Concordia University

Engineering and Computer Science

COMP 6721

Applied Artificial Intelligence

Project-2 Report

Submitted To: Prof. Dr. René Witte

Team Id: RN_08

Raj Mistry (40119206) - Data Specialist

Yash Jayeshkumar Pandya (40119272) - Training Specialist

Piyush Thummar (40125029) - Evaluation Specialist

Dataset:

AI is the field that its projects are mostly dependent on datasets. By that, it can be deduced that to get good results from AI projects, we need a very well balanced and good dataset, because then only, the model can be trained for almost every probable outcome/result/class. Dataset performs the key role in the training and testing phase.

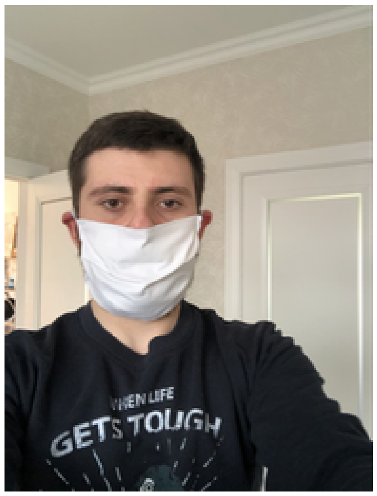
Here, we need to divide our dataset in the balanced manner for these three phases for precision and accuracy. In the ideal case, the whole dataset can be divided into training dataset and testing dataset.

- **Training phase:** Here, the model is trained using a dataset and algorithm or pattern is built by the AI to distinguish objects.
- **Testing phase:** In this step, the trained model gets tested for the unknown data, that is isolated from the dataset to measure the performance of the model or algorithm using several performance matrices.

In this project, we have used 3000 samples for each three classes that are person with mask, person without mask and not a person. We have used the balanced dataset with different varieties such as age, color, gender, etc. Some dataset has been taken from the data-repositories, for example, kaggle. And for some, we cannot find such repositories, where we have to come up with plenty of images from different sources from google or the internet.

Note : *Dataset links are added in the reference section.*

For example the training dataset contains following types of images for each category:



Person with Mask

Person Without Mask

Not a Person

CNN Architecture:

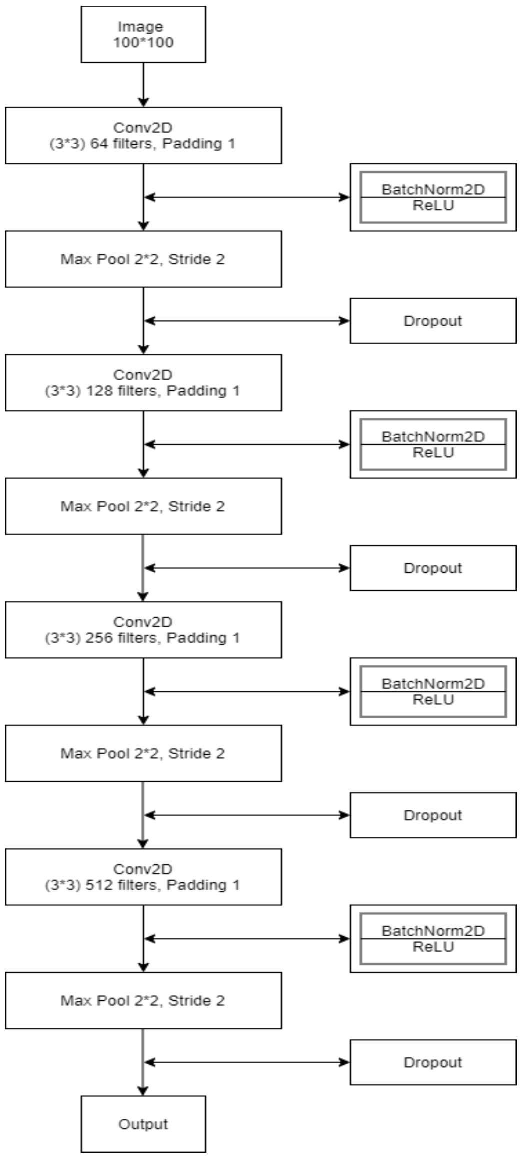
Here, we have implemented the Convolutional Neural Network to train the model for given classes. First of all, the given dataset must be preprocessed to make them balanced and of equal size(100*100). Here, 3 different datasets for each class will be processed, and their paths will be set. Those classes are labeled as below:

0: Person with Face Mask
1: Person without Face Mask
2: Not a Person

Now, we divide our dataset in 10 groups to do 10-fold cross validation. The weight information is passed to the loss function, which can be done via assigning the weight to each class according to its representability in the dataset.

After this step, we load our training and testing dataset, and convert it to tensor with a label, and then load it using a data loader with batch size. Here, we load data with training and testing data loaders.

As per the CNN diagram, it can be depicted that the firstly, input image with input channel of 100 is supplied to our 1st Convolutional 2D layer with RGB channel and its output channel of 64, and padding 1. Now, this is normalized and provided as input to ReLU activation function. Then, its output is provided for MaxPooling with stride 2 along with dropout, where its size will be minimized and then it will be passed to the 2nd Convolutional 2D Layer. Here, its output channel will be 128 with padding 1, and then next steps are as above, which are ReLU activation function and MaxPooling along with dropout. Then, it is passed to our 3rd Convolutional Layer, which produces the output with 256 channels and padding 1. And again, ReLU activation function and MaxPooling takes place along with dropout.Then, it is passed to our 4th Convolutional Layer, which produces the output with 512 channels and padding 1. And again, ReLU activation function and MaxPooling takes place along with dropout. Then, at last we get the output (0/1/2) label, which represents “Masked Person”, “Person without Mask”, and “Not a Person”.



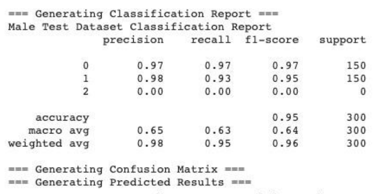
Bias:

Here, in this project, we can introduce a lot of bias such as gender, age groups, or race. For our project, we considered the gender bias to analyze the project. Firstly, we balanced the dataset by undertaking an almost equal size of data for male and female categories in all classes: person and person with mask. We trained the model again with this new dataset while considering the gender bias, with 10-fold cross validation for 10 epochs at each fold. We achieved almost the same accuracy for both the genders, and that is how, we can say that we eliminated gender bias.

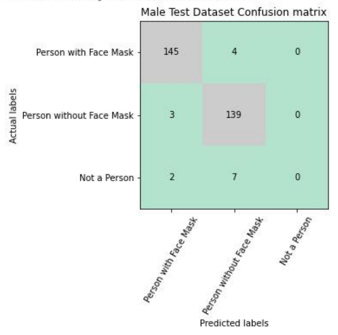
We have an equal amount of 1500 Male images and 1500 Female images for each label class such as a person with mask and without mask. 300 Images in Male and Female categories used for testing purpose and remaining are used in the training process.

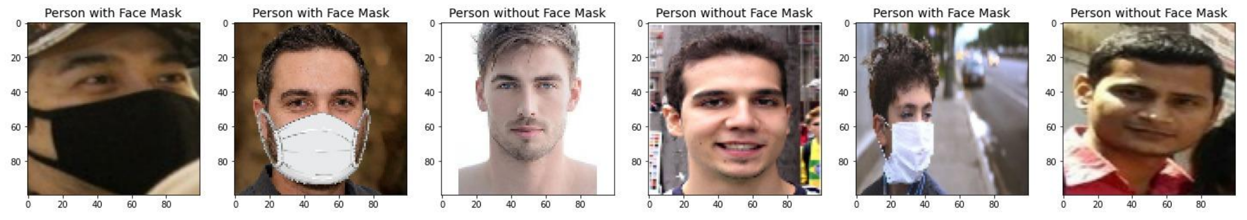
We have also performed separate testing on the images which are not part of any training process in KFold to review the results of the bias removal feature. Below are the results from the data passed to the KFold, please note that these results do not include the test data which is not part of any analysis process except final testing on the model obtained after the 10th Fold.

The confusion matrix and classification report considering Precision, Recall and F1-Score with supporting samples after bias elimination is shown as below:

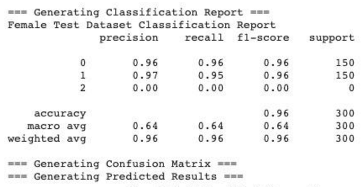


```
=== Generating Classification Report ===
Male Test Dataset Classification Report
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       150
           1       0.98      0.93      0.95       150
           2       0.00      0.00      0.00         0

    accuracy                           0.95       300
   macro avg       0.65      0.63      0.64       300
weighted avg       0.98      0.95      0.96       300

=== Generating Confusion Matrix ===
=== Generating Predicted Results ===
```

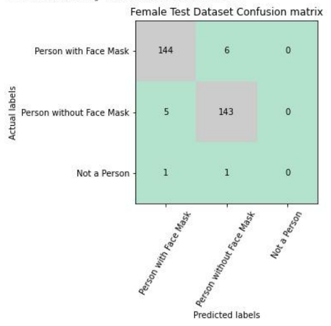


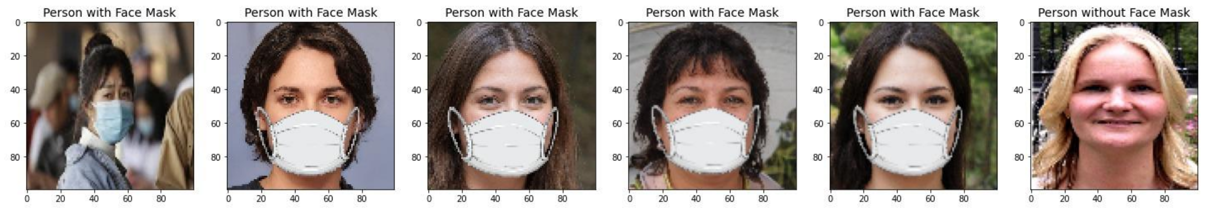


Classification report and Confusion Matrix of Male test data for bias of our project-part-2 model



```
=== Generating Classification Report ===
Female Test Dataset Classification Report
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       150
           1       0.97      0.95      0.96       150
           2       0.00      0.00      0.00         0

    accuracy                           0.96       300
   macro avg       0.64      0.64      0.64       300
weighted avg       0.96      0.96      0.96       300

=== Generating Confusion Matrix ===
=== Generating Predicted Results ===
```





Classification report and Confusion Matrix of Female test data for bias of our project-part-2 model

Here, we can clearly deduce from the given statistics and classification report that our model eliminated the gender bias smoothly by having properly distributed and balanced dataset. Numbers indicated that 95% of accuracy is achieved for Male dataset, which is almost equal to accuracy of Female dataset, that is 96%. Hence, it can be said that our trained model of project part 2 is not gender biased.

k-Fold Cross Validation:

Cross-validation is a statistical method used to estimate the skill of machine learning models. This resampling procedure used to evaluate machine learning models on a limited data sample. It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

Here, in this part 2 of the project, we performed the k-Fold (Here, we used k=10. So, 10-Fold) Cross validation to train our model more accurately. To do so, our dataset is divided into 10 groups, where 9 of them are used for training and the last one is for testing. Now, considering other 9 for training, and the second last is for testing, and so on. This way our model gets more training for the dataset, and we may achieve better performance afterwards for real testing. Here, after applying 10-fold cross validation, we got more accuracy for training and testing both.

The results are attached below for reference.

Fold No	Training Accuracy
1	95.92
2	99.97
3	99.97
4	99.98
5	99.95
6	99.97
7	99.98
8	100
9	99.98
10	99.98

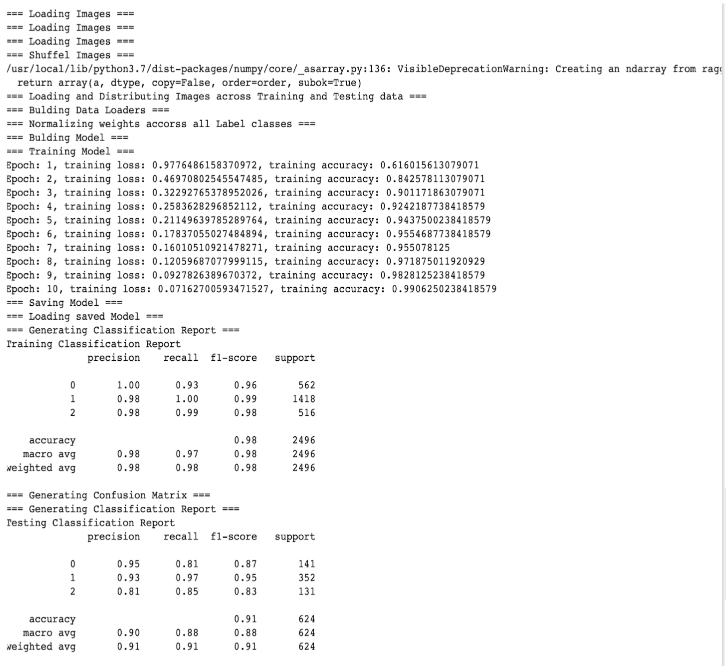
Here, for this project, we have used the KFold class of the sklearn library's model_selection package to divide the dataset in 10 folds(because here, we used k=10). Moreover, every fold will be executed in 10 epochs. So, the model gets a bunch of training data. Hence, we can see from the above classification report that for training, the model is giving 100% accuracy; however, 96% of accuracy is achieved for testing also compared to 98% and 91% for training and testing accuracy in the previous model from project part 1

We are also having improved confusion matrices with most likely accurate predictions with the latest split procedure. The main reason can be that the datasets for all 3 classes are equally balanced and as mentioned before, each class is given an equal chance to be trained and tested which helps the model to predict more accurately.

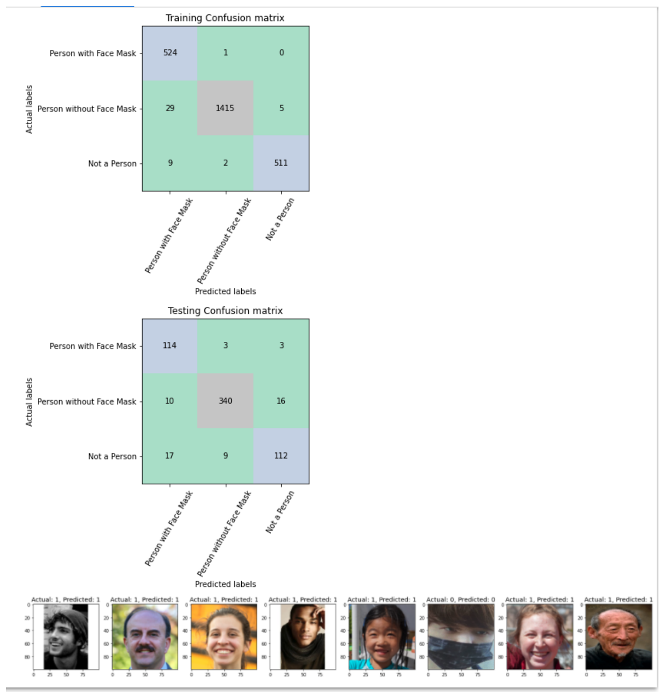
Evaluation:

Part-1

After these three phases, we can evaluate the algorithm or the model by using confusion matrix, and several other matrices for its performance. In this project, the model is trained for 10 epochs. In every epoch, there is learning at the rate of 0.0001. Here, we evaluated the performance using recall, precision, accuracy and f1-measure for all phases. By using a confusion matrix, we can say some of the images have been misclassified due to imbalance data, that means we have 3rd class (not a person) as a more generic one, where we may not have enough data. Which can lead the model for this confusion and misclassification. Here, we can see that our model was **trained with the accuracy of 98%**; whereas, for **testing, we have the accuracy of 91%**.



```
=== Loading Images ===
=== Loading Images ===
=== Loading Images ===
=== Shuffel Images ===
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:136: VisibleDeprecationWarning: Creating an ndarray from rag
  return array(a, dtype, copy=False, order=order, subok=True)
=== Loading and Distributing Images across Training and Testing data ===
=== Bulding Data Loaders ===
=== Normalizing weights accorss all Label classes ===
=== Bulding Model ===
=== Training Model ===
Epoch: 1, training loss: 0.9776486158370972, training accuracy: 0.616015613079071
Epoch: 2, training loss: 0.46970802545547485, training accuracy: 0.842578113079071
Epoch: 3, training loss: 0.32292765378952026, training accuracy: 0.901171863079071
Epoch: 4, training loss: 0.2583628296852112, training accuracy: 0.9242187738418579
Epoch: 5, training loss: 0.21149639785289764, training accuracy: 0.9437500238418579
Epoch: 6, training loss: 0.17837055027484894, training accuracy: 0.9554687738418579
Epoch: 7, training loss: 0.16010510921478271, training accuracy: 0.955078125
Epoch: 8, training loss: 0.12059687077999115, training accuracy: 0.971875011920929
Epoch: 9, training loss: 0.0927826389670372, training accuracy: 0.9828125238418579
Epoch: 10, training loss: 0.07162700593471527, training accuracy: 0.9906250238418579
=== Saving Model ===
=== Loading saved Model ===
=== Generating Classification Report ===
Training Classification Report
              precision    recall  f1-score   support

           0       1.00      0.93      0.96       562
           1       0.98      1.00      0.99      1418
           2       0.98      0.99      0.98       516

    accuracy                           0.98      2496
   macro avg       0.98      0.97      0.98      2496
weighted avg       0.98      0.98      0.98      2496

=== Generating Confusion Matrix ===
=== Generating Classification Report ===
Testing Classification Report
              precision    recall  f1-score   support

           0       0.95      0.81      0.87       141
           1       0.93      0.97      0.95       352
           2       0.81      0.85      0.83       131

    accuracy                           0.91       624
   macro avg       0.90      0.88      0.88       624
weighted avg       0.91      0.91      0.91       624
```

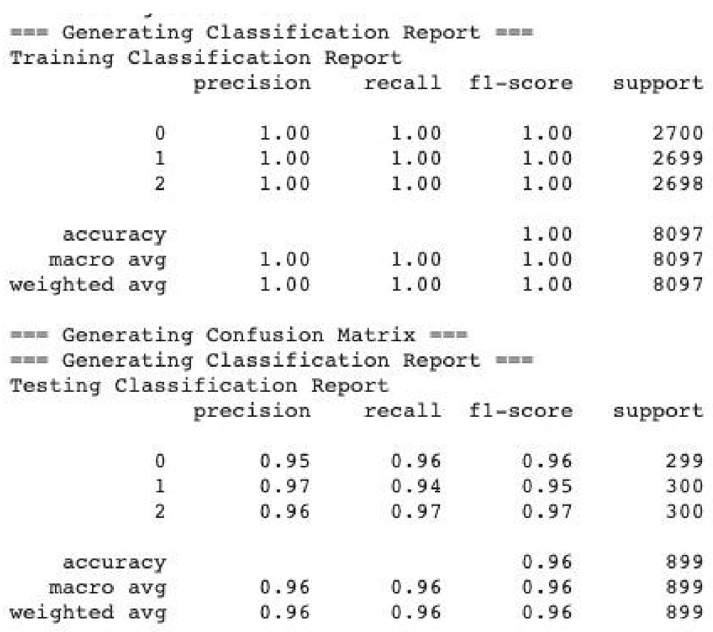


Classification report and Confusion Matrix of our project-part-1 model

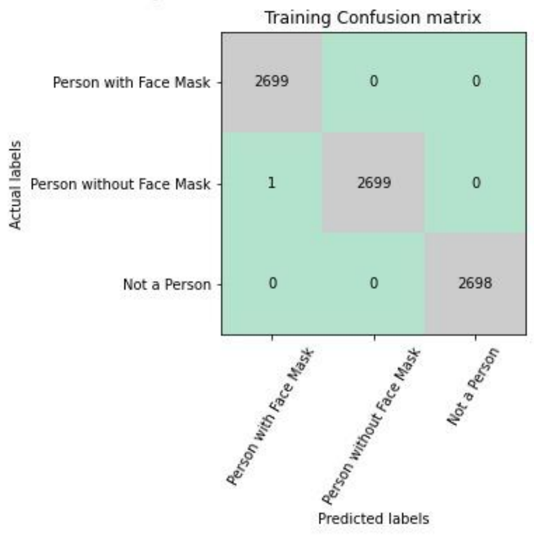
Part-2

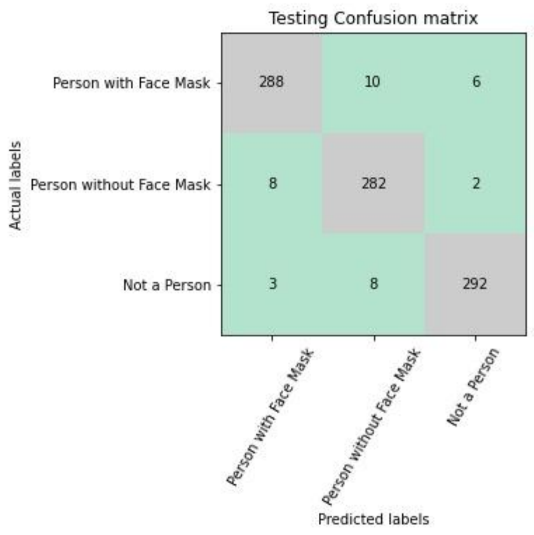
For the second part of the project, we made the statistics and evaluation in the k-Fold and bias sections. To sum it up, it can be indicated that by implementing k-Fold and eliminating the gender bias, we got a significant improvement in our model. To support that, we can use evaluation matrices for both the parts of the project.

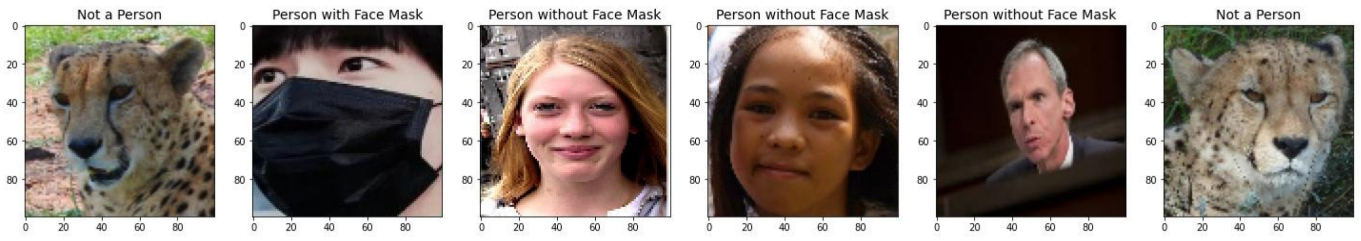
For instance, we improved the overall accuracy of the model for **training friom 98% to 100%**, and for **testing, it is from 91% to 96%**.



```
=== Generating Classification Report ===
Training Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2700
           1       1.00      1.00      1.00      2699
           2       1.00      1.00      1.00      2698

    accuracy                           1.00      8097
   macro avg       1.00      1.00      1.00      8097
weighted avg       1.00      1.00      1.00      8097

=== Generating Confusion Matrix ===
=== Generating Classification Report ===
Testing Classification Report
              precision    recall  f1-score   support

           0       0.95      0.96      0.96       299
           1       0.97      0.94      0.95       300
           2       0.96      0.97      0.97       300

    accuracy                           0.96       899
   macro avg       0.96      0.96      0.96       899
weighted avg       0.96      0.96      0.96       899
```

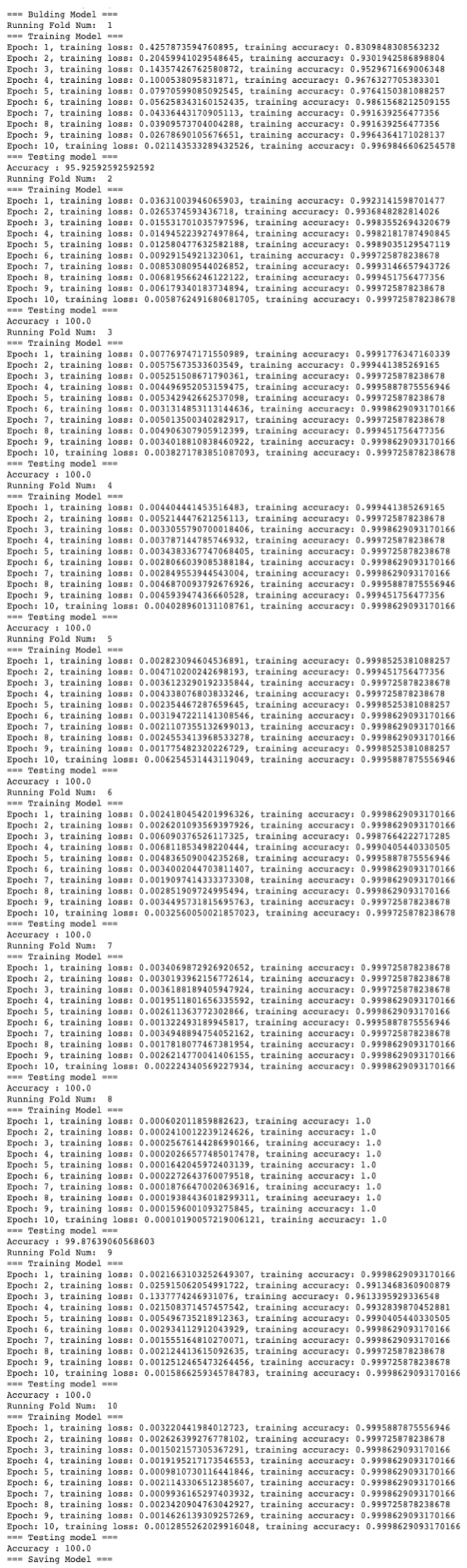






Classification report and Confusion Matrix of our project-part-2 model

Appendix:



```
=== Bulding Model ===
Running Fold Num:  1
=== Training Model ===
Epoch: 1, training loss: 0.4257873594760895, training accuracy: 0.8309848308563232
Epoch: 2, training loss: 0.20459941029548645, training accuracy: 0.9301942586898804
Epoch: 3, training loss: 0.14357426762580872, training accuracy: 0.9529671669006348
Epoch: 4, training loss: 0.1000538095831871, training accuracy: 0.9676327705383301
Epoch: 5, training loss: 0.07970599085092545, training accuracy: 0.9764150381088257
Epoch: 6, training loss: 0.056258343160152435, training accuracy: 0.9861568212509155
Epoch: 7, training loss: 0.04336443170905113, training accuracy: 0.991639256477356
Epoch: 8, training loss: 0.03909573704004288, training accuracy: 0.991639256477356
Epoch: 9, training loss: 0.02678690105676651, training accuracy: 0.9964364171028137
Epoch: 10, training loss: 0.021143533289432526, training accuracy: 0.9969846606254578
=== Testing model ===
Accuracy : 95.92592592592592
Running Fold Num:  2
=== Training Model ===
Epoch: 1, training loss: 0.03631003946065903, training accuracy: 0.9923141598701477
Epoch: 2, training loss: 0.0265374593436718, training accuracy: 0.9936848282814026
Epoch: 3, training loss: 0.015531701035797596, training accuracy: 0.9983552694320679
Epoch: 4, training loss: 0.014945223927497864, training accuracy: 0.9982181787490845
Epoch: 5, training loss: 0.012580477632582188, training accuracy: 0.9989035129547119
Epoch: 6, training loss: 0.00929154921323061, training accuracy: 0.999725878238678
Epoch: 7, training loss: 0.008530809544026852, training accuracy: 0.9993146657943726
Epoch: 8, training loss: 0.006819566246122122, training accuracy: 0.999451756477356
Epoch: 9, training loss: 0.006179340183734894, training accuracy: 0.999725878238678
Epoch: 10, training loss: 0.0058762491680681705, training accuracy: 0.999725878238678
=== Testing model ===
Accuracy : 100.0
Running Fold Num:  3
=== Training Model ===
Epoch: 1, training loss: 0.007769747171550989, training accuracy: 0.9991776347160339
Epoch: 2, training loss: 0.005756735336O3549, training accuracy: 0.999441385269165
Epoch: 3, training loss: 0.005251508671790361, training accuracy: 0.999725878238678
Epoch: 4, training loss: 0.004496952053159475, training accuracy: 0.9995887875556946
Epoch: 5, training loss: 0.005342942662537098, training accuracy: 0.999725878238678
Epoch: 6, training loss: 0.0031314853113144636, training accuracy: 0.9998629093170166
Epoch: 7, training loss: 0.005013500340282917, training accuracy: 0.999725878238678
Epoch: 8, training loss: 0.004906307905912399, training accuracy: 0.999451756477356
Epoch: 9, training loss: 0.0034018810838460922, training accuracy: 0.9998629093170166
Epoch: 10, training loss: 0.0038271783851087093, training accuracy: 0.999725878238678
=== Testing model ===
Accuracy : 100.0
Running Fold Num:  4
=== Training Model ===
Epoch: 1, training loss: 0.004404441453516483, training accuracy: 0.999441385269165
Epoch: 2, training loss: 0.005214447621256113, training accuracy: 0.999725878238678
Epoch: 3, training loss: 0.0033055790700018406, training accuracy: 0.9998629093170166
Epoch: 4, training loss: 0.003787144785746932, training accuracy: 0.999725878238678
Epoch: 5, training loss: 0.0034383367747068405, training accuracy: 0.999725878238678
Epoch: 6, training loss: 0.0028066039085388184, training accuracy: 0.9998629093170166
Epoch: 7, training loss: 0.002849553944543004, training accuracy: 0.9998629093170166
Epoch: 8, training loss: 0.0046870093792676926, training accuracy: 0.9995887875556946
Epoch: 9, training loss: 0.004593947436660528, training accuracy: 0.999451756477356
Epoch: 10, training loss: 0.0040289601311O8761, training accuracy: 0.9998629093170166
=== Testing model ===
Accuracy : 100.0
Running Fold Num:  5
=== Training Model ===
Epoch: 1, training loss: 0.002823094604536891, training accuracy: 0.9998525381088257
Epoch: 2, training loss: 0.004710200242698193, training accuracy: 0.999451756477356
Epoch: 3, training loss: 0.0036123290192335844, training accuracy: 0.999725878238678
Epoch: 4, training loss: 0.004338076803833246, training accuracy: 0.999725878238678
Epoch: 5, training loss: 0.002354467287659645, training accuracy: 0.9998525381088257
Epoch: 6, training loss: 0.0031947221141308546, training accuracy: 0.9998629093170166
Epoch: 7, training loss: 0.0021107355132699013, training accuracy: 0.9998629093170166
Epoch: 8, training loss: 0.0024553413968533278, training accuracy: 0.9998629093170166
Epoch: 9, training loss: 0.001775482320226729, training accuracy: 0.9998525381088257
Epoch: 10, training loss: 0.006254531443119049, training accuracy: 0.9995887875556946
=== Testing model ===
Accuracy : 100.0
Running Fold Num:  6
=== Training Model ===
Epoch: 1, training loss: 0.0024180454201996326, training accuracy: 0.9998629093170166
Epoch: 2, training loss: 0.0026201093569397926, training accuracy: 0.9998629093170166
Epoch: 3, training loss: 0.006090376526117325, training accuracy: 0.9987664222717285
Epoch: 4, training loss: 0.006811853498220444, training accuracy: 0.9990405440330505
Epoch: 5, training loss: 0.004836509004235268, training accuracy: 0.9995887875556946
Epoch: 6, training loss: 0.0034002044703811407, training accuracy: 0.9998629093170166
Epoch: 7, training loss: 0.0019097414333373308, training accuracy: 0.9998629093170166
Epoch: 8, training loss: 0.002851909724995494, training accuracy: 0.9998629093170166
Epoch: 9, training loss: 0.0034495731815695763, training accuracy: 0.999725878238678
Epoch: 10, training loss: 0.0032560050021857023, training accuracy: 0.999725878238678
=== Testing model ===
Accuracy : 100.0
Running Fold Num:  7
=== Training Model ===
Epoch: 1, training loss: 0.0034069872926920652, training accuracy: 0.999725878238678
Epoch: 2, training loss: 0.0030193962156772614, training accuracy: 0.999725878238678
Epoch: 3, training loss: 0.0036188189405947924, training accuracy: 0.999725878238678
Epoch: 4, training loss: 0.0019511801656335592, training accuracy: 0.9998629093170166
Epoch: 5, training loss: 0.002611363772302866, training accuracy: 0.9998629093170166
Epoch: 6, training loss: 0.001322493189945817, training accuracy: 0.9995887875556946
Epoch: 7, training loss: 0.0034948894754052162, training accuracy: 0.999725878238678
Epoch: 8, training loss: 0.0017818077467381954, training accuracy: 0.9998629093170166
Epoch: 9, training loss: 0.0026214770041406155, training accuracy: 0.9998629093170166
Epoch: 10, training loss: 0.002224340569227934, training accuracy: 0.9998629093170166
=== Testing model ===
Accuracy : 100.0
Running Fold Num:  8
=== Training Model ===
Epoch: 1, training loss: 0.000602011859882623, training accuracy: 1.0
Epoch: 2, training loss: 0.0002410012239124626, training accuracy: 1.0
Epoch: 3, training loss: 0.00025676144286990166, training accuracy: 1.0
Epoch: 4, training loss: 0.00020266577485017478, training accuracy: 1.0
Epoch: 5, training loss: 0.0001642045972403139, training accuracy: 1.0
Epoch: 6, training loss: 0.0002272643760079518, training accuracy: 1.0
Epoch: 7, training loss: 0.00018766470020636916, training accuracy: 1.0
Epoch: 8, training loss: 0.00019384436018299311, training accuracy: 1.0
Epoch: 9, training loss: 0.0001596001093275845, training accuracy: 1.0
Epoch: 10, training loss: 0.00010190057219006121, training accuracy: 1.0
=== Testing model ===
Accuracy : 99.87639060568603
Running Fold Num:  9
=== Training Model ===
Epoch: 1, training loss: 0.0021663103252649307, training accuracy: 0.9998629093170166
Epoch: 2, training loss: 0.025915062054991722, training accuracy: 0.9913468360900879
Epoch: 3, training loss: 0.1337774246931076, training accuracy: 0.9613395929336548
Epoch: 4, training loss: 0.021508371457457542, training accuracy: 0.9932839870452881
Epoch: 5, training loss: 0.005496735218912363, training accuracy: 0.9990405440330505
Epoch: 6, training loss: 0.002934112912043929, training accuracy: 0.9998629093170166
Epoch: 7, training loss: 0.001555164810270071, training accuracy: 0.9998629093170166
Epoch: 8, training loss: 0.002124413615092635, training accuracy: 0.999725878238678
Epoch: 9, training loss: 0.0012512465473264456, training accuracy: 0.999725878238678
Epoch: 10, training loss: 0.0015866259345784783, training accuracy: 0.9998629093170166
=== Testing model ===
Accuracy : 100.0
Running Fold Num:  10
=== Training Model ===
Epoch: 1, training loss: 0.003220441984012723, training accuracy: 0.9995887875556946
Epoch: 2, training loss: 0.002626399276778102, training accuracy: 0.999725878238678
Epoch: 3, training loss: 0.001502157305367291, training accuracy: 0.9998629093170166
Epoch: 4, training loss: 0.0019195217173546553, training accuracy: 0.9998629093170166
Epoch: 5, training loss: 0.0009810730116441846, training accuracy: 0.9998629093170166
Epoch: 6, training loss: 0.0021143306512385607, training accuracy: 0.9998629093170166
Epoch: 7, training loss: 0.0009936165297403932, training accuracy: 0.9998629093170166
Epoch: 8, training loss: 0.0023420904763042927, training accuracy: 0.999725878238678
Epoch: 9, training loss: 0.0014626139309257269, training accuracy: 0.9998629093170166
Epoch: 10, training loss: 0.0012855262029916048, training accuracy: 0.9998629093170166
=== Testing model ===
Accuracy : 100.0
=== Saving Model ===
```

References:

- https://medium.com/swlh/introduction-to-cnn-image-classification-using-cnn-in-pytorch-11eefae6d83c
- https://towardsdatascience.com/how-i-built-a-face-mask-detector-for-covid-19-using-pytorch-lightning-67eb3752fd61
- https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529
- https://core.ac.uk/reader/328808130
- https://machinelearningmastery.com/k-fold-cross-validation/

Dataset is collected from the links below.

- https://www.kaggle.com/andrewmvd/animal-faces
- https://www.kaggle.com/slothkong/10-monkey-species
- https://www.kaggle.com/hugozanini1/kangaroodataset?select=images
- https://www.kaggle.com/sumansid/facemask-dataset
- https://www.kaggle.com/dhruvmak/face-mask-detection
- https://www.kaggle.com/prithwirajmitra/covid-face-mask-detection-dataset
- https://www.kaggle.com/prasoonkottarathil/face-mask-lite-dataset