Facemask Detection Using CNN to contain the spread of COVID-19

Report Organization:

1. Introduction

The COVID-19 pandemic has wreaked havoc all over the planet. It had an unforeseen consequence, not only economically, but also socially and in terms of human lives. Between January 3rd and February 7th, 2022, there were 75,725,243 confirmed cases of COVID-19 in the United States, with 893,870 deaths recorded to the World Health Organization [1]. Each of the several techniques used to combat the illness's contagiousness has been demonstrated to be crucial. The usage of facemasks is one of the most essential precautionary measures. A study of an epidemic onboard the USS Theodore Roosevelt, which is known for its crowded living quarters and intimate working environs, discovered that wearing facial covers on board lowered the chance of infection by 70% [2]. Many public service providers have mandated clients to wear suitable masks to use their services. Detecting face masks and warning people in the contagious zones can prove to be vital in eradicating the spread of COVID-19. Our project takes live input from the camera and detects facemasks. Boxes are generated around faces and the probability of a person wearing a mask is shown.

2. Data Exploration

Dataset consists of two labels in a folder namely: with_mask and without_mask. With_mask folder contains 3725 images of people wearing masks from different angles. The second folder i.e. - without_mask contains 3828 images. The image title is a text feature in our data frame, with_mask and without_mask. To train, 6043 images are utilized, and there are 1510 images to test. The data exploration phase involved general quality check of the images visually where they were checked if they are corrupt or have any faults such as being black. HTML rendering with python was used to visualize and explore the quality of images. Inspecting all the statistics of the images such as aspect ratios, image heights and widths lead us to make few decisions like resizing the images as the models used required strict input image size. Lastly, the most important part was assigning a class label to every image, with_mask and without mask. There are multiple images with various types of masks; the color of the masks would not matter, but the algorithm would examine the size and form of the mask in the image, as different masks cover the face in various shapes and sizes.

3. Methodology

3.1 Preprocessing

All the images are fetched using OpenCV and Pre-Processed according to the need of the algorithm, these are the steps: -

1. Converting BGR image to RGB - The OpenCV method imread() is used to read the image file, which has default color order as BGR (blue, green, red). In Pillow, which is a Python Imaging Library, on the other hand, the color order is presumed to be RGB (red, green, blue). As a result, converting BGR and RGB is required to use both the Pillow and OpenCV functions.

2. Resizing the image to 160*160*3 - Machine learning algorithms, in general, train faster on smaller pictures. Furthermore, many deep learning models designs demand that our photos have the same size, despite the fact that our raw gathered images may differ in size. Because all photos must be adjusted to a predetermined size before being fed into the Convolutional Neural Networks (CNN), we resize all images to a fixed size before being fed into the CNN. The larger the fixed size, the less shrinking required. Less shrinkage means less distortion of the image's features and patterns.

3. Converting the image into NumPy matrix and rescaling the pixels from 0 to 1 - Normalization calibrates different pixel intensities into normal distribution which makes computation efficient by reducing values between 0 and 1, also making the image look better to the visualizer. As a result, it's important to normalize the pixel values such that each one is between 0 and 1.

4. Converting target labels into one-hot vectors, as the class labels of the images are categorical, and CNN requires numerical form of input data thus one-hot encoding is utilized.

5. Splitting into train set, validation set and test set - The set of data on which the actual training takes place is referred to as training data. By fine-tuning the model after each epoch, validation split helps to enhance model performance. Following the training phase, the test set tells us of the model's ultimate accuracy.

3.2 Mining the data

Face Detector - The Viola-Jones algorithm is used to detect faces; it discovers the position on the colored image after detecting the face on the grayscale image. Viola-Jones draws a box and searches within it for a face. It works based on Haar Feature selection.

We have executed classification of this project using two algorithms, CNN and SVM.

1] **SVM Method -** Algorithm model construction is done in two parts: a) Create Support Vector Classifier and b) With the help of GridSearchCV and parameters grid, create a model. C is the regularization parameter of the error term. In our case, the best suited values are 0.1, 1, 10, 100. Kernel specifies the kernel type to be used in the algorithm. We use ‘rbf’ and ‘poly’ for our algorithm. Gamma is the kernel coefficient for ‘rbf’ and ‘poly’. Values for the same are 0.0001,

0.001, 0.1, 1. Using this method gives decent accuracy of 95% but it takes too long to train the model on this dataset as compared to CNN.

2] **CNN Method -** The VVG16 architecture of CNN has been used to classify the faces with masks. On ImageNet, the transfer learning approach is to add 64 units as a fully connected layer (ReLU activation) and 2 units as an output layer to a pre-trained VGG16 (SoftMax activation). Only the last two layers of model (with mask and without mask) are configured to be trained on our dataset, since all previous levels are pre-trained. The model has been trained on ImageDataGenerator(), which is a type of data augmentation on the dataset. The catergorical_cross-entropy is used as a loss function with ADAM (Adaptive Moment Estimation) as the optimizer. The model has completed 11 epochs with the training set. This method gives an accuracy of 98.41% using the given parameters. The training is much faster than SVM which comes around 32 mins, whereas SVM takes 120 mins. Thus, because of poor execution time of SVM we selected CNN to train our model.

3.3 Model Performance

3.3.1 Performance measures

The performance measures used in the project are accuracy, confusion matrix and classification report.

3.3.2 Meaning of the performance measures

A. Confusion matrix - A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. A confusion matrix is a technique for summarizing the performance of a classification algorithm particularly.

True Positive (TP): It is predicted to be positive, and it is true.
True Negative (TN): It is predicted to be negative, and it is true.
False Positive (FP): It is predicted to be positive and is false.
False Negative (FN): It is predicted to be negative and is false.

B. Accuracy - Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy is the number of correct predictions divided by the total number of predictions. For binary classification, accuracy can also be calculated in terms of positives and negatives as follows: Accuracy = TP +TN/TP+TN+FP+FN

C. Classification report - The classification report displays the precision, recall, F1, and support scores for the model.

Precision — Precision is the ability of a classifier not to label an instance positive that is negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive.
Precision = TP/ (TP + FP)

Recall — Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.
Recall = TP/(TP+FN)

F1 score — The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.
F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Support — Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process

3.3.3 Motivation behind choosing performance measures

Almost all the performance metrics are based on Confusion Matrix and the numbers inside it so, the confusion matrix is necessary to calculate other metrics. It is the most fundamental metric to calculate the performance measure of a classification algorithm. The accuracy is the most common metric used for classification accuracy. Classification accuracy involves first using a classification model to make a prediction for each example in a test dataset. The predictions are then compared to the known labels for those examples in the test set. Accuracy is then calculated as the proportion of examples in the test set that were predicted correctly, divided by all predictions that were made on the test set. But this accuracy is not the accurate way of measuring performance of the model. When the skew in the class distributions is severe, accuracy can become an unreliable measure of model performance. Thus, a classification report is used which calculates other performance metrics to support the accuracy.

3.3.4 Estimate of performance measures

The performance measures were estimated using the percentage split method. The data was split into training and testing dataset where training data consisted of 80% and testing 20%. The testing data set was further split into a validation dataset. The validation dataset is used to tune the hyper parameters of the classifier.

3.3.5 Performance of different modes

	Precision	Recall	F1-score	Support
0	0.98	0.99	0.98	387
1	0.99	0.98	0.98	369
Accuracy			0.98	756
Macro Avg	0.98	0.98	0.98	756
Weighted Avg	0.98	0.98	0.98	756

Recall, Precision, F1-score and support are generated for both the classes. Overall accuracy comes out to be 98%. The metrics are calculated by using true and false positives, true and false negatives.

4. Conclusion

We have used CNN algorithm to train models. Performance is compared with SVM based on various parameters. SVM involves a lot of training, and we'll be dealing with an ensemble of classifiers that don't know anything about each other. Other than that, SVM takes 30 mins on an average to train 400 images, while CNN trains 7000 images in the meantime of 45 mins which is way faster than SVM. These are some of the patterns we observed in different models. SVM gave accuracy of 95% which is more or less like CNN which is 98.41%. Thus, both the algorithms are similar, but SVM takes much more time which is why we have chosen CNN. *When CNN and SVM are compared, each has advantages and disadvantages. CNN is particularly suited to image recognition. CNN can certainly be used for sequence data, but they excel at sifting through massive amounts of picture data and detecting non-linear relationships. SVMs are margin classifiers that may be used with a variety of kernels to perform these classifications. When the size of the class labels is large, SVM struggles to forecast them. Furthermore, SVM is difficult to parallelize, whereas the CNN design supports parallelization by default.

As this pandemic became the new normal it is important to wear a face mask and make sure whether people around us wear a face mask. The problem is people not wearing masks even though the rules are implied. It is a basic yet effective strategy to help control the spread of the coronavirus. Our model gives an excellent accuracy of 98.4%. It is evaluated using various performance parameters like accuracy, confusion matrix and classification report. The model can be improved by giving a variety of image data. The images which are given as input to the model for learning should be included with different types, colors of masks. *If this project is to be continued, a complete system can be developed for video analytics with an improved set of models and their optimizations. These systems can take feeds from various CCTV cameras and face mask detection can be applied. As our project works on only one camera feed, feeds from multiple cameras can be accommodated in the future with a higher frame rate. In addition to this more models can be trained with respect to the power of resources of underlying machines used for this system. Toolkits like Intel’s OpenVino can be used for non-GPU machines to enhance the performance and serve a higher FPS. And Nvidia’s CUDA can be used for the machines who house a GPU to enhance performance and throughput. This project can be applied with edge analytics also, where the computing is done on the camera itself. This project has a tremendous potential to be used by many companies who have their own cameras and Video Management Servers/Services.

5. Appendix:

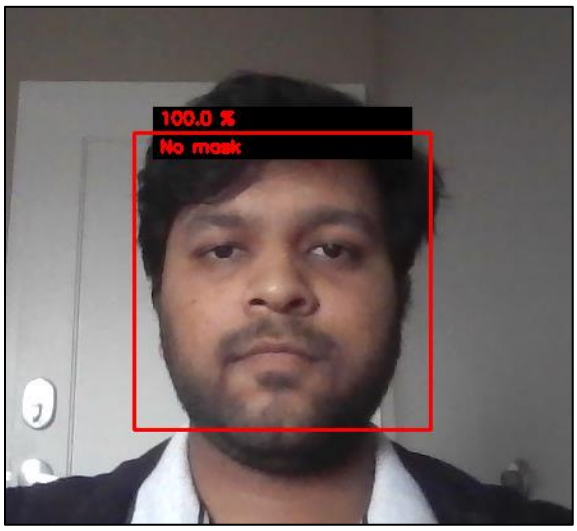

Fig1. Face detected and classified without mask

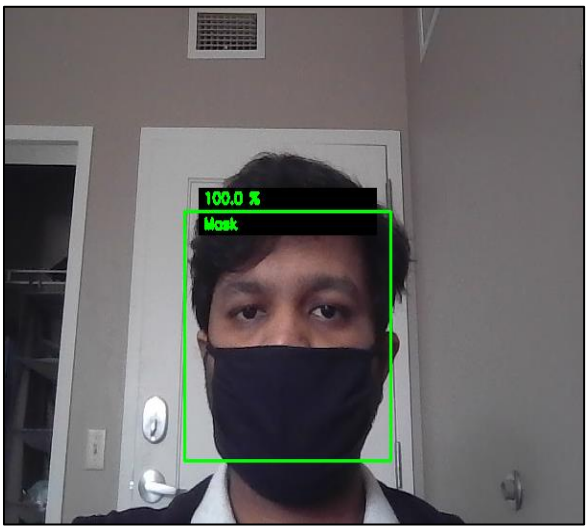

Fig2. Face detected and classified with mask

```
loading... category : with_mask
loaded category:with_mask successfully
loading... category : without_mask
loaded category:without_mask successfully
Splitted Successfully
The Model is trained well with the given images
The predicted Data is :
[0 1 0 0 1 0 0 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 1 0 0 0 0 1 1 0 0 0 1 1
 0 1 1 0 1 0 0 0 1 0 1 0 1 1 1 0 0 1 0 1 0 0 0 0 1 0 1 0 0 1 1 1 1 0 1 1 1
 0 1 0 1 0 0]
The actual data is:
[0 0 0 0 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 0 0 0 0 1 1 0 0 0 1 1
 0 0 1 0 1 0 0 0 1 0 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0 1 0 0 1 1 1 1 0 1 1 1
 0 1 0 1 0 0]
The model is 95.0% accurate
```

Fig3. Output of SVM Model

```
[[355   7]
 [ 10 384]]
```

Fig4. Confusion Matrix

	precision	recall	f1-score	support
0	0.97	0.98	0.98	362
1	0.98	0.97	0.98	394
accuracy			0.98	756
macro avg	0.98	0.98	0.98	756
weighted avg	0.98	0.98	0.98	756

Fig5. Classification Report

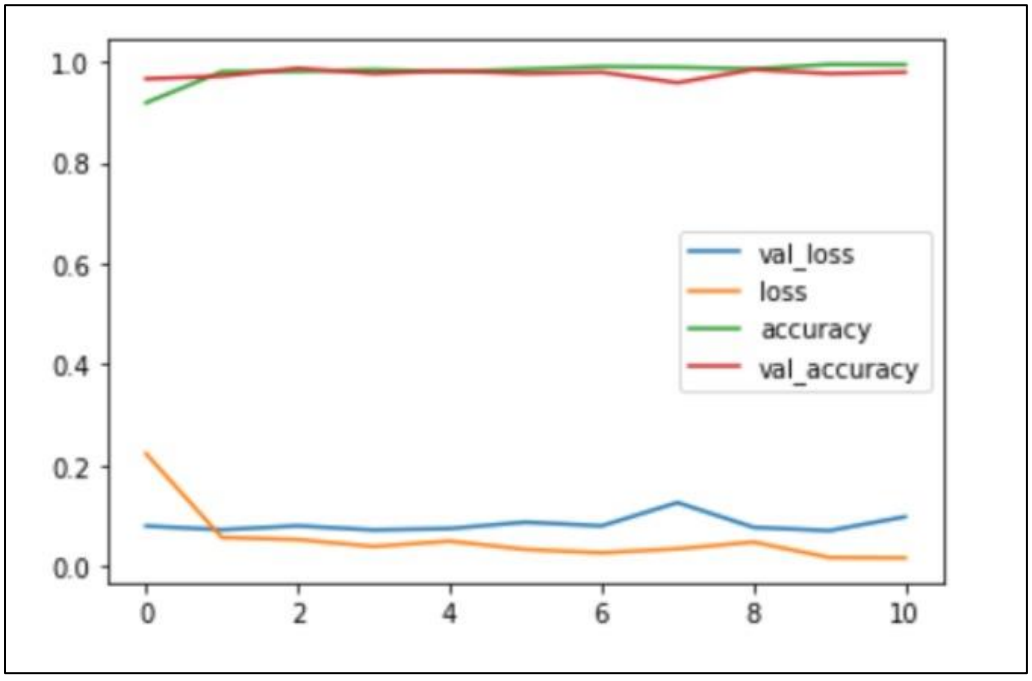

Fig6. Accuracies and losses plotted for CNN

6. References:

[1] *United States of America: WHO Coronavirus Disease (COVID-19) Dashboard With Vaccination Data*. (n.d.). Retrieved February 7, 2022, from https://covid19.who.int

[2] CDC. (2020, February 11). *Coronavirus Disease 2019 (COVID-19)*. Centers for Disease Control and Prevention. https://www.cdc.gov/coronavirus/2019-ncov/science/science-briefs/masking-science-sars-cov2.html

[3] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. (2016). Data Mining: Practical Machine Learning Tools and Techniques. Elsevier Science & Technology. http://ebookcentral.proquest.com/lib/rit/detail.action?docID=4708912

[4] Zhang, D. (2019). Fundamentals of Image Data Mining: Analysis, Features, Classification and Retrieval. Springer International Publishing. https://doi.org/10.1007/978-3-030-17989-2

[5] Introduction to Data Mining. (n.d.). Retrieved April 18, 2022, from https://www-users.cse.umn.edu/~kumar001/dmbook/index.php

YouTube link to the presentation: https://www.youtube.com/watch?v=fPjFgNWfnVQ