COMP6721 Artificial Intelligence - Face Mask Detection

Project Part 2 Report

Under the guidance of
Prof. Rene Witte

Submitted by,
Team name: SS_G07

Team Members:
Nikitha Jayant Bangera - 40088393
Yashika Khurana - 40094722
Tushar Jain - 40094872

November 2020

Contents

List of Figures

Chapter 1

Introduction

The purpose of this Artificial Intelligence project is to develop a face mask detector by applying some of the Artificial Intelligence methodologies. We build a Deep Learning Convolutional Neural Network with the help of PyTorch library and train the created CNN model on three different classes such as:

1. Person without a face mask
2. Person with a face mask
3. Not a person (i.e., any other image)

We collected relevant data-sets i.e. images for training and testing the model, built the Convolutional Neural Network model, trained the CNN model using the collected dataset, and finally evaluated the overall performance of our trained model with the help of different evaluation metrics. We also presented the results to demonstrate the hyper parameters selection for the better prediction on the model.

For a better evaluation of our CNN model, we will be applying K-fold Cross Validation(a re-sampling technique) procedure, in which each data in the given dataset is given a fair chance to act as a training set as well as the validation/test set. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. For this project we have used the value of k as 10[**15**]. The Cross validation technique is very helpful when there limited samples in the given dataset.

Chapter 2

Dataset Creation

Data building and cleaning is important because it helps to improves our data quality, ultimately leading to an increase in the overall productivity of the model. For this project, three different classes are considered, which are as follows,

1. Person with a mask
2. Person without a mask
3. Not a person

Dataset for the category "With Mask" and "Without Mask" are gathered from the Kaggle website (links can be found in the references section [**10**], [**11**], [**12**], [**13**]), and for the category "Not a Person" we chose the CIFAR10 dataset images [**14**]. We stored individual category images in different folders, so in total we have 3 different folders representing each category as mentioned above. We also included some augmented images to train our Model with different angles of the images.

The classes "With Mask" and "Without Mask" consists of different varieties of images, including male, female, child and augmented images. The class "Not a Person" consists of airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck images. The number of images present in each of the three classes are as follows:

1. 'with mask': 6027 images
2. 'without mask': 5978 images
3. 'not a person': 6000 images

We split our Dataset into train, validation and test sets. The train size is 60%, the Test size is 20% and the Validation size is 20% of the total number of datasets where as total data set size is 18000 of images.
During the second phase of the project, as we will be applying the k-fold so we combined our training set as 60% of train set and 20% test set.
The new train set size is 80% as we will be applying the k fold i.e. k as 10 and it will automatically split our data into train and validation. Test size is 20% of the original data set which will be used in the evaluation process of the model.

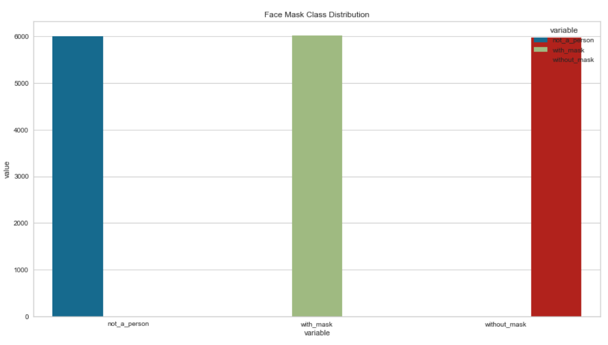


Figure 2.1: Class Statistics figure.

Large datasets are indispensable in the world of machine learning and deep learning these days. However, working with large datasets requires loading them into memory all at once. This leads to memory outage and slowing down of programs.

PyTorch offers a solution for parallelizing the data loading process with the support of automatic batching as well.[**7**] So, we used PyTorch's Dataloader library to load our custom dataset on the disk with the following important arguments:

1. batch_size, which denotes the number of samples contained in each generated batch.
2. shuffle T̄rue, we will get a new order of exploration at each pass (or just keep a linear exploration scheme otherwise). Shuffling the order in which examples are fed to the classifier is helpful so that batches between epochs do not look alike. Doing so will eventually make our model more robust.
3. num_workers, which denotes the number of processes that generate batches in parallel. A high enough number of workers assures that CPU computations are efficiently managed, i.e. that the bottleneck is indeed the neural network's forward and backward operations on the GPU (and not data generation).[**6**]

There are plots in the code to visualize the dataset for each class labels. We sequentially added dataset images, initially we started with 10,000 images samples to train our model and now we have 18,005 images in total. We also tried to train our model with different batch sizes and observed that higher the number of batch size, the less time it takes to train our model.

For the test set, we omitted the batch size because we want to test our images in one go instead of creating the batches, whereas in the training part we kept the batch size as 50. After loading the images, we also normalized the images, which is a part of pre-processing the dataset. Figure 2.2 below shows samples of the transformed images .

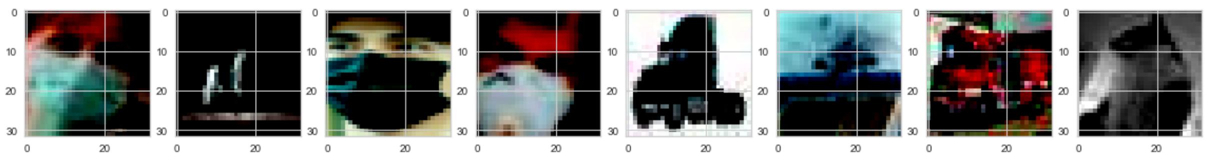


Figure 2.2: Samples of Normalized images.

First Images were loaded from the folder where each folder contains the data of each class labels. After that images were resized to the size of 32*32 and then we applied the normalization on channels i.e. 0.5 and also converted the image into the gray scale of 0.5 for all three channels.

Chapter 3

CNN Architecture

3.1 CNN definition

A Convolutional Neural Network(CNN or ConvNet) belongs to a class of deep neural networks which are used in the analysis of visual imagery.[**2**] A CNN is made up of an input layer, an output layer and a bunch of hidden layers. These hidden layers are formed of convolutional layers which convolves the images to a more abstract feature map.

The role of the ConvNet is to reduce the images into a more abstract form which makes it easier for processing, without losing features which are critical for getting a good prediction.[**3**]

3.2 Model Architecture

The architecture of the CNN used in this project, which is shown in the Figure 3.1, consists of three blocks of Convolutional layers. Each block consists of two Conv2d layers, two Batch Normalization layers, two LeakyReLU activation layers, followed by Maximum pooling layer. The first block creates a feature map of the dimension (32x16x16). The second block takes the (32x16x16) feature map and outputs another feature map of the dimension (64x8x8). The last block takes the (64x8x8) feature map as input, convolves into a better abstract feature map of the dimension (128x4x4). The reason for using Leaky ReLU layer instead of a ReLU is because the former activation function is a more balanced one, as it has a small slope for negative values, which also makes it to learn the data faster.[**4**]

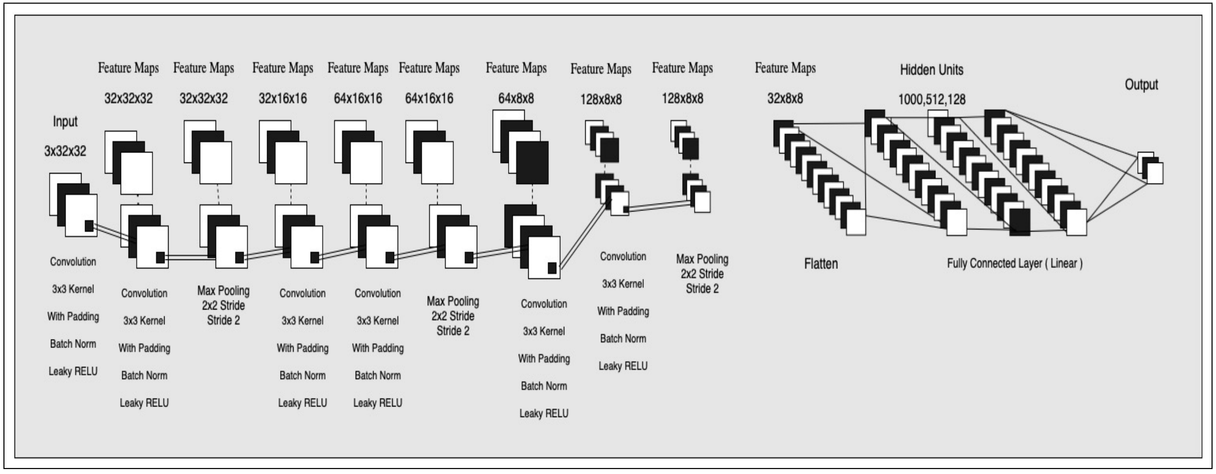


Figure 3.1: CNN architecture

Now the (128x4x4) feature map is passed into a flattening layer so that the feature map is ready to enter the fully connected linear layers. The fully connected layers consist three hidden units, each having a combination of Dropout layer, linear layer and ReLU activation layer. The purpose of using the Dropout layer is to perform regularization of the larger neural network in order to avoid over-fitting the training data. During the model training, some of the layer outputs are randomly ignored, thereby making the active layer treated-like a layer with a different number of nodes and connectivity to the prior layer.[**5**]

3.3 Training the CNN model

In the training phase, our CNN model makes use of the Cross Entropy loss for implementing the loss function. For the optimization of the model, we make use of Stochastic Gradient Descent(SGD) as the optimizer.

According to Wikipedia, SGD is an iterative method for optimizing an objective function with suitable smoothness properties.[**8**] The learning rate value used by the optimizer is 0.001, along with the momentum value of 0.9. The purpose of including a momentum in the SGD optimizer is sometimes the SGD oscillates across the steep slopes, thus slowing down the progress towards the right direction. With the help of momentum, the SGD is made to accelerate in the relevant direction, thereby reducing the oscillations.

During each epoch, the model is set to training mode so that the model learns the features of the training data. Forward propagation on the CNN model takes place, followed by the computation of loss using the Cross Entropy Loss function. Then the model goes through the backward propagation phase to update the weights of each input units. We also calculate the loss in each epoch along with the accuracy computation.

In the first phase of the project, we trained our model for 3, 5, 10, and 15 epochs, in order to evaluate how well the CNN model learns the features from the training data. For each of the 3, 5, 10, and, 15 epochs, the minimum loss value was computed in each epoch as well as the total validation accuracy for the epochs. The loss values and the accuracy values are depicted in the form of plots.

During the second phase of the project, the model we used is same as the project phase 1 but the we trained our model on 2, 3, 5 and 10 epochs with 10 folds. For the epochs, we experimented with different batch sizes in the dataloader and computed the evaluation results. The combination of 10 folds, 10 epochs and batch size 50 gave good results than its previous counterparts. For the 10 folds, The graphs are plotted for the training loss and the validation accuracy respectively.

Chapter 4

K-fold Cross-Validation

4.1 Definition

K-fold Cross Validation is a technique of re-sampling the data, where the samples of the dataset act as training set as well as validation set in each fold. This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining k-1 folds.[**15**] It is a technique used to evaluate the performance of machine learning models. The K-fold Cross Validation is very useful when there is limited dataset available to train the model. We used scikit- learns' implementation for splitting our datasets.

4.2 K-fold Cross Validation steps

Following are the steps involved in evaluating the machine learning model using K-fold Cross Validation:

1. The dataset is shuffled in random order.
2. The entire dataset is decomposed into k groups.
3. Each k group goes through the following steps:
   - Each group is divided into training set and validation/test set.
   - In each group, a small portion is set aside as a validation set and the remaining portion is used as the training set.
   - The machine learning model is trained on the training set.
   - The trained model is evaluated using the validation/test set.
   - Finally, the accuracy score is evaluated and the trained model is discarded.
4. In the last step, the mean of the scores of all the K groups is computed, which shows the performance of the model.

4.3 Results

Figure 4.1 shows the Average training loss during the epochs vs No. of K folds used during the training of the mode. During the training k is taken as 10 where as no. of epochs taken as 10 to do the analysis. We can see that in K fold 1 and K fold 2 our training loss significantly decreased. During k fold 9 and 10 there are very less change in terms of the training loss.

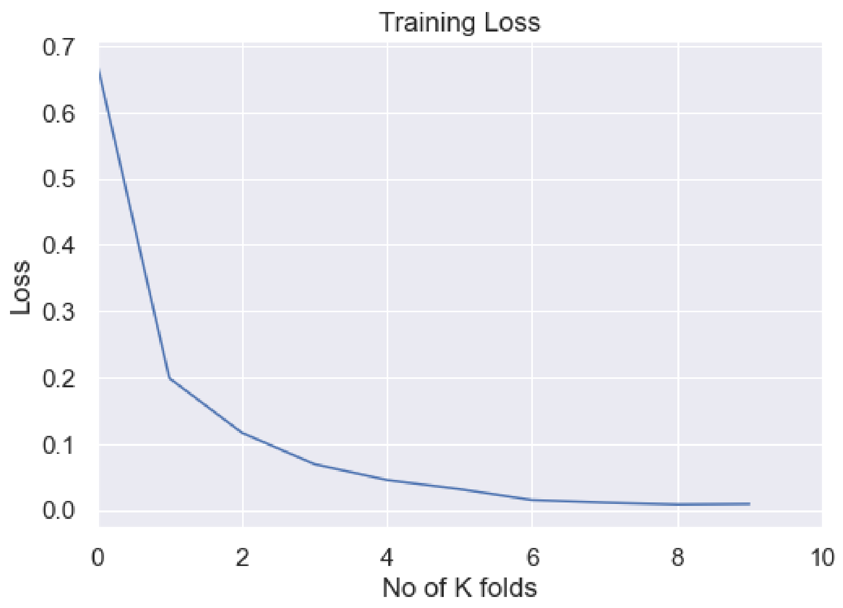


Figure 4.1: Average Training loss vs No. of K folds

.

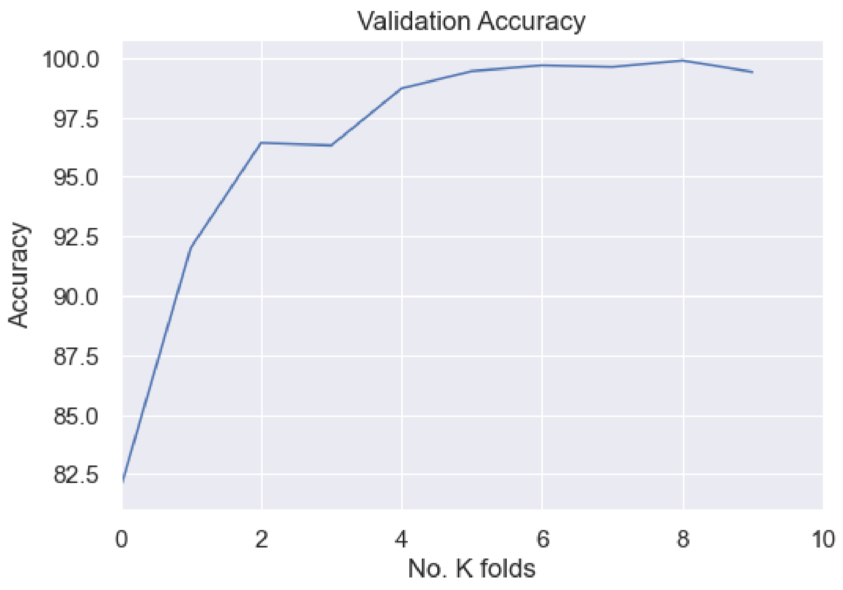


Figure 4.2: Average Accuracy on Validation set vs No. of K folds.

Figure 4.2 shows the Average accuracy on the validation set during the epochs vs No. of K folds used during the training of the mode. During the training k is taken as 10 where as no. of epochs taken as 10 to do the analysis. We can see that from K fold 1 to K fold 4 our accuracy significantly increased. During k fold 9 and 10 there are very less change in terms of the accuracy on the validation set.

Chapter 5

Evaluation

5.1 Model Evaluation

Evaluation on the model is done by examining the Test data set which consists of 1801 images, with three different classes in total, which are split randomly. During the project phase 1, We evaluated the accuracy, loss, precision, recall, F1 score and confusion matrix for different number of epochs such as 3, 5, 10, and 15. After performing the analysis with the help of Validation set of the model, epoch 15 is considered as the final epoch to be considered for the results. Training accuracy and loss for the model results are plotted which help us in selection of epochs. It can be clearly seen from the plots that after 15 epochs iteration, accuracy and loss of training data is not improving much.

In the testing phase of the project, we measure the accuracy of the testing set. To include the results of the model for the testing set, the first 10 samples of images are evaluated in the code i.e. Actual vs Predicted. For this, we have used the max which simply selects the greatest value and ignores the others, so max is the identity operation for that one element.

We also generated the plot for the confusion matrix with the help of Seaborn library, which is a Python data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics[9]

Confusion matrix shows the number of classes under Actual vs Predicted, and it can be clearly seen that class "Not a person" is predicted 17 times as "Without a Mask"class, whereas class "Without a Mask" is predicted 6 times as "Not a person" class.

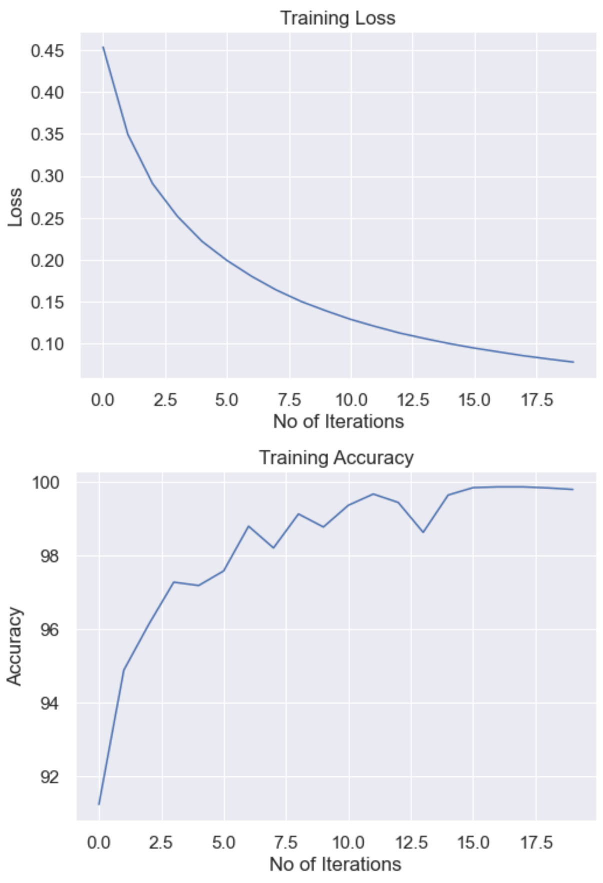


Figure 5.1: Training Loss & Accuracy.

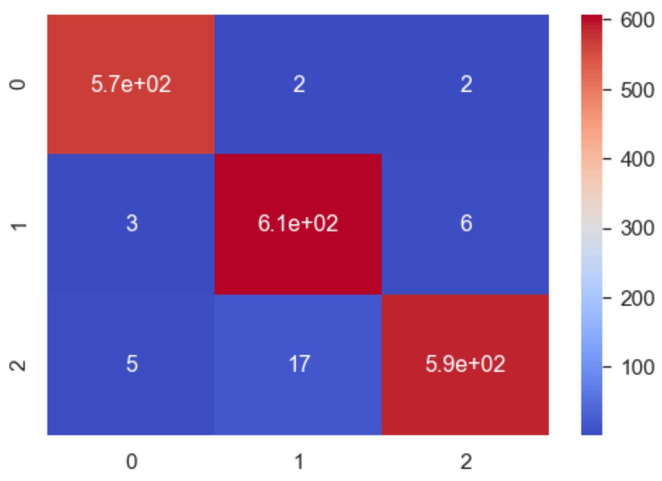
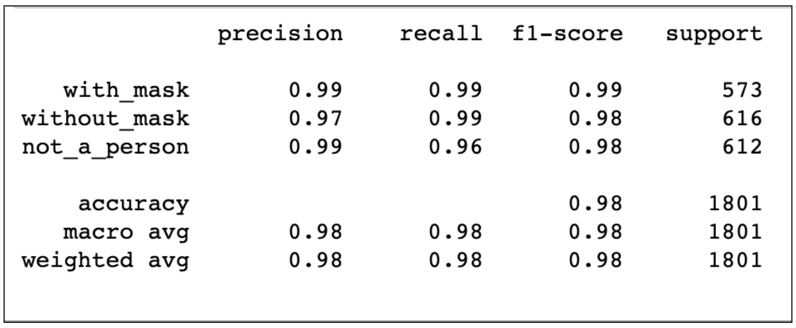


Figure 5.2: Confusion Matrix.



```
              precision    recall  f1-score   support

   with_mask       0.99      0.99      0.99       573
without_mask       0.97      0.99      0.98       616
not_a_person       0.99      0.96      0.98       612

    accuracy                           0.98      1801
   macro avg       0.98      0.98      0.98      1801
weighted avg       0.98      0.98      0.98      1801
```

Figure 5.3: Classification Report.

Classification report contains the accuracy of the testing, precision, recall and F1-score of the evaluated model.According to the classification report, class ”With a Mask” did well, that is it scored 0.99 in terms of precision, recall and f1-score whereas class ”Without a Mask” did less well in precision and class ”Not a Person” did less well in terms of Recall. Accuracy of the testing data set is 98%.

During the second phase of the project, we evaluated the accuracy, loss, precision, recall, F1 score and confusion matrix for the k folds, where the value of k used is 10 and no. of epochs used as 10.

When it comes to evaluating the accuracy of our trained CNN model, for the testing dataset, our model achieved the overall accuracy of 99.44%, whereas in the phase 1 of the project the overall accuracy achieved was 98%.

To observe the results of the model for the testing set, the first 8 samples of images are evaluated in the code, i.e. Actual vs Predicted, and in the second phase of the project, we also included the actual images for the comparison. Figure 5.4 shows sample of the test images and where as Figure 5.5 Actual vs Predicted class labels of the images.

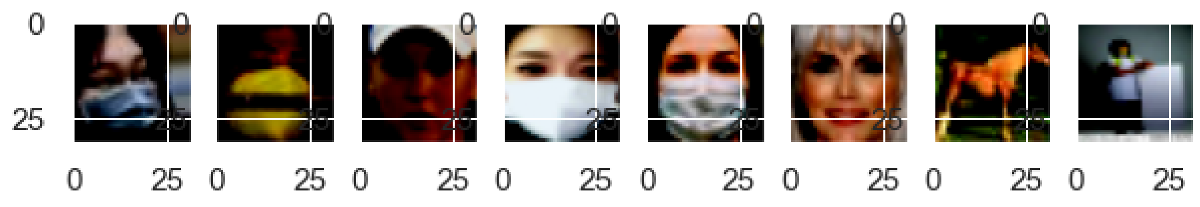
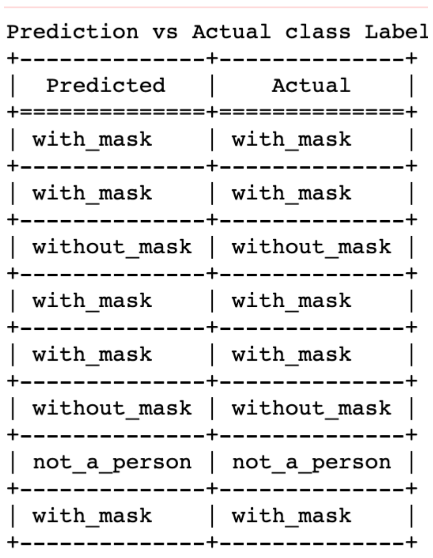


Figure 5.4: Sample Images of the Test dataset



```
Prediction vs Actual class Label
+--------------+--------------+
|  Predicted   |    Actual    |
+==============+==============+
| with_mask    | with_mask    |
+--------------+--------------+
| with_mask    | with_mask    |
+--------------+--------------+
| without_mask | without_mask |
+--------------+--------------+
| with_mask    | with_mask    |
+--------------+--------------+
| with_mask    | with_mask    |
+--------------+--------------+
| without_mask | without_mask |
+--------------+--------------+
| not_a_person | not_a_person |
+--------------+--------------+
| with_mask    | with_mask    |
+--------------+--------------+
```

Figure 5.5: Testing data sets Actual vs Predicted class labels.

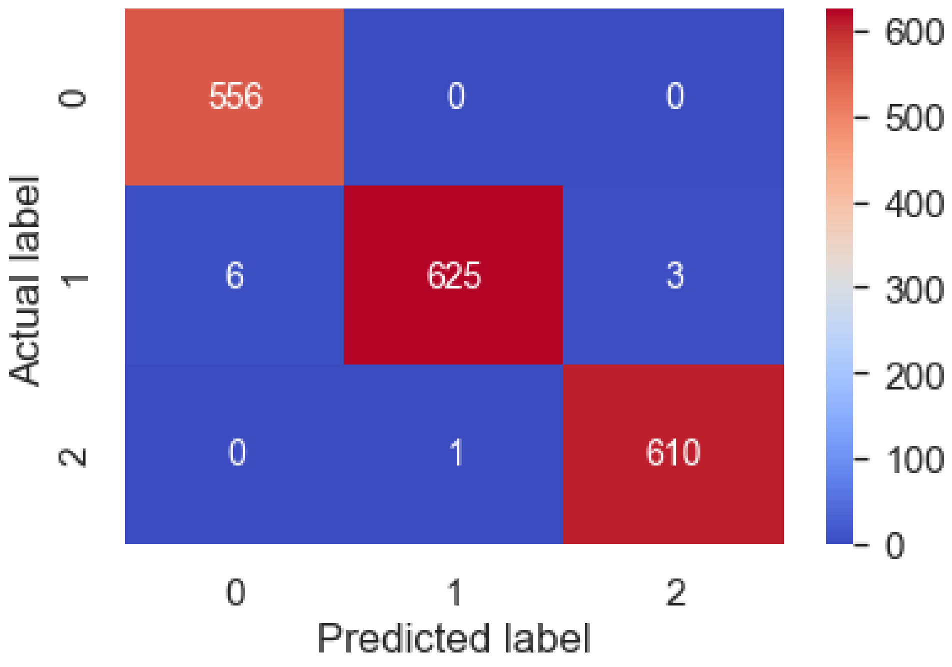
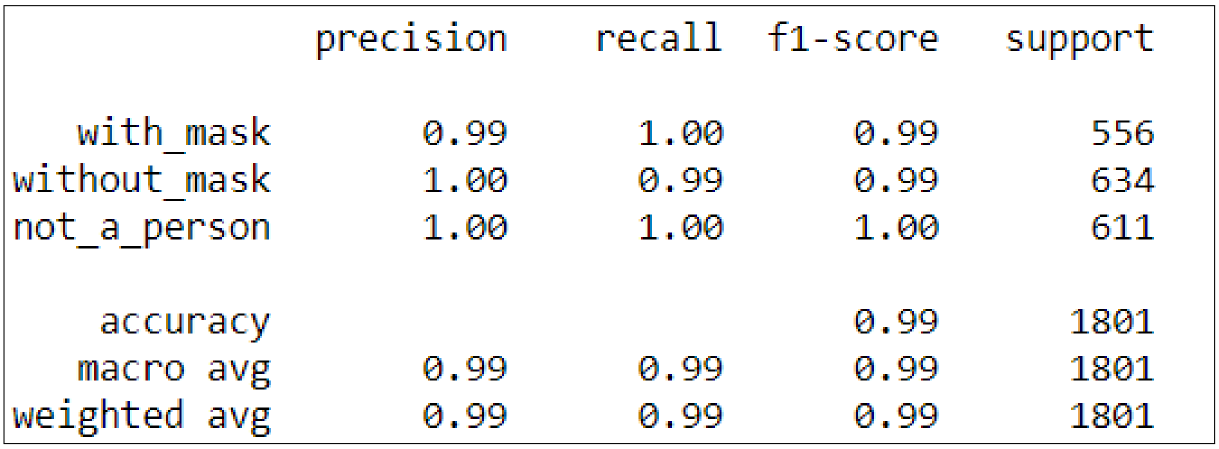


Figure 5.6: Confusion Matrix with 10 K fold.

We also generated the plot for the confusion matrix with the help of Seaborn library which can be seen in as shown in Figure 5.6 for 10 k fold with the iteration taken as 10 i.e. no of epochs during each k fold.We can observe from the confusion matrix that 0 times our model predicted ’with mask’ as ’without mask’ and 5 ’not a person’ which is better than the project phase 1 whereas in total 4 images were predicted wrong. In comparison to ’not a person’ predicted 0 times as ’with mask’ which is better than the previous phase 1 where model predicted 5 times. The model significantly improved in the predictions of ’not a person’ as ’without mask’ from 17 incorrect results to 1.

Figure 5.7 shows the Classification report contains the accuracy of the testing, precision, recall and F1-score of the evaluated model on the testing data set with 10 k fold and the no.of epochs taken as 10 for each k fold.



```
              precision    recall  f1-score   support

   with_mask       0.99      1.00      0.99       556
without_mask       1.00      0.99      0.99       634
not_a_person       1.00      1.00      1.00       611

    accuracy                           0.99      1801
   macro avg       0.99      0.99      0.99      1801
weighted avg       0.99      0.99      0.99      1801
```

Figure 5.7: Classification Report with 10 K fold.

According to the classification report, class ”With a Mask” did well, that is it scored 0.99 in terms of precision, and 1 in terms of recall and 0.99 in f1-score whereas class ”Without a Mask” and ’not a person’ did less well in precision. Specifically class ’not a person’ did well in this iteration with the score 1 in precision, recall and f1 score compare to previous phase where the results of the recall were 0.96, precision were 0.99 and f1 score was 0.98.

In general we can say in the first phase our model was able to predict good if given the image of ’with mask’ but it was not good enough for the class ’not a person’ and ’without a mask’. But in project phase 2, using k fold cross validation technique, our model is now better for examining all the class labels.

Bibliography

[1] https://www.kaggle.com/charlessamuel/face-mask-detection-pytorch

[2] https://en.wikipedia.org/wiki/Convolutional_neural_network

[3] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional
-neural-networks-the-eli5-way-3bd2b1164a53

[4] https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

[5] https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks

[6] https://stanford.edu/ shervine/blog/pytorch-how-to-generate-data-parallel

[7] https://www.journaldev.com/36576/pytorch-dataloader

[8] https://en.wikipedia.org/wiki/Stochastic_gradient_descent

[9] https://seaborn.pydata.org/generated/seaborn.heatmap.html

[10] https://www.kaggle.com/omkargurav/face-mask-dataset

[11] https://www.kaggle.com/prithwirajmitra/covid-face-mask-detection-dataset

[12] https://www.kaggle.com/sumansid/facemask-dataset

[13] https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset

[14] https://github.com/YoongiKim/CIFAR-10-images

[15] https://machinelearningmastery.com/k-fold-cross-validation/