COMP 6721

Project Phase I

TEAM FL_07

Name	Student Id	Specialization
Amritpreet Singh	40150231	Data Specialist
Nikhil Nikhil	40151391	Training Specialist
Pushpa Gautam Ojha	40151892	Evaluation Specialist

Table of Contents

Dataset

For the dataset creation Images were collected from various open-source datasets available online and those images were manually classified into the classes for the training and testing of the model. For detailed information of the image sources please check the ***datasourceInfo.pdf*** file.

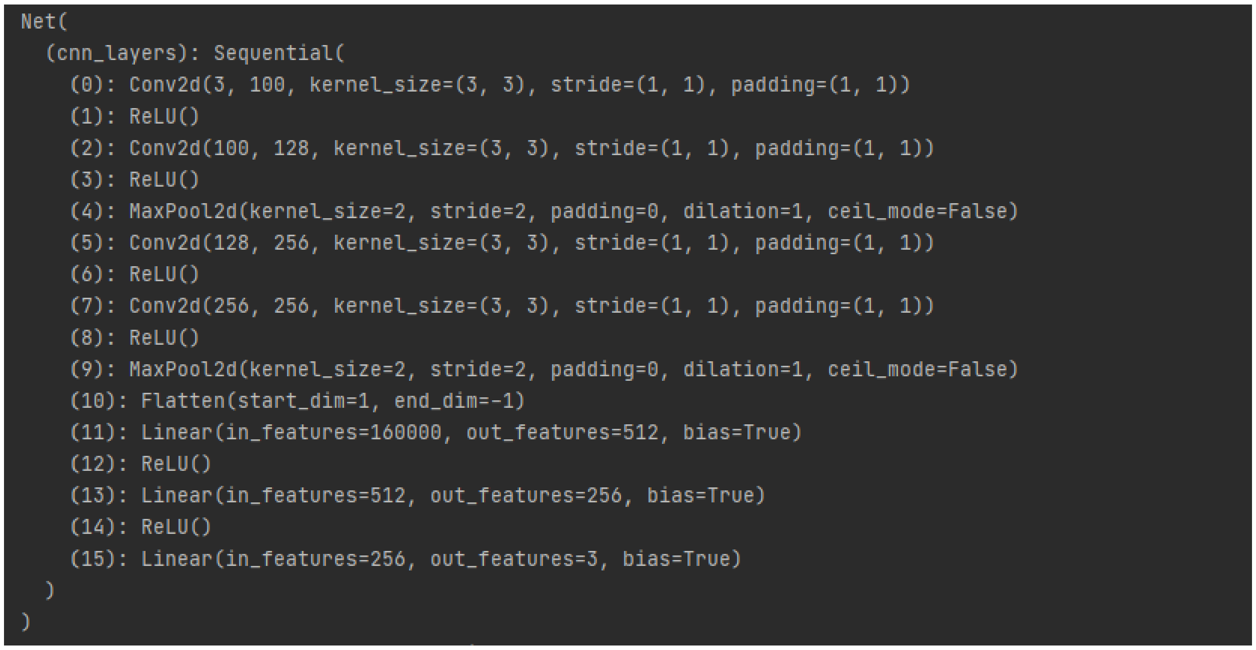
Since images were gathered from the various open sources available online because of which image sizes were inconsistent to resolve. Hence, before loading the dataset we fixed the image dimensions to 100X100. So once image size was fixed, we were able to process images and create our training and test datasets.

We have a total of 3582 images which are divided into training and test data sets. For training we have kept 2952 images across all classes and for testing total comprises 630 images across all classes. Further details are as follows:

Class	Training Dataset (Image Count)	Test Dataset (Image Count)
Class 0 (Unmasked)	1100	275
Class 1 (Masked)	1197	209
Class 2 (Not a person)	655	146
Total	2952	630

CNN Architecture

The following picture gives the detail about the layers that are used in our CNN model.

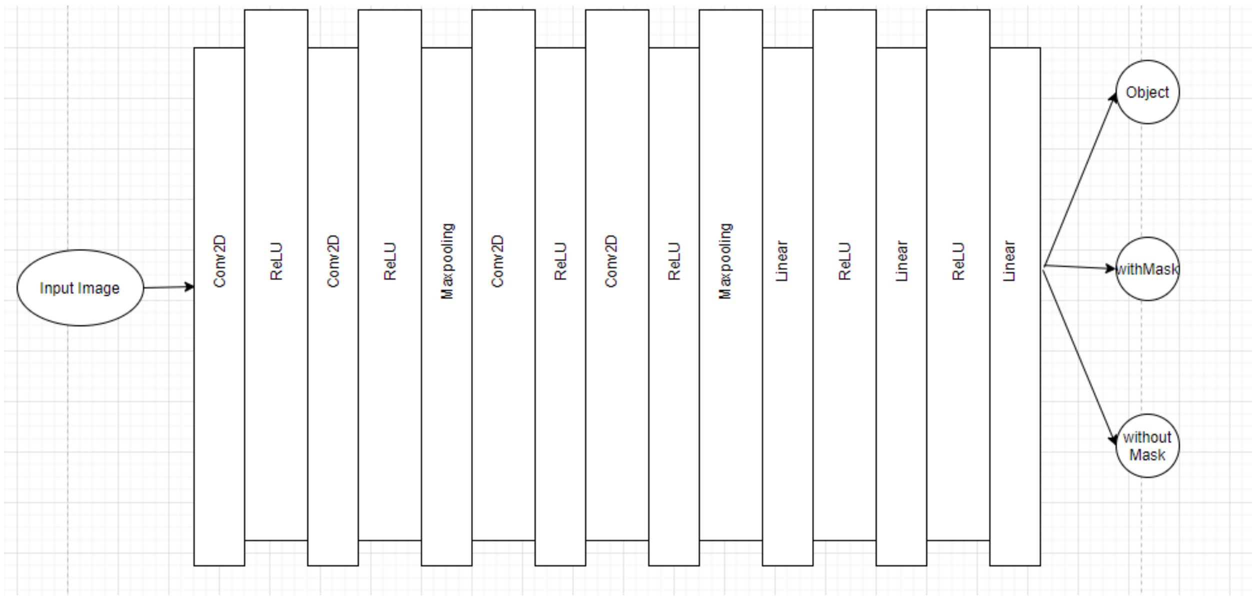


```
Net(
  (cnn_layers): Sequential(
    (0): Conv2d(3, 100, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(100, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU()
    (7): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Flatten(start_dim=1, end_dim=-1)
    (11): Linear(in_features=160000, out_features=512, bias=True)
    (12): ReLU()
    (13): Linear(in_features=512, out_features=256, bias=True)
    (14): ReLU()
    (15): Linear(in_features=256, out_features=3, bias=True)
  )
)
```

We use our pre-processed images to feed our model. Our model has totally 15 layers. We use ReLU as an activation function because of its faster computability and fewer vanishing gradients. There are four types of layers:

- **Conv2d**: This is the convolution layer we convolute the input the weight matrix to create activation maps.
- **ReLU (Rectified Linear Unit):** This is an activation function which use the activation maps to give the outputs to the next layer. The activation function defined as f(x) =max (0, x).
- Pooling: we use max pooling to reduce the number of activation maps.
- Linear: This layer basically maps the input to output linearly (n inputs to m output). it can be described as Ax = b function.

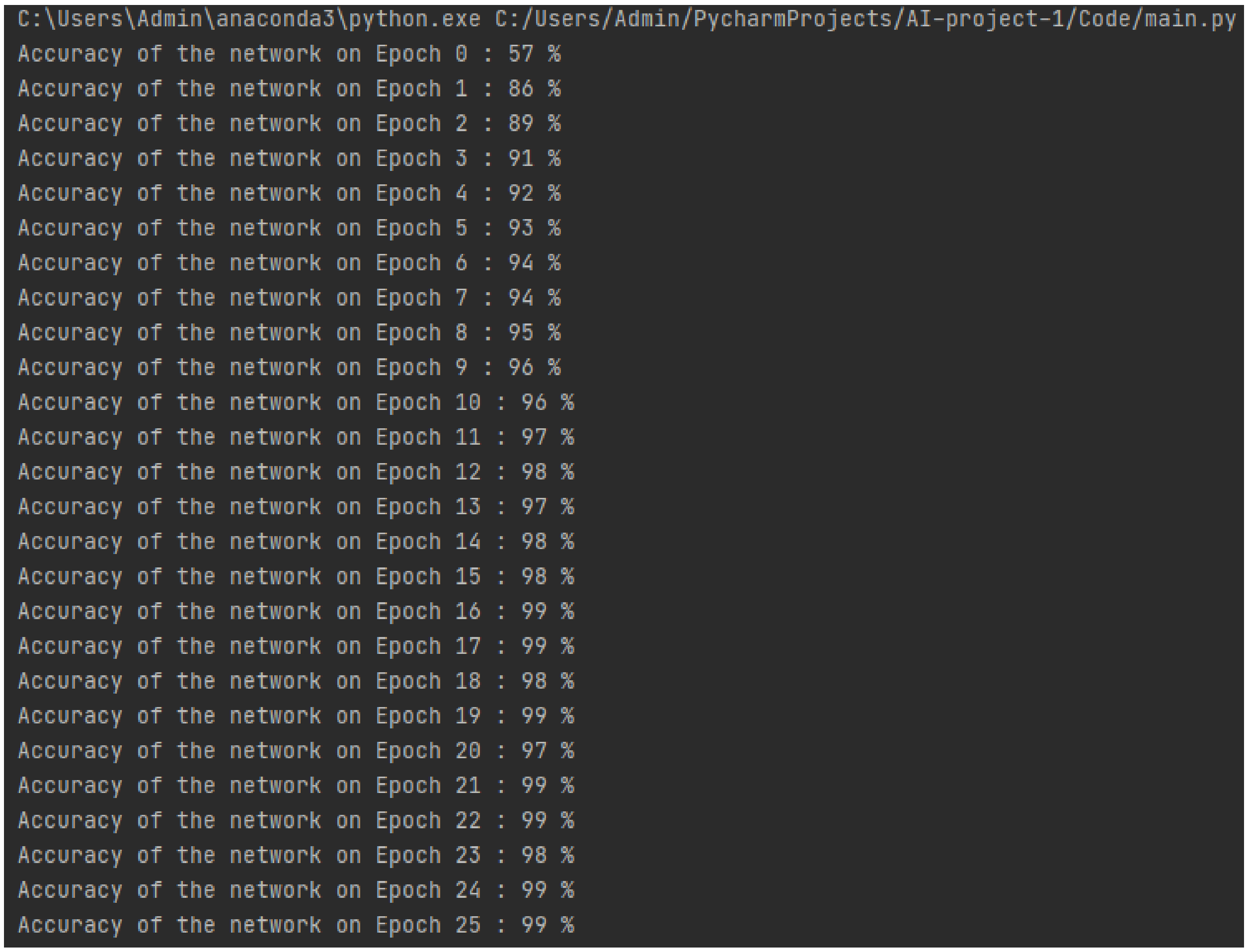
The structure of the model:



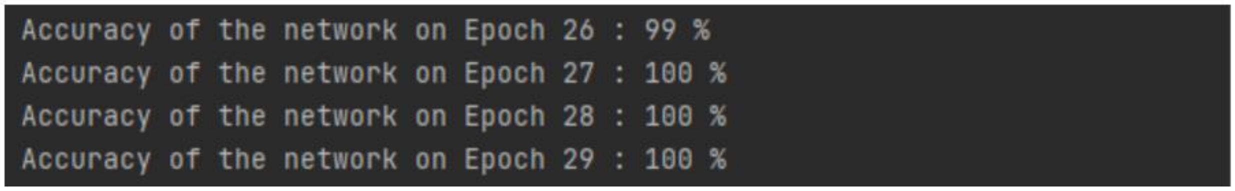
We also use cross Entropy loss or log loss that measures the difference between the actual and predicted values. This loss is then back propagated through the network to update the weights.

We use SGD (Stochastic gradient descent) to calculate gradients to update the weights with a learning rate of 0.005 and a momentum of .9.

We train the model for 30 epochs to make it more robust. At the 28th epoch, the model shows the training accuracy as 100%. Below is the image how the accuracy for the training data increases in each epoch.



```
C:\Users\Admin\anaconda3\python.exe C:/Users/Admin/PycharmProjects/AI-project-1/Code/main.py
Accuracy of the network on Epoch 0 : 57 %
Accuracy of the network on Epoch 1 : 86 %
Accuracy of the network on Epoch 2 : 89 %
Accuracy of the network on Epoch 3 : 91 %
Accuracy of the network on Epoch 4 : 92 %
Accuracy of the network on Epoch 5 : 93 %
Accuracy of the network on Epoch 6 : 94 %
Accuracy of the network on Epoch 7 : 94 %
Accuracy of the network on Epoch 8 : 95 %
Accuracy of the network on Epoch 9 : 96 %
Accuracy of the network on Epoch 10 : 96 %
Accuracy of the network on Epoch 11 : 97 %
Accuracy of the network on Epoch 12 : 98 %
Accuracy of the network on Epoch 13 : 97 %
Accuracy of the network on Epoch 14 : 98 %
Accuracy of the network on Epoch 15 : 98 %
Accuracy of the network on Epoch 16 : 99 %
Accuracy of the network on Epoch 17 : 99 %
Accuracy of the network on Epoch 18 : 98 %
Accuracy of the network on Epoch 19 : 99 %
Accuracy of the network on Epoch 20 : 97 %
Accuracy of the network on Epoch 21 : 99 %
Accuracy of the network on Epoch 22 : 99 %
Accuracy of the network on Epoch 23 : 98 %
Accuracy of the network on Epoch 24 : 99 %
Accuracy of the network on Epoch 25 : 99 %
```



```
Accuracy of the network on Epoch 26 : 99 %
Accuracy of the network on Epoch 27 : 100 %
Accuracy of the network on Epoch 28 : 100 %
Accuracy of the network on Epoch 29 : 100 %
```

Evaluation:

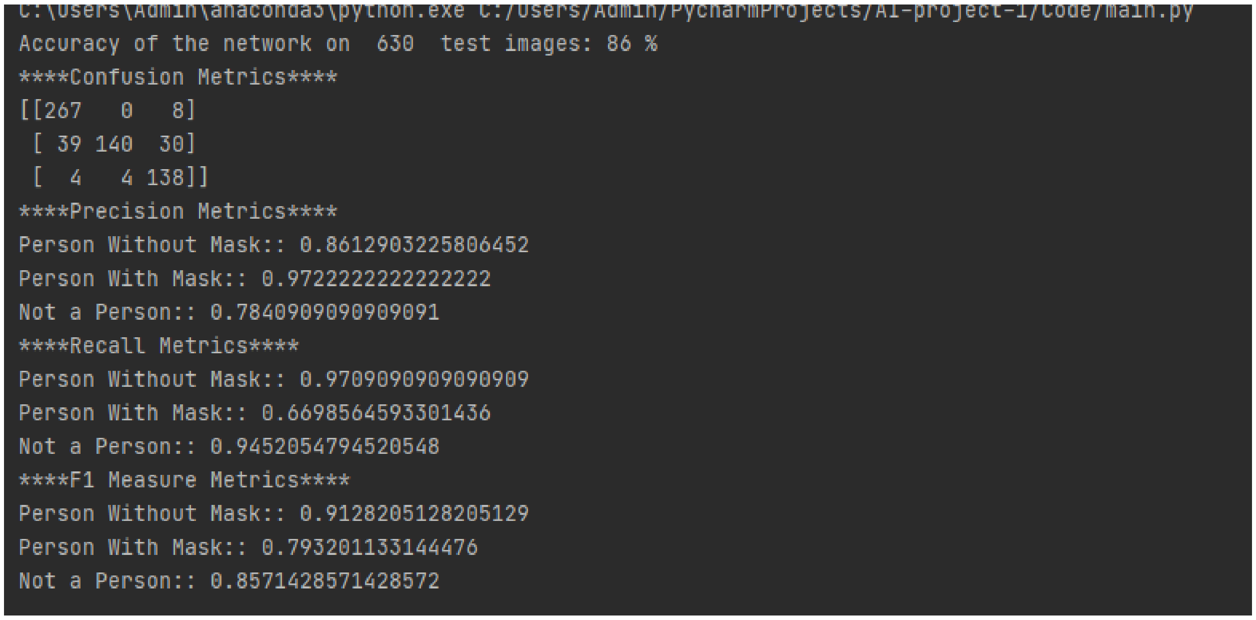
The model designed for this project has been evaluated using mainly 5 metrics, which are described below:

1. **Accuracy:** For computing accuracy, we are comparing the output labels, class 0,1 or 2, representing classes *Person With Mask*, *Person Without Mask* and *Not a Person* respectively, from CNN model with the actual test data labels. We then compute the sum of the test data that has been evaluated correctly by the model and divide it with the total test data size to get accuracy rate.
2. **Recall:** This is the ratio of true positive results and sum of true positive and false negatives result from CNN model. For our model it is computed by using *sklearn.metrics.recall_score*

3. **Precision**: This is the ratio of true positive results and sum of true and false positives result from CNN model. For our model it is computed by using *sklearn.metrics.precision_score*.3.4 F1 Measure: This is the ratio of twice the product of recall and precision over sum of recall and precision. For our model it is computed by using *sklearn.metrics.f1_score*.3.5
4. **Confusion Matrix**: This provides the accuracy of classification of each of the results by the CNN model. For our model it is computed by *sklearn.metrics.confusion_matri*x

For Computing above four metrics, i.e., *precision*, *recall*, *f1 measure* and *computation metrics*, we are first preparing the list of the *test data*, that have the labels represented by 0,1 or 2 (*0 for People Without Mask* class, *1* for *People With Mask* class *and 2* for *Not a person* class). Then we prepare the list of predicted results by the CNN model against these test data, which also have elements representing class by 0,1 or 2. We are then using *sklearn.metrics* library to compute these metrics for each of the classes, by sending labels as input i.e. *labels=[0,1,2]*.

Accuracy for 630 tested images as predicted by model was 86%. We are planning to work on the various aspects of the model in Project phase II for improving model's performance. Below is the CNN model output for 3 classes of the test data for each evaluation metrics as specified above.



```
C:\Users\Admin\anaconda3\python.exe C:/Users/Admin/PycharmProjects/AI-project-1/Code/main.py
Accuracy of the network on  630  test images: 86 %
****Confusion Metrics****
[[267   0   8]
 [ 39 140  30]
 [  4   4 138]]
****Precision Metrics****
Person Without Mask:: 0.8612903225806452
Person With Mask:: 0.9722222222222222
Not a Person:: 0.7840909090909091
****Recall Metrics****
Person Without Mask:: 0.9709090909090909
Person With Mask:: 0.6698564593301436
Not a Person:: 0.9452054794520548
****F1 Measure Metrics****
Person Without Mask:: 0.9128205128205129
Person With Mask:: 0.793201133144476
Not a Person:: 0.8571428571428572
```

GIT Link

https://github.com/nikpat9/AI-project-1

References:

- https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- https://stats.stackexchange.com/questions/360157/epoch-vs-iteration-in-cnn-training
- https://www.analyticsvidhya.com/blog/2020/07/how-to-train-an-image-classification-model-in-pytorch-and-tensorflow/
- https://medium.com/jovianml/how-i-created-a-simple-mask-detector-using-gpu-in-pytorch-bd13f3542f46
- https://medium.com/@i2i-blog/facial-mask-detection-using-deep-learning-and-computer-vision-c0966e14dd94
- https://towardsdatascience.com/understanding-pytorch-with-an-example-a-step-by-step-tutorial-81fc5f8c4e8e
- https://www.researchgate.net/publication/267269542_SWU-OFDB_A_database_for_occluded_face_detection_research
- https://deeplizard.com/learn/video/MasG7tZj-hw
- https://www.tutorialspoint.com/pytorch/pytorch_convolutional_neural_network.htm