International Journal of Scientific Research & Engineering Trends
Volume 7, Issue 4, July-Aug-2020, ISSN (Online): 2395-566X

Face Mask Detector

Keerthi Yaramala, Jamula Keerthi Sameera, Harshitha Yarlagadda, Bommina Naga Pranathi
Department of Information Technology,
Velagapudi Ramakrishna Siddhartha Engineering College,
Vijayawada, India.
188w1a12b4@vrsiddhartha.ac.in, 188w1a1277@vrsiddhartha.ac.in, 188w1a12b5@vrsiddhartha.ac.in,
198w5a1209@vrsiddhartha.ac.in

***Abstract-* In this new era where we are experiencing a pandemic and people all around the world are advised to wear masks, some people are not used to it and are avoiding to wear masks. The motivation behind this projects is that if we can take help of AI to detect people wearing or not wearing masks in public places, it would be helpful to increase our safety. If deployed correctly, the mask detector could potentially be used to help ensure our safety.**

***Keywords-* Mask Detection, Face Mask Detection, Neural network for mask detection, Computer vision.**

I. INTRODUCTION

Coronavirus Disease 2019 (COVID-19) unexpectedly broke out in 2019 and has seriously affected the whole world. As of 26 March 2021, COVID-19 has infected more than 125 million people worldwide and caused over 2.7 million deaths [1]. One of the transmission routes of COVID-19 is through droplets of saliva or nasal secretions when an infected person coughs or sneezes, which is highly infectious and could be worse in crowded places.

Since there is no specific treatment for COVID-19 [2], infections have to be limited through prevention methods. Studies have shown that wearing masks can reduce the risk of coronavirus transmission [3], which means wearing masks is currently one of the effective prevention methods [4]. According to the World Health Organization (WHO), the right way to wear a mask is by adjusting the mask to cover the mouth, nose, and chin [5].

The protection will be greatly reduced if masks are not worn properly. At present, security guards are arranged in public places to remind people to wear masks. However, this measure not only exposes the guards to the air that may contain the virus, but also leads to overcrowding at the entrances due to its inefficiency. Therefore, a fast and effective method is needed to address the situation.

Computer vision is an interdisciplinary scientific field that involves how computers gain advanced understanding from digital images or videos [6]. Traditional computer vision tasks include image processing, image classification, object detection, and image recognition. Object detection can detect instances of visual objects of a certain class in the images [7], which is a proper solution for the problem mentioned above. Consequently, mask detection has become a vital computer vision task to help the global society.

In this paper we proposed a cnn based model to detect the face mask. The model in this paper uses the Convolutional Neural Network. It is a deep neural network model used for analyzing any visual imagery. It takes the image data as input, captures all the data, and send to the layers of neurons. It has a fully connected layer, which processes the final output that represents the prediction about the image. The Convolutional neural network model used here is the MobileNetV2 architecture. MobileNet model is a network model using depth wise separable convolution as its basic unit. Its depth wise separable convolution has two layers: depth wise convolution and point convolution [1] .

It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depth wise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck.

II. BACKGROUND

We used some existing research and applications as references for our project. [1] A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. A model named as SSDMNV2 has been proposed in this paper for face mask detection using OpenCV Deep Neural Network (DNN), TensorFlow, Keras, and MobileNetV2 architecture which is used as an image classifier.SSDMNV2 performs competently in differentiating images having frontal faces with masks from images having frontal faces without masks.

To impede the COVID-19 transmission the proposed model can be integrated with surveillance cameras so that it can be used for the detection of people who are not wearing face masks. This paper also keeps complete attention towards the removal of various inaccurate predictions mainly in cases of real world datasets that occurred in different other proposed models. Detection of face masks is an extremely challenging task for the present proposed models of face.

This is because faces with masks have varied accommodations, various degrees of obstructions, and diversified mask types. They are used to facilitate self-focusing, the interaction between humans and computers and managing image database. Even after having such extraordinary and exceptional results in the existing face detectors, there is still high rising scrutiny in the development of more advanced face detectors as for existing models, event analysis and video surveillance is always a challenging job.

Several reasons were found for the poor achievement of existing face mask detection model as compared to the normal ones, two of them were First due to lack of suitable datasets with properly masked faces and facial recognition. Secondly, the presence of masks on the face brings a certain kind of noise, which further deteriorates the detection process. These issues have been studied in some existing research papers such still, there is an excellent challenge for a vast dataset so that an efficient face mask detection model can be easily developed. [2]

A Real-Time Integrated Face Mask Detector to Curtail Spread of Coronavirus. The proposed model is based on object recognition benchmark in. According to this benchmark, all the tasks related to an object recognition problem can be ensemble under three main components: Backbone, Neck and Head as depicted in. Here, the backbone corresponds to baseline convolutional neural network capable of extracting features from images. Since training a convolutional neural network is expensive in terms of computational power and time; transfer learning is applied here. Transfer learning allows to transfer the trained knowledge of the pre-trained neural network in terms of parametric weights to the new model.

In order to obtain best results for facemask detection, the experiment is setup with three popular pre-trained models namely ResNet50, MobileNet and AlexNet separately. Although emerging in 2012, AlexNet was considered in our experiments to show the progress in term of performances by comparison with more recent architectures. It is experimentally observed that ResNet50 is optimal choice in terms of accuracy, inference time and memory as compared to AlexNet and MobileNet for our targeted problem.

III. PROPOSED ARCHITECTURE

Our proposed model uses the convolutional neural network which is capable of extracting features from images. Since training a convolutional neural network is expensive in terms of computational power and time, transfer learning is applied here.

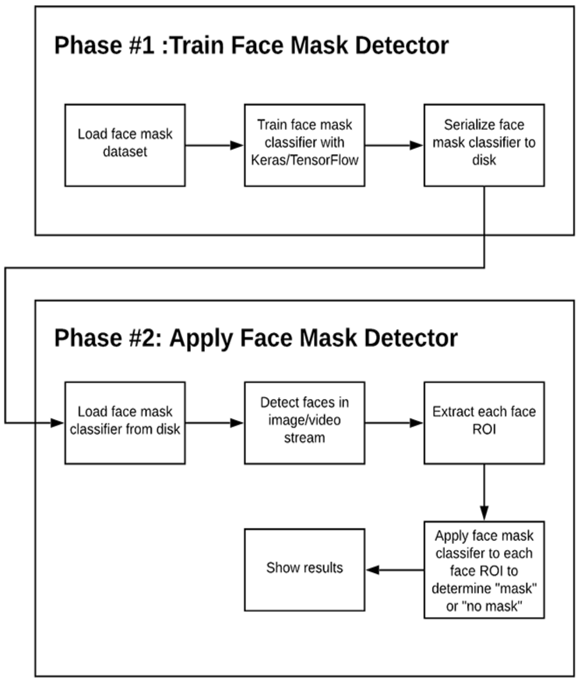

Phase #1 :Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifer to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

Fig 1. Proposed Architecture.

Transfer learning allows to transfer the trained knowledge of the pre-trained neural network in terms of parametric weights to the new model. In order to obtain best results for facemask detection, the experiment is setup with three popular pre-trained models namely ResNet50, MobileNet.

1. Construction of Face Masked Dataset:

- Bing search api is an microsoft application which provides services like web images,videos and news and many more.
- RMFD(Real-World Masked Face Dataset).
- kaggle which is an community for data scientists which allow users to publish and find data sets.

2. Transfer Learning:

Due to scarcity of large facemask-centric datasets and to employ the strengths of influential deep convolutional neural networks, the proposed model is built over ResNet50 and able to transfer functionality, features and weights learnt from first 49 layers to newly added fully connected convolutional layer with dimension 1×64.

Transfer learning leads to achieve good results even with optimal dataset, since basic face features have already been learnt by ResNet50 from a much larger dataset like ImageNet. In proposed model, only the parameters of backbone and head are initialized using ResNet50 and neck is customized to perform face and mask detection tasks.

For this work, the last layer of ResNet50 is replaced by adding four more layers. These are: a pooling layer of pool size=5×5, a flattering layer, a dense layer of 128 neurons and finally a decisive layer with softmax activation function are added to classify a face into mask/non-mask.

3. Bounding Box Regression:
After detecting a face in the search proposal, we apply affine transformation to obtain a bounding box of fixed size irrespective of the aspect ratio of the candidate region. The purpose of applying bounding box is to improve localization performance during mask detection. The technique is similar to deformable part models described in [12]. The primary difference between the two methods is that the proposed model regress from features computed by CNN rather than just applying geometric features taken from DPM part locations.

Let each region proposal (face) be represented by a pair (R, G), where R=(Rx,Ry,Rw,Rh) represents the pixel coordinates of the center of proposals along with width and height. Each ground truth bounding box is also represented in the same way, i.e., G=(Gx,Gy,Gw,Gh). So, the goal is to learn a transformation that can map region proposal (R) to ground-truth bounding box (G) without loss of information. We propose to apply a scale-invariant transformation on pixels coordinates of R and log space transformation on width and height of R. The corresponding four transformation functions are represented as Tx(R), Ty(R), Tw(R) and Th(R). So, coordinates of the ground truth box can be obtained by Eqs. (2)–(5).

Gx=Tx(Rx)+Rx(2)

Gy=Ty(Ry)+Ry(3)

Gw=Tw(Rw)+Rw(4)

Gh=Th(Rh)+Rh(5)

Here, each Ti (where i denotes one of x, y, w, h) is applied as a linear function of the Pool 6 feature of R denoted by f6 (R). Here, the dependence of f6 (R) on R is implicitly assumed. Thus, Ti(R) can be obtained by Eq. (6).

Ti(R)=wif6(R)(6)

where Wi denotes the weight learned by optimizing the regularized least square objective of ridge regression and is computed by Eq. (7). Ridge regression is used here, to penalize the variables with minor contribution to the outcome; have their coefficient close to zero. One popular penalty is to panelize the model based on sum of squared coefficient values; this is called L2 penalty (w^i). The w^i minimize the size of all coefficients and preventing the coefficient from being removed from the model.

Further, a hyperparameter called tuning parameter or penalty parameter (λ) is used to control the weighing of penalty to the loss function. A default value of λ=1.0 will fully weight the penalty whereas λ=0 excludes the penalty. The scikit-learn library in Python is used to automatically finds good value for λ via RidgeCV class. For our model. λ is set to 0.51.

wi=∑n∈R(tin-w^f6(Rn))2+λ|w^i|2(7)

The regression target (ti) related to coordinates, width and height of region proposal pair (R, G) are defined by Eqs. (8)–(11), respectively.

tx=(Gx-Rx)/Rw(8)

ty=(Gy-Ry)/Rh(9)

tw=log(Gw/Rw)(10)

th=log(Gh/Rh)(11)

4. Loss Function and Optimization:
Defining the loss function for the classification problem is among the most important part of the convolutional neural network design. In classification theory, a loss function or objective function is defined as a function that maps estimated distribution onto true distribution. It is desirable for an optimization algorithm to minimize the output of this function. The stochastic gradient descent optimization algorithm is applied to update the model parameters with a learning rate of 0.03. Further, there exist numerous loss functions in PyTorch but one which is most suitable with balance data is a cross-entropy loss. Furthermore, an activation function is required at the output layer to transform the output in such a way that would be easier to interpret for the loss function.

Since the formula for cross-entropy loss given in Eq. (12) takes two distributions, t(x), the true distribution and e(x), the estimated distribution defined over discrete variables x [58], thus activation functions that are not interpretable as probabilities (i.e., negative or greater than 1 or sum of output not equals to 1) should not be selected.

Since Softmax guarantees to generate well behaved probabilities distribution over categorial variable so it is chosen in our proposed model.

Loss=∑∀xt(x)log(e(x))(12)

Further, the loss function over N images (also known as cost function over the complete system) in binary classification can be formulated as given in Eq. (13).

Loss=1N∑x∑n=1Ntn(x)log(en(x))(13)

5. Performance Analysis of Proposed Model:
The performance of the proposed model using ResNet50 is further evaluated using various metrics such as Accuracy, Intersection over Union (IoU) and AU-ROC using Eqs. (14)–(17), respectively.

Accuracy=(TP+TN)(TP+TN+FP+FN)(14)

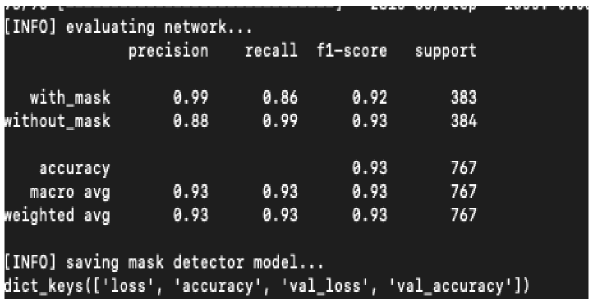
IoU=TP(TP+FP+FN)(15)

Sensitivity=TP(TP+FN)(16)

Specificity=TN(TN+FP)(17)

Where TP, FP, TN and FN represent True Positive, False Positive, True Negative and False Negative, respectively. TP, FP, TN and FN are obtained through a confusion matrix.

Classification report for our model is as follows:



```
[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'])
```

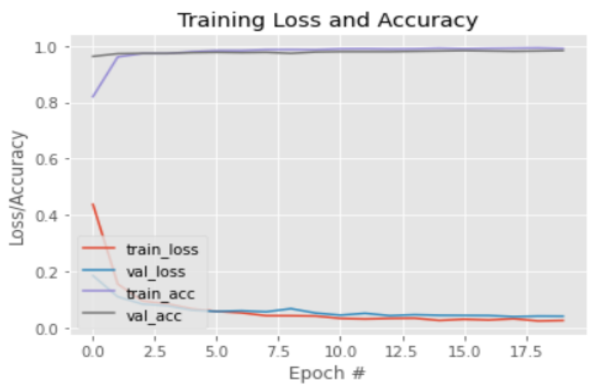


Fig 2. Accuracy/loss training curve plot.

IV. MODULES

1. Module_1: Detect_mask_image:

- Constructing the argument parsers for path to input image, face_detector and Mask_detector_model.
- Loading the serialized face detector model.
- Loading the serialized face mask detector model.
- Loading the input image from disk, clone it, and grab the image spatial dimensions.
- Constructing a blob from the image and passing the blob through the network and obtain the face detections.
- Extract the confidence (i.e., probability) associated with the detection.
- Filtering out weak detections by ensuring the confidence is greater than the minimum confidence.
- Compute the (x, y)-coordinates of the bounding box for the object.
- Extract the face ROI, convert it from BGR to RGB channel.
- Ordering, resize it to 224x224, and preprocess it.
- Passing the face through the model to determine if the face has a mask or not.
- Determining the class label including the probability in the label.
- Displaying the label and bounding box rectangle on the output frame.
- Finally showing the output image with bounding box rectangle .

2. Module_2: detect_mask_video:

- Grab the dimensions of the frame and then construct a blob from it.
- Pass the blob through the network and obtain the face detections.
- Initialize our list of faces, their corresponding locations, and the list of predictions from our face mask network.
- constructing the argument parsers for path to input image, face_detector and Mask_detector_model. Loading the serialized face detector model.
- Loading the serialized face mask detector model.
- Initialize the video stream and allow the camera sensor to warm up.
- Grab the frame from the threaded video stream and resize it to have a maximum width of 400 pixels.
- Compute the (x, y)-coordinates of the bounding box for the object.
- Extract the face ROI, convert it from BGR to RGB channel.
- Passing the face through the model to determine if the face has a mask or not.
- Loop over the detected face locations and their corresponding locations.
- Determining the class label including the probability in the label.
- Displaying the label and bounding box rectangle on the output frame.
- If the `q` key was pressed, break from the loop.

V. IMPLEMENTATION

The Implementation part of the Project Management method puts the project into action. It is the most important phase as all the main tasks including the execution of the project are done here. Some of the software and frameworks which we used for the implementation of various steps of the project are.

1. Kaggle:
Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data

science challenges. We have extracted the datasets from kaggle which are used for model training.

2. Tensorflow:
Tensorflow is an open-source library for numerical computation and large-scale machine learning that ease Google Brain TensorFlow, the process of acquiring data, training models, serving predictions, and refining future results. Tensorflow bundles together Machine Learning and Deep Learning models and algorithms. Here we used the tensorflow for creating our neural network model.

3. Keras:
Keras is a powerful and easy-to-use free open source Python library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in just a few lines of code. Here we used the keras for ease evaluation of our model using tensorflow.

4. Imutil:
Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

5. Opencv:
OpenCV-Python is a library of Python bindings designed to solve computer vision problems. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

6. Numpy:
NumPy is an open-source numerical Python library. NumPy contains a multi-dimensional array and matrix data structures. It can be utilised to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.

7. Scipy:
SciPy is an open-source Python library which is used to solve scientific and mathematical problems. It is built on the NumPy extension and allows the user to manipulate and visualize data with a wide range of high-level commands.

8. Scikit-Learn:
Scikit-learn is the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

9. Pillow:
Python pillow library is used to image class within it to show the image. The image modules that belong to the pillow package have a few inbuilt functions such as load images or create new images, etc.

10. Streamlit:
Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. In just a few minutes you can build and deploy powerful data apps. Here we used the streamlit for creating a web application that is used to deploy our model.

11. Matplotlib:
Matplotlib. pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

VI. CONCLUSION AND FUTURE WORK

To conclude, the above results are sample results and denote that they need much future development like we can introduce videos classification in web app and try to use different algorithms for further best results. We have achieved an accuracy of 97% and we can do best by trying alternate algorithms.

For the future development, we can focus on improving accuracy of the model by adding more data sets to training data and test data. Also we can add video future in the web app to identify the masked people in video sample. We can use different face detection models to increase the detection of people in case of crowded places.

REFERENCES

[1] "A survey on Face Detection in the Wild: Past , Present and future" Cha Zhang .
[2] "Neural Network - based Face Detection" - S. Baluja.
[3] "Facial Mask detection using semantic Segmentation" - T. Meenpal.
[4] Python documentation - https://docs.python.org/3.8/
[5] Richard Duda, Peter Hart and David Stork, "Pattern Classification", 2 nd ed., John Wiley and Sons.
[6] Support Vector Machines (SVM), http://www.site.uottawa.ca/~stan/csi5387/SVMtransp.pdf
[7] Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C. Ghemawat S. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467. [Google Scholar]
[8] Ahmed I., Ahmad M., Rodrigues J.J., Jeon G., Din S. A deep learning-based social distance monitoring framework for COVID-19. Sustainable Cities and Society. 2020 [PMC free article] [PubMed] [Google Scholar].
[9] Alom M.Z., Taha T.M., Yakopcic C., Westberg S., Sidike P., Nasrin M.S. Asari V.K. 2018. The history

began from alexnet: A comprehensive survey on deep learning approaches. arXiv preprint arXiv: 1803. 01164. [Google Scholar].

[10] Anisimov D., Khanova T. Towards lightweight convolutional neural networks for object detection. 2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS); IEEE; 2017. pp. 1–8. August. [Google Scholar].

[11] Chen D., Ren S., Wei Y., Cao X., Sun J. European conference on computer vision. Springer; Cham: 2014. Joint cascade face detection and alignment; pp. 109–122. September. [Google Scholar].

[12] Chen D., Hua G., Wen F., Sun J. European conference on computer vision. Springer; Cham: 2016. Supervised transformer network for efficient face detection; pp. 122–138. October. [Google Scholar]

[13] Ge S., Li J., Ye Q., Luo Z. Detecting masked faces in the wild with lle-cnns. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017:2682–2690. [Google Scholar].

[14] Ge X.Y., Pu Y., Liao C.H., Huang W.F., Zeng Q., Zhou H.…Chen H.L. Evaluation of the exposure risk of SARS-CoV-2 in different hospital environment. Sustainable Cities and Society. 2020;61 [PMC free article] [PubMed] [Google Scholar].

[15] Ghiasi G., Fowlkes C.C. Occlusion coherence: Localizing occluded faces with a hierarchical deformable part model. Proceedings of the IEEE conference on computer vision and pattern recognition. 2014:2385–2392. [Google Scholar].

[16] Gulcehre C., Moczulski M., Denil M., Bengio Y. International conference on machine learning. 2016. Noisy activation functions; pp. 3059–3068. June. [Google Scholar].

[17] Hahnloser R.H., Sarpeshkar R., Mahowald M.A., Douglas R.J., Seung H.S. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature. 2000;405(6789):947–951. [PubMed] [Google Scholar].