COMP 6721

Applied Artificial Intelligence

Project-1 Report

Submitted To: Prof. Dr. René Witte

Team Id: NS_04

Ananya Varsha (40197012) - Data Specialist

Manimaran Palani (40167543) - Evaluation Specialist

William Moses Stalin Jebadoss (40186129) – Training Specialist

1. Dataset:

The dataset used in this project is picked from different reference sites: Kaggle and Google Images [1]. The collected dataset is segregated into a dataset consisting of (1) Person without a face mask, (2) Person with a “community” (cloth) face mask, (3) Person with a “surgical” (procedural) mask, (4) Person with a “FFP2/N95/KN95”-type mask (you do not have to distinguish between them), and (5) Person with a FFP2/N95/KN95 mask that has a valve.

Following is the table representing dataset with number of images collected.

	Number of Images
No Mask	400
Cloth face mask	400
Surgical mask	400
N95 mask	400
N95 with valve	350
TOTAL	1950

Figure 1.1: Table representing Number of Images for each model





2. CNN Architecture

2.1 Characterizing the model

In this case, we used a Convolutional Neural Network to train the model for certain classes. First and foremost, the provided dataset must be preprocessed to ensure that it is balanced and equal in size. Five different datasets were collected for each of the classes and labeled as ‘0’ for the person denoting no mask, ‘1’ for the person with cloth mask, ‘2’ for the person with surgical mask, ‘3’ for the person with N95 mask and ‘4’ for the person with N95 with valve mask.

Now we divide our dataset into two parts: training and testing, with training accounting for 80% of the data and testing accounting for the remaining 20%. To extract features from the images we have used 4 convolution layers with input size of [3*32*32] where 3 is the RGB channels and 32*32 is the height and width of the image. Each of these four convolution layers has a kernel size of (3,3), a stride of (1,1), and a padding of (1,1). The convolution layer is followed by the Batch Norm Layer which make neural networks faster and more stable through adding extra layers in a deep neural network. Its job is to take the outputs from the first hidden layer and normalize them before passing them on as the input of the next hidden layer. After batch normalization, there is an activation function. Here, we used ReLU as the activation function. There are two pooling layers, each after every 2 convolution layers. A filter of (2,2) and a maximum pooling with a step size of 2 were used. Padding helps hold more information around the edges of the image. Dropout functions are used to avoid overfitting the model. Finally, the output of the last pooling layer is flattened and fed as input to a fully connected layer. Neurons in the fully connected layer are connected to the activation in the previous layer. This helps to classify the data into multiple classes.

2.2 Preparing the Data

The dataset created is stored using Pickle file and torch vision is used for transformations that helps transforming the library in PyTorch. After resizing of images into (32,32), functions namely normalization and totensor is applied on the data. Split is tested using K Fold where k refers to the number of groups that a given data sample is to be split into. We have used K Fold because it results in a less biased or less optimistic estimate of the model skill than other methods.

2.3 Loading data using Data Loaders

We have used Data Loaders for parallelizing the data loading process with automatic batching. With a batch size of 32, the neural network model is trained. When the Shuffle attribute is set to true, the model is better trained since it receives data that is not in a repeating pattern.

2.4 Modification using Optimizer Configuration

To modify the attributes of Neural Network, we have used the optimizer function to reduce loses. Adam optimizer is utilized, with a learning rate of 0.001.

2.5 Training the Model

Followed by optimizer configuration we train the data. The number of epochs is set to 10 and for each epoch the data is loaded using Data loader. The model is trained using face_mask_detector model as mentioned in 2.1. Entropy loss is used to calculate the loss value. Optimizer.zero_grad() function is used to set gradients to zero. Back propagation loss is calculated using backward() function. A screenshot is attached below which gives an idea about how CNN model is trained.

```
print("Metrics")
print(metrics_df.mean())
print()
print("Across 10-folds")
conf_mat(fold_confusion_matrix, classes)
```

```
Training Loss after epoch 8 : 17.024662017822266 Accuracy: 90.17%
Training Loss after epoch 9 : 17.27050018310547 Accuracy: 89.63%
Running Fold : 10
Training Loss after epoch 0 : 74.94190216064453 Accuracy: 50.08%
Training Loss after epoch 1 : 47.223594665527344 Accuracy: 69.73%
Training Loss after epoch 2 : 38.45515441894531 Accuracy: 75.47%
Training Loss after epoch 3 : 32.51390075683594 Accuracy: 79.44%
Training Loss after epoch 4 : 28.301776885986328 Accuracy: 82.07%
Training Loss after epoch 5 : 25.180147171020508 Accuracy: 84.43%
Training Loss after epoch 6 : 21.773462295532227 Accuracy: 86.04%
Training Loss after epoch 7 : 19.055694580078125 Accuracy: 87.49%
Training Loss after epoch 8 : 16.73674964904785 Accuracy: 89.86%
Training Loss after epoch 9 : 11.880767822265625 Accuracy: 93.08%

Metrics
accuracy     0.653940
precision    0.673588
recall       0.654705
f-score      0.656636
dtype: float64
```

Fig 2.1: Results after training

3. Evaluation

Following these three steps, we may assess the method or model's performance using the confusion matrix and numerous other matrices. The model is trained for ten epochs in this project. Learning occurs at a rate of 0.0001 in every epoch. For all stages, we used recall, precision, accuracy, and f1-measure to assess performance. We can state that some of the photos have been misclassified due to imbalance data by utilizing a confusion matrix. This suggests that we have a 3rd class (not a human) as a more generic one, where we may not have enough data. Which could be the model for all this muddle and misclassification.

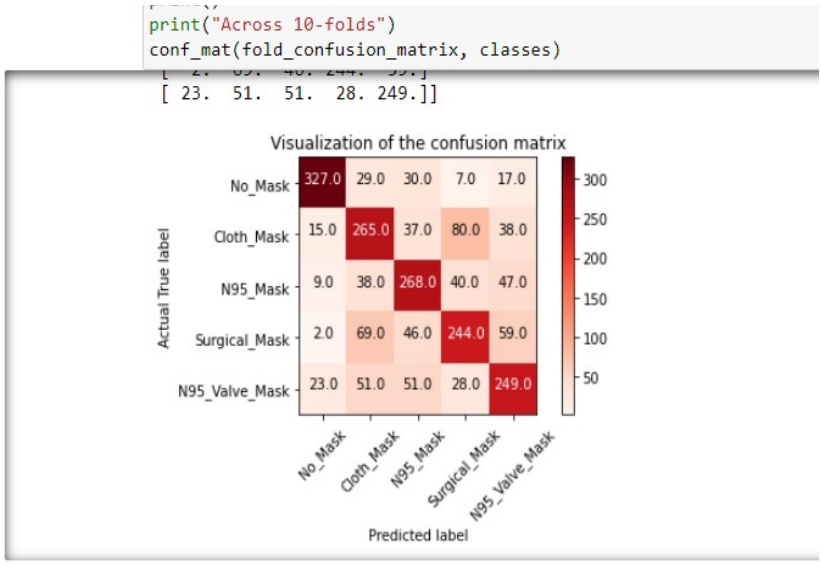

Fig 3.1: Visualization of Confusion Matrix

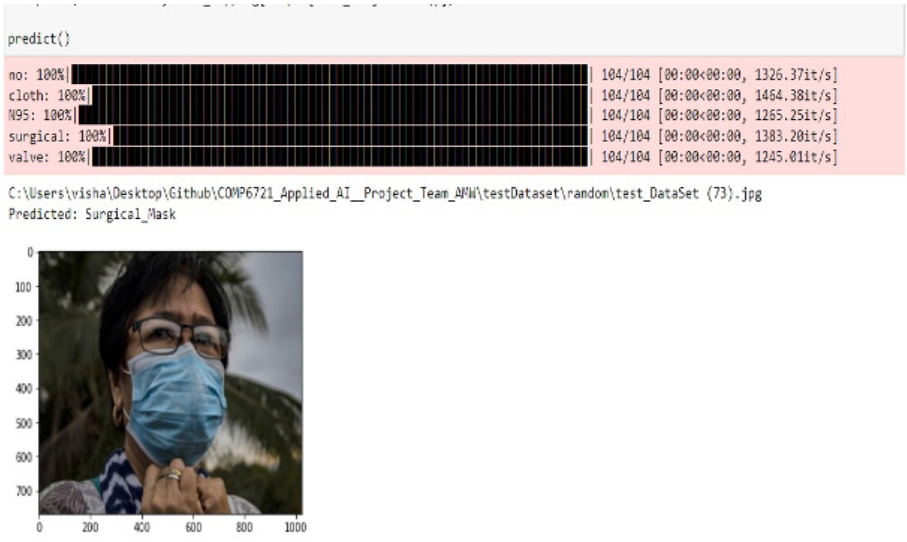

Fig 3.2: Test Result after training images

As previously stated, certain data are misclassified due to imbalanced data for each of the three classes. Humans with masks and without masks can be classified using numerous factors, however the non-human class is very broad, encompassing animals, non-living objects, statues, plants, and so on.

We will enhance the performance by adding more convolution layers as it will extract more features. Also, we can observe from the convolution matrix that the model doesn't predict well for Surgical Mask.

References

[1] https://www.kaggle.com/omkargurav/face-mask-dataset
[2] https://www.kaggle.com/prithwirajmitra/covid-face-mask-detection-dataset
[3] https://www.kaggle.com/sumansid/facemask-dataset
[4] https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset
[5] https://www.kaggle.com/omkargurav/face-mask-dataset
[6] https://humansintheloop.org/resources/datasets/medical-mask-dataset/
[7] https://www.shutterstock.com/search/ffp2?number_of_people=1&mreleased=true
[8] https://www.gettyimages.ca/photos/ffp2-mask?assettype=image&sort=mostpopular&
[9] https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529