IE-406 Machine Learning

Face Mask Detection

Mentor

Prof. Manjunath V. Joshi

Group - 41

Ishang Kumar(201801071)

Abhishek Solanki(201801088)

Paresh Chauhan(201801200)

Aditya Goyal(201801253)

Introduction

In the current times, people can't think of going in public places without the face mask, due to the COVID. Our aim in this project, as the name suggests, is to prepare a Machine Learning model that can help the users in detecting whether people are wearing face masks or not, and also if the mask is worn incorrectly. Through this model users can identify and segregate the different types of people on the three bases,

1. Wearing a Face Mask,
2. Wearing a Face Mask incorrectly, and
3. Not Wearing a Face Mask, by simply testing the image through the model.

Motivation

During the dire situation of the COVID pandemic, it is necessary to wear the face mask for the safety of everyone, but many people don't follow the regulations and choose not to wear or wear the face mask according to their comfort, or wear it improperly. In the public places with a large footfall, it is very hard to detect and segregate such people from the crowd. So to tackle such a situation the Face Mask Detection Model can help the authorities by identifying and segregating people who are wearing the face mask improperly or not wearing the mask, from the ones who are wearing.

Problem Statement

The project will deliver a model to tackle a situation such as improper position of masks on face, and to help authorities segregate them from the ones wearing them properly.

As for the input, we will take an image.

After passing the input image through our system, one can categorize images into 3 sets as mentioned above.

Datasets

We will be using the dataset on the Kaggle website description of which is as follows:

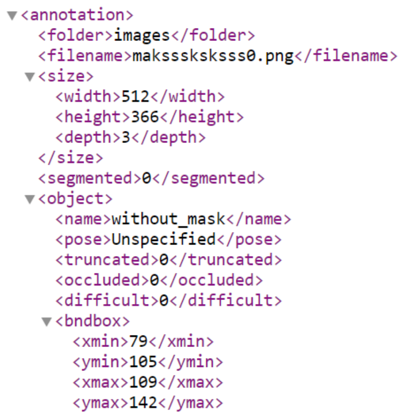
Andrewmvd, 2020-05-22, Face Mask Detection, version 1, from Face Mask Detection

The Dataset for this Face Mask Detection model consists of several images with bounding boxes, these boxes are labeled in three different classes. The classes are:

- With mask
- Without mask
- Mask is worn incorrectly

Our dataset contains 853 images belonging to these 3 classes, as well as their bounding boxes in the PASCAL VOC format. The images in the dataset contain multiple or single people in it.

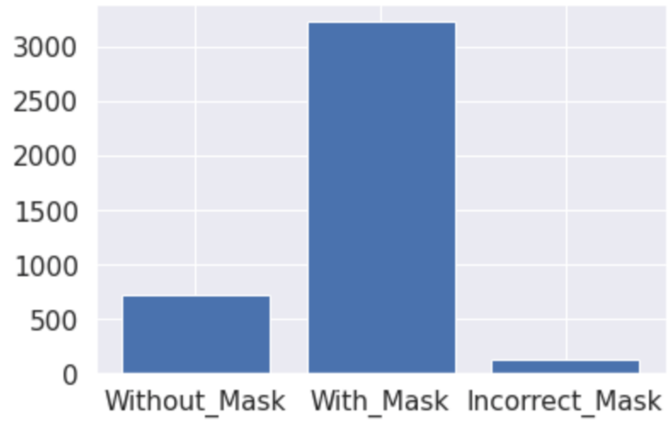
Below is an example of image from dataset along with the corresponding XML file:



```
▼<annotation>
   <folder>images</folder>
   <filename>maksssksksss0.png</filename>
 ▼<size>
     <width>512</width>
     <height>366</height>
     <depth>3</depth>
   </size>
   <segmented>0</segmented>
 ▼<object>
     <name>without_mask</name>
     <pose>Unspecified</pose>
     <truncated>0</truncated>
     <occluded>0</occluded>
     <difficult>0</difficult>
   ▼<bndbox>
       <xmin>79</xmin>
       <ymin>105</ymin>
       <xmax>109</xmax>
       <ymax>142</ymax>
```

The 853 images in total consist of 4072 faces, of these faces 3232 are with masks, 717 are without masks, and 123 are with incorrect masks.

```
{0: 717, 1: 3232, 2: 123}
```



Methodology

Preprocessing Data

Preprocessing consists of the following steps.

1. Finding coordinates of the faces, and the number of faces in each image
2. Extracting faces from the image and their corresponding label.
3. Crop the image and store the faces and their labels.
4. Resizing image and make image array
5. Encoding labels in one hot encode form
6. Performing data augmentation using ImageDataGenerator
   Image data augmentation is the most well-known type of data augmentation and involves creating transformed versions of images in the training dataset that belong to the same class as the original image.
7. Splitting the data into training and testing set

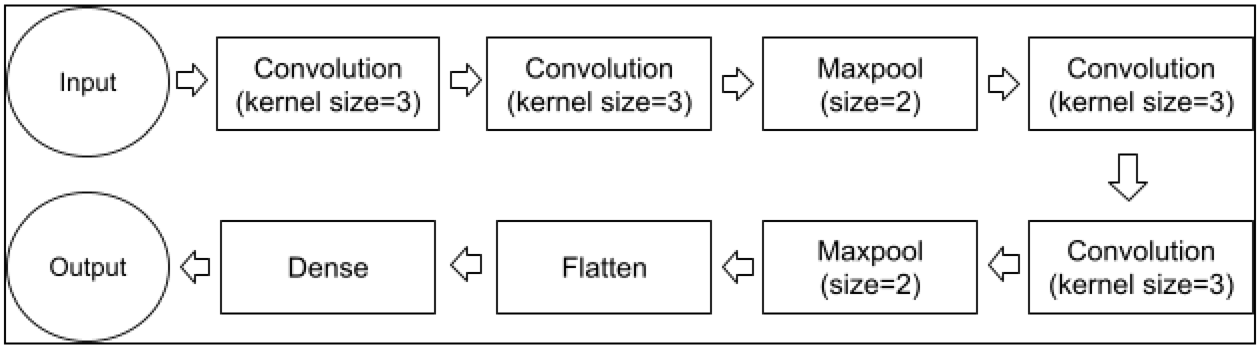
Implementation

The implementation of the dataset can be divided into two halves:

1. Model Training
2. Data Prediction

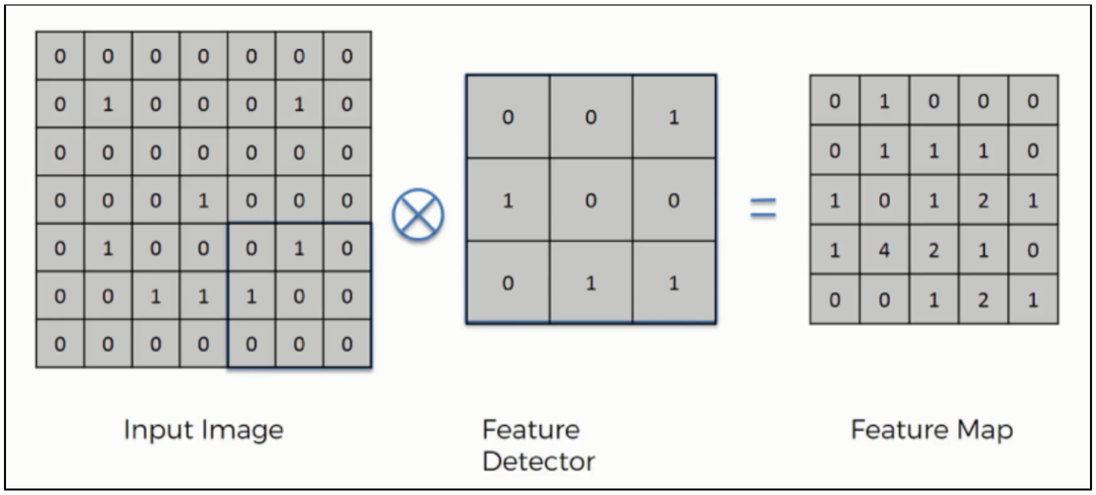
Model Training

The model used for training is the Keras Sequential model, in this model we can add multiple layers of our choice, the more the layers the deeper is the model. Our model is a 8 layer sequential model. Initially there are two convolution layers with kernel size of 3, followed by a max pool layer of size 2, again followed by two convolution layers with kernel size of 3, followed by maxpool layer, then finally for the final classification mapping it is followed by a flatten layer and then dense.


Input
Convolution (kernel size=3)
Convolution (kernel size=3)
Maxpool (size=2)
Convolution (kernel size=3)
Convolution (kernel size=3)
Maxpool (size=2)
Flatten
Dense
Output

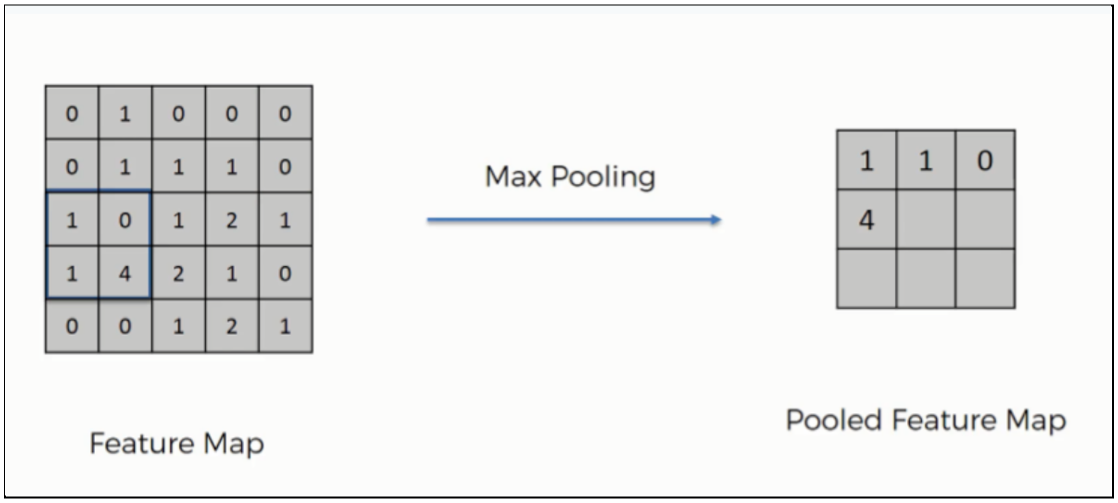
Convolution

In convolution we take a matrix of kernel size and then multiply it with the whole matrix starting from top-left and then moving towards right. Then the data is passed for further layers.



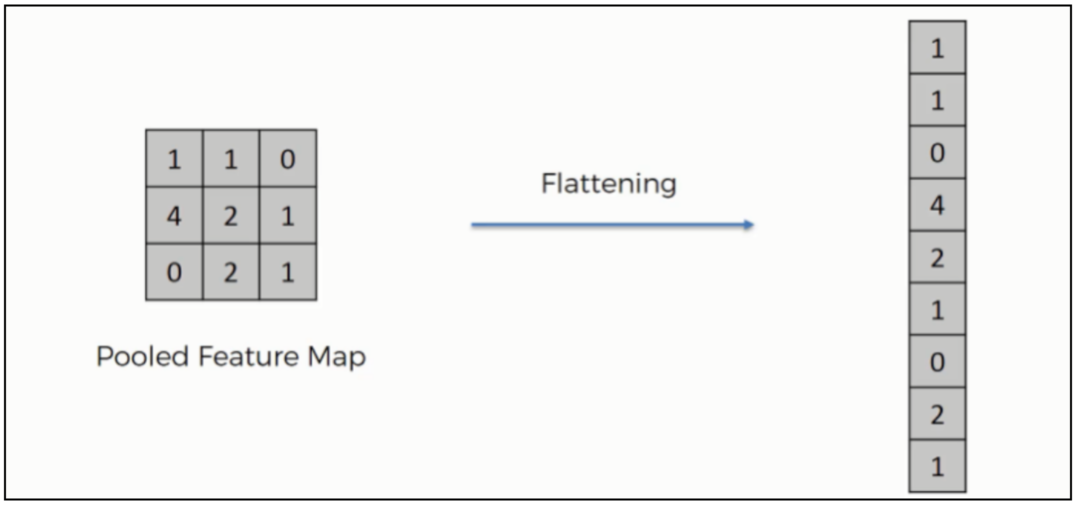
Maxpool

In a max pool the max of the square matrix of the given size is assigned to the particular cell, this significantly reduces the size of the matrix.



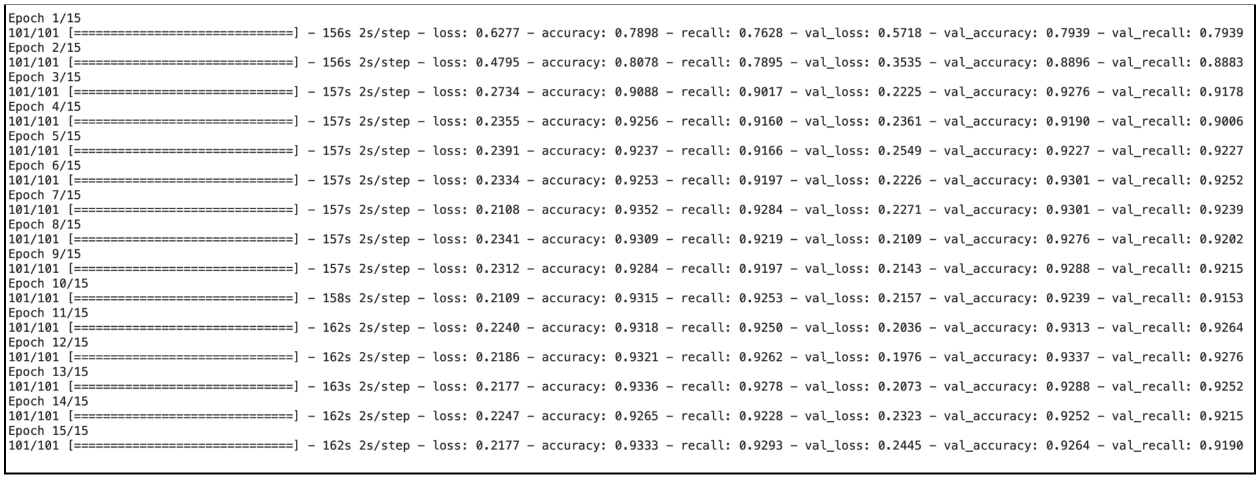
Flatten

Flatten simply converts the matrix data in linear form.



After preparing the sequential model we compile the model to make it useful for our prediction. For compilation we use ADAM optimizer along with accuracy and recall as the metrics.

Post compilation we run our model for 15 epochs, with the augmented data by ImageData Generator. The result of each epoch are as follows:



```
Epoch 1/15
101/101 [==============================] - 156s 2s/step - loss: 0.6277 - accuracy: 0.7898 - recall: 0.7628 - val_loss: 0.5718 - val_accuracy: 0.7939 - val_recall: 0.7939
Epoch 2/15
101/101 [==============================] - 156s 2s/step - loss: 0.4795 - accuracy: 0.8078 - recall: 0.7895 - val_loss: 0.3535 - val_accuracy: 0.8896 - val_recall: 0.8883
Epoch 3/15
101/101 [==============================] - 157s 2s/step - loss: 0.2734 - accuracy: 0.9088 - recall: 0.9017 - val_loss: 0.2225 - val_accuracy: 0.9276 - val_recall: 0.9178
Epoch 4/15
101/101 [==============================] - 157s 2s/step - loss: 0.2355 - accuracy: 0.9256 - recall: 0.9160 - val_loss: 0.2361 - val_accuracy: 0.9190 - val_recall: 0.9006
Epoch 5/15
101/101 [==============================] - 157s 2s/step - loss: 0.2391 - accuracy: 0.9237 - recall: 0.9166 - val_loss: 0.2549 - val_accuracy: 0.9227 - val_recall: 0.9227
Epoch 6/15
101/101 [==============================] - 157s 2s/step - loss: 0.2334 - accuracy: 0.9253 - recall: 0.9197 - val_loss: 0.2226 - val_accuracy: 0.9301 - val_recall: 0.9252
Epoch 7/15
101/101 [==============================] - 157s 2s/step - loss: 0.2108 - accuracy: 0.9352 - recall: 0.9284 - val_loss: 0.2271 - val_accuracy: 0.9301 - val_recall: 0.9239
Epoch 8/15
101/101 [==============================] - 157s 2s/step - loss: 0.2341 - accuracy: 0.9309 - recall: 0.9219 - val_loss: 0.2109 - val_accuracy: 0.9276 - val_recall: 0.9202
Epoch 9/15
101/101 [==============================] - 157s 2s/step - loss: 0.2312 - accuracy: 0.9284 - recall: 0.9197 - val_loss: 0.2143 - val_accuracy: 0.9288 - val_recall: 0.9215
Epoch 10/15
101/101 [==============================] - 158s 2s/step - loss: 0.2109 - accuracy: 0.9315 - recall: 0.9253 - val_loss: 0.2157 - val_accuracy: 0.9239 - val_recall: 0.9153
Epoch 11/15
101/101 [==============================] - 162s 2s/step - loss: 0.2240 - accuracy: 0.9318 - recall: 0.9250 - val_loss: 0.2036 - val_accuracy: 0.9313 - val_recall: 0.9264
Epoch 12/15
101/101 [==============================] - 162s 2s/step - loss: 0.2186 - accuracy: 0.9321 - recall: 0.9262 - val_loss: 0.1976 - val_accuracy: 0.9337 - val_recall: 0.9276
Epoch 13/15
101/101 [==============================] - 163s 2s/step - loss: 0.2177 - accuracy: 0.9336 - recall: 0.9278 - val_loss: 0.2073 - val_accuracy: 0.9288 - val_recall: 0.9252
Epoch 14/15
101/101 [==============================] - 162s 2s/step - loss: 0.2247 - accuracy: 0.9265 - recall: 0.9228 - val_loss: 0.2323 - val_accuracy: 0.9252 - val_recall: 0.9215
Epoch 15/15
101/101 [==============================] - 162s 2s/step - loss: 0.2177 - accuracy: 0.9333 - recall: 0.9293 - val_loss: 0.2445 - val_accuracy: 0.9264 - val_recall: 0.9190
```

Data Prediction

We have used two ways of classification of faces:

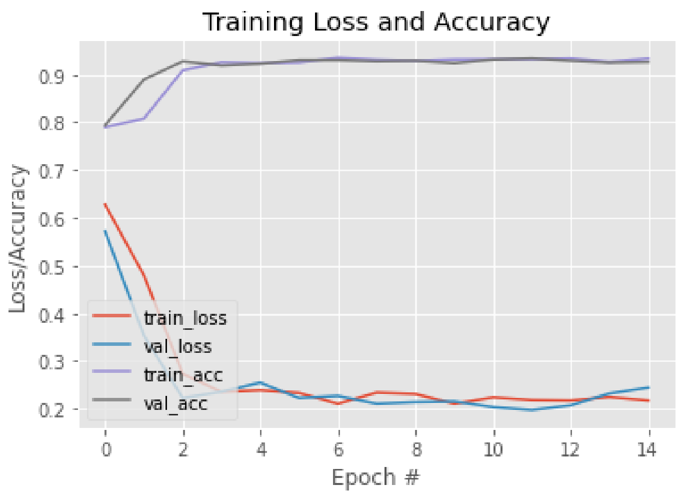
1. Through XML file
2. Through cascade classifier

In XML file, we are given the coordinates of each face in the image(as in the dataset), through this we extract the face and predict it.

In the cascade classifier, we first detect the face with the cascade classifier and then we classify the face in the given categories.

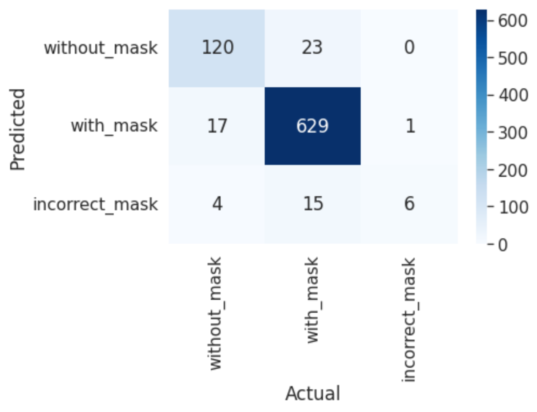
Results and Discussion

While training our CNN model, Training and validation accuracy increase with each epoch, and Training and validation loss decrease with each epoch. This can be seen with below Loss/Accuracy vs Epoches graph:



We got the following result while evaluating/testing the model:

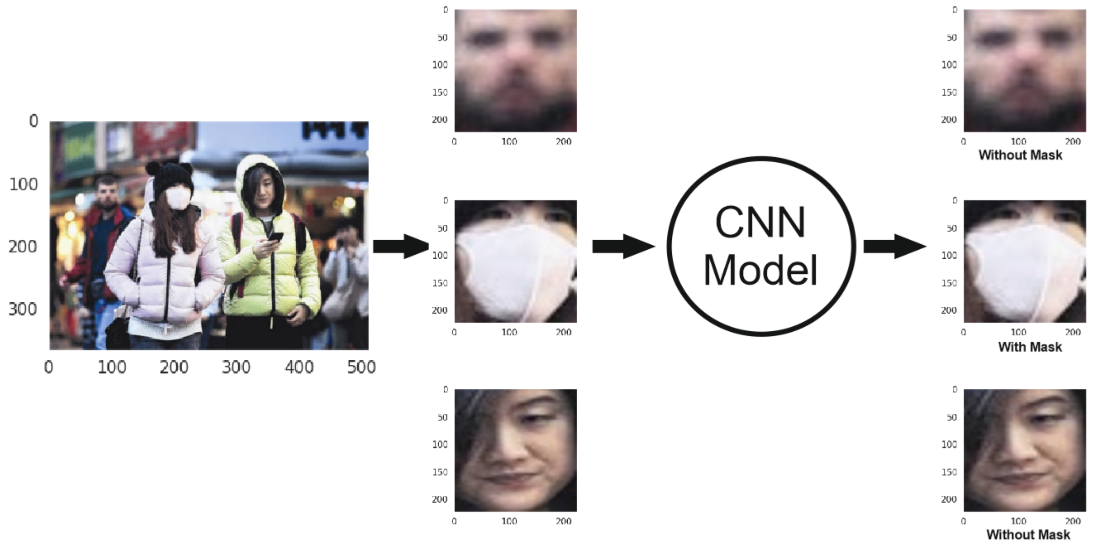
```
              precision    recall  f1-score   support

           0       0.85      0.84      0.85       143
           1       0.94      0.97      0.96       647
           2       0.86      0.24      0.38        25

    accuracy                           0.93       815
   macro avg       0.88      0.68      0.73       815
weighted avg       0.92      0.93      0.92       815
```



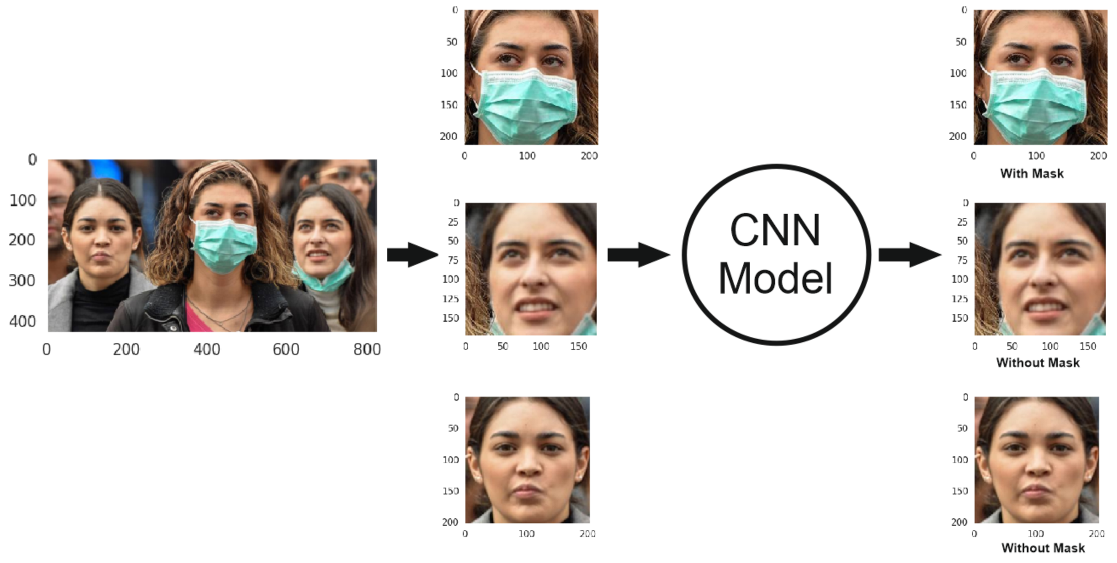
We can see that our model has **93%** accuracy. Also, our model predicts class 0 (Without_mask) with 85% precision, class 1 (with_mask) with 94% precision and class 2 (Incorrect_mask) with 86% precision.

Below are the results for predicting a single image after training our model. First, we have extracted faces from images, after that we passed it to our CNN model and got the predictions:

(1) When face annotation is given in the image:



(2) Using Haar Cascade classifier for extracting faces from an image:



Conclusion

After testing our model for several images, we concluded that our model performs very well in prediction “With Mask” and “Without Mask” faces but sometimes in the case of “Incorrect Mask”, it gives predictions as “With Mask”. This is mainly because of the imbalance in the dataset. This prediction can be improved by adding more images of faces with an Incorrect mask.

References

- Andrewmvd, 2020-05-22, Face Mask Detection, version 1, from Face Mask Detection Dataset
- A. Das, M. Wasif Ansari and R. Basak, "Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV," 2020 IEEE 17th India Council International Conference (INDICON), 2020, pp. 1-5, doi:10.1109/INDICON49873.2020.9342585
- https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn