Both the networks were evaluated on the dataset using the same training hyperparameters. I experimented with different batch sizes like 100, 256 and 512. The batch size of 256 gave consistent results. The learning rate was 0.01. The optimizer as used in both the networks according to the original paper was selected for this evaluation which is a stochastic gradient descent optimizer with a momentum of 0.9.
The training dataset was split into two parts with 10% for validation and 90% for training. The trained model was evaluated on the test dataset. All the image pixel values were standardized by subtracting the mean of the whole dataset and dividing by the standard deviation. No resizing was done for the images. The training and validation were done on their original sizes.

a) Googlenet training for MNIST
Within the first 10 epochs the model reached an accuracy of above 99% for both training and validation. The trained model after 10 epochs showed 99.11% accuracy on the test dataset.

b) GoogleNet training for CIFAR10
After 100 epochs of training, the training and validation accuracy for GoogleNet on CIFAR 10 was around 98% and 78% respectively. The test dataset accuracy was 77.10% after 100 epochs.

c) VGG-11 training for MNIST
As with GoogleNet, VGG-11 also showed above 99% accuracy within the first few epochs for training and validation. After 20 epochs, it showed 99.15% accuracy on test dataset.

d) VGG-11 training for CIFAR10
On CIFAR 10, this model showed around 90% accuracy on training and 80% accuracy on validation after 100 epochs. On test dataset, the final accuracy it showed was 79.14%
For MNIST dataset, both the models demonstrated outstanding performance within a few epochs of training with specified hyperparameters. But for CIFAR 10, the models started overfitting. So, using two augmentation techniques of horizontal flipping and horizontal and vertical shifting of the images, training was done on the augmented version of the datasets which reduced the overfitting to a big extent and showed superior accuracy.

i) GoogleNet training for CIFAR10 with augmentation:
Within 70 epochs, the model displayed an accuracy of 84.76% on the validation dataset and above 90% accuracy on training. This shows how the overfitting was reduced.

ii) VGG-11 training for CIFAR10 with augmentation:
Within 80 epochs, the model showed an accuracy of 87.48% on the validation dataset and 93% accuracy on training. The VGG-11 was more generalized compared to GoogleNet

**The results and screenshot for training are put in the appendix section from next page.

Appendix

```
[10] history = model.fit(x=X_train,
                         y=[Y_train,Y_train,Y_train],
                         validation_data=(X_val,[Y_val,Y_val,Y_val]),
                         epochs=5,
                         batch_size=256
                         )
     training_history.append(history)
```

```
Epoch 1/5
211/211 [==============================] - 31s 146ms/step - loss: 0.0562 - output_loss: 0.0191 - aux1_loss: 0.0181 - aux2_loss: 0.0190 - output_accuracy: 0.9945 - aux1_accuracy: 0.994
Epoch 2/5
211/211 [==============================] - 31s 146ms/step - loss: 0.0370 - output_loss: 0.0120 - aux1_loss: 0.0122 - aux2_loss: 0.0129 - output_accuracy: 0.9965 - aux1_accuracy: 0.996
Epoch 3/5
211/211 [==============================] - 31s 146ms/step - loss: 0.0380 - output_loss: 0.0125 - aux1_loss: 0.0121 - aux2_loss: 0.0133 - output_accuracy: 0.9963 - aux1_accuracy: 0.996
Epoch 4/5
211/211 [==============================] - 31s 146ms/step - loss: 0.0337 - output_loss: 0.0114 - aux1_loss: 0.0107 - aux2_loss: 0.0116 - output_accuracy: 0.9967 - aux1_accuracy: 0.996
Epoch 5/5
211/211 [==============================] - 31s 146ms/step - loss: 0.0278 - output_loss: 0.0094 - aux1_loss: 0.0087 - aux2_loss: 0.0096 - output_accuracy: 0.9972 - aux1_accuracy: 0.997
```

Fig - Screenshot for GoogleNet training on MNIST

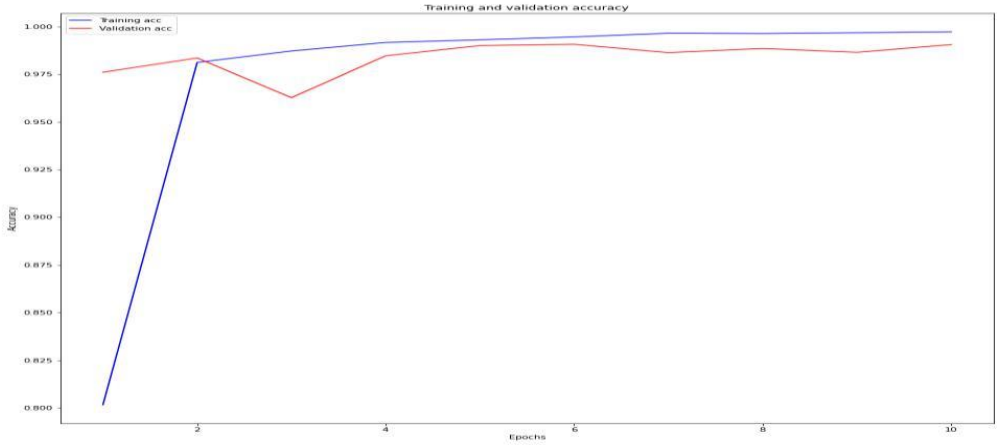

Fig - GoogleNet training on MNIST Accuracy curve

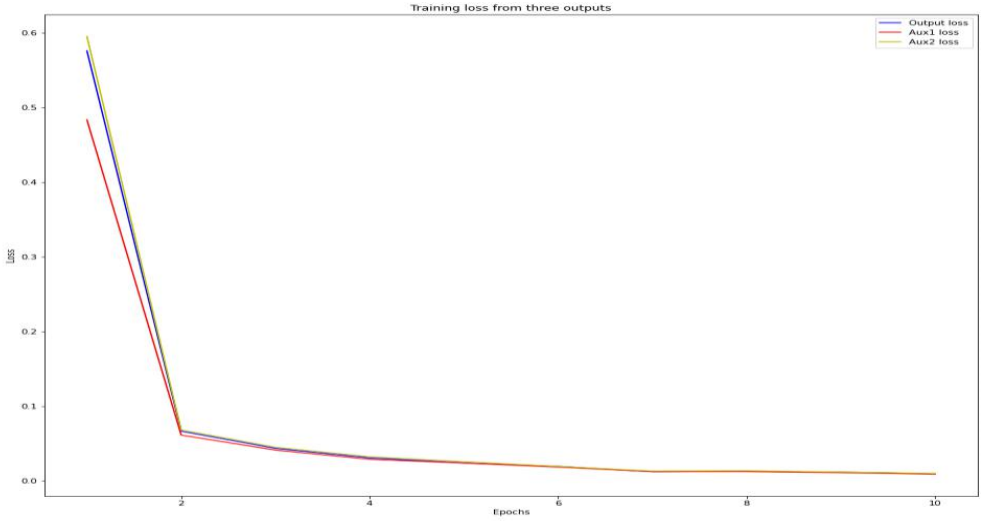

Fig - GoogleNet training on MNIST Loss curve

```
history = model.fit(x=X_train,
                    y=[Y_train,Y_train,Y_train],
                    validation_data=(X_val,[Y_val,Y_val,Y_val]),
                    epochs=20,
                    batch_size=256
                    )
training_history.append(history)
```

```
Epoch 1/20
176/176 [==============================] - 9s 51ms/step - loss: 0.1759 - output_loss: 0.0578 - aux1_loss: 0.0590 - aux2_loss: 0.0592 - output_accuracy: 0.9812 - aux1_accuracy: 0.9808
Epoch 2/20
176/176 [==============================] - 9s 51ms/step - loss: 0.1798 - output_loss: 0.0592 - aux1_loss: 0.0605 - aux2_loss: 0.0601 - output_accuracy: 0.9802 - aux1_accuracy: 0.9804
Epoch 3/20
176/176 [==============================] - 9s 52ms/step - loss: 0.1442 - output_loss: 0.0471 - aux1_loss: 0.0478 - aux2_loss: 0.0493 - output_accuracy: 0.9836 - aux1_accuracy: 0.9836
Epoch 4/20
176/176 [==============================] - 9s 52ms/step - loss: 0.1532 - output_loss: 0.0498 - aux1_loss: 0.0521 - aux2_loss: 0.0513 - output_accuracy: 0.9831 - aux1_accuracy: 0.9825
Epoch 5/20
176/176 [==============================] - 9s 52ms/step - loss: 0.1501 - output_loss: 0.0486 - aux1_loss: 0.0513 - aux2_loss: 0.0502 - output_accuracy: 0.9842 - aux1_accuracy: 0.9834
Epoch 6/20
176/176 [==============================] - 9s 53ms/step - loss: 0.1606 - output_loss: 0.0531 - aux1_loss: 0.0534 - aux2_loss: 0.0541 - output_accuracy: 0.9816 - aux1_accuracy: 0.9823
Epoch 7/20
176/176 [==============================] - 9s 52ms/step - loss: 0.1347 - output_loss: 0.0438 - aux1_loss: 0.0464 - aux2_loss: 0.0446 - output_accuracy: 0.9850 - aux1_accuracy: 0.9844
Epoch 8/20
176/176 [==============================] - 9s 52ms/step - loss: 0.1219 - output_loss: 0.0403 - aux1_loss: 0.0405 - aux2_loss: 0.0410 - output_accuracy: 0.9865 - aux1_accuracy: 0.9866
Epoch 9/20
176/176 [==============================] - 9s 52ms/step - loss: 0.1107 - output_loss: 0.0363 - aux1_loss: 0.0377 - aux2_loss: 0.0368 - output_accuracy: 0.9884 - aux1_accuracy: 0.9877
Epoch 10/20
176/176 [==============================] - 9s 52ms/step - loss: 0.1391 - output_loss: 0.0457 - aux1_loss: 0.0463 - aux2_loss: 0.0470 - output_accuracy: 0.9853 - aux1_accuracy: 0.9849
Epoch 11/20
```

Fig - Screenshot for GoogleNet training on CIFAR10

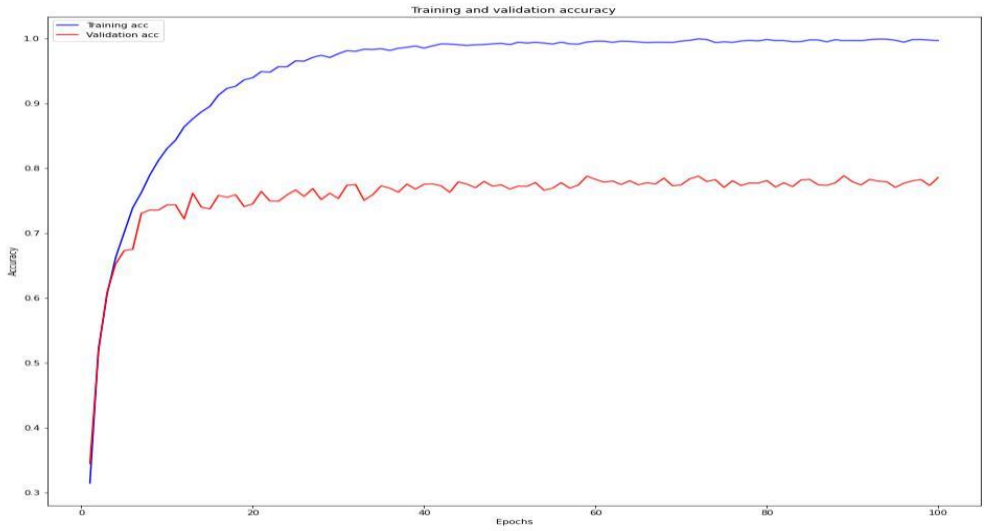

Fig - GoogleNet training on CIFAR10 Accuracy curve

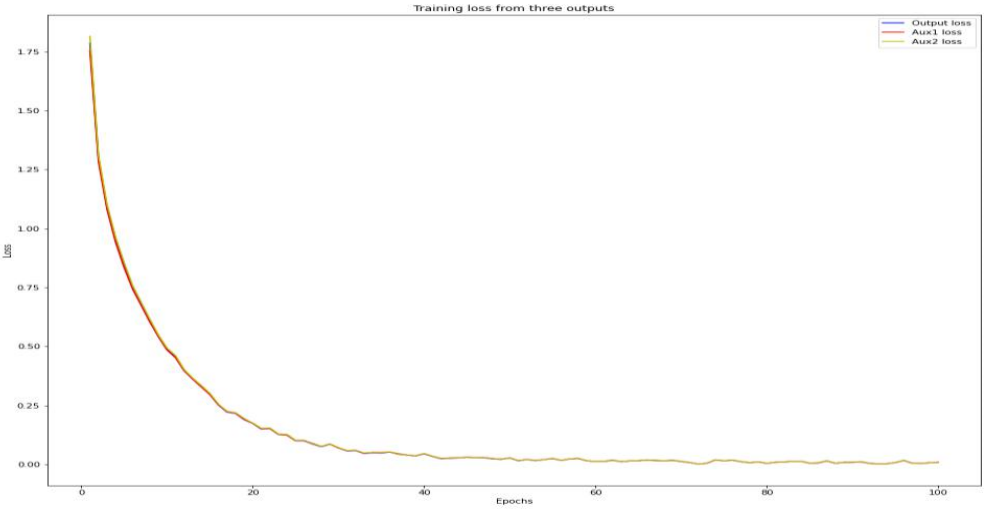

Fig - GoogleNet training on CIFAR10 Loss curve

```
batch_size=256
history = model.fit_generator(datagen.flow(X_train,Y_train,batch_size=batch_size),
                              steps_per_epoch=math.ceil(X_train.shape[0]/batch_size),
                              validation_data=(X_val,Y_val),
                              epochs=30)
```

```
WARNING:tensorflow:From <ipython-input-12-4696e8bd7b3d>:5: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/30
176/176 [==============================] - 42s 237ms/step - loss: 5.6773 - output_loss: 1.8936 - aux1_loss: 1.8651 - aux2_loss: 1.9186 - output_accuracy: 0.2692 - aux1_accuracy: 0.
Epoch 2/30
176/176 [==============================] - 36s 207ms/step - loss: 4.4285 - output_loss: 1.4749 - aux1_loss: 1.4644 - aux2_loss: 1.4892 - output_accuracy: 0.4564 - aux1_accuracy: 0.
Epoch 3/30
176/176 [==============================] - 37s 208ms/step - loss: 3.9175 - output_loss: 1.3061 - aux1_loss: 1.2952 - aux2_loss: 1.3162 - output_accuracy: 0.5304 - aux1_accuracy: 0.
Epoch 4/30
176/176 [==============================] - 37s 208ms/step - loss: 3.5542 - output_loss: 1.1869 - aux1_loss: 1.1735 - aux2_loss: 1.1938 - output_accuracy: 0.5754 - aux1_accuracy: 0.
Epoch 5/30
176/176 [==============================] - 37s 208ms/step - loss: 3.2265 - output_loss: 1.0744 - aux1_loss: 1.0672 - aux2_loss: 1.0849 - output_accuracy: 0.6185 - aux1_accuracy: 0.
Epoch 6/30
176/176 [==============================] - 37s 208ms/step - loss: 3.0489 - output_loss: 1.0163 - aux1_loss: 1.0084 - aux2_loss: 1.0243 - output_accuracy: 0.6416 - aux1_accuracy: 0.
Epoch 7/30
176/176 [==============================] - 37s 207ms/step - loss: 2.8675 - output_loss: 0.9542 - aux1_loss: 0.9501 - aux2_loss: 0.9631 - output_accuracy: 0.6652 - aux1_accuracy: 0.
Epoch 8/30
176/176 [==============================] - 36s 207ms/step - loss: 2.7134 - output_loss: 0.9020 - aux1_loss: 0.8991 - aux2_loss: 0.9123 - output_accuracy: 0.6854 - aux1_accuracy: 0.
Epoch 9/30
176/176 [==============================] - 36s 207ms/step - loss: 2.6073 - output_loss: 0.8672 - aux1_loss: 0.8643 - aux2_loss: 0.8757 - output_accuracy: 0.6969 - aux1_accuracy: 0.
Epoch 10/30
176/176 [==============================] - 36s 207ms/step - loss: 2.5036 - output_loss: 0.8336 - aux1_loss: 0.8270 - aux2_loss: 0.8430 - output_accuracy: 0.7091 - aux1_accuracy: 0.
Epoch 11/30
176/176 [==============================] - 36s 207ms/step - loss: 2.3984 - output_loss: 0.7984 - aux1_loss: 0.7944 - aux2_loss: 0.8056 - output_accuracy: 0.7230 - aux1_accuracy: 0.
Epoch 12/30
```

Fig - Screenshot for GoogleNet training on CIFAR10 with augmentation

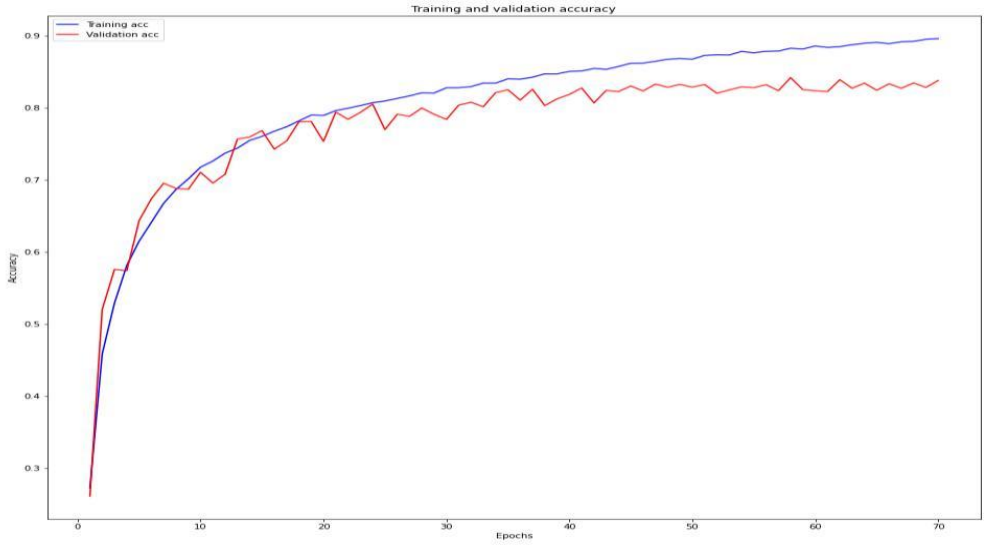

Fig - GoogleNet training on CIFAR10 with augmentation Accuracy Curve

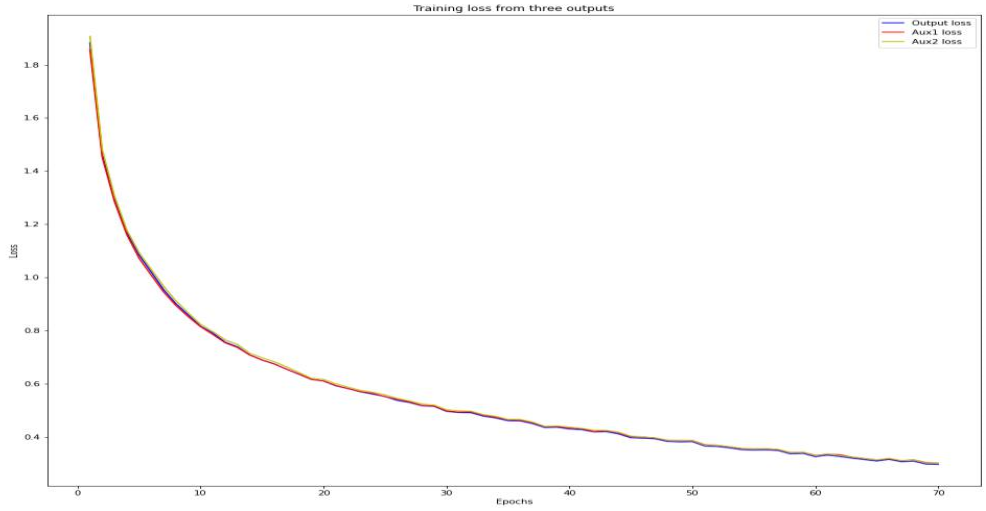

Fig - GoogleNet training on CIFAR10 with augmentation Loss Curve

```
8]  batch_size = 256
    history = model.fit(x=X_train,
                        y=Y_train,
                        validation_data=(X_val,Y_val),
                        epochs=10,
                        batch_size=256
                        )
    training_history.append(history)

    Epoch 1/10
    211/211 [==============================] - 8s 36ms/step - loss: 0.0124 - accuracy: 0.9964 - val_loss: 0.0478 - val_accuracy: 0.9885
    Epoch 2/10
    211/211 [==============================] - 8s 36ms/step - loss: 0.0112 - accuracy: 0.9963 - val_loss: 0.0469 - val_accuracy: 0.9897
    Epoch 3/10
    211/211 [==============================] - 8s 36ms/step - loss: 0.0095 - accuracy: 0.9968 - val_loss: 0.0399 - val_accuracy: 0.9902
    Epoch 4/10
    211/211 [==============================] - 8s 36ms/step - loss: 0.0072 - accuracy: 0.9977 - val_loss: 0.0480 - val_accuracy: 0.9902
    Epoch 5/10
    211/211 [==============================] - 8s 36ms/step - loss: 0.0055 - accuracy: 0.9983 - val_loss: 0.0472 - val_accuracy: 0.9897
    Epoch 6/10
    211/211 [==============================] - 8s 36ms/step - loss: 0.0046 - accuracy: 0.9987 - val_loss: 0.0495 - val_accuracy: 0.9897
    Epoch 7/10
    211/211 [==============================] - 8s 36ms/step - loss: 0.0036 - accuracy: 0.9988 - val_loss: 0.0449 - val_accuracy: 0.9898
    Epoch 8/10
    211/211 [==============================] - 8s 36ms/step - loss: 0.0039 - accuracy: 0.9988 - val_loss: 0.0441 - val_accuracy: 0.9908
    Epoch 9/10
    211/211 [==============================] - 8s 36ms/step - loss: 0.0049 - accuracy: 0.9983 - val_loss: 0.0478 - val_accuracy: 0.9903
    Epoch 10/10
    211/211 [==============================] - 8s 36ms/step - loss: 0.0040 - accuracy: 0.9988 - val_loss: 0.0482 - val_accuracy: 0.9912
```

Fig - Screenshot for VGG-11 training on MNIST

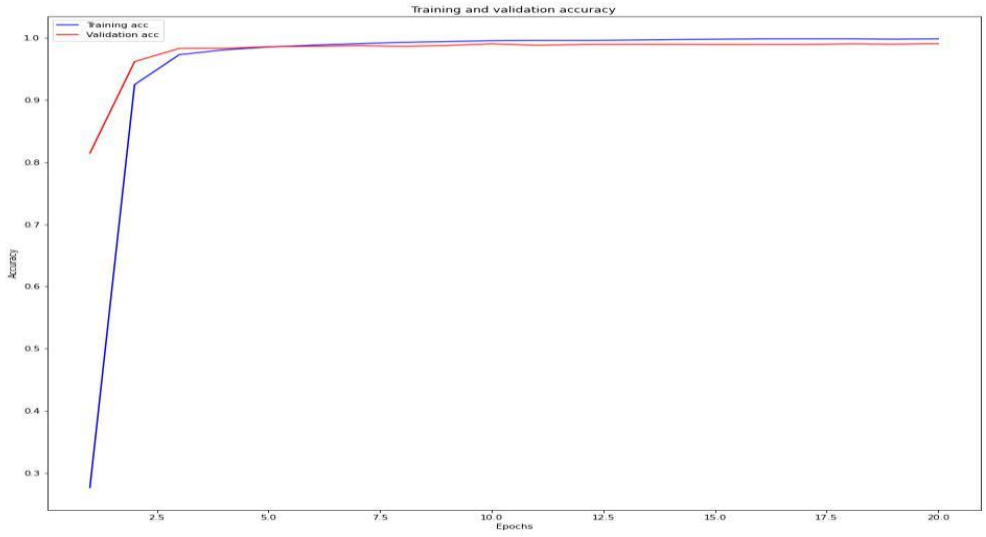

Fig – VGG-11 training on MNIST Accuracy Curve

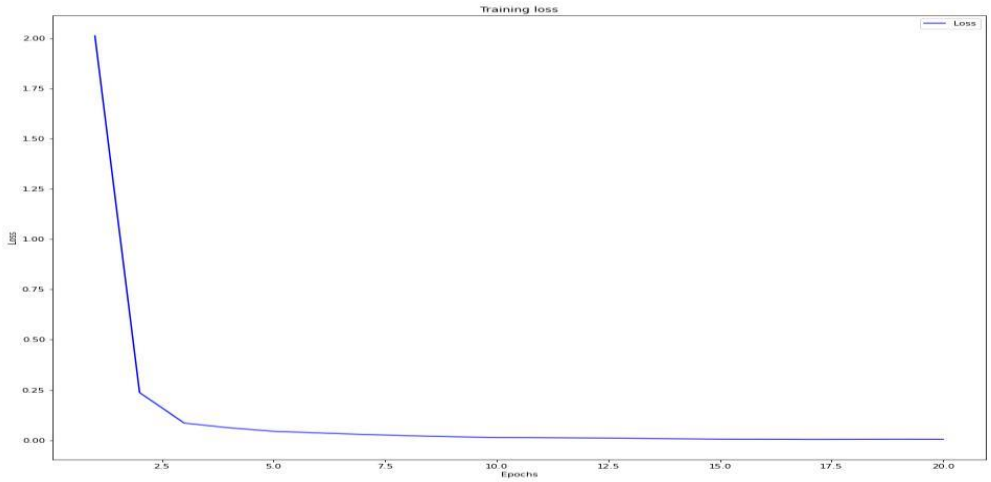

Fig 16 – VGG-11 training on MNIST Loss Curve

```
[6]  history = model.fit(x=X_train,
                         y=Y_train,
                         validation_data=(X_val,Y_val),
                         epochs=30,
                         batch_size=256
                         )
     training_history.append(history)

     Epoch 1/30
     176/176 [==============================] - 8s 43ms/step - loss: 2.2858 - accuracy: 0.1318 - val_loss: 2.1487 - val_accuracy: 0.2188
     Epoch 2/30
     176/176 [==============================] - 7s 38ms/step - loss: 1.9187 - accuracy: 0.2698 - val_loss: 1.7141 - val_accuracy: 0.3680
     Epoch 3/30
     176/176 [==============================] - 7s 38ms/step - loss: 1.5935 - accuracy: 0.4007 - val_loss: 1.4797 - val_accuracy: 0.4570
     Epoch 4/30
     176/176 [==============================] - 7s 38ms/step - loss: 1.3798 - accuracy: 0.4908 - val_loss: 1.2724 - val_accuracy: 0.5358
     Epoch 5/30
     176/176 [==============================] - 7s 38ms/step - loss: 1.1872 - accuracy: 0.5661 - val_loss: 1.1584 - val_accuracy: 0.5834
     Epoch 6/30
     176/176 [==============================] - 7s 39ms/step - loss: 1.0408 - accuracy: 0.6192 - val_loss: 0.9662 - val_accuracy: 0.6582
     Epoch 7/30
     176/176 [==============================] - 7s 39ms/step - loss: 0.9035 - accuracy: 0.6732 - val_loss: 0.8962 - val_accuracy: 0.6742
     Epoch 8/30
     176/176 [==============================] - 7s 39ms/step - loss: 0.7837 - accuracy: 0.7190 - val_loss: 0.8448 - val_accuracy: 0.7080
     Epoch 9/30
     176/176 [==============================] - 7s 38ms/step - loss: 0.6782 - accuracy: 0.7588 - val_loss: 0.7383 - val_accuracy: 0.7432
     Epoch 10/30
     176/176 [==============================] - 7s 38ms/step - loss: 0.5778 - accuracy: 0.7953 - val_loss: 0.7826 - val_accuracy: 0.7328
     Epoch 11/30
     176/176 [==============================] - 7s 38ms/step - loss: 0.4776 - accuracy: 0.8315 - val_loss: 0.7105 - val_accuracy: 0.7624
     Epoch 12/30
```

Fig - Screenshot for VGG-11 training on CIFAR-10

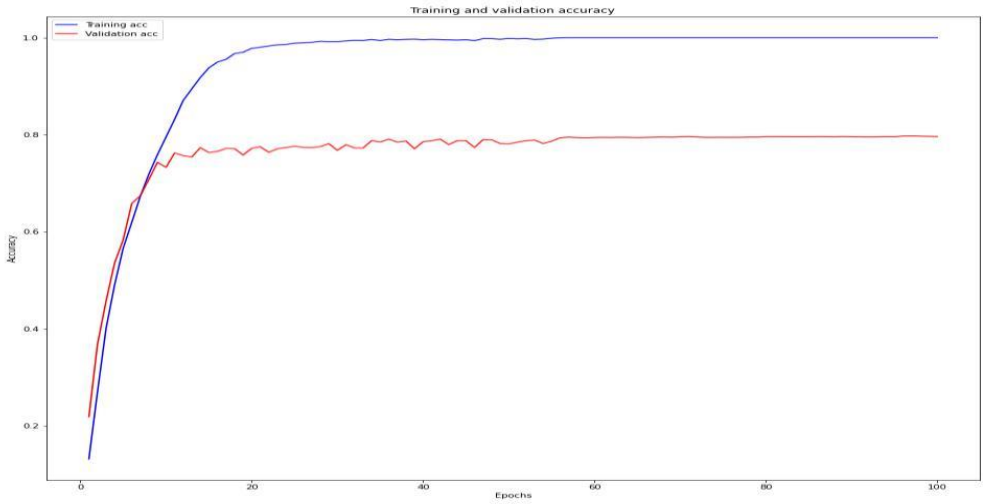

Fig – VGG-11 training on CIFAR10 Accuracy Curve

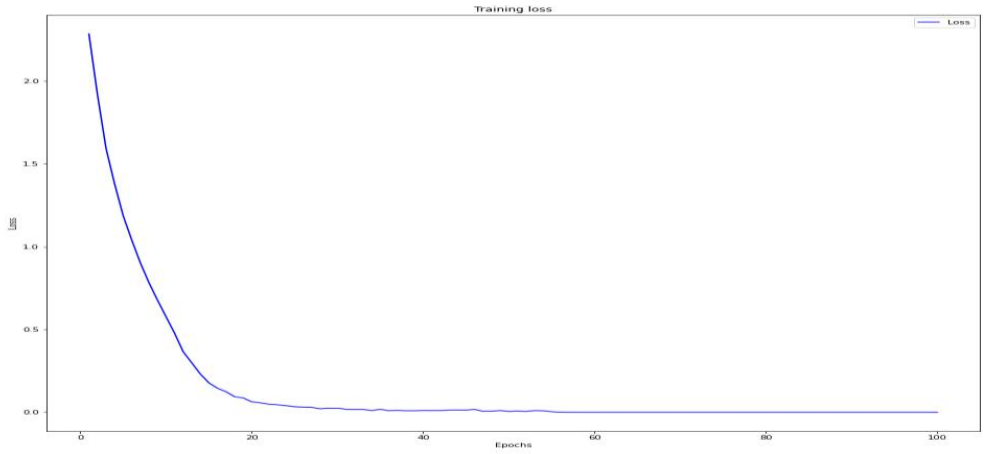

Fig – VGG-11 training on CIFAR10 Loss Curve

```
[29] batch_size=256
     history = model.fit_generator(datagen.flow(X_train,Y_train,batch_size=batch_size),
                                   steps_per_epoch=math.ceil(X_train.shape[0]/batch_size),
                                   validation_data=(X_val,Y_val),
                                   epochs=20)
     training_history.append(history)

     Epoch 1/20
     176/176 [==============================] - 19s 110ms/step - loss: 0.2879 - accuracy: 0.9010 - val_loss: 0.4833 - val_accuracy: 0.8462
     Epoch 2/20
     176/176 [==============================] - 19s 109ms/step - loss: 0.2812 - accuracy: 0.9034 - val_loss: 0.4983 - val_accuracy: 0.8444
     Epoch 3/20
     176/176 [==============================] - 19s 111ms/step - loss: 0.2803 - accuracy: 0.9027 - val_loss: 0.4652 - val_accuracy: 0.8544
     Epoch 4/20
     176/176 [==============================] - 19s 109ms/step - loss: 0.2650 - accuracy: 0.9072 - val_loss: 0.4275 - val_accuracy: 0.8676
     Epoch 5/20
     176/176 [==============================] - 19s 109ms/step - loss: 0.2626 - accuracy: 0.9096 - val_loss: 0.4877 - val_accuracy: 0.8532
     Epoch 6/20
     176/176 [==============================] - 19s 109ms/step - loss: 0.2503 - accuracy: 0.9136 - val_loss: 0.4815 - val_accuracy: 0.8526
     Epoch 7/20
     176/176 [==============================] - 19s 109ms/step - loss: 0.2360 - accuracy: 0.9175 - val_loss: 0.4708 - val_accuracy: 0.8658
     Epoch 8/20
     176/176 [==============================] - 19s 109ms/step - loss: 0.2373 - accuracy: 0.9182 - val_loss: 0.4505 - val_accuracy: 0.8612
     Epoch 9/20
     176/176 [==============================] - 19s 109ms/step - loss: 0.2320 - accuracy: 0.9210 - val_loss: 0.4497 - val_accuracy: 0.8678
     Epoch 10/20
     176/176 [==============================] - 19s 108ms/step - loss: 0.2128 - accuracy: 0.9269 - val_loss: 0.4443 - val_accuracy: 0.8654
     Epoch 11/20
     176/176 [==============================] - 19s 109ms/step - loss: 0.2114 - accuracy: 0.9263 - val_loss: 0.4378 - val_accuracy: 0.8748
     Epoch 12/20
```

Fig - Screenshot for VGG-11 training on CIFAR-10 with augmentation

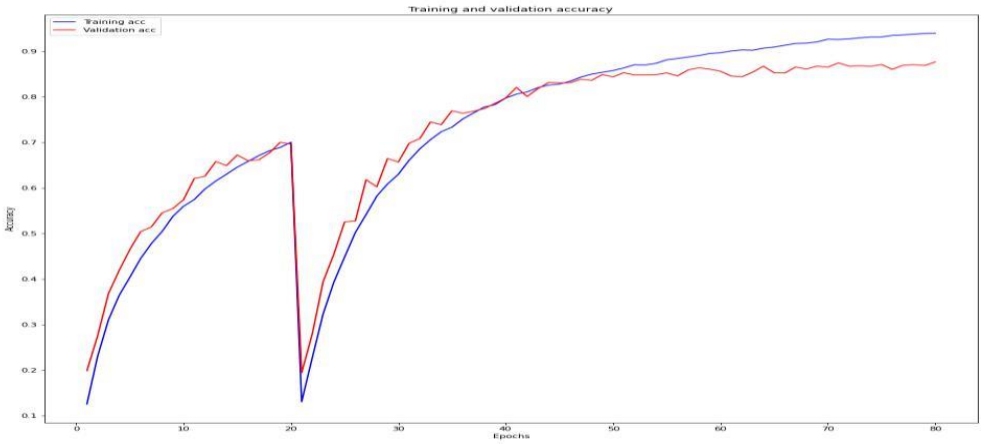

Fig – VGG-11 training on CIFAR10 with augmentation Accuracy Curve

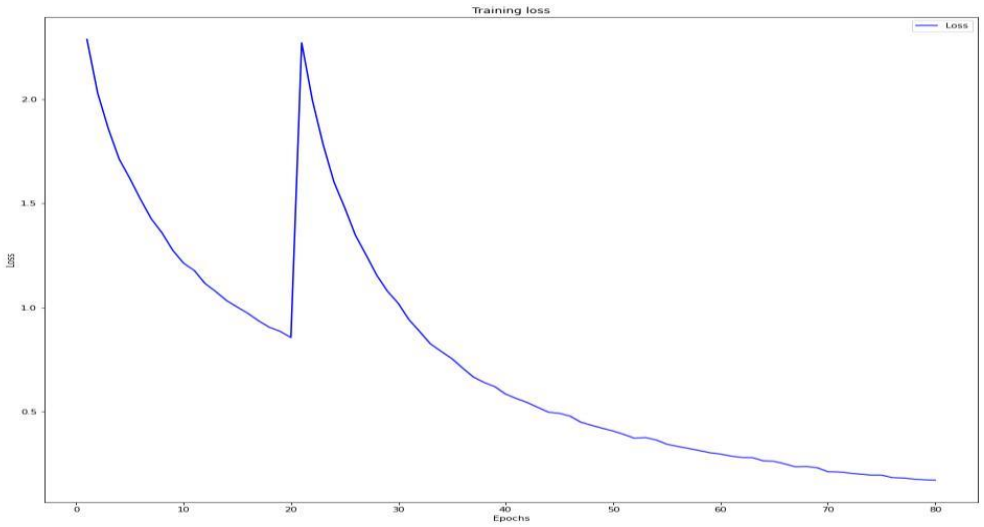

Fig – VGG-11 training on CIFAR10 with augmentation Loss Curve