The George Washington University

Machine Learning 2

DATS 6203
Fall 2022

Instructor: Dr. Amir Jafari

Face Mask Detection

Authors :
Mahtab Barkhordarian

Submission Date:
December 13, 2022

Introduction:
The CDC continues to monitor the spread of COVID-19 and advises people who are completely vaccinated as well as those who are not fully vaccinated to wear face masks. When visiting the doctor's office, hospitals, or long-term care institutions, the CDC recommends wearing masks and keeping a safe distance.
Manually monitoring people entering such institutions is tedious and requires a workforce. In this project, we will learn how we can automate this process through deep learning techniques which will automatically detect people not wearing masks to prevent their entry.

Dataset:

We use the dataset from Kaggle. The dataset has 5155 images of people with masks divided into test and train and validation. 75% of data is given to train. The half of the remaining 25% is given to validation and the other half to test . It has three target classes: masked_weared_incorrect, without_mask and with_mask. The dataset is highly imbalanced.

Individual Description :

To improve the result of this project, I have done some preprocessing and creating some models.

Data Preprocessing:

Imbalanced data causes lots of issue , even if you have the accuracy of 90% it doesn't mean that the model working 90% accurate. So we made the dataset balanced in the beginning. Two methods were tried : one is joined the without mask to incorrectly wearing the mask because both were so small and the second method is to do over sampling . with that I wrote the function which could make the data balance.

```
with_mask                3232
without_mask              717
mask_weared_incorrect     122
Name: name, dtype: int64
3232
839
4071
```

Before merging the without mask group with masked wear incorrectly the number of faces which were extracted with mask is 3232, without masks are 717, masks were

incorrectly 122 and after the group without mask and maske-weared-incorrectly groups joined the number is 839. Then after this part we have just two classes with masks and without masks.

Class balancing:

For learning in a balanced way we have added the class_weights argument in model.fit(), which is used to make the model learn more from the minority class. Using the Scikit learn module's compute_class_weight method we calculated the weights required for the balancing train set.

```
compute_class_weight(class_weight='balanced', classes=np.unique(train_generator.classes),y=train_generator.classes)
```

Which gives us the following weights:

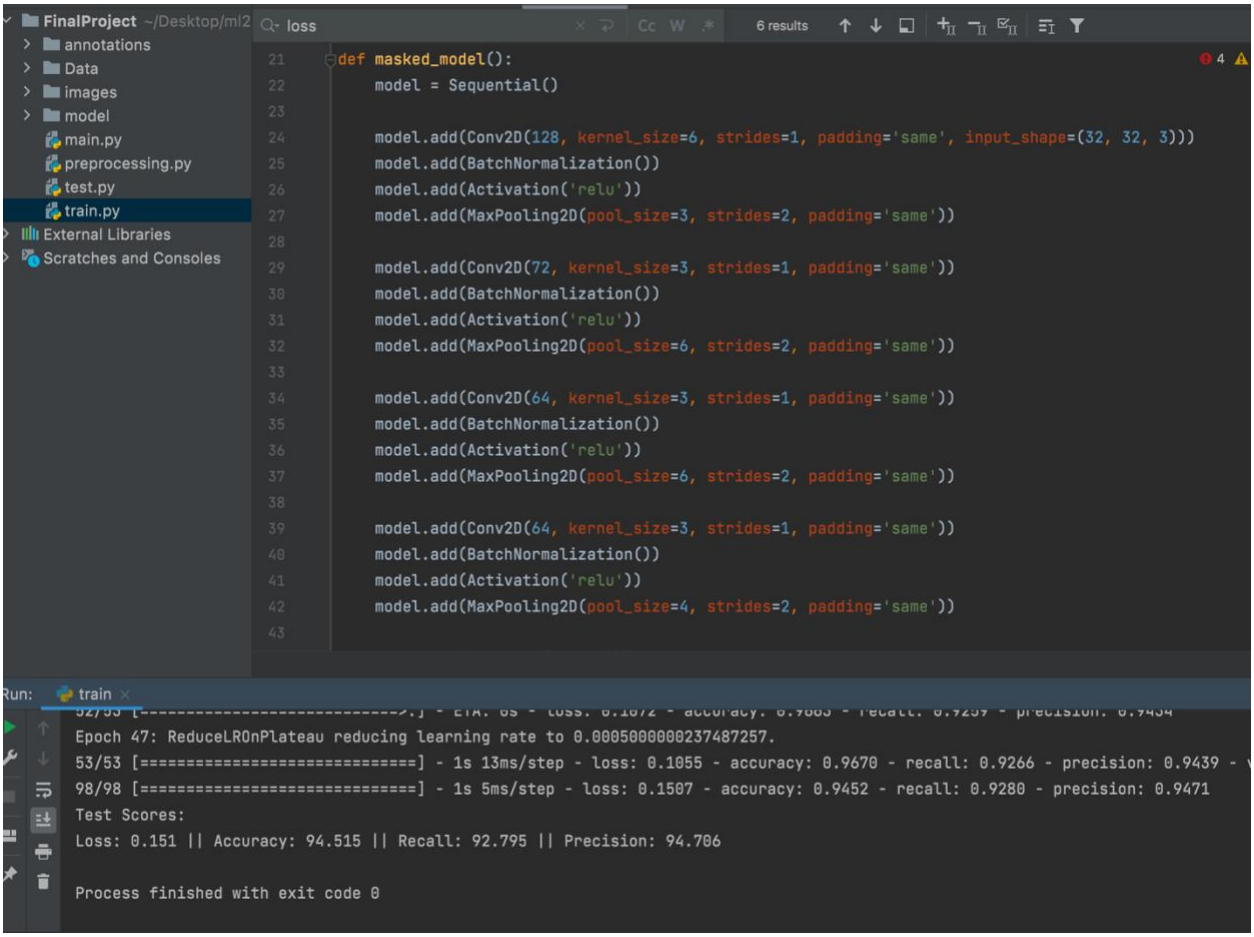
{0: 0.6297872340425532, 1: 2.4262295081967213}

Our dataset has been balanced using class weight.

Here, we can see that class 0 has more data than class 1, thus our model will give this class an estimated weight of 0.63, while class 1 has less data than class 0 and will receive an approximate weight of 2.43. Simply put, the model will take into account around 4 (2.43/0.63) images of class 1 for every 1 image of class 0.

Base Model (CNN):

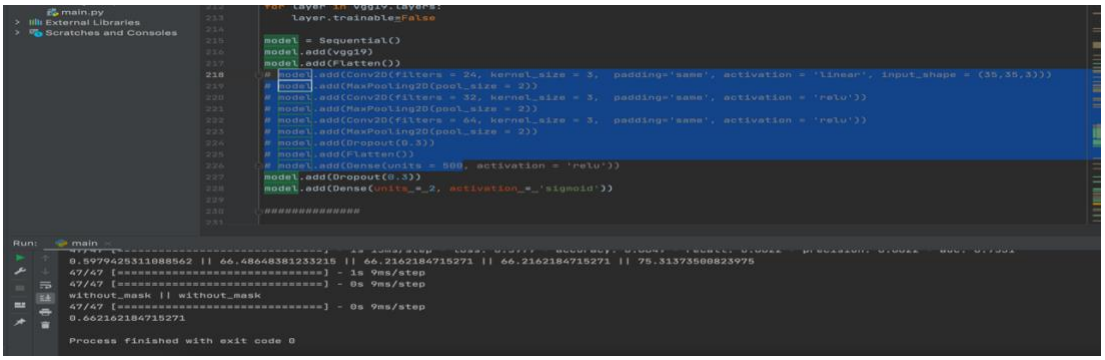
We used CNN which has 4 convolution layers. For each convolution layer we added a Batch normalization layer to catch the instability. For the CNN activation layer we used 'Relu'. In comparison to sigmoid and tanh, ReLU is more reliable and accelerates the convergence by six times. In the input convolution layer we reshaped all the images to 32*32* 3 to be the same size. We got the size from our preprocessing and 3 channels because the data are pictures. We used a 2D convolution layer because the data is picture not text, audio or video. After we were done with the CNN as our model the F1 increased to 88.



The above screenshot was my first CNN model.

I tried 2 CNN model, one with accuracy of 92 and the other with accuracy of 66 for train.

The second one was the highlighted one which had 3 convolution layers, for each layer I had pooling layer and there is one drop out layer to avoid overfitting, then flatten the output and pass it to the dense layer. However because the accuracy with this model was so low, I removed it.



Final Model: CNN+LSTM :

In this study, the structure of this architecture was designed by combining CNN and LSTM networks, where the CNN is used to extract complex features from images, and LSTM is used as a classifier. We have used 4 convolution layers each with batch normalization and activation function of 'ReLU', followed by MaxPooling layer. After that we reshaped the data to feed it into the LSTM model followed by a fully connected layer to get our target output with activation function 'Sigmoid'.

Some other models were tried which the can be observed in the following table.

Result:

Model	Optimizer	Epoch	Batch Size	Learning Rate	Train Accuracy	Test Accuracy
CNN	Adam	100	8	0.001	66	53
CNN	Adam	100	8	0.00001	92	94
CNN+LSTM	**Adam**	**150**	**8**	**0.0009**	**96**	**93.75**
CNN+LSTM	Adam	150	16	0.001	93.23	1.0

Conclusion:
My individual participation in this project made the dataset be balanced as using the Compute_class_weight technique to make the dataset balanced and it increased the F1 score up to 20 percent. After several models used finally the bet model is the combination of CNN+LSTM with Adam optimizer with the train accuracy of 96% and test accuracy of 93.75.

Percentage of code written:

The code which are developed by me got marked by commenting on top of each portion in the individual code part in preprocessing and train file.

100 lines of code developed by me. 30 lines copied and modified.

((294-30)/394)* 100 = 67% (it's approximately)

References:

[1]https://www.kaggle.com/code/chinmayaudupa/face-mask-detection-using-cnn-and-deep-learning

[2]https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network

[3]https://www.upgrad.com/blog/basic-cnn-architecture/

[4]https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[5]https://www.researchgate.net/figure/The-proposed-deep-LSTM-network-with-three-LSTM-layers-and-two-feedforward-layers-For_fig1_283748673

[6]https://en.wikipedia.org/wiki/Long_short-term_memory

[7]https://analyticsindiamag.com/guide-to-tensorflow-keras-optimizers/

Dataset Link: https://www.kaggle.com/datasets/andrewmvd/face-mask-detection

https://becominghuman.ai/building-a-convolutional-neural-network-cnn-model-for-image-classification-116f77a7a236

Reference Code Link: face-mask-detection CNN | Kaggle

Project GitHub Link: https://github.com/NusratNawshin/ML2ImageClassification