Concordia University
Engineering and Computer Science

COMP 6721
Applied Artificial Intelligence

Project Report

Submitted to: Prof. Dr. René Witte

Team Id: RN_05

Deep Patel (40185585) – Data Specialist
Rutwik Patel (40160646) – Training Specialist
Virag Vora (40168195) – Evaluation Specialist

Table of Content

Table of Images

Dataset

Collection of data is called a dataset. In the field of Artificial Intelligence, high quality and accurate data is key to success. It is also important that AI interpret data correctly and train accordingly to generate an accurate model.

For this project, we need to collect data for 4 different categories namely Person **"without a mask**", Person with a "**community(cloth) mask**", Person with a "**Surgical mask"** and person with a **"FPP2 mask"**.

Person with Surgical Mask

Person with FFP2 Mask

Person with Cloth Mask

Person without a Mask

1.1: Four categories of Mask

Our dataset contains 2736 images collected mainly from 3 different sources, namely:

1. Kaggle Datasets
2. Google Images
3. GitHub Repositories

It is important to create dataset in balanced manner to generate unbiased model. In our dataset, all category contains almost equal number of images. In total we have 3500 images, among which 2736 images are for training and rest 800 images for testing dataset and bias. Among training, 727 images of Cloth Masks, 612 images of FFP2 Masks, 701 images of Surgical Masks and 700 images of Person not Wearing Mask. Among testing dataset, 400 images(all categories) and 100 images for each section(age, gender). In this second build of the project, we corrected flaws from our first build. Firstly, by removing some redundant images from the dataset, mentioning per class statistics of each category, adding some more images and increasing the accuracy. We were successfully able to correct listed notes in this second build of our project.

Approaching the second build of the project, we have increased our dataset from 1326 total images to 2736 images(training) and 800 distinct images for testing. We have implemented k-fold cross validation, by which accuracy of our model is increased considerably. We did not get enough images for testing dataset, so for that section we have generated images of people wearing masks. We accomplish this by using a **Har Cascading** script to place mask on the face.

In the training phase of the project, the model takes labelled data as an input and decides for finding out patterns for the future predictions and in Testing phase is like an exam of the model. In which, the model will be given unknown data(unlabelled) and it will try to generate output based on the training pattern.

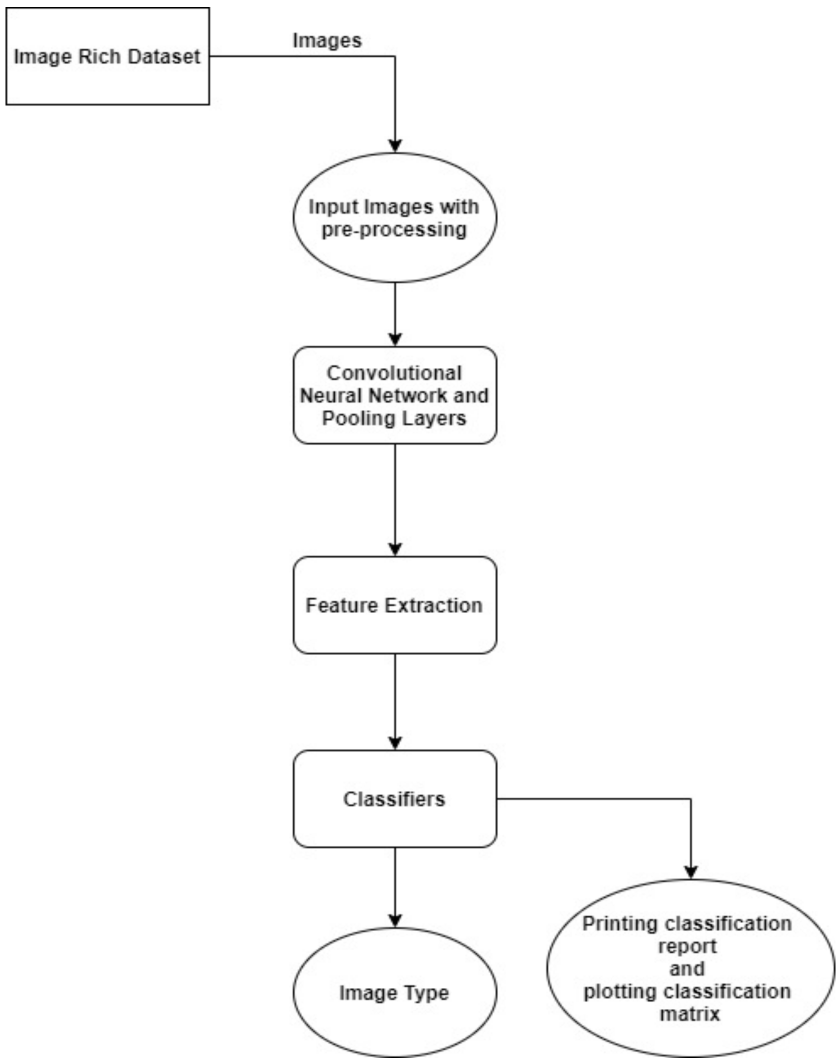


Image Rich Dataset
Images
Input Images with pre-processing
Convolutional Neural Network and Pooling Layers
Feature Extraction
Classifiers
Image Type
Printing classification report and plotting classification matrix

1.2: General overview of the project.

As shown in image 1.2, raw images from an image rich dataset will be loaded and pre-processed, passed into Convolutional Neural Network. There can be more than one CNN layer. After this, features can be extracted from the trained model and thus one can generate classification report and plot a confusion matrix and can also classify an unlabelled image.

CNN Architecture

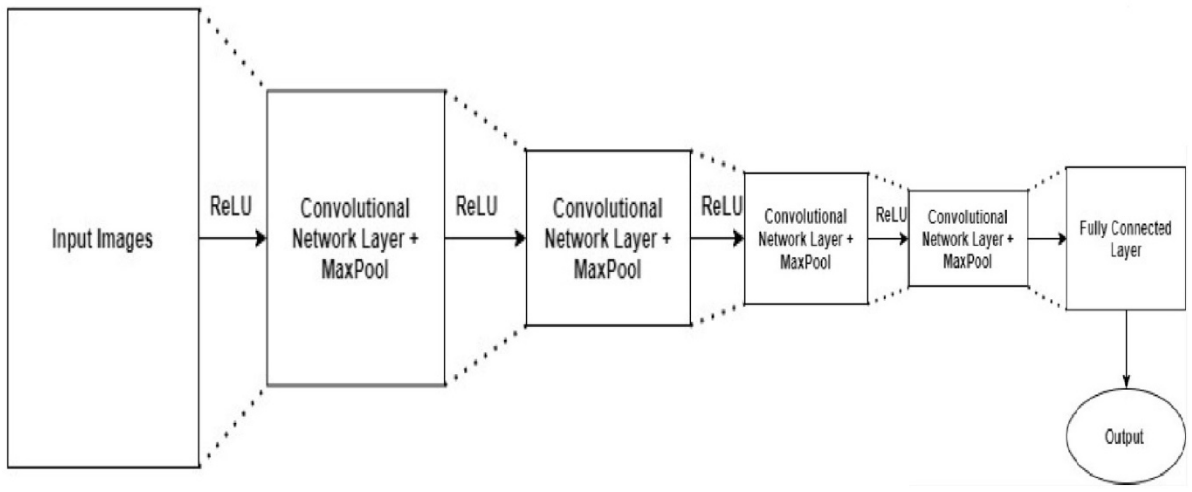
Convolutional Neural Network (CNN) are special kind of multi-layer neural networks, designed to extract key features from an image like visual patterns with very low pre-processing of data.

We have implemented CNN to train our model for given categories. Since we have collected data from different sources, the dimensions of all the images we had, were not in proper size. For example, some were too large, or some were small. To make all these images look same, we had to do some pre-processing. While loading the image from dataset, we converted all the images to a same size of (100 * 100). Also, we normalize the data using standard deviation and means of images to change the pixel intensity. To bring some randomness, we have also shuffled the data before passing it into training phase. By all these, training speed was increased considerably.

As we had four different categories, we labelled them as: {0: "cloth_mask", 1: "ffp2_mask", 2: "surgical_mask", 3: "without_mask"}. We then pass this dataset to 10-fold iterations, where training and testing dataset will be separated according to the number of folds. Then we load these datasets using DataLoader method with batch size of 64. Thus, training and testing datasets will be loaded.

We have used a very simple CNN architecture with just three convolutional layers to extract features from our dataset. Later we added one more layer(4th layer) to overcome overfitting problem. Initially, images of size (100 * 100) will be passed into first convolutional 2D layer with kernel size of (3 * 3), having stride and padding equals to 1. Out channel of this layer is of 64. Then the data will be normalized and then pass onto ReLU activation function as an input. From here the output of ReLU activation function will act as an input to MaxPooling with stride 2 and kernel size (2 * 2). The output of this will be then passed to the next convolutional 2D layer. Here the input channel will be 64, having all other parameter same as first convolutional 2D layer. Following steps will be same as it were before namely, normalization, ReLU activation function, and MaxPooling. After this, its output will then be input for third convolutional 2D layer, whose input channels are of 128 and output channel of 256. Again, all the steps will be followed like normalization, ReLU activation function and MaxPooling. In the end, we added one more hidden layer to our cnn model whose output channel id 512. We have added dropouts between each layer in order to eliminate nodes with less weight.

Then, we have used a fully connected dense layer to classify those extracted features from our data into their respective categories. At the end of the training process, we get the output as label of the category, in our case it is, (0, 1, 2, 3), which stands for ("Cloth Mask", "FFP2 Mask", "Surgical Mask", "Without Mask").


Input Images
ReLU
Convolutional Network Layer + MaxPool
ReLU
Convolutional Network Layer + MaxPool
ReLU
Convolutional Network Layer + MaxPool
ReLU
Convolutional Network Layer + MaxPool
Fully Connected Layer
Output

2.1: CNN Architecture

Initially, when we train the model with the dataset, model's testing accuracy was around 50-55% and training accuracy was 99%. We knew the model is overfit to the training data. Thus, to overcome the problem, we changed some of the conflicting images again and add some more distinct images to the dataset. We then tune our hyperparameters like learning rate (changed from 0.001 to 0.0001) and epoch number (limiting to 10 epochs). increasing the batch size (from 32 to 64) for both training and testing dataset and at last increasing one more hidden layer in our CNN model.

The main reason to add a hidden layer and decrease the learning rate is to give model some more time to learn and extract features from the training dataset. In addition to this we have also added dropout (set dropout = 0.02) between the layers, as it will randomly set the outgoing edges of hidden units to zero at each epoch in training phase, which will rescue the model from overfitting.

As a result of performing this step, our model was trained with accuracy of 96% and testing accuracy 90%. We saw significant change in the accuracy by changing some data and tuning hyperparameters.

Bias

With increasing number of data, a possible chance of bias might be hovering on the model. Bias is the data which does not give proper prediction of some specific input as such data might not be introduced while training that data.

There are many examples where not eliminating bias pave problems for big companies. For example, Amazon Scraps AI recruiting tool that showed bias against women and Google cloud's Computer Vision Algorithms to be biased against black people.

To check for bias is one of the important thing while building the dataset. To check bias in our project, we have considered two categories namely, age and gender. To check that, I have collected different testing images for different categories to check for bias. For each category we have selected 100 images to test for bias. If they print correct accuracy of all those categories, we can say that there is no bias in the training dataset. Precisely, we have parted gender into two section i.e. male and female. In the same way we have parted age in three sections namely, child(Age: 1 – 25), Adult(Age: 26 – 50) and Old(51 – 90). Thus, we separated our testing images according to these sections and their corresponding category. Roughly, we have 600-700 images for testing all these sections.

We are annotating the data with its pre-defined labels and then loading these images to check for bias. We have calculated accuracy, precision, F1-measure and Recall with the help of classification matrix and have also showed this data in form of figure with the help of confusion matrix. Here I am sharing the resultant data which shows bias information of each category. Firstly looking at Gender Section, which consist of two parts, male and female. Both male and female had 400 images each for testing the dataset. Here is a glance on the reports:

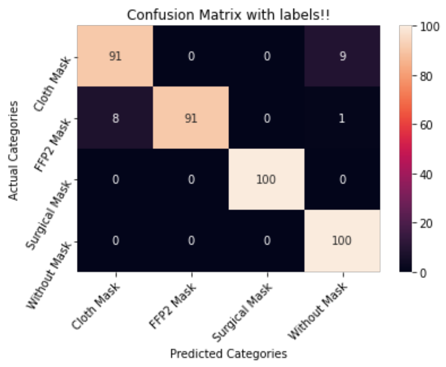
Male

```
------------Generate Matrix of Male------------
Mask Male Image Classification Report
              precision    recall  f1-score   support

           0       0.91      0.92      0.91        99
           1       0.91      1.00      0.95        91
           2       1.00      1.00      1.00       100
           3       1.00      0.91      0.95       110

    accuracy                           0.95       400
   macro avg       0.96      0.96      0.95       400
weighted avg       0.96      0.95      0.96       400
```

Mask Male Image Confusion Matrix



4.1 'Male' wearing mask Testing Confusion Matrix

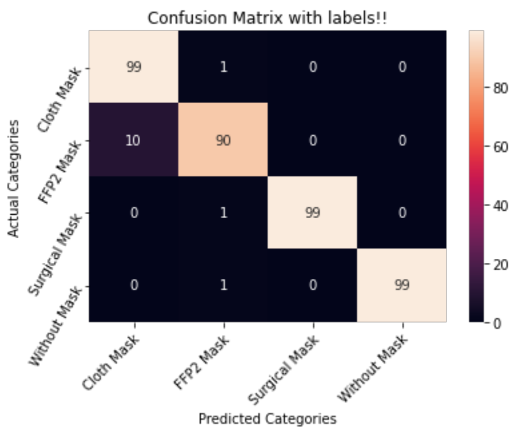
Female

```
------------Generate Matrix of Female------------
Mask Female Image Classification Report
              precision    recall  f1-score   support

           0       0.99      0.91      0.95       109
           1       0.90      0.97      0.93        93
           2       0.99      1.00      0.99        99
           3       0.99      1.00      0.99        99

    accuracy                           0.97       400
   macro avg       0.97      0.97      0.97       400
weighted avg       0.97      0.97      0.97       400
```

Mask Female Image Confusion Matrix



4.2 'Female' wearing mask Testing Confusion Matrix

As you can see in the image that there is no bias in the Gender section. Coming to the other section of the Bias assessment i.e., Age, which consists of Child, Young and Old.

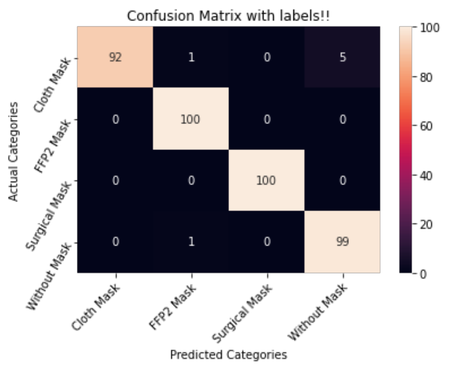
Child

```
------------Generate Matrix of Child------------
Mask Child Image Classification Report
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        92
           1       1.00      0.98      0.99       102
           2       1.00      1.00      1.00       100
           3       0.99      0.95      0.97       104

    accuracy                           0.98       398
   macro avg       0.98      0.98      0.98       398
weighted avg       0.98      0.98      0.98       398
```

Mask Child Image Confusion Matrix



4.3 'Child' people wearing mask Testing Confusion Matrix

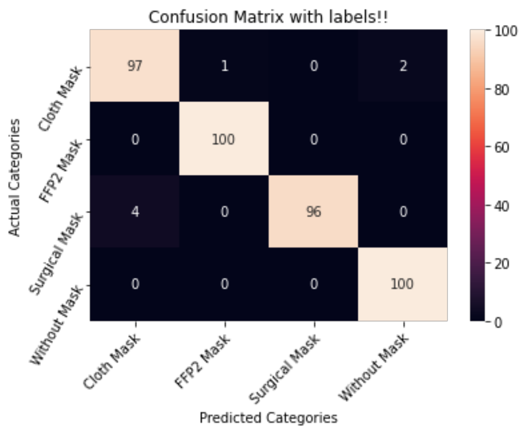
Young

```
------------Generate Matrix of Young/Adult------------
Mask Young Image Classification Report
              precision    recall  f1-score   support

           0       0.97      0.96      0.97       101
           1       1.00      0.99      1.00       101
           2       0.96      1.00      0.98        96
           3       1.00      0.98      0.99       102

    accuracy                           0.98       400
   macro avg       0.98      0.98      0.98       400
weighted avg       0.98      0.98      0.98       400
```

Mask Young Image Confusion Matrix



4.4 'Young' people wearing mask Testing Confusion Matrix

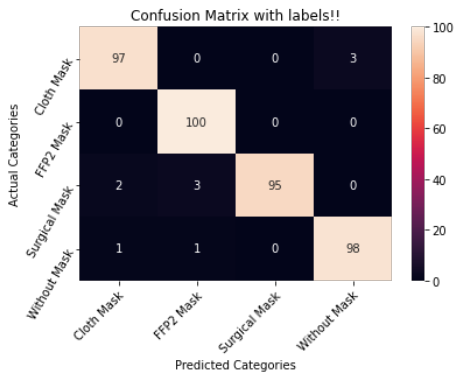
OLD

```
------------Generate Matrix of Old------------
Mask Old Image Classification Report
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       100
           1       1.00      0.96      0.98       104
           2       0.95      1.00      0.97        95
           3       0.98      0.97      0.98       101

    accuracy                           0.97       400
   macro avg       0.97      0.98      0.97       400
weighted avg       0.98      0.97      0.98       400
```

Mask Old Image Confusion Matrix



4.5 'Old' people wearing mask Testing Confusion Matrix

Here one can notice that the testing data is consistent for all over the training dataset, which gives us the assurance that there is no presence of bias in the training dataset. As we all were collecting data for our datasets, we kept in mind this particular scenario and collected images base on it.

Thus, no bias was found from our training dataset. Soon after our project 1 presentation we all the team mates were in search of data. Also we keep in mind various things like bias, quantity of dataset, number of testing images, etc. By taking all these precautionary steps we did not see any bias in our dataset. Of course, with the use of K-fold cross validation, taking care of bias and by increasing the number of images have shown tremendous effect on our model’s accuracy. Now moving towards Evaluation part.

K-fold Cross Validation

K-fold cross validation is a method that attempts to maximize the use of available data for training and then testing a model. Here, k refers the number of split of the dataset. In our case, we have 2736 images, so with 10 folds, in each fold training dataset will get 2462 images from the dataset and rest 274 will go to testing dataset. These images will shuffle in every fold and an image in training dataset in fold number 1, can go in testing dataset in some other fold.

we have implemented 10 K-folds(i.e. there will be total 10 folds). Here, we have kept 10 epochs in each fold. We are using 400 different images(not trained and of all categories mixed) to test the trained model.

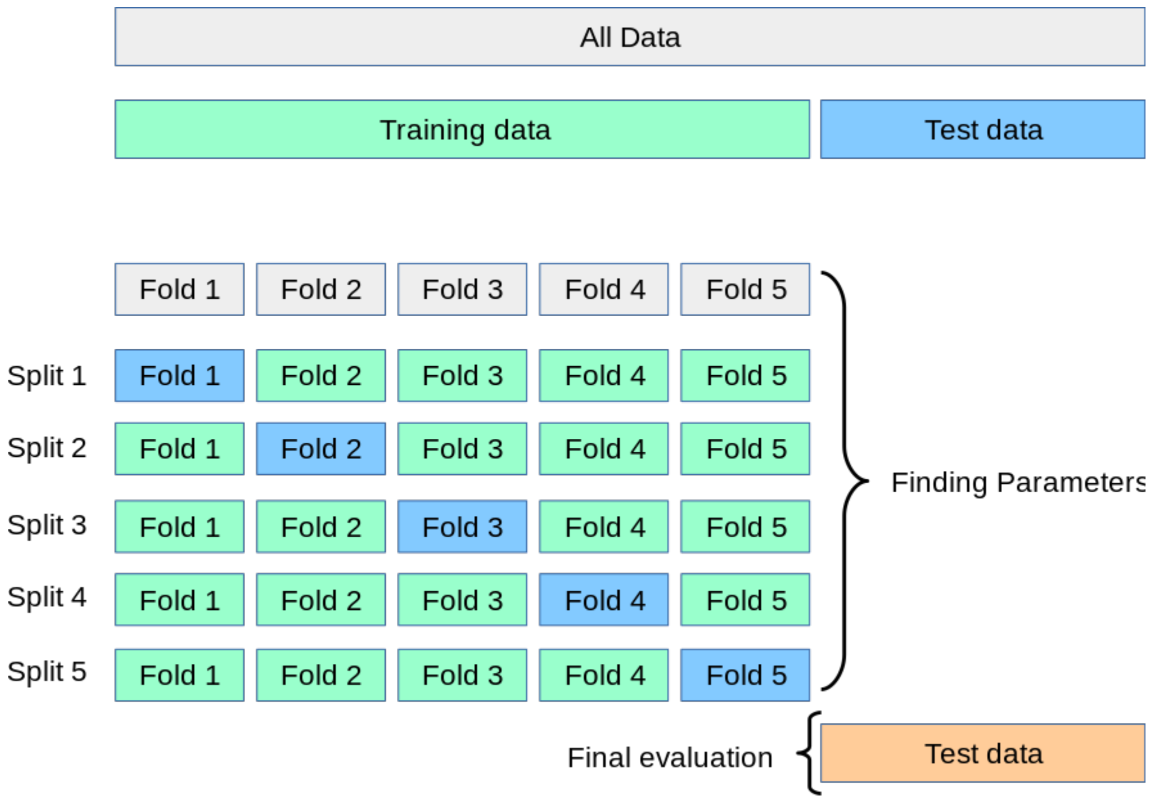


Figure 4.1: K-fold Cross Validation General Overview

The figure 3.1 gives a clear picture of how k-fold works. How the data is being separated initially and then on the training data how split is performed in each fold. And then in the end the trained data is being tested with some unseen data.

Our model's accuracy before k-fold cross validation was around 76%. Now by using k-fold cross validation, accuracy is hiked up to xx%. Basically, total 10 folds will be iterated in which, training dataset will get different images to train and testing dataset will get different images to test. Thus, maximize use of training dataset is being done to improve the accuracy of the model.

Analysis

In standard method of splitting the data i.e. “train_test_split” method of “sckitlearn” package, dataset was separated right before training the dataset(70% training and 30% testing) and the whole process was done once. Whereas in K-fold cross validation, dataset which we pass to the k-fold, it will take different images from available images, which helps model to train better which in turn helps model to improve its testing accuracy against unseen testing images.

Result of “train_test_split”

In project part, our dataset was not so much consistent and many images in the dataset were redundant. For that reason we added new images in that dataset and then checked for biases. As there was no bias in the model, re-training of data was not needed.

Here we are showing testing accuracy of “train_test_split” result where dataset was divide into 70 and 30% ratio.

Testing Classification Report (Part – 1):

```
              precision    recall  f1-score   support

           0       0.93      0.57      0.70       161
           1       0.80      0.89      0.84        84
           2       0.86      0.94      0.90        96
           3       0.46      0.81      0.58        57

    accuracy                           0.76       398
   macro avg       0.76      0.80      0.76       398
weighted avg       0.82      0.76      0.76       398
```

Result After applying k-fold cross validation

Here we are displaying the result of k-fold cross validation, where k = 10. In each fold, we have set 10 epochs. Also, at the end of each fold, we are applying “classification_method” to print accuracy, precision, recall and f1-score.

Fold Number: 1

```
Epoch [1/10], Loss: 0.4736, Accuracy: 76.67%
Epoch [2/10], Loss: 0.2583, Accuracy: 90.00%
Epoch [3/10], Loss: 0.2516, Accuracy: 90.00%
Epoch [4/10], Loss: 0.1601, Accuracy: 96.67%
Epoch [5/10], Loss: 0.1473, Accuracy: 100.00%
Epoch [6/10], Loss: 0.0719, Accuracy: 100.00%
Epoch [7/10], Loss: 0.1282, Accuracy: 96.67%
Epoch [8/10], Loss: 0.0461, Accuracy: 100.00%
Epoch [9/10], Loss: 0.0177, Accuracy: 100.00%
Epoch [10/10], Loss: 0.0142, Accuracy: 100.00%
```

Model_Number_1 Classification Report

```
              precision    recall  f1-score   support

           0       0.85      0.90      0.88        63
           1       0.86      0.86      0.86        73
           2       1.00      1.00      1.00        74
           3       0.95      0.89      0.92        64

    accuracy                           0.92       274
   macro avg       0.92      0.91      0.91       274
weighted avg       0.92      0.92      0.92       274
```

Fold Number: 2

```
Epoch [1/10], Loss: 0.3399, Accuracy: 86.67%
Epoch [2/10], Loss: 0.3323, Accuracy: 86.67%
Epoch [3/10], Loss: 0.2720, Accuracy: 90.00%
Epoch [4/10], Loss: 0.2107, Accuracy: 93.33%
Epoch [5/10], Loss: 0.1277, Accuracy: 96.67%
Epoch [6/10], Loss: 0.2095, Accuracy: 90.00%
Epoch [7/10], Loss: 0.1228, Accuracy: 93.33%
Epoch [8/10], Loss: 0.0698, Accuracy: 96.67%
Epoch [9/10], Loss: 0.0805, Accuracy: 96.67%
Epoch [10/10], Loss: 0.0807, Accuracy: 96.67%
```

Model_Number_2 Classification Report

```
              precision    recall  f1-score   support

           0       0.81      0.75      0.78        67
           1       0.71      0.88      0.79        51
           2       1.00      0.99      0.99        75
           3       0.97      0.90      0.94        81

    accuracy                           0.88       274
   macro avg       0.87      0.88      0.87       274
weighted avg       0.89      0.88      0.89       274
```

Fold Number: 3

```
Epoch [1/10], Loss: 0.4712, Accuracy: 83.33%
Epoch [2/10], Loss: 0.3407, Accuracy: 83.33%
Epoch [3/10], Loss: 0.2422, Accuracy: 96.67%
Epoch [4/10], Loss: 0.1515, Accuracy: 96.67%
Epoch [5/10], Loss: 0.2276, Accuracy: 93.33%
Epoch [6/10], Loss: 0.1220, Accuracy: 96.67%
Epoch [7/10], Loss: 0.2464, Accuracy: 86.67%
Epoch [8/10], Loss: 0.0784, Accuracy: 96.67%
Epoch [9/10], Loss: 0.0345, Accuracy: 100.00%
Epoch [10/10], Loss: 0.0670, Accuracy: 96.67%
```

Model_Number_3 Classification Report

```
              precision    recall  f1-score   support

           0       0.87      0.87      0.87        77
           1       0.86      0.79      0.82        53
           2       0.99      0.99      0.99        76
           3       0.94      1.00      0.97        68

    accuracy                           0.92       274
   macro avg       0.91      0.91      0.91       274
weighted avg       0.92      0.92      0.92       27
```

Fold Number: 4

```
Epoch [1/10], Loss: 0.4593, Accuracy: 86.67%
Epoch [2/10], Loss: 0.3125, Accuracy: 86.67%
Epoch [3/10], Loss: 0.2206, Accuracy: 93.33%
Epoch [4/10], Loss: 0.4172, Accuracy: 80.00%
Epoch [5/10], Loss: 0.1134, Accuracy: 96.67%
Epoch [6/10], Loss: 0.0239, Accuracy: 100.00%
Epoch [7/10], Loss: 0.0532, Accuracy: 100.00%
Epoch [8/10], Loss: 0.1276, Accuracy: 93.33%
Epoch [9/10], Loss: 0.0531, Accuracy: 100.00%
Epoch [10/10], Loss: 0.0616, Accuracy: 100.00%
```

Model_Number_4 Classification Report

```
              precision    recall  f1-score   support

           0       0.84      0.91      0.88        70
           1       0.90      0.82      0.86        66
           2       1.00      1.00      1.00        62
           3       0.99      0.99      0.99        76

    accuracy                           0.93       274
   macro avg       0.93      0.93      0.93       274
weighted avg       0.93      0.93      0.93       274
```

Fold Number: 5

```
Epoch [1/10], Loss: 0.4683, Accuracy: 83.33%
Epoch [2/10], Loss: 0.3522, Accuracy: 83.33%
Epoch [3/10], Loss: 0.3071, Accuracy: 83.33%
Epoch [4/10], Loss: 0.1279, Accuracy: 100.00%
Epoch [5/10], Loss: 0.0757, Accuracy: 100.00%
Epoch [6/10], Loss: 0.1729, Accuracy: 96.67%
Epoch [7/10], Loss: 0.0543, Accuracy: 96.67%
Epoch [8/10], Loss: 0.1021, Accuracy: 96.67%
Epoch [9/10], Loss: 0.0664, Accuracy: 100.00%
Epoch [10/10], Loss: 0.0613, Accuracy: 100.00%
```

Model_Number_5 Classification Report

```
              precision    recall  f1-score   support
```

```
           0       0.88      0.73      0.80        86
           1       0.55      0.79      0.65        39
           2       1.00      1.00      1.00        70
           3       0.99      0.95      0.97        79

    accuracy                           0.87       274
   macro avg       0.85      0.87      0.85       274
weighted avg       0.89      0.87      0.88       274
```

Fold Number: 6

```
Epoch [1/10], Loss: 0.3756, Accuracy: 90.00%
Epoch [2/10], Loss: 0.3853, Accuracy: 83.33%
Epoch [3/10], Loss: 0.4318, Accuracy: 76.67%
Epoch [4/10], Loss: 0.1174, Accuracy: 93.33%
Epoch [5/10], Loss: 0.0450, Accuracy: 100.00%
Epoch [6/10], Loss: 0.1500, Accuracy: 93.33%
Epoch [7/10], Loss: 0.0551, Accuracy: 100.00%
Epoch [8/10], Loss: 0.0725, Accuracy: 96.67%
Epoch [9/10], Loss: 0.1788, Accuracy: 90.00%
Epoch [10/10], Loss: 0.0183, Accuracy: 100.00%
```

Model_Number_6 Classification Report

```
              precision    recall  f1-score   support

           0       0.89      0.85      0.87        60
           1       0.85      0.92      0.88        60
           2       1.00      1.00      1.00        74
           3       0.99      0.96      0.97        80

    accuracy                           0.94       274
   macro avg       0.93      0.93      0.93       274
weighted avg       0.94      0.94      0.94       274
```

Fold Number: 7

```
Epoch [1/10], Loss: 0.2761, Accuracy: 90.32%
Epoch [2/10], Loss: 0.4362, Accuracy: 83.87%
Epoch [3/10], Loss: 0.2172, Accuracy: 93.55%
Epoch [4/10], Loss: 0.1935, Accuracy: 96.77%
Epoch [5/10], Loss: 0.0938, Accuracy: 96.77%
Epoch [6/10], Loss: 0.1900, Accuracy: 90.32%
Epoch [7/10], Loss: 0.1208, Accuracy: 96.77%
Epoch [8/10], Loss: 0.0319, Accuracy: 100.00%
Epoch [9/10], Loss: 0.1488, Accuracy: 93.55%
Epoch [10/10], Loss: 0.0298, Accuracy: 100.00%
```

Model_Number_7 Classification Report

```
              precision    recall  f1-score   support

           0       0.85      0.82      0.83        88
           1       0.73      0.84      0.78        55
           2       1.00      0.97      0.98        63
           3       0.97      0.93      0.95        67

    accuracy                           0.88       273
   macro avg       0.89      0.89      0.89       273
weighted avg       0.89      0.88      0.88       273
```

Fold Number: 8

```
Epoch [1/10], Loss: 0.3956, Accuracy: 80.65%
Epoch [2/10], Loss: 0.2223, Accuracy: 93.55%
Epoch [3/10], Loss: 0.3033, Accuracy: 90.32%
Epoch [4/10], Loss: 0.3159, Accuracy: 90.32%
Epoch [5/10], Loss: 0.0892, Accuracy: 96.77%
Epoch [6/10], Loss: 0.1025, Accuracy: 96.77%
Epoch [7/10], Loss: 0.0473, Accuracy: 100.00%
Epoch [8/10], Loss: 0.0486, Accuracy: 100.00%
Epoch [9/10], Loss: 0.0231, Accuracy: 100.00%
Epoch [10/10], Loss: 0.0467, Accuracy: 100.00%
```

Model_Number_8 Classification Report

```
              precision    recall  f1-score   support

           0       0.85      0.85      0.85        82
           1       0.77      0.80      0.78        61
           2       1.00      1.00      1.00        65
           3       0.95      0.91      0.93        65

    accuracy                           0.89       273
   macro avg       0.89      0.89      0.89       273
weighted avg       0.89      0.89      0.89       273
```

Fold Number: 9

```
Epoch [1/10], Loss: 0.5078, Accuracy: 80.65%
Epoch [2/10], Loss: 0.2683, Accuracy: 87.10%
Epoch [3/10], Loss: 0.3082, Accuracy: 90.32%
Epoch [4/10], Loss: 0.3970, Accuracy: 83.87%
Epoch [5/10], Loss: 0.1169, Accuracy: 100.00%
Epoch [6/10], Loss: 0.1197, Accuracy: 93.55%
Epoch [7/10], Loss: 0.1487, Accuracy: 93.55%
Epoch [8/10], Loss: 0.0319, Accuracy: 100.00%
Epoch [9/10], Loss: 0.0572, Accuracy: 96.77%
Epoch [10/10], Loss: 0.0203, Accuracy: 100.00%
```

Model_Number_9 Classification Report

```
              precision    recall  f1-score   support

           0       0.96      0.86      0.91        85
           1       0.78      0.98      0.87        43
           2       1.00      1.00      1.00        85
           3       0.95      0.92      0.93        60

    accuracy                           0.93       273
   macro avg       0.92      0.94      0.93       273
weighted avg       0.94      0.93      0.93       273
```

Fold Number: 10

```
Epoch [1/10], Loss: 0.2333, Accuracy: 93.55%
Epoch [2/10], Loss: 0.3288, Accuracy: 93.55%
Epoch [3/10], Loss: 0.3170, Accuracy: 83.87%
Epoch [4/10], Loss: 0.2759, Accuracy: 87.10%
Epoch [5/10], Loss: 0.1428, Accuracy: 93.55%
Epoch [6/10], Loss: 0.3104, Accuracy: 87.10%
Epoch [7/10], Loss: 0.1477, Accuracy: 96.77%
Epoch [8/10], Loss: 0.1286, Accuracy: 93.55%
Epoch [9/10], Loss: 0.0533, Accuracy: 100.00%
Epoch [10/10], Loss: 0.0513, Accuracy: 100.00%
```

Model_Number_10 Classification Report

```
              precision    recall  f1-score   support

           0       0.82      0.83      0.83        72
           1       0.79      0.79      0.79        63
           2       1.00      0.97      0.98        61
           3       0.96      0.97      0.97        77

    accuracy                           0.89       273
   macro avg       0.89      0.89      0.89       273
weighted avg       0.89      0.89      0.89       273
```

Here we saw all the 10 folds of training of our dataset. So from the data, one get a glimpse of training accuracy, precision, recall, f1-score, and training loss at the end of each fold.

As we can notice that by using k-fold cross validation in our project, we get a very good accuracy. Initially accuracy of our model was about 76% but now because of using k-fold cross validation and increasing the number of count of images in the dataset, led us to get a good accuracy.

Evaluation

After training the model, we need to check whether the model was trained well by evaluating various factors like accuracy, precision, f1-score and support. The model is trained in 10 different folds, where in each fold 10 epochs will be executed, the learning rate was 0.0001. We have calculated all this with factors with the help of classification_report() method. At last, we build the confusion matrix using confusion_matrix() method. The confusion matrix of the model can be useful to know the precision which in our case shows that many testing set images were misclassified due to imbalanced data. The confusion matrix of the model can be useful to know the precision. In our case, accuracy of training dataset is 100% and testing dataset is 89%.

Training Phase

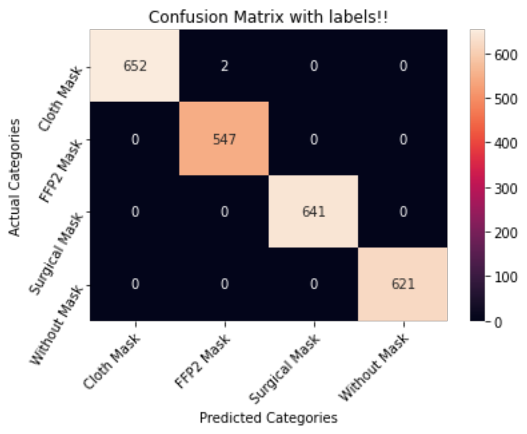
Training Classification Report

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       652
           1       1.00      1.00      1.00       549
           2       1.00      1.00      1.00       641
           3       1.00      1.00      1.00       621

    accuracy                           1.00      2463
   macro avg       1.00      1.00      1.00      2463
weighted avg       1.00      1.00      1.00      2463
```

Training Confusion Matrix



5.1 Training Confusion Matrix

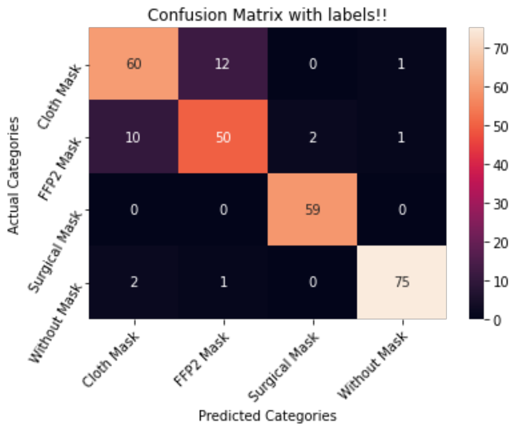
Testing Classification Report

```
              precision    recall  f1-score   support

           0       0.82      0.83      0.83        72
           1       0.79      0.79      0.79        63
           2       1.00      0.97      0.98        61
           3       0.96      0.97      0.97        77

    accuracy                           0.89       273
   macro avg       0.89      0.89      0.89       273
weighted avg       0.89      0.89      0.89       273
```

Testing Confusion Matrix:



5.2: Testing Confusion Matrix

References

Dataset References

- https://www.kaggle.com/dhruvmak/face-mask-detection
- https://www.kaggle.com/niharika41298/withwithout-mask
- https://www.kaggle.com/vijaykumar1799/face-mask-detection
- https://www.kaggle.com/omkargurav/face-mask-dataset
- https://www.google.com/search?q=people+wearing+ffp2+mask&rlz=1C1CHBF_enIN840IN840&sxsrf=AOaemvKKrYHhWB4Ir1t47qNJt73xqnFn-g:1636268752963&source=lnms&tbm=isch&sa=X&ved=2ahUKEwj_tJuO2IX0AhV3mHIEHaB6CYgQ_AUoAXoECAEQAw&biw=1366&bih=610&dpr=1
- https://www.google.com/search?q=people+wearing+cloth+mask&tbm=isch&ved=2ahUKEwiT-a6P2IX0AhUyg3IEHQGDATMQ2-cCegQIABAA&oq=people+wearing+cloth+mask&gs_lcp=CgNpbWcQAzIFCAAQgAQ6BwgjEO8DECc6BggAEAcQHjoICAAQCBAHEB5Q5whY_Rlgjx1oAHAAeACAAVmIAYoFkgECMTCYAQCgAQGqAQtnd3Mtd2l6LWltZ8ABAQ&sclient=img&ei=03qHYdPDFrKGytMPgYaGmAM&bih=610&biw=1366&rlz=1C1CHBF_enIN840IN840
- https://www.google.com/search?q=people+wearing+N95+mask&rlz=1C1CHBF_enIN840IN840&source=lnms&tbm=isch&sa=X&ved=2ahUKEwi15KyF_Ib0AhUQheAKHUQ6BsYQ_AUoAXoECAEQAw&biw=1366&bih=635&dpr=1
- https://github.com/ostibhisma/surgical-mask-detection-system/tree/master/dataset/dataset

Other References

- https://towardsdatascience.com/machine-learning-part-20-dropout-keras-layers-explained-8c9f6dc4c9ab
- https://medium.com/sfu-cspmp/model-transparency-fairness-552a747b444
- https://scikit-learn.org/stable/modules/cross_validation.html
- https://github.com/aqeelanwar/MaskTheFace/
- https://susanqq.github.io/UTKFace/