Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV

Arjya Das
Department of Information Technology
Jadavpur University
Kolkata, India
arjyadas1999@gmail.com

Mohammad Wasif Ansari
Department of Information Technology
Jadavpur University
Kolkata, India
razamoeezraza@gmail.com

Rohini Basak
Department of Information Technology
Jadavpur University
Kolkata, India
visitrohinihere@gmail.com

2020 IEEE 17th India Council International Conference (INDICON) | 978-1-7281-6916-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/INDICON49873.2020.9342585

***Abstract*—COVID-19 pandemic has rapidly affected our day-to-day life disrupting the world trade and movements. Wearing a protective face mask has become a new normal. In the near future, many public service providers will ask the customers to wear masks correctly to avail of their services. Therefore, face mask detection has become a crucial task to help global society. This paper presents a simplified approach to achieve this purpose using some basic Machine Learning packages like TensorFlow, Keras, OpenCV and Scikit-Learn. The proposed method detects the face from the image correctly and then identifies if it has a mask on it or not. As a surveillance task performer, it can also detect a face along with a mask in motion. The method attains accuracy up to 95.77% and 94.58% respectively on two different datasets. We explore optimized values of parameters using the Sequential Convolutional Neural Network model to detect the presence of masks correctly without causing over-fitting.**

***Keywords*—Coronavirus, Covid-19, Machine Learning, Face Mask Detection, Convolutional Neural Network, TensorFlow**

I. Introduction

According to the World Health Organization (WHO)'s official Situation Report – 205, coronavirus disease 2019 (COVID-19) has globally infected over 20 million people causing over 0.7million deaths [1]. Individuals with COVID-19 have had a wide scope of symptoms reported – going from mellow manifestations to serious illness. Respiratory problems like shortness of breath or difficulty in breathing is one of them. Elder people having lung disease can possess serious complications from COVID-19 illness as they appear to be at higher risk [2]. Some common human coronaviruses that infect public around the world are 229E, HKU1, OC43, and NL63. Before debilitating individuals, viruses like 2019-nCoV, SARS-CoV, and MERS-CoV infect animals and evolve to human coronaviruses [3]. Persons having respiratory problems can expose anyone (who is in close contact with them) to infective beads. Surroundings of a tainted individual can cause contact transmission as droplets carrying virus may withal arrive on his adjacent surfaces [4].

To curb certain respiratory viral ailments, including COVID-19, wearing a clinical mask is very necessary. The public should be aware of whether to put on the mask for source control or aversion of COVID-19. Potential points of interest of the utilization of masks lie in reducing vulnerability of risk from a noxious individual during the "pre-symptomatic" period and stigmatization of discrete persons putting on masks to restraint the spread of virus. WHO stresses on prioritizing medical masks and respirators for health care assistants[4]. Therefore, face mask detection has become a crucial task in present global society.

Face mask detection involves in detecting the location of the face and then determining whether it has a mask on it or not. The issue is proximately cognate to general object detection to detect the classes of objects. Face identification categorically deals with distinguishing a specific group of entities i.e. Face. It has numerous applications, such as autonomous driving, education, surveillance, and so on [5]. This paper presents a simplified approach to serve the above purpose using the basic Machine Learning (ML) packages such as TensorFlow, Keras, OpenCV and Scikit-Learn.

The rest of the paper is organized as follows: Section II explores related work associated with face mask detection. Section III discusses the nature of the used dataset. Section IV presents the details of the packages incorporated to build the proposed model. Section V gives an overview of our method. Experimental results and analysis are reported in section VI. Section VII concludes and draws the line towards future works.

II. Related Work

In face detection method, a face is detected from an image that has several attributes in it. According to [21], research into face detection requires expression recognition, face tracking, and pose estimation. Given a solitary image, the challenge is to identify the face from the picture. Face detection is a difficult errand because the faces change in size, shape, color, etc and they are not immutable. It becomes a laborious job for opaque image impeded by some other thing not confronting camera, and so forth. Authors in [22] think occlusive face detection comes with two major challenges: 1) unavailability of sizably voluminous

datasets containing both masked and unmasked faces, and 2) exclusion of facial expression in the covered area. Utilizing the locally linear embedding (LLE) algorithm and the dictionaries trained on an immensely colossal pool of masked faces, synthesized mundane faces, several mislaid expressions can be recuperated and the ascendancy of facial cues can be mitigated to great extent. According to the work reported in [11], convolutional neural network (CNNs) in computer vision comes with a strict constraint regarding the size of the input image. The prevalent practice reconfigures the images before fitting them into the network to surmount the inhibition.

Here the main challenge of the task is to detect the face from the image correctly and then identify if it has a mask on it or not. In order to perform surveillance tasks, the proposed method should also detect a face along with a mask in motion.

III. Dataset

Two datasets have been used for experimenting the current method. Dataset 1 [16] consists of 1376 images in which 690 images with people wearing face masks and the rest 686 images with people who do not wear face masks. Fig. 1 mostly contains front face pose with single face in the frame and with same type of mask having white color only.

Fig. 1. Samples from Dataset 1 including faces without masks and with masks

Dataset 2 from Kaggle [17] consists of 853 images and its countenances are clarified either with a mask or without a mask. In fig. 2 some face collections are head turn, tilt and slant with multiple faces in the frame and different types of masks having different colors as well.

Fig. 2. Samples from Dataset 2 including faces without masks and with masks

IV. Incorporated Packages

A. TensorFlow

TensorFlow, an interface for expressing machine learning algorithms, is utilized for implementing ML systems into fabrication over a bunch of areas of computer science, including sentiment analysis, voice recognition, geographic information extraction, computer vision, text summarization, information retrieval, computational drug discovery and flaw detection to pursue research [18]. In the proposed model, the whole Sequential CNN architecture (consists of several layers) uses TensorFlow at backend. It is also used to reshape the data (image) in the data processing.

B. Keras

Keras gives fundamental reflections and building units for creation and transportation of ML arrangements with high iteration velocity. It takes full advantage of the scalability and cross-platform capabilities of TensorFlow. The core data structures of Keras are layers and models [19]. All the layers used in the CNN model are implemented using Keras. Along with the conversion of the class vector to the binary class matrix in data processing, it helps to compile the overall model.

C. OpenCV

OpenCV (Open Source Computer Vision Library), an open-source computer vision and ML software library, is utilized to differentiate and recognize faces, recognize objects, group movements in recordings, trace progressive modules, follow eye gesture, track camera actions, expel red eyes from pictures taken utilizing flash, find comparative pictures from an image database, perceive landscape and set up markers to overlay it with increased reality and so forth [20]. The proposed method makes use of these features of OpenCV in resizing and color conversion of data images.

V. The Proposed Method

The proposed method consists of a cascade classifier and a pre-trained CNN which contains two 2D convolution layers connected to layers of dense neurons. The algorithm for face mask detection is as follows:

Algorithm 1: Face Mask Detection

Input: Dataset including faces with and without masks
Output: Categorized image depicting the presence of face mask
for *each image in the dataset* **do**
    Visualize the image in two categories and label them
    Convert the RGB image to Gray-scale image
    Resize the gray-scale image into 100 x 100
    Normalize the image and convert it into 4 dimensional array
end
for *building the CNN model* **do**
    Add a Convolution layer of 200 filters
    Add the second Convolution layer of 100 filters
    Insert a Flatten layer to the network classifier
    Add a Dense layer of 64 neurons
    Add the final Dense layer with 2 outputs for 2 categories
end
Split the data and train the model

A. Data Processing

Data preprocessing involves conversion of data from a given format to much more user friendly, desired and meaningful format. It can be in any form like tables, images, videos, graphs, etc. These organized information fit in with an information model or composition and captures relationship between different entities [6]. The proposed method deals with image and video data using Numpy and OpenCV.

a) Data Visualization: Data visualization is the process of transforming abstract data to meaningful representations using knowledge communication and insight discovery through encodings. It is helpful to study a particular pattern in the dataset [7].

The total number of images in the dataset is visualized in both categories – 'with mask' and 'without mask'.

The statement *categories=os.listdir(data_path)* categorizes the list of directories in the specified data path. The variable *categories* now looks like: ['with mask', 'without mask']

Then to find the number of labels, we need to distinguish those categories using *labels=[i for i in range(len(categories))]*. It sets the labels as: [0, 1]

Now, each category is mapped to its respective label using *label_dict=dict(zip(categories,labels))* which at first returns an iterator of tuples in the form of zip object where the items in each passed iterator is paired together consequently. The mapped variable *label_dict* looks like: {'with mask': 0, 'without mask': 1}

b) Conversion of RGB image to Gray image: Modern descriptor-based image recognition systems regularly work on grayscale images, without elaborating the method used to convert from color-to-grayscale. This is because the color-to-grayscale method is of little consequence when using robust descriptors. Introducing nonessential information could increase the size of training data required to achieve good performance. As grayscale rationalizes the algorithm and diminishes the computational requisites, it is utilized for extracting descriptors instead of working on color images instantaneously [8].

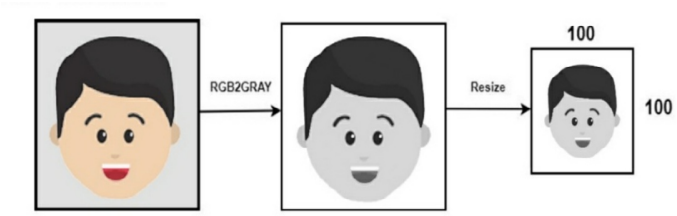

Fig. 3. Conversion of a RGB image to a Gray Scale image of 100x100 size

We use the function *cv2.cvtColor(input_image, flag)* for changing the color space. Here flag determines the type of conversion [9]. In this case, the flag *cv2.COLOR_BGR2GRAY* is used for gray conversion.

Deep CNNs require a fixed-size input image. Therefore we need a fixed common size for all the images in the dataset. Using *cv2.resize()* the gray scale image is resized into 100 x 100.

c) Image Reshaping: The input during relegation of an image is a three-dimensional tensor, where each channel has a prominent unique pixel. All the images must have identically tantamount size corresponding to 3D feature tensor. However, neither images are customarily coextensive nor their corresponding feature tensors [10]. Most CNNs can only accept fine-tuned images. This engenders several problems throughout data collection and implementation of model. However, reconfiguring the input images before augmenting them into the network can help to surmount this constraint. [11].

The images are normalized to converge the pixel range between 0 and 1. Then they are converted to 4 dimensional arrays using *data=np.reshape(data,(data.shape[0], img_size,img_size,1)*) where 1 indicates the Grayscale image. As, the final layer of the neural network has 2 outputs – with mask and without mask i.e. it has categorical representation, the data is converted to categorical labels.

B. Training of Model

a) Building the model using CNN architecture: CNN has become ascendant in miscellaneous computer vision tasks [12]. The current method makes use of Sequential CNN.

The First Convolution layer is followed by Rectified Linear Unit (ReLU) and MaxPooling layers. The Convolution layer learns from 200 filters. Kernel size is set to 3 x 3 which specifies the height and width of the 2D convolution window. As the model should be aware of the shape of the input expected, the first layer in the model needs to be provided with information about input shape. Following layers can perform instinctive shape reckoning [13]. In this case, *input_shape* is specified as *data.shape[1:]* which returns the dimensions of the data array from index 1. Default padding is "valid" where the spatial dimensions are sanctioned to truncate and the input volume is non-zero padded. The activation parameter to the Conv2D class is set as "relu". It represents an approximately linear function that possesses all the assets of linear models that can easily be optimized with gradient-descent methods. Considering the performance and generalization in deep learning, it is better compared to other activation functions [14]. Max Pooling is used to reduce the spatial dimensions of the output volume. Pool_size is set to 3 x 3 and the resulting output has a shape (number of rows or columns) of: shape_of_output = (input_shape - pool_size + 1) / strides), where strides has default value (1,1) [15].

As shown in fig, 4, the second Convolution layer has 100 filters and Kernel size is set to 3 x 3. It is followed by ReLu and MaxPooling layers. To insert the data into CNN, the long vector of input is passed through a Flatten layer which transforms matrix of features into a vector that can be fed into a fully connected neural network classifier. To reduce

overfitting a Dropout layer with a 50% chance of setting inputs to zero is added to the model. Then a Dense layer of 64 neurons with a ReLu activation function is added. The final layer (Dense) with two outputs for two categories uses the Softmax activation function.

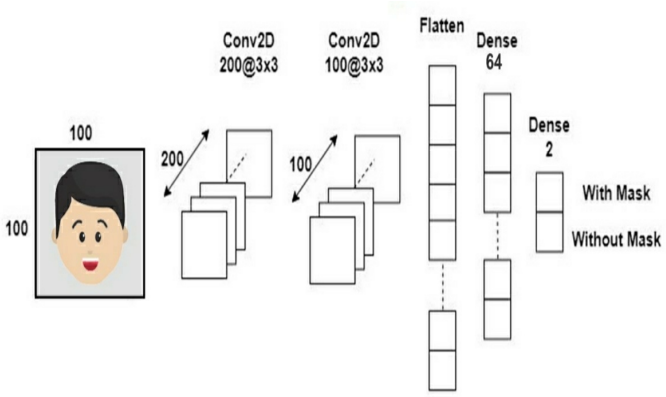

Fig. 4. Convolutional Neural Network architecture

The learning process needs to be configured first with the compile method [13]. Here "adam" optimizer is used. *categorical_crossentropy* which is also known as multiclass log loss is used as a loss function (the objective that the model tries to minimize). As the problem is a classification problem, metrics is set to "accuracy".

b) Splitting the data and training the CNN model: After setting the blueprint to analyze the data, the model needs to be trained using a specific dataset and then to be tested against a different dataset. A proper model and optimized *train_test_split* help to produce accurate results while making a prediction. The test_size is set to 0.1 i.e. 90% data of the dataset undergoes training and the rest 10% goes for testing purposes. The validation loss is monitored using *ModelCheckpoint*. Next, the images in the training set and the test set are fitted to the Sequential model. Here, 20% of the training data is used as validation data. The model is trained for 20 epochs (iterations) which maintains a trade-off between accuracy and chances of overfitting. Fig. 5 depicts visual representation of the proposed model.

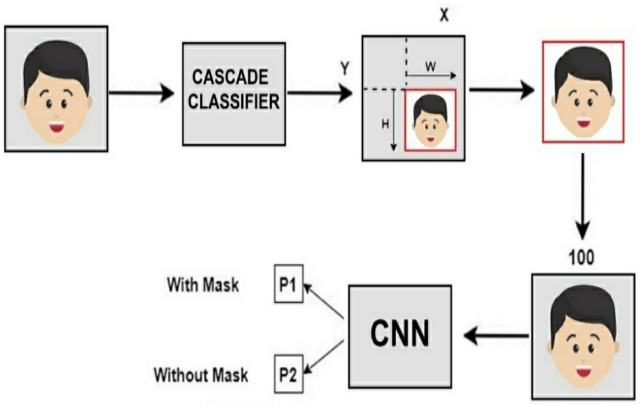

Fig. 5. Overview of the Model

VI. Result And Analysis

The model is trained, validated and tested upon two datasets. Corresponding to dataset 1, the method attains accuracy up to 95.77% (shown in fig. 7). Fig. 6 depicts how this optimized accuracy mitigates the cost of error. Dataset 2 is more versatile than dataset 1 as it has multiple faces in the frame and different types of masks having different colors as well. Therefore, the model attains an accuracy of 94.58% on dataset 2 as shown in Fig. 9. Fig. 8 depicts the contrast between training and validation loss corresponding to dataset 2. One of the main reasons behind achieving this accuracy lies in *MaxPooling*. It provides rudimentary translation invariance to the internal representation along with the reduction in the number of parameters the model has to learn. This sample-based discretization process down-samples the input representation consisting of image, by reducing its dimensionality. Number of neurons has the optimized value of 64 which is not too high. A much higher number of neurons and filters can lead to worse performance. The optimized filter values and pool_size help to filter out the main portion (face) of the image to detect the existence of mask correctly without causing over-fitting.

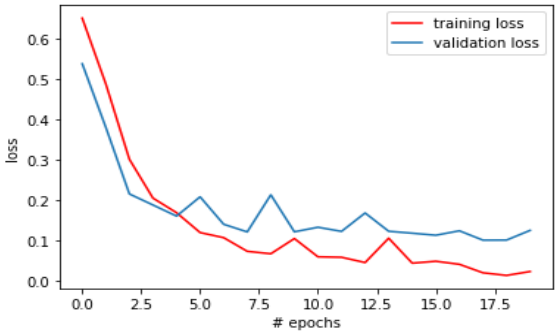

Fig. 6. # epochs vs loss corresponding to dataset 1

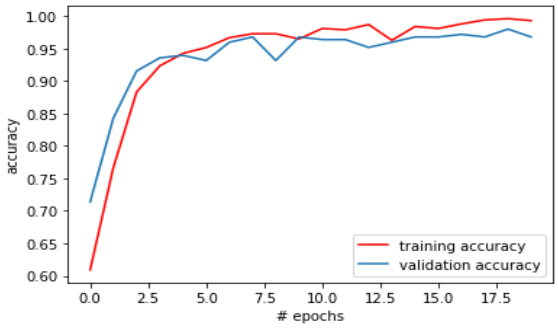

Fig. 7. # epochs vs accuracy corresponding to dataset 1

The system can efficiently detect partially occluded faces either with a mask or hair or hand. It considers the occlusion degree of four regions – nose, mouth, chin and eye to differentiate between annotated mask or face covered by hand. Therefore, a mask covering the face fully including nose and chin will only be treated as "with mask" by the model.

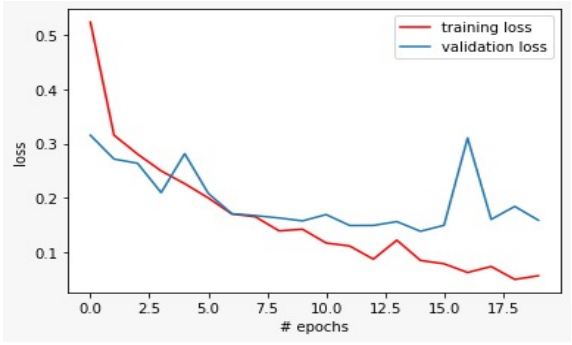

Fig. 8. # epochs vs loss corresponding to dataset 2

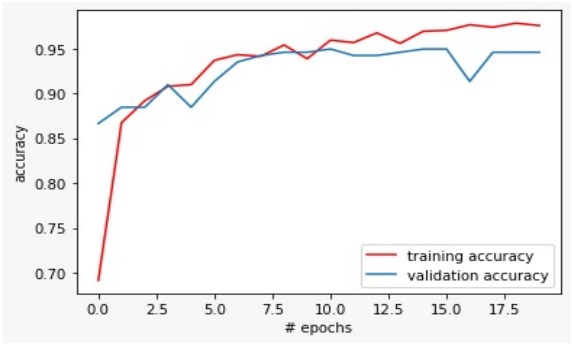

Fig. 9. # epochs vs accuracy corresponding to dataset 2

The main challenges faced by the method mainly comprise of varying angles and lack of clarity. Indistinct moving faces in the video stream make it more difficult. However, following the trajectories of several frames of the video helps to create a better decision – "with mask" or "without mask".

VII. Conclusions

In this paper, we briefly explained the motivation of the work at first. Then, we illustrated the learning and performance task of the model. Using basic ML tools and simplified techniques the method has achieved reasonably high accuracy. It can be used for a variety of applications.Wearing a mask may be obligatory in the near future, considering the Covid-19 crisis. Many public service providers will ask the customers to wear masks correctly to avail of their services. The deployed model will contribute immensely to the public health care system. In future it can be extended to detect if a person is wearing the mask properly or not. The model can be further improved to detect if the mask is virus prone or not i.e. the type of the mask is surgical, N95 or not.

References

[1] W.H.O., "Coronavirus disease 2019 (covid-19): situation report, 205". 2020

[2] "Coronavirus Disease 2019 (COVID-19) – Symptoms", Centers for Disease Control and Prevention, 2020. [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/symptoms-testing/symptoms.html. 2020.

[3] "Coronavirus — Human Coronavirus Types — CDC", Cdc.gov, 2020. [Online]. Available: https://www.cdc.gov/coronavirus/types.html. 2020.

[4] W.H.O., "Advice on the use of masks in the context of COVID-19: interim guidance", 2020.

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

[6] B. Suvarnamukhi and M. Seshashayee, "Big Data Concepts and Techniques in Data Processing", International Journal of Computer Sciences and Engineering, vol. 6, no. 10, pp. 712-714, 2018. Available: 10.26438/ijcse/v6i10.712714.

[7] F. Hohman, M. Kahng, R. Pienta and D. H. Chau, "Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers," in IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 8, pp. 2674-2693, 1 Aug. 2019, doi: 10.1109/TVCG.2018.2843369.

[8] C. Kanan and G. Cottrell, "Color-to-Grayscale: Does the Method Matter in Image Recognition?", PLoS ONE, vol. 7, no. 1, p. e29740, 2012. Available: 10.1371/journal.pone.0029740.

[9] Opencv-python-tutroals.readthedocs.io. 2020. Changing Colorspaces — Opencv-Python Tutorials 1 Documentation. [online] Available at:https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html. 2020.

[10] M. Hashemi, "Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation", Journal of Big Data, vol. 6, no. 1, 2019. Available: 10.1186/s40537-019-0263-7 . 2020.

[11] S. Ghosh, N. Das and M. Nasipuri, "Reshaping inputs for convolutional neural network: Some common and uncommon methods", Pattern Recognition, vol. 93, pp. 79-94, 2019. Available: 10.1016/j.patcog.2019.04.009.

[12] R. Yamashita, M. Nishio, R. Do and K. Togashi, "Convolutional neural networks: an overview and application in radiology", Insights into Imaging, vol. 9, no. 4, pp. 611-629, 2018. Available: 10.1007/s13244-018-0639-9.

[13] "Guide to the Sequential model - Keras Documentation", Faroit.com, 2020. [Online]. Available: https://faroit.com/keras-docs/1.0.1/getting-started/sequential-model-guide/. 2020.

[14] Nwankpa, C., Ijomah, W., Gachagan, A. and Marshall, S., 2020. Activation Functions: Comparison Of Trends In Practice And Research For Deep Learning. [online] arXiv.org. Available at: https://arxiv.org/abs/1811.03378. 2020.

[15] K. Team, "Keras documentation: MaxPooling2D layer", Keras.io, 2020. [Online]. Available: https://keras.io/api/layers/pooling_layers/max_pooling2d/. 2020.

[16] "prajnasb/observations", GitHub, 2020. [Online]. Available: https://github.com/prajnasb/observations/tree/master/experiements/data. 2020.

[17] "Face Mask Detection", Kaggle.com, 2020. [Online]. Available: https://www.kaggle.com/andrewmvd/face-mask-detection. 2020.

[18] "TensorFlow White Papers", TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/about/bib. 2020.

[19] K. Team, "Keras documentation: About Keras", Keras.io, 2020. [Online]. Available: https://keras.io/about. 2020.

[20] "OpenCV", Opencv.org, 2020. [Online]. Available: https://opencv.org/. 2020.

[21] D. Meena and R. Sharan, "An approach to face detection and recognition," 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Jaipur, 2016, pp. 1-6, doi: 10.1109/ICRAIE.2016.7939462.

[22] S. Ge, J. Li, Q. Ye and Z. Luo, "Detecting Masked Faces in the Wild with LLE-CNNs," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 426-434, doi: 10.1109/CVPR.2017.53.