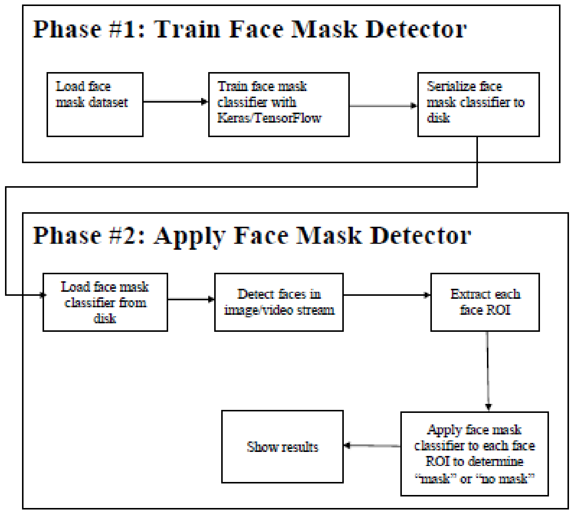

Phase #1: Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifier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

Fig.1 Phases and individual steps for building a COVID-19 face mask detector with computer vision and deep learning using Python, OpenCV, TensorFlow, Keras.

## 4. Algorithms

### (i) Feature Extraction using ConvNets

Feature extraction is used in traditional machine learning methods for image By using global feature descriptors like Local Binary Patterns (LBP), Histogram of Oriented Gradients (HoG), Color Histograms, etc. or local feature descriptors like SIFT, SURF, ORB, etc., the traditional machine learning approach extracts features from images. These are custom features that call for domain-specific knowledge. Convolutional neural networks, however, are here (CNN). Deep Neural Nets automatically learn these features from photos in a hierarchical manner as opposed to using manually created features. While middle levels teach colour, shape, and other low-level features, higher layers teach high-level attributes that represent the image object. Lower layers teach low-level features like corners and edges. By using the activations accessible prior to the last fully connected layer of the network, we can use a CNN as a Feature Extractor rather than creating one as a model to categorise photos (i.e. before the final softmax classifier). These activations will serve as the feature vector for a classifier that uses machine learning to further refine its classification abilities. A pre-trained CNN could be utilised as a Feature Extractor - Transfer Learning in place of training a CNN from scratch, which is time-consuming and laborious, for Image Classification tasks.

### (ii) Transfer Learning Algorithm

A model created for one task is used as the basis for another using the machine learning technique known as transfer learning. The concept is to get past the isolated learning paradigm and use the knowledge you learn to solve one problem to others that are related.
Traditional learning is solitary and relies only on a limited number of tasks, datasets, and the training of distinct, isolated models. No information that can be applied to another model is maintained. With transfer learning, you may train newer models using knowledge (features, weights, etc.) from older models that have already been trained, even if the new task requires less data!
Transfer learning is essentially the process of training and making predictions on a new dataset using a model that has already been trained on one dataset. Learning is a difficult process for both people and machines. It was vital to develop a method that would avoid a

model from forgetting the learning curve that it obtained from a given dataset and also enable it learn more from fresh and varied datasets because it is a laborious, resource-intensive, and time-consuming operation.
"A pre-trained model is a saved network that was previously trained on a large dataset, typically on a large-scale image-classification task such as the ImageNet."

## 5. Methodologies

**OpenCV's "deep neural networks" (dnn) module:** OpenCV (3.3 or later) comprises of the highly efficient dnn module supported by a number of deep learning frameworks such as Caffe, TensorFlow, and Torch/PyTorch. This module has a more accurate Caffe-based face detector. In this project, we will be training our deep learning model using Caffe, hence we need the following files:
a) The **.**prototxt file(s) which will define the *model architecture* (i.e., the layers)
b) The .caffemodel file which will contain the *weights* for the actual layers

## 6. Detection of Masks in Real-Time Video Streams using Webcam.

**Step 1: Training the model with Keras and TensorFlow.**
We will use Keras, TensorFlow, and deep learning to train our face mask detector model in this stage. The following file constructs our unique mask detector model using our input dataset.
The learning hyperparameter constants are declared including:

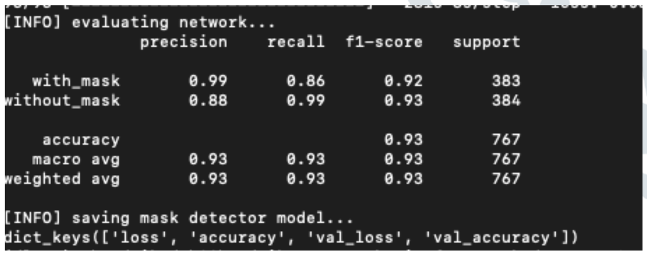
- Initial learning rate (INIT_LR)
- Number of training epochs (EPOCHS)
- Batch Size (BS)

- Class label encoding in a single pass: Categorical variables are essentially represented as binary vectors in one-hot encoding. First, integer values are assigned to these categorical values. Then, each integer value is represented as an all-zero binary vector (except the index of the integer which is marked as 1).

- • Dataset partitioning: We divide our dataset into 80% for training and the remaining 20% for testing using scikit-train test split() learn's method.
- Dataset partitioning: Using scikit-train test split() learn's approach, we divide our dataset into 80% for training and the remaining 20% for testing.

- The Adam optimizer, a time-based learning rate scheduler employing the "decay" parameter, and "binary cross-entropy" are used to create the model (because 2 classes classifier).
- After that, we train the network's head using batches of augmented data provided by the "aug" object.

**Step 2: Execute the following command in Terminal after going into the project folder to train the model.**
*$ python train_mask_detector.py --dataset dataset*



```
[INFO] evaluating network...
              precision    recall  f1-score   support

   with_mask       0.99      0.86      0.92       383
without_mask       0.88      0.99      0.93       384

    accuracy                           0.93       767
   macro avg       0.93      0.93      0.93       767
weighted avg       0.93      0.93      0.93       767

[INFO] saving mask detector model...
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

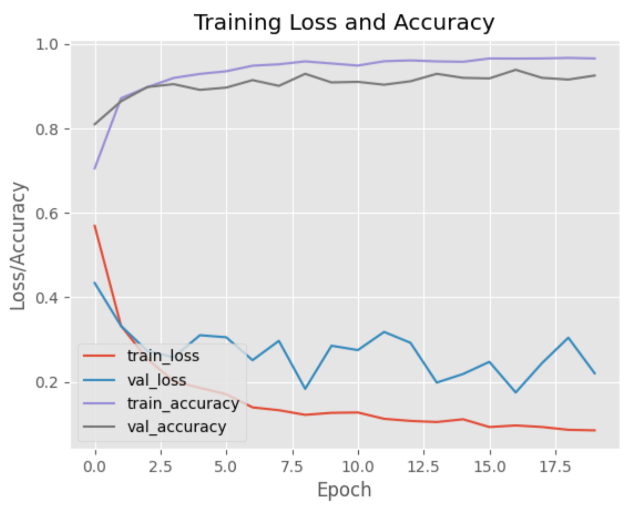


Fig 2: Loss/accuracy versus epochs

We now receive a separate file called "plot.png" that contains the loss/accuracy curves from our training history. We can infer that the curves exhibit great accuracy and little overfitting symptoms.
We can now use our trained model to accurately identify face masks in still photos and live video streams outside of our input dataset.
**Step 3: Implementing our COVID-19 face mask detector model for static images with OpenCV**
The following file loads our model and hence detects faces and classifies them as **'Mask'** or **'No Mask'** in static images given as an input.
File: detect_mask_image.py
we require the TensorFlow/Keras imports for loading the model and pre-processing the input image. We import OpenCV for image display and modifications.

We make a copy of the input image and grab the frame dimensions for scaling and display.

- Image pre-processing is done by OpenCV's 'blobFromImage' method. We resize the image to 300px*300px and mean subtraction is performed.
- Then we perform face detection to locate all the faces in the input image provided.

**Step 4: Execute the following command in Terminal to detect masks in static images present in 'images' folder.**
*$ python detect_mask_image.py --image images/10.jpeg*

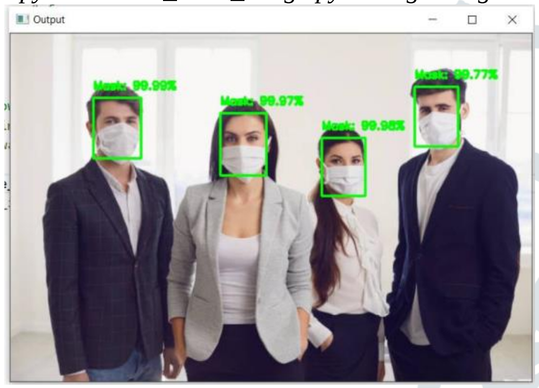


Fig3: Output file showing with mask
*$ python detect_mask_image.py --image images/4.jpg*

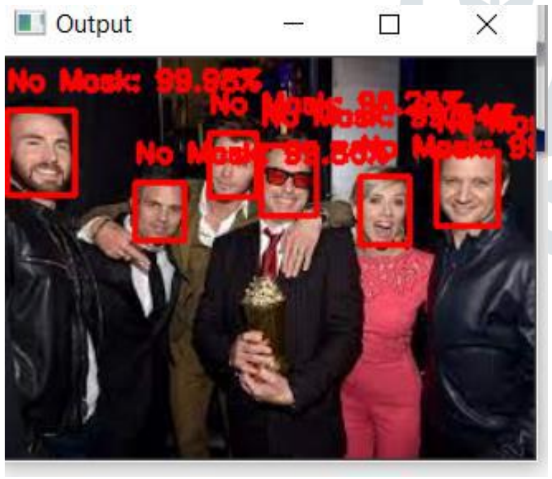


Fig4:Output file showing without mask
**Step 5: Implementing our COVID-19 face mask detector model in real-time video streams with OpenCV.**
The following file loads our model and hence detects faces and classifies them as **'Mask'** or **'No Mask'** in real-time video streams.

File: detect_mask_video.py

- So, we import the VideoStream class along with the time module.
- The imutils module helps for its aspect aware resizing method.
- The detect_and_predict_mask function detects faces and then applies the face mask classifier to each face ROI.
- This function accepts three parameters:
- *frame*: A frame from our stream
- *faceNet*: The model used to detect where in the image faces are
- *maskNet*: Our COVID-19 face mask classifier model

- We then construct a blob, detect faces and initialize the lists including faces (i.e., ROIs), locs (the face locations), and preds (the list of mask/no mask predictions).
- We perform the inference on entire batch of faces in the frame so that our pipeline is faster.
- At last, we return the face bounding box locations and corresponding mask/non-mask predictions to the caller function.

## 7. Results and Conclusion

**(i)** Detection of masks in real-time video streams using webcam.

*$ python detect_mask_video.py*

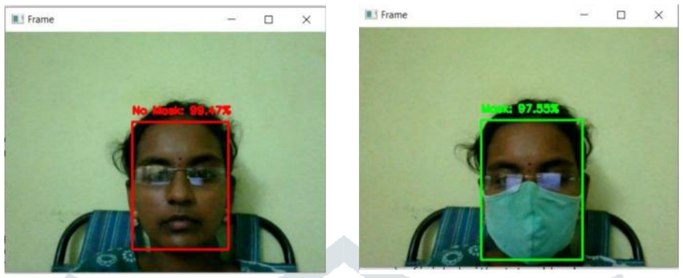


Fig 5:Detection of Mask using webcam with and without mask

A face mask identification system that achieves equivalent metrics to the current state-of-the-art technology has been created. Recent advances in computer vision and deep learning are used in this research. The evaluation of the model on the test dataset was found to be consistent. A custom dataset was created from scratch utilising the Bing Search API, Kaggle datasets, and RMFD dataset. The algorithm accurately identified the presence of face masks on human faces in both still photos and streaming video.
A two-class model was trained using photos of persons wearing masks and those who were not in order to construct a face mask detector. After using MobileNetV2 to refine the model on our mask/no mask dataset, we were able to produce an image classifier that was 93% accurate. Due to the Covid-19 outbreak, this technology can therefore be employed in real-time applications that demand face-mask detection for security reasons. To ensure that public safety regulations are fulfilled, this work can be combined with embedded technologies for application in airports, train stations, offices, schools, and public spaces.

## 8. Future scope

My Current method of detecting whether a person is wearing a mask or not is a two-step process:
**Step 1:** Perform face detection
**Step 2:** Apply our face mask detector to each face
The issue with this strategy is that a face mask by nature covers a portion of the face. The face mask detector will not be used if enough of the face is covered up to prevent detection of the face.
We should train a two-class object detector with a "with mask" class and a "without mask" class to get around that problem. The model can be improved in two ways by combining an object detector with a specific "with mask" class.
First, because too much of the face is hidden, the face detector would not have been able to recognise people wearing masks. Instead, the object detector will be able to do so.
Second, this strategy condenses our computer vision pipeline to a single step, eliminating the need for face detection and our face mask detector model. Instead, we only need to apply the object detector to obtain bounding boxes for both people with and without masks in a single forward pass of the network.
Such a procedure is not only more end-to-end and "beautiful," but also more computationally effective. In the future, we can also create an Android or web application for the same.

**References:**

[1] Wenxuan Han, Zitong Huang, Alifu.kuerban, Meng Yan, Haitang Fu, "A Mask Detection Method for Shoppers Under the Threat of COVID-19 Coronavirus", in 2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL), pp. 442-447, doi: 10.1109/CVIDL51233.2020.00-54.

[2] Susanto, Febri Alwan Putra, Riska Analia, Ika Karlina Laila Nur Suciningtyas, "The Face Mask Detection For Preventing the Spread of COVID-19 at Politeknik Negeri Batam", in 2020 3rd International Conference on Applied Engineering (ICAE), 2020 IEEE, doi: 10.1109/ICAE50557.2020.9350556.

[3] Abdellah Oumina, Noureddine El Makhfi, Mustapha Hamdi, "Control The COVID-19 Pandemic: Face Mask Detection Using Transfer Learning", 2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS), 2020 IEEE.

[4] R. Girshick, J. Donahue, T. Darrel and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", *Conference on Computer Vision and Pattern Recognition*, pp. 580-587, 2014.

[5] J. R.R. Uijlings, K. E. A. V. D. Sande, T. Gevers and A. W.M. Smeulders, "Selective search for object recognition", *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154-171, 2013.

[6] A. Sharifara, M. S. Mohd Rahim and Y. Anisi, "A general review of human face detection including a study of neural networks and Haar feature-based cascade classifier in face detection", *Proc: - 2014 Int. Symp. Biometrics Secur. Technol. ISBAST 2014*, pp. 73-78, 2015.

[7] R. Samet and M. Tanriverdi, "Face recognition-based mobile automatic classroom attendance management system", *Proc. - 2017 Int. Conf. Cyberworlds CW 2017 - Coop. with Eurographics Assoc. Int. Fed. Inf. Process. ACM SIGGRAPH*, vol. 2017-Janua, pp. 253-256, 2017.

[8] N. Muthukumaran, R. Aiswarya, S. Anna Sankari and K. Divya Bharathi, "Deep Learning Neural Network Based Human Emotion Classification with ANFIS Algorithm", *Irish Interdisciplinary Journal of Science & Research*, vol. 4, no. 3, pp. 105-111, July-September 2020.