Project Report

Introduction to Artificial Intelligence – Comp 6721
Fall 2020

Project: Face Mask Detector

Group Name:

SS_G03

Group Members:

Ahsan Saleem (40181654)

Prateek Bagora (40156671)

Mahsa Raeiszadeh (40176173)

1. Dataset

Our dataset consists of 15,000 images belonging to three classes. We train a CNN model to recognize three different classes: (1) Person without a face mask, (2) Person with a face mask, and (3) Other image. We gather the dataset for the three classes from three different sources. The distribution of our dataset is in Table 1.

The Mask faces [1] dataset contains 67,193 images, but we have selected 5000 images by taking, in consideration, the processing power of the system. While the dataset of faces without mask is taken from [2], it's not published on any form but cited by many GitHub projects. The selection of dataset is based on variations of images e.g., faces of different age group and with having different facial expression, illumination condition and occlusions. Caltier 101 [3], is selected for the third class, we have selected 5000 random object images belonging to 96 different categories from the dataset for our model.

The data structure of the dataset is shown in Figure 1. All the images of the dataset are stored in the "Dataset" directory and the format of all images is *jpeg* except face images which is *png*. The dataset includes three classes and each class contains 5000 images. Since the image resolution of each class is different, we used *transforms.Resize* function of *torchvision* to resize all the images to equal size of 200 x 200. Also, we converted all the images to 'RGB' format using *PIL library*. Using *numpy* library, the dataset is converted to array and saved along with the label in *.npy* file, then the mean and standard deviation is calculated using the created array.

After the data preprocessing, we split the dataset randomly to train and test parts and use 80% of the data for training phase and the remaining 20% for testing phase. Then, we apply the 10-fold cross-validation on the train set. Afterward, we transform the data into a tensor form and create data loader for training and testing part to help for iterating over the dataset.

Table 1: Number of Images in each class.

Categories	Number of Images
Person with Face Mask	5000
Person without Mask Face	5000
Not a Person	5000
Total Images	**15000**

Figure 1: Dataset Structure.

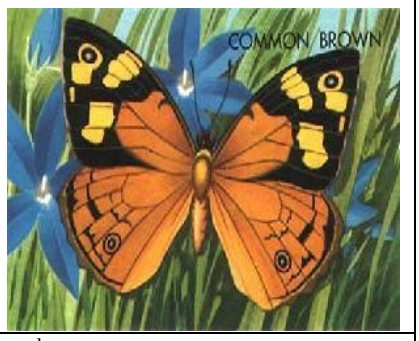
		
1st Class: With Mask [1]	2nd Class: Without Mask [2]	3rd Class: Other Images [3]

Figure 2: Sample Images from three classes.

2. CNN Architecture:

We have used a Mini Residual CNN Network (Mini-ResNet) [4], inspired by Deep Residual Learning for Image Recognition, to strengthen feature propagation and encourage feature reuse. We use this network because it has shown very impressive performance in image classification application and it can extract highly discriminative features for image identification. The network is based on the idea of "skip-connections" and implements heavy batch-normalization, that help it in training over thousands of layers effectively, without degrading the performance in the long run. The problem rose with the training of deeper networks. The problem of "vanishing gradient" where repeated multiplication being done, as the gradient is being backpropagated, makes the gradient infinitely small. This results in degradation of performance. The idea that was infused in this architecture was "identity shortcut connection" that implies transferring the results of a few layers to some deeper layers skipping some of the other layers in between.

Also, in this network, the deeper layers do not produce higher training errors than its shallower counterparts. The skip-connections were done to implement this idea. The developers of this network implemented a pre-activation variant of the residual block, in which gradients can flow through the shortcut connection to the earlier layers, thus reducing the "vanishing gradient" problem.
The architecture, as shown in Figure 3, is a 14 layered Mini-ResNet, implemented in PyTorch, which has 13 convolutional layers, followed by average pooling, and a fully connected layer. Each of the convolutional layers uses 3x3 filters, with fixed padding of 1 and a stride of either 1 or 2.

The first convolutional layer takes in a 3-channeled 200x200 image and convolves it into 16 channels of size 200x200, using a stride of 1. This is followed by batch normalization and a rectified linear unit (ReLU) activation function. Then we have 6 residual blocks, each composed of 2 convolutional layers. Each convolutional layer is followed by batch normalization and a rectified linear unit (ReLU) activation function except at summation points. At each summation point, the output from the last convolutional layer in the residual block is fed through batch normalization. The output from batch normalization is then summed up with the input fed to the residual block, which is followed by the ReLU activation function.

At the summation point, the input to the residual block may need to be down sampled to match the residual block's output's dimensions, before the summation. There may be two cases when

down sampling is required. Firstly, whenever the first convolutional layer in the residual block has a stride of 2, the size of the activation function will be reduced by a factor of 2. Secondly, when the first convolutional layer in the residual block increases the number of channels. The down sampling is done with the help of a 3x3 convolution.

The first and second residual blocks maintain the same dimensions they are fed. The first convolutional layer in the third residual block increases the number of channels to 32 from 16. Also, it reduces the activation map size to 100x100 by using a stride of 2. The fourth residual block maintains the same dimensions it is fed. The fifth residual block increases the number of channels to 64 from 32. Also, it reduces the activation map size to 50x50 by using a stride of 2. The sixth residual block maintains the same dimensions it is fed.

An average pooling layer of kernel size 50x50 reduces each of the 64 channels received from the sixth residual block to size 1x1, giving out 64 values as output, one for each channel. These are fed to a fully connected layer consisting of three neurons, each representing one of the three classifications. While making the predictions, the image is classified according to the highest-scoring neuron. The details for the selected parameters for model, are shown in Table 2.

Table 2: Details about Parameters of training

Hyperparameters	Value
Image size	200 x 200
Number of Epochs	15
Learning Rate	0.001
Learning Rate Decay Factor	0.5
Learning Rate Decay Frequencies (in number of epochs)	10
Batch Size	20
Training Ratio	80%
Testing Ratio	20%
K-fold Validation	10 folds

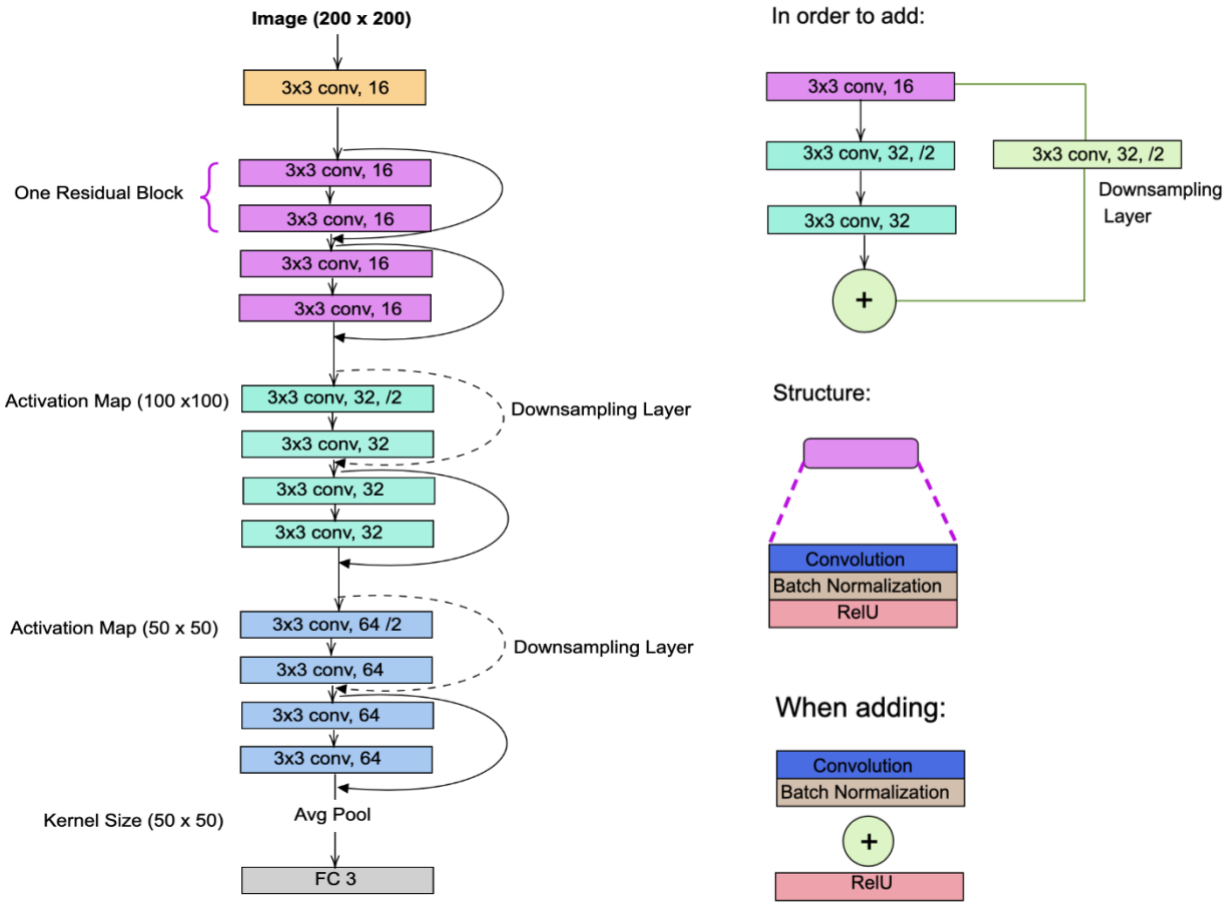


Figure 1: Residual CNN Architecture[5].

3. Evaluation:

In the previous phase, we use 80% of the data for training phase and the remaining 20% for testing phase. Once we are done with training our model, we got a good result on our training and testing data. However, we cannot be sure that the model will have the desired accuracy in other environments. We need some kind of assurance of the accuracy of the predictions that our model is putting out. For this, we need to validate our model.

To evaluate the performance of any machine learning model we need to test it on some unseen data. Based on the model's performance on unseen data, we can say weather our model is Under-fitting, Over-fitting or Well generalized. Cross validation (CV) [6] is one of the technique used to test the effectiveness of our models. To perform CV, we need to keep aside a sample of the data on which is not used to train the model, later use this sample for testing. So, we use one of the techniques of CV naming as 10-Fold Cross Validation. In 10-Fold Cross Validation a given data set is split into a 10 number of folds, where each fold is the 10% of the dataset and used as a validation set at some point. In the first iteration, the first fold (10% of the data) is used to validate the model and the rest

are used to train the model. In the second iteration, second fold is used as the validation set while the rest serve as the training set. This process is repeated until each fold of the 10 folds have been used as the validation set.

We apply the 10-fold cross-validation on the 80% of the dataset. Our model is tested on the 20% of the dataset. The hyperparameters that we used have the same configuration as the training with an input image size as 200. The evaluation result of our model on each of 10-fold is shown in Table 3. We have a balanced dataset, and we are dealing with multi-class dataset, so, we consider weighted versions of the precision, recall, and F1-score metrics [7]. The results from precision, recall, and F1 score show that our model performs well on each fold. Also, the results from the average of 10-fold cross validation and the result of testing data are reported in Table 4. The model performs equally well on the testing set, so there is no indication of overfitting.

We plot the Confusion Matrix for each of 10-fold [8], along with Accuracy and Loss curve of training phase to show the performance of our model which is shown in Table 6.

Furthermore, we plot the Confusion Matrix for the average of 10-fold cross validation and confusion matrix for the testing data. The plots are shown in Table 5.

Considering the Confusion Matrix for final testing:

- The confusion matrix is generated for three possible predicted classes: "With Mask", "Without Mask", and "Other Images".
- The classifier made a total of 3000 predictions.
- The predication is based on, 1000 images from the faces with mask, 1000 images are from faces without mask, and 1000 images are not person images.
- Out of those 3000 images, the classifier predicted "Without Mask" 882 times, "With Mask" 986 times, and "Other Images" 999” times.

Table 3: Evaluation result on 10-fold Cross Validation

K in 10-Fold CV	Accuracy	Precision	Recall	F1-Score
1	95.66%	95.98%	95.66%	95.62%
2	95.91%	96.36%	95.91%	95.91%
3	97.58%	97.71%	97.58%	97.58%
4	97.66%	97.74%	97.66%	97.65%
5	95.66%	96.09%	95.66%	95.63%
6	97.75%	97.84%	97.75%	97.75%
7	97.50%	97.62%	97.50%	97.49%
8	97.66%	97.79%	97.66%	97.65%
9	93.58%	94.49%	93.58%	93.54%
10	94.25%	95.09%	94.25%	94.21%

Table 4: Evaluation result of the average of 10-fold Cross Validation and testing data

	Accuracy	Precision	Recall	F1-Score
Mean result of 10-fold Cross Validation	96.32%	96.61%	96.32%	96.31%
Final Testing	95.56%	96.04%	95.56%	95.57%

Table 5: Confusion matrix for the average of 10-fold Cross Validation and Confusion Matrix of Final Testing

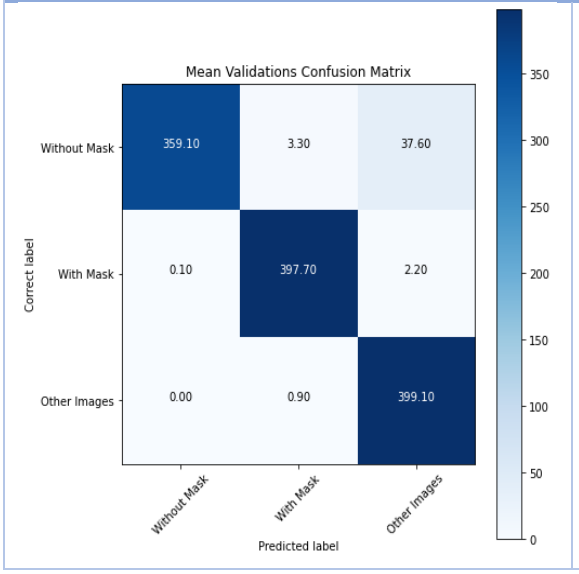
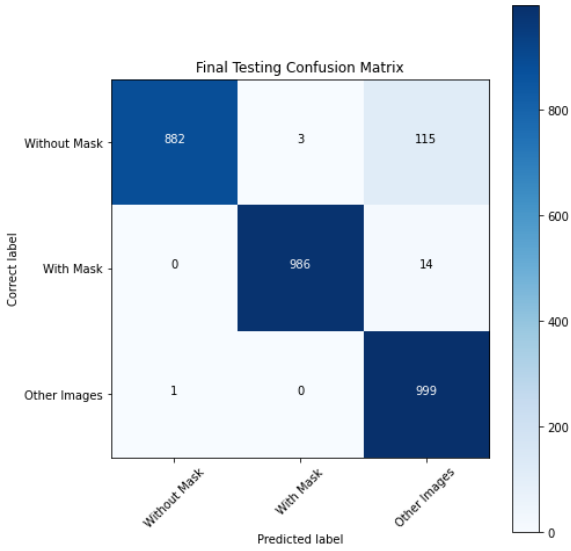
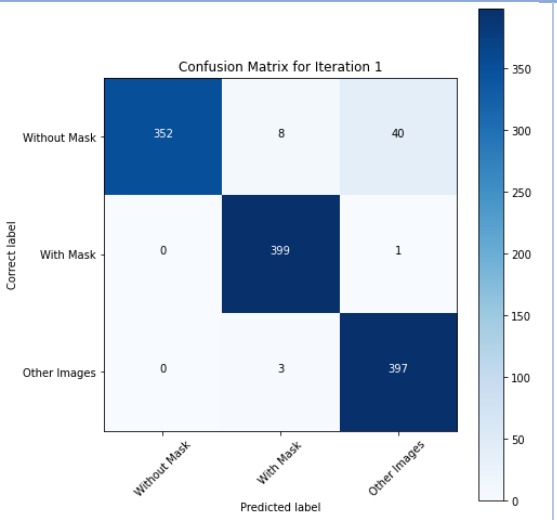
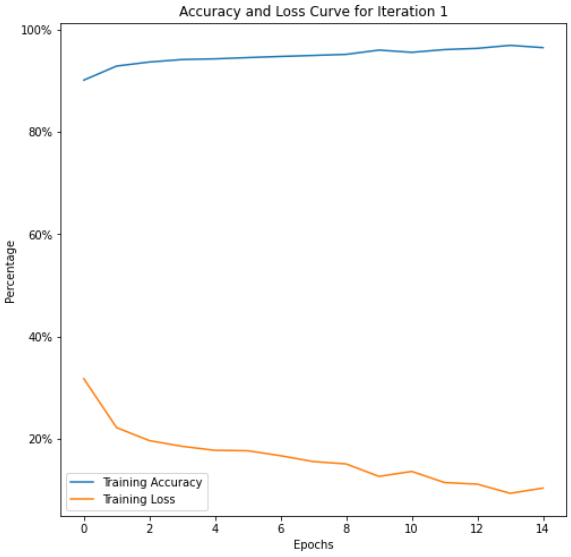
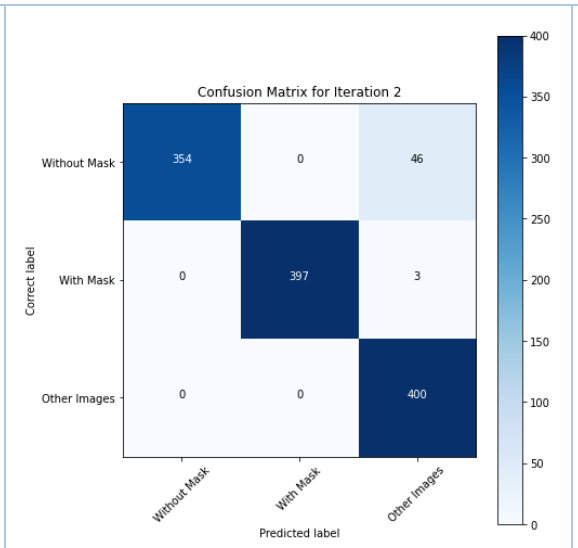
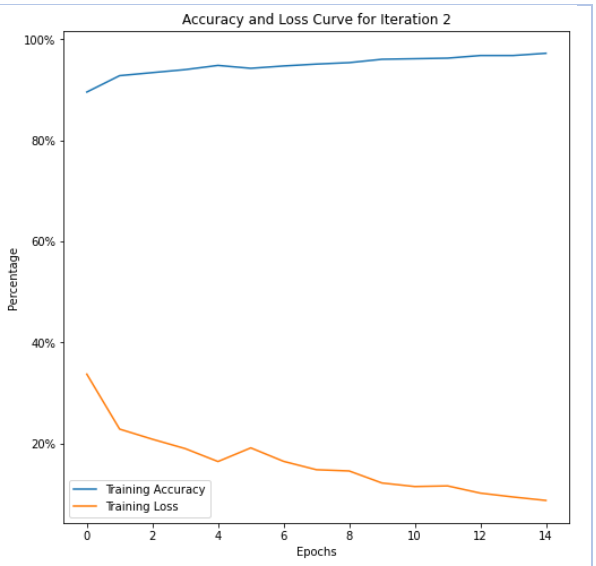
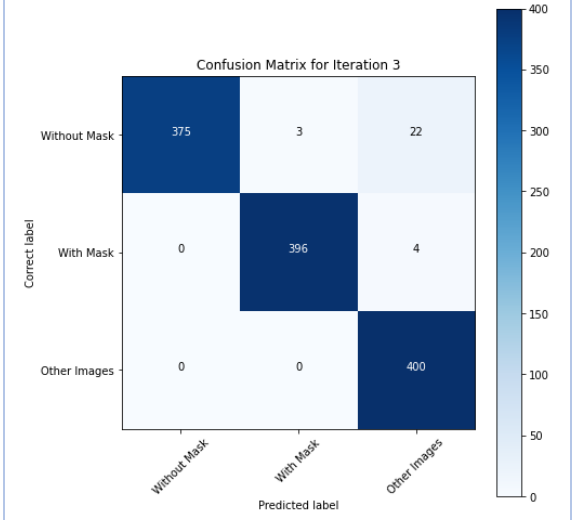
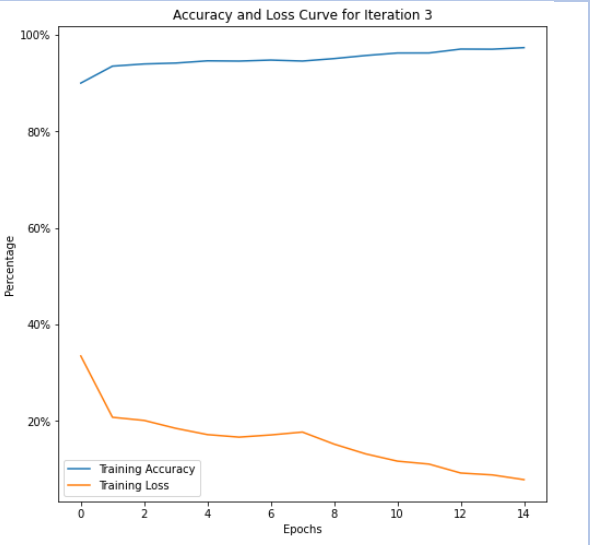
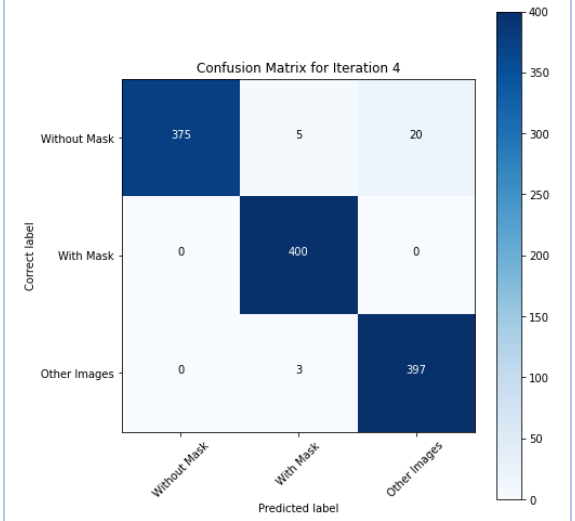
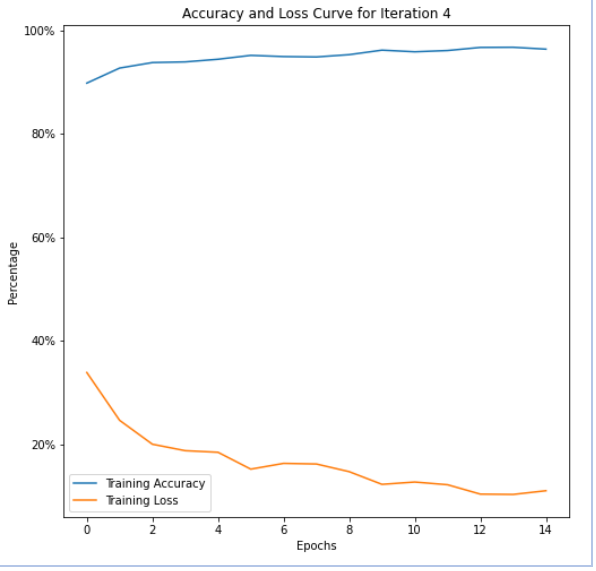
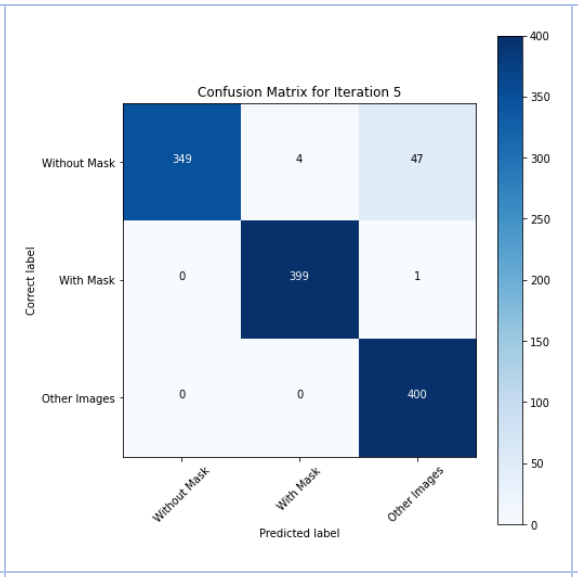
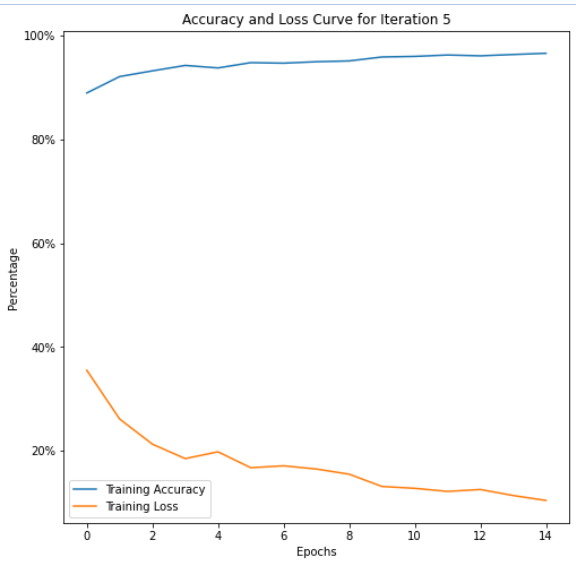
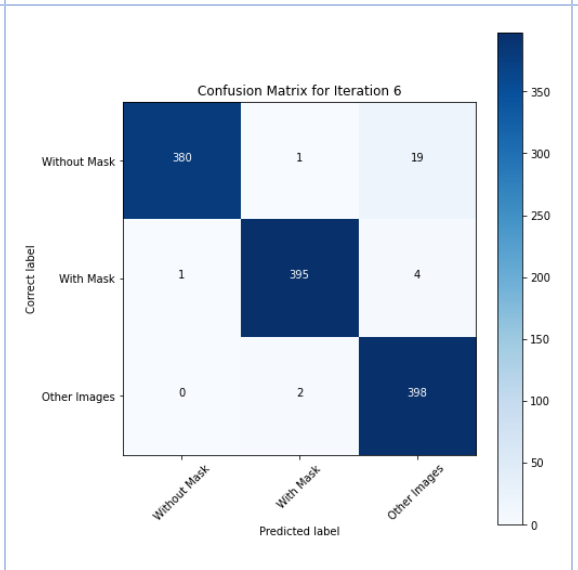
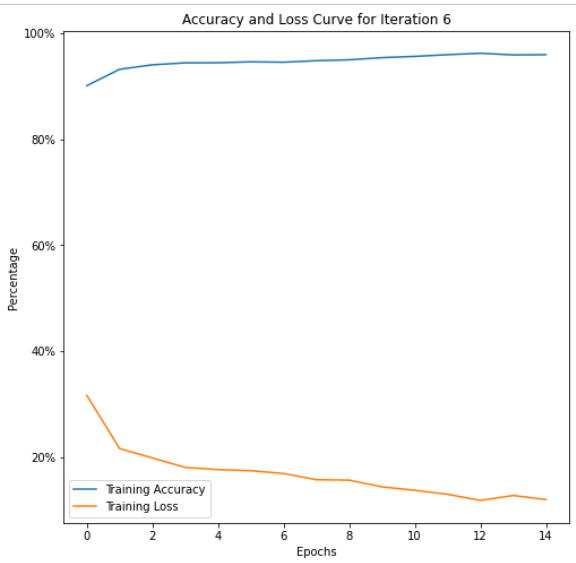
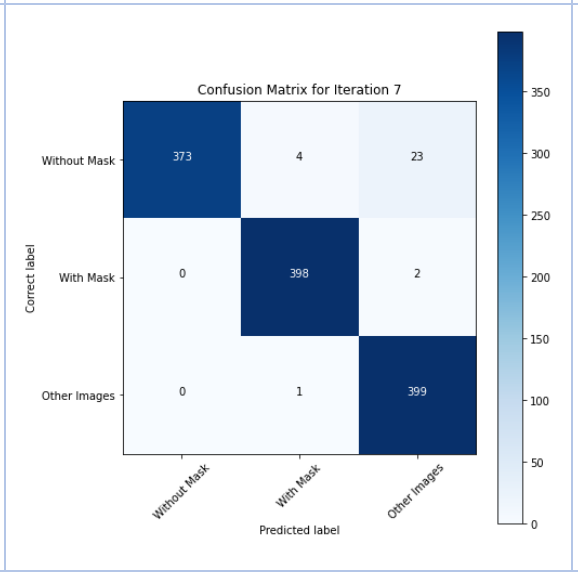
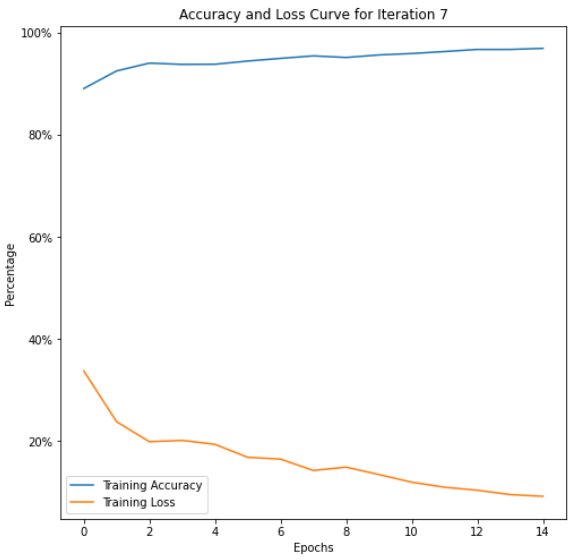
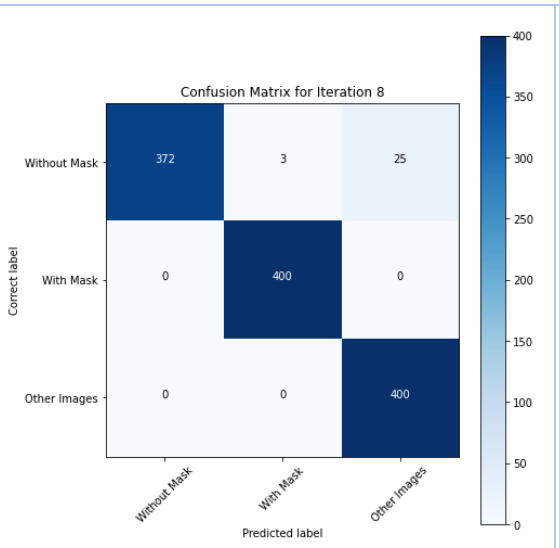
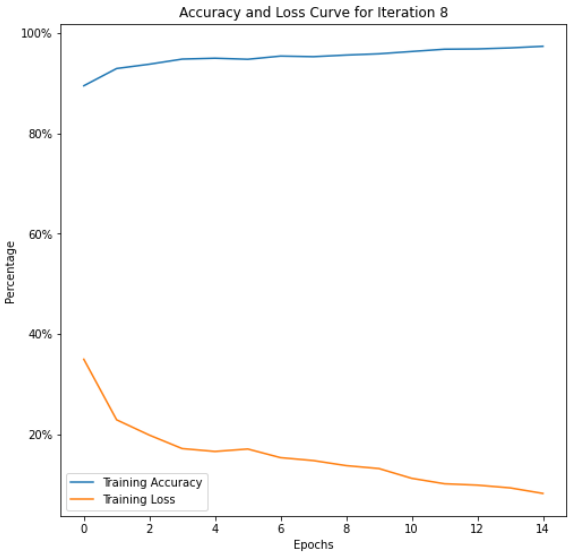
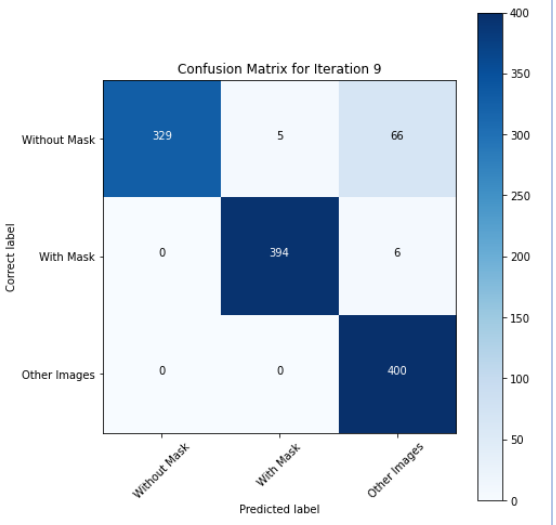
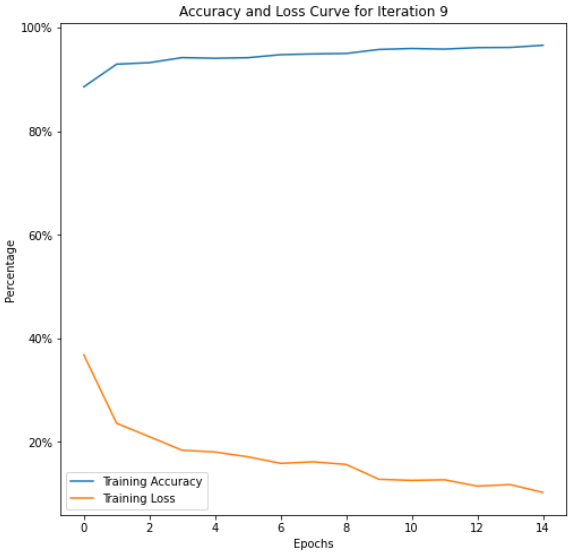
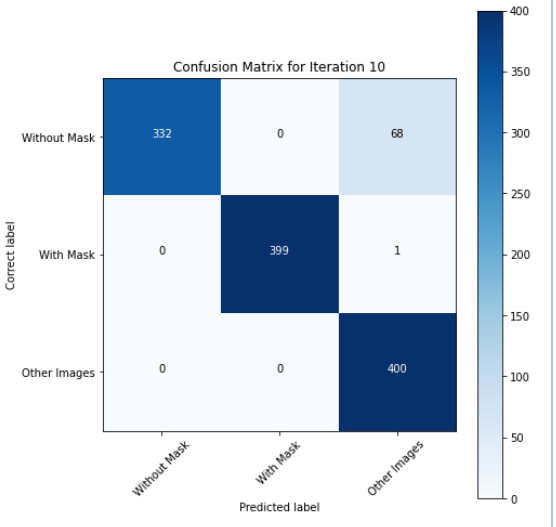
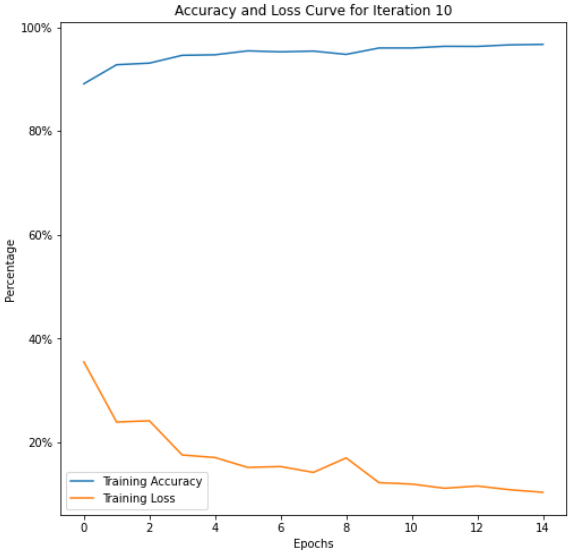
Mean Validation Confusion Matrix	Final Testing Confusion Matrix
	

Table 6: Confusion matrix for 10-fold Cross Validation and accuracy & loss curve over training data at 10-fold

K in 10-fold CV	Confusion Matrix	Accuracy & Loss curve
1		

2	 Confusion Matrix for Iteration 2 Without Mask With Mask Other Images Correct label Predicted label 354 0 46 0 397 3 0 0 400 400 350 300 250 200 150 100 50 0	 Accuracy and Loss Curve for Iteration 2 Percentage Epochs Training Accuracy Training Loss 100% 80% 60% 40% 20% 0 2 4 6 8 10 12 14
3	 Confusion Matrix for Iteration 3 Without Mask With Mask Other Images Correct label Predicted label 375 3 22 0 396 4 0 0 400 400 350 300 250 200 150 100 50 0	 Accuracy and Loss Curve for Iteration 3 Percentage Epochs Training Accuracy Training Loss 100% 80% 60% 40% 20% 0 2 4 6 8 10 12 14
4	 Confusion Matrix for Iteration 4 Without Mask With Mask Other Images Correct label Predicted label 375 5 20 0 400 0 0 3 397 400 350 300 250 200 150 100 50 0	 Accuracy and Loss Curve for Iteration 4 Percentage Epochs Training Accuracy Training Loss 100% 80% 60% 40% 20% 0 2 4 6 8 10 12 14

5	 Confusion Matrix for Iteration 5 Without Mask With Mask Other Images Correct label Predicted label 349 4 47 0 399 1 0 0 400	 Accuracy and Loss Curve for Iteration 5 Percentage Epochs Training Accuracy Training Loss
6	 Confusion Matrix for Iteration 6 Without Mask With Mask Other Images Correct label Predicted label 380 1 19 1 395 4 0 2 398	 Accuracy and Loss Curve for Iteration 6 Percentage Epochs Training Accuracy Training Loss
7	 Confusion Matrix for Iteration 7 Without Mask With Mask Other Images Correct label Predicted label 373 4 23 0 398 2 0 1 399	 Accuracy and Loss Curve for Iteration 7 Percentage Epochs Training Accuracy Training Loss

8	 Confusion Matrix for Iteration 8 Without Mask With Mask Other Images Correct label Predicted label 372 3 25 0 400 0 0 0 400	 Accuracy and Loss Curve for Iteration 8 Percentage Epochs Training Accuracy Training Loss
9	 Confusion Matrix for Iteration 9 Without Mask With Mask Other Images Correct label Predicted label 329 5 66 0 394 6 0 0 400	 Accuracy and Loss Curve for Iteration 9 Percentage Epochs Training Accuracy Training Loss
10	 Confusion Matrix for Iteration 10 Without Mask With Mask Other Images Correct label Predicted label 332 0 68 0 399 1 0 0 400	 Accuracy and Loss Curve for Iteration 10 Percentage Epochs Training Accuracy Training Loss

4. Reference

1. Cabani, A., et al., *MaskedFace-Net--A Dataset of Correctly/Incorrectly Masked Face Images in the Context of COVID-19.* 2020.
2. Spaeh, J.L. *Face Mask ~12K Images Dataset*. Available from: https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset/discussion.
3. L. Fei-Fei, R.F.a.P.P. *Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories.* 2004; Available from: http://www.vision.caltech.edu/Image_Datasets/Caltech101/.
4. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
5. Sammani, F. *The Complete Neural Networks Bootcamp: Theory, Applications*.
6. Learn, S. *Cross-validation: evaluating estimator performance*. Available from: https://scikit-learn.org/stable/modules/cross_validation.html.
7. Learn, S. *Sklearn Metrics F1_score*. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.
8. SalRite. *Demystifying 'Confusion Matrix' Confusion*. 2015; Available from: https://towardsdatascience.com/demystifying-confusion-matrix-confusion-9e82201592fd.