Face mask Detection using Convolutional Neural Network

Ashiqur Rahman - 1712389642;
Jubayer Hossain Arnob - 1813124642

North South University

Abstract. After the new Coronavirus disease (COVID-19) case spread rapidly in Wuhan-China in December 2019, World Health Organization (WHO) [1] confirmed that this is a dangerous virus which can be spreading from humans to humans through droplets and airborne. The basic aim of the project is to detect the presence of a face mask on human faces on live streaming video as well as on images. We have used deep learning to develop our face detector model. A dataset from kaggle [2] of 6200 images is used to build this face mask detector using Python, OpenCV [3], TensorFlow [4] and Keras [5]. Some irresponsible people refuse to wear face mask with so many excuses. Moreover, developing the face mask detector is very crucial in this case. The experimental results have done very well and we believe that this project can be merged with embedded application systems at airports, train stations, workplaces, schools, and public places to ensure compliance with the guidelines for public safety. From the experimental results, this device is able to detect the people who wear or do not wear the face mask accurately even if they are moving to various position.

1 Introduction

Public use of face masks has been common in China and other nations in the world since the beginning of the new coronavirus disease outbreak. The recent information also gives trace of a new strain of corona virus, the mutant corona virus which, in which the virus has changed its structure and become mutant. The new strain is not even able to detect using the RT-PCR [6] test we use now. So it is inevitable for the people of an overpopulated country like Bangladesh to wear masks and let the work go on. No one can keep an eye on each person wearing a mask in the workplace. Thus came the necessity to identify face masks. It is a deep neural network model that is utilized for visual imagery analysis. The visual data is used as input, all data is captured and the layers of neurons are sent. It features a completely linked layer that works with the final output representing the image prediction. This paper presents a simplified approach to serve the above purpose using the Convolutional Neural Networks - Python, Keras [5], Tensorflow, OpenCV [3]. This network is composed of two pairs of layers Conv and MaxPool to extract features from dataset. Which is then followed by a Flatten and Dropout layer to convert the data in 1D and

avoid overfitting. In the next section we briefly discuss about the related work on Face Mask detection projects. We will address briefly the related work on face mask detection projects in the following part. In section 3, we explore in detail and describe the architecture of the system our suggested framework and tools. In addition the results of the recovery system may be found in the next part and, before summarizing the report in the last portion of section 5, we highlight the limits and future work.

2 Related Work

Projects with similar intent have been quite popular due to the ongoing pandemic. In the paper by Adnane Cabani [7] and his colleagues from Universite de Haute-Alsace, a method was proposed to utilize haar-cascade based feature detectors to individually determine the presence of nose and mouth from a detected face. Its reasoning follows that no mask is worn if one can successfully identify one's mouth, the mask is miscarried if one cannot detect one's nose by one's mouth, and the mask is correctly worn if either one's nose or mouth cannot be identified. This technique is effective and intuitive, but it does have serious restrictions - only complete face can process and one may deceive the detector easy by handing over one's lips and nose.

Another approach is proposed by Isunuri, Jagadeesh, Shree in their project. In [8] they have proposed a pre-trained MobileNet [9] with a global pooling block for face mask detection. The pre- prepared MobileNet takes a shading picture and creates a multi-dimensional component map. The worldwide pooling block that has been used in the proposed model changes the element map into an element vector of 64 highlights. At long last, the softmax layer performs paired order utilizing the 64 highlights.

Chen et al., in [10], have created a smartphone application that allows us to determine the service life of a facemask, indicating what period it is in and informing us how effective it is after a time of usage. They achieve this by collecting four gray features from microphotographs of the face mask and using co-occurrence matrices. Three outcomes are obtained using KNN. These have an accuracy of 82.87 percent. The accuracy of "normal use" and "not recommended" is 92.00 and 92.59 percent, respectively.

3 Proposed Framework (Or Methodology)

Our proposed framework for face mask detection is divided into three modules where the first module is "Data Preprocessing", the second module is "Model Training" and the third module is "Retrieval". We analyzed the data and transformed it into a comprehensible form for the training module in data preprocessing. This classifier distinguishes between two classes. They are "with mask" and "without mask". The data preprocessing module assists in the cleaning, structuring, and organization of raw data, preparing it to be used in the training module.This module categorized the data, reduced the number of image

channels, and gave each sample a distinct shape. Finally, used NumPy [11] to transform the images into usable numerical arrays.

Model training exploits deep learning technique, more specifically convolutional neural network to train a classification model using publicly available face mask image dataset. The classifier predicts the user input (i.e. frames from live video with face in it) and classifies as one of the classes among the two classes the model is trained with. The prediction is then forwarded to the retrieval module, which exhibits it in real time. The softmax vector was employed in the output layer of the training module, which transmitted the prediction to the retrieval module, which displayed it to the user. For training procedure we used Keras [5] deep learning framework with Tensorflow [4] in the backend.

The "Retrieval" module captures frames from a live video feed and passes them to the "Model Training" module's trained model in real time. Then it takes the predictions from the trained model and broadcasts them on the user's screen in real time.

Dataset The dataset used for training the model was acquired from Kaggle [2]. The dataset consists of 5988 images belonging to two categories. The classes are "with mask" and "without mask". Each class consists of 2994 images. Following a study of the data samples, we realized that there were enough samples for model training but not enough samples to identify occlusion. Then, to deal with occlusions we added some more samples. We created the added samples by our own to defend the occlusion. We used 2 videos to extract frames for samples. We used "Free video to JPG Converter" [12] to extract the frames. Thus resulting the total dataset consists of 6200 samples. Then we split the resulting dataset into train, test and validation datasets. As a result for final training we end up with 4464 training samples, 620 test samples and 1116 validation images all belonging to two different categories.

3.1 Data Preprocessing

The dataset we used to train our model had 6200 images after finalizing it. In both classes, all of the samples were 128*128 pixels in size. The samples were color images. They had three channels: red, green, and blue. The dataset folder had two sub-folders to denote the classes. Sub-folders were "with mask" and "without mask". In this preprocessing module we first retrieved the subfolder categories and added numerical labels to them. Our "with mask" class had a label of 0 and our "without mask" class had a label of 1. The samples were then converted from color to grayscale images. The channel size was lowered from three to one, and our feature vector contained far fewer elements than color images. The feature vector for a single 128*128 color image has 49152 elements. A single 128*128 grayscale image shrinks the feature vector element size from 49152 to 16384, which is three times less than the color image. Using the cv2.COLORBRG2GRAY function from OpenCV [3], we transformed the color image to grayscale. We inserted the numerical values of gray images to NumPy

[11] arrays. We added the equivalent target vector to our feature vector. Then finally saved the feature and target vectors for the "Model Training" module. Fig. 1 reflects the visual representation of our "Data Preprocessing" module.

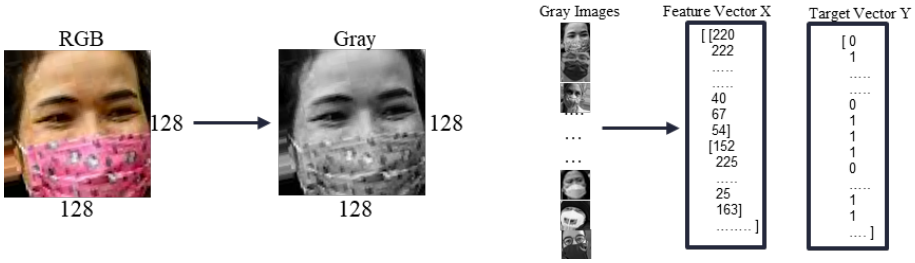

Fig. 1. Illustration of data preprocessing

3.2 Model Training

Our model training was carried out following keras deep learning framework with tensorflow on the backend. In convolutional neural network [13], the convolutional layers together with the max-pooling layers can cope up with the random distortion and translation. Additionally, the max-pooling layers can simplify the process by offering abstraction for the objects in the pooling. The input layer was set of grayscale images all resized in 128X128 pixels in one single channel. The input layer is then processed with two consecutive pairs of convolution and max pooling layers. Where the first convolutional layer had 200 filters, which are of size 3x3 and the max pooling layer filter is of size 2x2. The first convolutional layer was the same convolution. Then our second convolutional layer was a valid convolution. It had 100 filters, which are of size 3x3 and the max pooling layer filter is of size 2x2. Between each convolution and max pooling, we incorporate Rectified Linear Unit(ReLU) as the activation function. After the second max-pool layer we introduced a layer in the network to flatten the output of the last max-pool layer then passed it onto the first fully connected dense layer. The first dense layer had 50 neurons. The activation function was the ReLU. Then we had the second dense layer with activation function Softmax before the output layer. During the network training we used Adam optimizer, categorical loss and took Accuracy as our cross entropy metric to train the final network. Fig. 2 illustrates the abstraction of our final trained network.

3.3 Retrieval

In the retrieval module we use the pretrained deep neural network classifier that we trained and saved in the previous training module to compare and classify unseen frames provided by the user. This module captures the real-time video

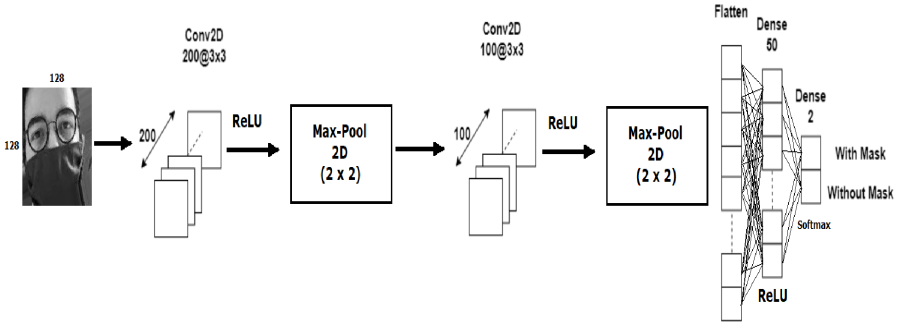

Fig. 2. Illustration of the Trained Network

frames as the user input. Then it passes those frames to the loaded pre-trained classifier. Each frame is then subsequently classified by the pre-trained classifier, which delivers the results back to this retrieval module. The numerical class values are then decoded, and the real text label is added to each frame by this module. Then the labeled frame is shown in real-time as predicted results for each frames. In the output frame the faces are detected in a rectangle box and the predictions are displayed in text forms. This module uses OpenCV [3]to capture video frames from input device and also to display the output on user's screen. Fig.4 illustrates the abstraction of the retrieval module of our sketch-based image retrieval system.

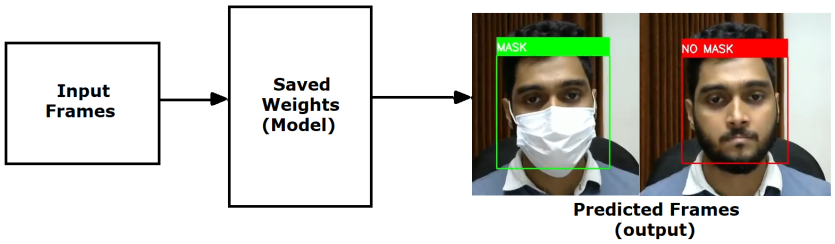

Fig. 3. Illustration of the Retrieval Module

4 Results

Although the validation accuracy during the training phase was relatively low compared to the train accuracy, the trained model performs well in terms of predicting with or without mask images in retrieving real images from prediction. After experimenting with the retrieval module with the trained classifier, we found out that predicting exact class from frames does perform well but not at the level of training phase, but eventually it gets the job done.

Fig.4 shows the training accuracy and training loss during the final model training phase. It can be concluded that the training accuracy increases over time with each increasing epochs whereas the validation accuracy does not perform likewise. On the other hand, the training loss decreases as the number of training epoch increases that corresponds to increasing training accuracy whereas the validation loss is relatively high due the same effect in the validation phase of the model training.

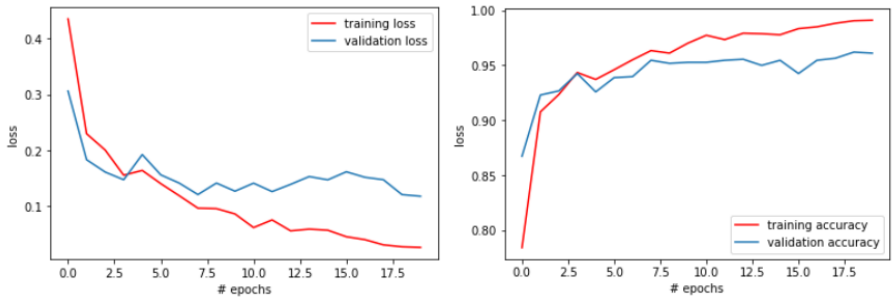

Fig. 4. Model training learning rate. The left figure shows the loss(training and validation) rate in training time per epoch and the right figure shows the accuracy rate both in training and in validation phase per epoch

4.1 Analysis

The reason behind using 20 epoch is that after 20 epochs the validation accuracy wasn't increasing much moreover starts decreasing. So, training was stopped. we chose model 20 as it was the best outcome in the manner of accuracy and loss differences between train sets and validation sets. Putting the optimal Bayes error as 0.01 percent. Comparing with the data from table 1, we can see the difference between train error and dev error is much higher than the gap between Bayes error and train error. Same goes for accuracy data. this is a sign of overfitting. So the system has high variance.

After being satisfied with training set and dev set, we tried our model on test set and the result of accuracy and loss is 95.33 percent and 13.55 percent following. In table 2 we can see the accuracy rate in each data set.

Model	Train	Dev	Difference
Accuracy	99.10	96.10	3.00
Loss	2.62	11.78	9.16

Table 1. Train and dev set results and deference

Sets	Train	Dev	Test
Accuracy	99.10	96.10	95.33

Table 2. Accuracy in each data set

5 Limitations

The technology created can identify live video streams but does not maintain a record. The admin can't replay, play or pause the CCTV camera recording. As when people are continually trying to violate a rigid system. Therefore, the head of the organization may be alerted via email if a person is identified without a mask that this individual entered without a mask. In order to record the individual who entered without masking, the suggested system may be incorporated in the databases of each company. A snapshot of the face of the individual can also be included to demonstrate more complicated functions. We also have several areas in which to work. It can be built to detect whether or not someone wears masks correctly. Many people, for example, wear masks bellow the nose. Our model cannot identify whether or not the mask is susceptible to the virus. Many individuals, for instance, use fashionable masks that won't function well against virus.

6 Conclusion

We discussed the motivation of the work briefly in this article first. Then we showed the model's learning and achievement challenge. The approach has obtained relatively good precision with use of Python, Keras, OpenCV Tensorflow. It may be used for a number of uses. In view of the Covid 19 problem, wearing a mask may be mandatory in the near future. Many public service providers may advise their clients to use their services appropriately using masks. The model used will make an enormous contribution to the public health system. It may be expanded in future to identify whether a person wears the mask correctly. The model can be further improved to detect if the mask is virus prone or not for example the type of the mask is surgical, N95 or not. The model we implemented is getting around 96 percent accuracy. We are satisfied with our current model and still trying to improve it over time.

Source Code Repository The processed dataset, source code for Data Preprocessing, Model Training and Retrieval can be found in the following link.
https://github.com/Ashiqur-Rahman-Tanzil/Face-Mask-detection-with-Convolutional-Neural-Netw

References

1. World Health Organization. Coronavirus disease (COVID-19). https://www.who.int/emergencies/diseases/novel-coronavirus-2019. Last accessed 17-Sep 2021
2. Find open datasets and machine learning projects — kaggle 2021, https://www.kaggle.com/datasets. Last accessed 17 Sep 2021
3. Home - OpenCV", OpenCV, 2021. https://opencv.org/. Last accessed 17 Sep 2021
4. Tensorflow Homepage, https://www.tensorflow.org/. Last accessed 19 May 2019
5. Keras Homepage, https://keras.io/. Last accessed 19 May 2019
6. Are RT-PCR tests failing to detect new Variants? https://shorturl.at/dnGV4. Last accessed 17-Sep 2021
7. A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi, "MaskedFace-Net – a dataset of CORRECTLY/INCORRECTLY MASKED face images in the context of covid-19," Smart Health, vol. 19, p. 100144, 2021.
8. I. B. Venkateswarlu, J. Kakarla, and S. Prakash, "Face mask detection USING MobileNet and Global Pooling Block," 2020 IEEE 4th Conference on Information and Communication Technology (CICT), 2020.
9. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and Linear bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.
10. Chen, Y.; Hu, M.; Hua, C.; Zhai, G.; Zhang, J.; Li, Q.; Yang, S.X. Face mask assistant: Detection of face mask service stage based on mobile phone. IEEE Sens. J. 2021.
11. Numpy.org. 2021. NumPy. https://numpy.org. Last Accessed 17 September 2021.
12. U. SL, "Free Video to JPG Converter (Windows)", Uptodown.com, 2021. https://free-video-to-jpg-converter.en.uptodown.com/windows. Last accessed 17 Sep 2021
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)