Deep Networks for Image Classification

Author: Niranjan Ganesan -170963389

1. Introduction

The goal of this coursework is to evaluate the performance of Image classification tasks by training the deep neural networks. The performance of VGG-16 and ResNet models are evaluated for large scale Image Recognition by training the models with MNIST digits dataset and CIFAR-10 dataset. With the help of the original architecture from [1] for VGG-16 and [2] for ResNet the models are trained and evaluated.

2. Critical Analysis/Related work

Convolutional Neural Networks are one of the best learning algorithms for understanding image content and shows tremendous performance in image segmentation, classification, detection and retrieval related tasks [3], [4]. The important feature of CNN is the ability to exploit spatial or temporal correlation in data.

The first successful CNN architecture called ConvNet[5] has been applied to image classification tasks such as handwritten digit and zip code recognition [6]. The ConvNet[5] architecture was further improved called as LeNet-5 [8] which proved to be successful in document recognition related applications [7], [8] but didn't perform well on other image recognition problems. Due to inefficient training images and less computational resources there was no significant improvement in CNN architecture. In 2003 an improved CNN architecture showed good results on a hand digit benchmark dataset called MNIST [7], [9], [10]. This improvement in performance extended the applications of CNNs to face detection [10], [11], customer tracking [12], medical image segmentation [13], anomaly detection [14] and robot vision [15].

Minor changes in the CNN architecture such as using max-pooling instead of subsampling [16] let to the improvement of image classification tasks. Training the deep neural models on Graphics processing units (GPUs) made the training much quicker which revived the research in CNN [17]. The establishment of large database of Images known as ImageNet [18], containing millions of annotated images belonging to a large number of classes increased the research opportunities in Image classification using deep CNNs.

The main breakthrough in CNN performance was brought by AlexNet[19], which showed exemplary performance in 2012-ILSVRC compared to conventional Convolution techniques[19]. CNNs such as AlexNet was considered as the first deep CNN architecture, which showed groundbreaking results for Image classification and recognition tasks by enhancing the learning capacity of the CCN, making it deeper and by applying several parameter optimization strategies.

The AlexNet depth was extended from 5 (LeNet) to 8 layers to make CNN applicable for diverse category of images. A very deep state of the art infrastructure was designed with 19 layers deep compared to AlexNet called as VGG [1] to simulate the relation of depth with the representational capacity of the network [20]. VGG network uses a stack of 3x3 small size filters which reduced the computational complexity by reducing the number of parameters. These findings set a new trend in research to work with smaller size filters in CNN.

A residual learning architecture called ResNet [2] was introduced which revolutionized the CNN architectural race by introducing the concept of residual learning in CNNs and devised an effective methodology for the training of deep networks. The architecture of ResNet is 20 times deeper than AlexNet and 8 times deeper than VGG but still showed less computational complexity than previous proposed networks [19], [21]. ResNet with 50/101/152 layers has less error on image classification task than 34 layers plain Net. Another state of the art very deep CNN called GoogleNet [22] was designed which achieved high accuracy with a reduced computational cost. It introduced the new concept of inception block in CNN, whereby it incorporates multi-scale convolutional transformations using split, transform and merge idea.

As previously mentioned, two very deep CNNs, VGG-16 and ResNet-50 is considered to be trained with MNIST and CIFAR10 datasets in this project. With a minor change in the original architecture the deep CNNs are trained and evaluated.

3. Model Description

In this paper, I use various deeper networks for evaluating the effectiveness of deeper CNN models for image classification on MNIST and CIFAR-10 datasets.

3.1 Model Architecture

[1] VGG-16

The deep CNN architecture named VGG, was made 19 layers deep (also present in 13 and 16 layers) is a simple and effective design principle for CNN architectures. During training, the input to CNN is a fixed size 28x28 Grayscale image. The image is passed through a stack of convolutional layers where filters with a very small receptive field 3x3 are used. The convolution stride is fixed to 1 pixel and padding is 1 pixel for 3 x 3 convolution layers. Max-pooling is performed over a 2 x 2 pixel window, with stride 2. A stack of convolutional layers is followed by three fully connected layers. The final layer is a soft-max layer. All hidden layers are equipped with the rectification (ReLU) non-linearity. The major architectural change is
The model architecture of VGG-16 on MNIST can be found in **Appendix A** and VGG-16 on CIFAR-10 can be found in **Appendix B.**

[2] ResNet

The entire architecture consists of two networks called as Plain Network and Residual Network. The plain network consists of convolutional layers with 3x3 filters and follows two simple design rules, for the same output feature map size, the layers have the same number of filters and if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer. Down sampling is performed directly at the convolutional layers that have a stride of 2. At the end a 10-way fully-connected layer with softmax is implemented. The model has fewer filters and lower complexity than VGG nets. The second network called Residual network is based on the plain network where shortcut connections are inserted which turns the network into its counterpart residual version. The skip connections play a major role is enhancing the performance of the residual network.

The model architecture of ResNet-50 on MNIST can be found in **Appendix C** and ResNet-50 on CIFAR-10 can be found in **Appendix D.**

4. Experiments

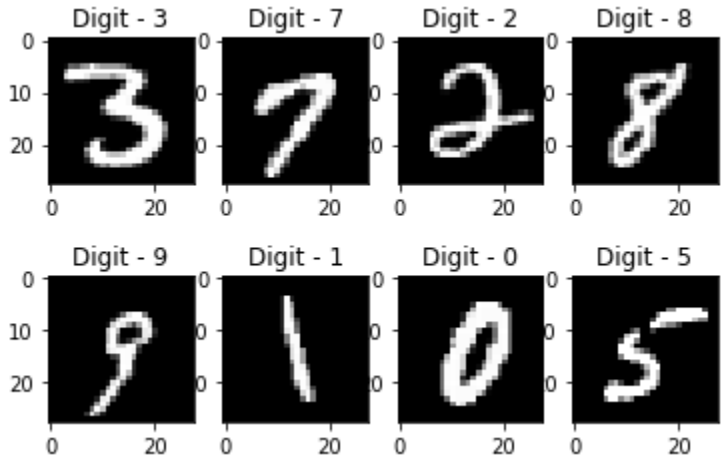
Four experiments are carried out with two different deep learning models and two different datasets.

4.1 Datasets

As previously mentioned, the models have been trained on MNIST database and CIFAR10 database. Below is the description of these databases,

- **MNIST**

The MNIST database [1] of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.



The size of the images is fixed to 28x28x1 where height = 28, width = 28 and 1 = image channel(grayscale).

- **CIFAR-10**

  The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

  The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

  Here are the classes in the dataset along with a random image from each class:

4.2 Training Procedure and Performance results

4.2.1 Experiment 1 - Model Training on VGG-16 architecture using MNIST:

- The VGG-16 model (Appendix A) has been trained on MNIST dataset. First the data is split into train and test images while loading the dataset. I have used Keras to load the dataset and train the VGG-16 model.
- After the dataset is split into training and test images, few training samples are visualized and can be found in Appendix I.
- Next the output training samples are preprocessed (converted to categorical variables) by one hot encoding method using Keras utils "to_categorical".
- The modified network architecture from original paper [1] is shown in the Appendix A.
- The total trainable parameters are 14,913,226. The experiment is run over 50 epochs and loss function used is 'binary crossentropy', optimizer is 'RMSprop' with learning rate of '1e-4'.
- The training performance results screenshots for every epoch can be found in Appendix E.

4.2.2 Experiment 2 – Model Training on VGG-16 architecture using CIFAR-10:

- The VGG-16 model (Appendix B) has been trained on CIFAR-10 dataset. First the data is split into train and test images while loading the dataset. I have used Keras to load the dataset and train the VGG-16 model.
- After the dataset is split into training and test images, few training samples are visualized and can be found in Appendix J.
- Next the output training samples are preprocessed (converted to categorical variables) by one hot encoding method using Keras utils "to_categorical".
- The modified network architecture from original paper [1] is shown in the Appendix B.
- The total trainable parameters are 18,924,618. The experiment is run over 50 epochs and loss function used is 'SGD', optimizer is 'Adam' with learning rate of '1e-3'.
- The training performance results screenshots for every epoch can be found in Appendix F.

4.2.3 Experiment 3 – Model Training on ResNet-50 architecture using MNIST:

- The ResNet-50 model (Appendix C) has been trained on MNIST dataset. First the data is split into train and test images while loading the dataset. I have used Keras to load the dataset and train the ResNet-50 model.
- After the dataset is split into training and test images, few training samples are visualized and can be found in Appendix I.
- Next the output training samples are preprocessed (converted to categorical variables) by one hot encoding method using Keras utils "to_categorical".
- The modified network architecture from original paper [2] is shown in the Appendix C.
- The total trainable parameters are 23,601,930. The experiment is run over 50 epochs with batch size of 128 and loss function used is 'categorical crossentropy', optimizer is 'Adam'.
- The training performance results screenshots for every epoch can be found in Appendix G.

4.2.4 Experiment 4 – Model Training on ResNet-50 architecture using CIFAR10:

- The ResNet-50 model (Appendix D) has been trained on CIFAR10 dataset. First the data is split into train and test images while loading the dataset. I have used Keras to load the dataset and train the ResNet-50 model.
- After the dataset is split into training and test images, few training samples are visualized and can be found in Appendix J.
- Next the output training samples are preprocessed (converted to categorical variables) by one hot encoding method using Keras utils "to_categorical".

- The modified network architecture from original paper [2] is shown in the Appendix D.
- The total trainable parameters are 23,608,202. The experiment is run over 50 epochs with batch size of 32 and loss function used is 'categorical_crossentropy', optimizer is 'Adam' with learning rate of '0.001'.
- The training performance results screenshots for every epoch can be found in Appendix H.

4.3 Testing results

4.3.1 Testing results of Experiment 1 – Model Training on VGG-16 architecture using MNIST

- The training loss and validation loss over 50 epochs are recorded and the graph which shows the decrease in loss with minor fluctuations over 50 epochs can be found in Appendix K.
- Training accuracy and validation accuracy over 50 epochs are recorded and the graph which shows the increase in accuracy with minor fluctuations can be found in Appendix L.
- Evaluating the loss and accuracy over 50 epoch it can be found the loss gradually decreases and accuracy increases. The overall testing accuracy achieved is 99.33 %.
- The final test results for incorrectly classified test samples and correctly classified test samples are visualized and can be found in Appendix M.

4.3.2 Testing results of Experiment 2 – Model Training on VGG-16 architecture using CIFAR-10

- The training loss and validation loss over 50 epochs are recorded and the graph which shows the decrease in loss with minor fluctuations over 50 epochs can be found in Appendix N.
- Training accuracy and validation accuracy over 50 epochs are recorded and the graph which shows the increase in accuracy with minor fluctuations can be found in Appendix O.
- Evaluating the loss and accuracy over 50 epoch it can be found the loss gradually decreases and accuracy increases. The overall testing accuracy achieved is 85.17 %.
- The final test results for incorrectly classified test samples and correctly classified test samples are visualized and can be found in Appendix P.

4.3.3 Testing results of Experiment 3 – Model Training on ResNet-50 architecture using MNIST

- The training loss and validation loss over 50 epochs are recorded and the graph which shows the decrease in loss with minor fluctuations over 50 epochs can be found in Appendix Q.
- Training accuracy and validation accuracy over 50 epochs are recorded and the graph which shows the increase in accuracy with minor fluctuations can be found in Appendix R.

- Evaluating the loss and accuracy over 50 epoch it can be found the loss gradually decreases and accuracy increases. The overall testing accuracy achieved is 98.96 %.
- The final test results for incorrectly classified test samples and correctly classified test samples are visualized and can be found in Appendix S.

4.3.4 Testing results of Experiment 4 – Model Training on ResNet-50 architecture using CIFAR-10

- The training loss and validation loss over 50 epochs are recorded and the graph which shows the decrease in loss with minor fluctuations over 50 epochs can be found in Appendix T.
- Training accuracy and validation accuracy over 50 epochs are recorded and the graph which shows the increase in accuracy with minor fluctuations can be found in Appendix U.
- Evaluating the loss and accuracy over 50 epoch it can be found the loss gradually decreases and accuracy increases. The overall testing accuracy achieved is 70 %.
- The final test results for incorrectly classified test samples and correctly classified test samples are visualized and can be found in Appendix V.

4.4 Further Evaluation

Observing the performance of VGG-16 and ResNet deep models using MNIST and CIFAR-10 datasets, a greater model performance can be achieved by following different training strategies such as choosing a different loss function or increasing the batch size along with greater number of epoch size. As these very deep CNNs takes longer time to train experimenting with different settings can be very challenging given the set of computational resources, but there is always room for improvement. The deep models can be further evaluated using different datasets such as CIFAR-1000 or Tiny-ImageNet or one can come up with their own dataset.

5. Conclusion

The different experiments on VGG-16 and ResNet deep CNNs shows that these models perform exceptionally well on Image classification tasks. Minor changes to the original network architecture along with various training strategies such as including activation, loss function, optimization, regularization can increase the model performance. While VGG-16 performed well on both MNIST and CIFAR-10, the ResNet performance on CIFAR-10 was less because of using the actual CIFAR-10 image size as input to the CNN. If the input images have been resized to 224x224x3 according to [1] a greater performance could be expected. Overall, comparing the performance of both VGG-16 and ResNet models both perform equally well on MNIST and CIFAR-10 datasets.

References:

[1] K. Simonyan and A. Zisserman, “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION,” ICLR, vol. 75, no. 6, pp. 398–406, 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” Multimed. Tools Appl., vol. 77, no. 9, pp. 10437–10453, Dec. 2015.

[3] X. Liu, Z. Deng, and Y. Yang, “Recent progress in semantic image segmentation,” Artif. Intell. Rev., vol. 52, no. 2, pp. 1089–1106, 2019

[4] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images,” in Advances in neural information processing systems, 2012, pp. 2843–2851.

[5] K. Fukushima and S. Miyake, “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition,” in Competition and cooperation in neural nets, Springer, 1982, pp. 267–285.

[6] X. Zhang and Y. LeCun, “Text understanding from scratch,” arXiv Prepr. arXiv1502.01710, 2015.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[8] Y. LeCun et al., “Learning algorithms for classification: A comparison on handwritten digit recognition,” Neural networks Stat. Mech. Perspect., vol. 261, p. 276, 1995.

[9] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, “Handwritten digit recognition: benchmarking of state-of-the-art techniques,” Pattern Recognit., vol. 36, no. 10, pp. 2271– 2285, 2003.

[10] K. Chellapilla, S. Puri, and P. Simard, “High performance convolutional neural networks for document processing,” in Tenth International Workshop on Frontiers in Handwriting Recognition, 2006.

[11] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Deep , Big , Simple Neural Nets for Handwritten,” Neural Comput., vol. 22, no. 12, pp. 3207–3220, 2010.

[12] Y. LeCun, K. Kavukcuoglu, C. C. Farabet, and others, “Convolutional networks and applications in vision,” in ISCAS, 2010, vol. 2010, pp. 253–256.

[13] M. Matsugu, K. Mori, M. Ishii, and Y. Mitarai, “Convolutional spiking neural network model for robust face detection,” in Neural Information Processing, 2002. ICONIP’02. Proceedings of the 9th International Conference on, 2002, vol. 2, pp. 660–664.

[14] Y.-N. Chen, C.-C. Han, C.-T. Wang, B.-S. Jeng, and K.-C. Fan, “The application of a convolution neural network on face and license plate detection,” in Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, 2006, vol. 3, pp. 552–555.

[15] B. Fasel, “Facial expression analysis using shape and motion information extracted by convolutional neural networks,” in Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on, 2002, pp. 607–616.

[16] F. J. Huang, Y.-L. Boureau, Y. LeCun, and others, “Unsupervised learning of invariant feature hierarchies with applications to object recognition,” in Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on, 2007, pp. 1–8.

[17] K.-S. Oh and K. Jung, “GPU implementation of neural networks,” Pattern Recognit., vol. 37, no. 6, pp. 1311–1314, 2004.
[18] O. Russakovsky et al., “ImageNet Large Scale Visual Recognition Challenge,” Int. J. Comput. Vis., 2015.
[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Adv. Neural Inf. Process. Syst., pp. 1–9, 2012.
[20] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” arXiv Prepr. arXiv1311.2901v3, vol. 30, no. 1–2, pp. 225–231, 2013.
[21] K. Simonyan and A. Zisserman, “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION,” ICLR, vol. 75, no. 6, pp. 398–406, 2015.
[22] C. Szegedy et al., “Going deeper with convolutions,” in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, vol. 57, no. 3, pp. 1–9.

Appendix A – VGG-16 Architecture using MNIST as input dataset

Below is the model summary of VGG-16 architecture built using MNIST as input dataset,

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape_1 (Reshape)          (None, 28, 28, 1)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 64)        640
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 128)       73856
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 256)         295168
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 7, 7, 256)         590080
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 7, 7, 256)         590080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 256)         0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 3, 3, 512)         1180160
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 3, 3, 512)         2359808
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 3, 3, 512)         2359808
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 1, 1, 512)         0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 1, 1, 512)         2359808
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 1, 1, 512)         2359808
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 1, 1, 512)         2359808
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                2570
_________________________________________________________________
activation_1 (Activation)    (None, 10)                0
=================================================================
Total params: 14,913,226
Trainable params: 14,913,226
Non-trainable params: 0
_________________________________________________________________
None
```

Appendix B – VGG-16 Architecture using CIFAR-10 as input dataset

Below is the model summary of VGG-16 architecture built using CIFAR-10 as input dataset,

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_29 (Conv2D)           (None, 32, 32, 64)        1792
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 32, 32, 64)        36928
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 16, 16, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
dropout_5 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 16, 16, 128)       73856
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 16, 16, 128)       147584
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 8, 8, 128)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
dropout_6 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 8, 8, 256)         295168
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 8, 8, 256)         590080
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 8, 8, 256)         590080
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 4, 4, 256)         0
_________________________________________________________________
batch_normalization_3 (Batch (None, 4, 4, 256)         1024
_________________________________________________________________
dropout_7 (Dropout)          (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 4, 4, 512)         1180160
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 4, 4, 512)         2359808
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 4, 4, 512)         2359808
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 2, 2, 512)         0
_________________________________________________________________
batch_normalization_4 (Batch (None, 2, 2, 512)         2048
_________________________________________________________________
dropout_8 (Dropout)          (None, 2, 2, 512)         0
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 2, 2, 512)         2359808
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 2, 2, 512)         2359808
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 2, 2, 512)         2359808
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 1, 1, 512)         0
_________________________________________________________________
batch_normalization_5 (Batch (None, 1, 1, 512)         2048
_________________________________________________________________
dropout_9 (Dropout)          (None, 1, 1, 512)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 512)               0
_________________________________________________________________
```

```
dense_7 (Dense)              (None, 4096)              2101248
_________________________________________________________________
dropout_10 (Dropout)         (None, 4096)              0
_________________________________________________________________
dense_8 (Dense)              (None, 512)               2097664
_________________________________________________________________
dropout_11 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_9 (Dense)              (None, 10)                5130
=================================================================
Total params: 18,924,618
Trainable params: 18,921,674
Non-trainable params: 2,944
_________________________________________________________________
None
```

Appendix C – ResNet-50 Architecture with MNIST as input dataset

Below is the model summary of ResNet-50 architecture built using MNIST as input dataset,

```
Model: "ResNet50"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            (None, 28, 28, 1)    0
__________________________________________________________________________________________________
zero_padding2d_2 (ZeroPadding2D (None, 34, 34, 1)    0           input_2[0][0]
__________________________________________________________________________________________________
conv1 (Conv2D)                  (None, 14, 14, 64)   3200        zero_padding2d_2[0][0]
__________________________________________________________________________________________________
bn_conv1 (BatchNormalization)   (None, 14, 14, 64)   256         conv1[0][0]
__________________________________________________________________________________________________
activation_50 (Activation)      (None, 14, 14, 64)   0           bn_conv1[0][0]
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 6, 6, 64)     0           activation_50[0][0]
__________________________________________________________________________________________________
res2a_branch2a (Conv2D)         (None, 6, 6, 64)     4160        max_pooling2d_2[0][0]
__________________________________________________________________________________________________
bn2a_branch2a (BatchNormalizati (None, 6, 6, 64)     256         res2a_branch2a[0][0]
__________________________________________________________________________________________________
activation_51 (Activation)      (None, 6, 6, 64)     0           bn2a_branch2a[0][0]
__________________________________________________________________________________________________
res2a_branch2b (Conv2D)         (None, 6, 6, 64)     36928       activation_51[0][0]
__________________________________________________________________________________________________
bn2a_branch2b (BatchNormalizati (None, 6, 6, 64)     256         res2a_branch2b[0][0]
__________________________________________________________________________________________________
activation_52 (Activation)      (None, 6, 6, 64)     0           bn2a_branch2b[0][0]
__________________________________________________________________________________________________
res2a_branch2c (Conv2D)         (None, 6, 6, 256)    16640       activation_52[0][0]
__________________________________________________________________________________________________
res2a_branch1 (Conv2D)          (None, 6, 6, 256)    16640       max_pooling2d_2[0][0]
__________________________________________________________________________________________________
bn2a_branch2c (BatchNormalizati (None, 6, 6, 256)    1024        res2a_branch2c[0][0]
__________________________________________________________________________________________________
bn2a_branch1 (BatchNormalizatio (None, 6, 6, 256)    1024        res2a_branch1[0][0]
__________________________________________________________________________________________________
add_17 (Add)                    (None, 6, 6, 256)    0           bn2a_branch2c[0][0]
                                                                 bn2a_branch1[0][0]
__________________________________________________________________________________________________
activation_53 (Activation)      (None, 6, 6, 256)    0           add_17[0][0]
__________________________________________________________________________________________________
res2b_branch2a (Conv2D)         (None, 6, 6, 64)     16448       activation_53[0][0]
__________________________________________________________________________________________________
bn2b_branch2a (BatchNormalizati (None, 6, 6, 64)     256         res2b_branch2a[0][0]
__________________________________________________________________________________________________
activation_54 (Activation)      (None, 6, 6, 64)     0           bn2b_branch2a[0][0]
__________________________________________________________________________________________________
res2b_branch2b (Conv2D)         (None, 6, 6, 64)     36928       activation_54[0][0]
__________________________________________________________________________________________________
bn2b_branch2b (BatchNormalizati (None, 6, 6, 64)     256         res2b_branch2b[0][0]
__________________________________________________________________________________________________
activation_55 (Activation)      (None, 6, 6, 64)     0           bn2b_branch2b[0][0]
__________________________________________________________________________________________________
res2b_branch2c (Conv2D)         (None, 6, 6, 256)    16640       activation_55[0][0]
__________________________________________________________________________________________________
bn2b_branch2c (BatchNormalizati (None, 6, 6, 256)    1024        res2b_branch2c[0][0]
__________________________________________________________________________________________________
add_18 (Add)                    (None, 6, 6, 256)    0           bn2b_branch2c[0][0]
                                                                 activation_53[0][0]
__________________________________________________________________________________________________
activation_56 (Activation)      (None, 6, 6, 256)    0           add_18[0][0]
__________________________________________________________________________________________________
```

res2c_branch2a (Conv2D)	(None, 6, 6, 64)	16448	activation_56[0][0]
bn2c_branch2a (BatchNormalizati	(None, 6, 6, 64)	256	res2c_branch2a[0][0]
activation_57 (Activation)	(None, 6, 6, 64)	0	bn2c_branch2a[0][0]
res2c_branch2b (Conv2D)	(None, 6, 6, 64)	36928	activation_57[0][0]
bn2c_branch2b (BatchNormalizati	(None, 6, 6, 64)	256	res2c_branch2b[0][0]
activation_58 (Activation)	(None, 6, 6, 64)	0	bn2c_branch2b[0][0]
res2c_branch2c (Conv2D)	(None, 6, 6, 256)	16640	activation_58[0][0]
bn2c_branch2c (BatchNormalizati	(None, 6, 6, 256)	1024	res2c_branch2c[0][0]
add_19 (Add)	(None, 6, 6, 256)	0	bn2c_branch2c[0][0] activation_56[0][0]
activation_59 (Activation)	(None, 6, 6, 256)	0	add_19[0][0]
res3a_branch2a (Conv2D)	(None, 3, 3, 128)	32896	activation_59[0][0]
bn3a_branch2a (BatchNormalizati	(None, 3, 3, 128)	512	res3a_branch2a[0][0]
activation_60 (Activation)	(None, 3, 3, 128)	0	bn3a_branch2a[0][0]
res3a_branch2b (Conv2D)	(None, 3, 3, 128)	147584	activation_60[0][0]
bn3a_branch2b (BatchNormalizati	(None, 3, 3, 128)	512	res3a_branch2b[0][0]
activation_61 (Activation)	(None, 3, 3, 128)	0	bn3a_branch2b[0][0]
res3a_branch2c (Conv2D)	(None, 3, 3, 512)	66048	activation_61[0][0]
res3a_branch1 (Conv2D)	(None, 3, 3, 512)	131584	activation_59[0][0]
bn3a_branch2c (BatchNormalizati	(None, 3, 3, 512)	2048	res3a_branch2c[0][0]
bn3a_branch1 (BatchNormalizatio	(None, 3, 3, 512)	2048	res3a_branch1[0][0]
add_20 (Add)	(None, 3, 3, 512)	0	bn3a_branch2c[0][0] bn3a_branch1[0][0]
activation_62 (Activation)	(None, 3, 3, 512)	0	add_20[0][0]
res3b_branch2a (Conv2D)	(None, 3, 3, 128)	65664	activation_62[0][0]
bn3b_branch2a (BatchNormalizati	(None, 3, 3, 128)	512	res3b_branch2a[0][0]
activation_63 (Activation)	(None, 3, 3, 128)	0	bn3b_branch2a[0][0]
res3b_branch2b (Conv2D)	(None, 3, 3, 128)	147584	activation_63[0][0]
bn3b_branch2b (BatchNormalizati	(None, 3, 3, 128)	512	res3b_branch2b[0][0]
activation_64 (Activation)	(None, 3, 3, 128)	0	bn3b_branch2b[0][0]
res3b_branch2c (Conv2D)	(None, 3, 3, 512)	66048	activation_64[0][0]

bn3b_branch2c (BatchNormalizati	(None, 3, 3, 512)	2048	res3b_branch2c[0][0]
add_21 (Add)	(None, 3, 3, 512)	0	bn3b_branch2c[0][0] activation_62[0][0]
activation_65 (Activation)	(None, 3, 3, 512)	0	add_21[0][0]
res3c_branch2a (Conv2D)	(None, 3, 3, 128)	65664	activation_65[0][0]
bn3c_branch2a (BatchNormalizati	(None, 3, 3, 128)	512	res3c_branch2a[0][0]
activation_66 (Activation)	(None, 3, 3, 128)	0	bn3c_branch2a[0][0]
res3c_branch2b (Conv2D)	(None, 3, 3, 128)	147584	activation_66[0][0]
bn3c_branch2b (BatchNormalizati	(None, 3, 3, 128)	512	res3c_branch2b[0][0]
activation_67 (Activation)	(None, 3, 3, 128)	0	bn3c_branch2b[0][0]
res3c_branch2c (Conv2D)	(None, 3, 3, 512)	66048	activation_67[0][0]
bn3c_branch2c (BatchNormalizati	(None, 3, 3, 512)	2048	res3c_branch2c[0][0]
add_22 (Add)	(None, 3, 3, 512)	0	bn3c_branch2c[0][0] activation_65[0][0]
activation_68 (Activation)	(None, 3, 3, 512)	0	add_22[0][0]
res3d_branch2a (Conv2D)	(None, 3, 3, 128)	65664	activation_68[0][0]
bn3d_branch2a (BatchNormalizati	(None, 3, 3, 128)	512	res3d_branch2a[0][0]
activation_69 (Activation)	(None, 3, 3, 128)	0	bn3d_branch2a[0][0]
res3d_branch2b (Conv2D)	(None, 3, 3, 128)	147584	activation_69[0][0]
bn3d_branch2b (BatchNormalizati	(None, 3, 3, 128)	512	res3d_branch2b[0][0]
activation_70 (Activation)	(None, 3, 3, 128)	0	bn3d_branch2b[0][0]
res3d_branch2c (Conv2D)	(None, 3, 3, 512)	66048	activation_70[0][0]
bn3d_branch2c (BatchNormalizati	(None, 3, 3, 512)	2048	res3d_branch2c[0][0]
add_23 (Add)	(None, 3, 3, 512)	0	bn3d_branch2c[0][0] activation_68[0][0]
activation_71 (Activation)	(None, 3, 3, 512)	0	add_23[0][0]
res4a_branch2a (Conv2D)	(None, 2, 2, 256)	131328	activation_71[0][0]
bn4a_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4a_branch2a[0][0]
activation_72 (Activation)	(None, 2, 2, 256)	0	bn4a_branch2a[0][0]
res4a_branch2b (Conv2D)	(None, 2, 2, 256)	590080	activation_72[0][0]
bn4a_branch2b (BatchNormalizati	(None, 2, 2, 256)	1024	res4a_branch2b[0][0]
activation_73 (Activation)	(None, 2, 2, 256)	0	bn4a_branch2b[0][0]

activation_73 (Activation)	(None, 2, 2, 256)	0	bn4a_branch2b[0][0]
res4a_branch2c (Conv2D)	(None, 2, 2, 1024)	263168	activation_73[0][0]
res4a_branch1 (Conv2D)	(None, 2, 2, 1024)	525312	activation_71[0][0]
bn4a_branch2c (BatchNormalizati	(None, 2, 2, 1024)	4096	res4a_branch2c[0][0]
bn4a_branch1 (BatchNormalizatio	(None, 2, 2, 1024)	4096	res4a_branch1[0][0]
add_24 (Add)	(None, 2, 2, 1024)	0	bn4a_branch2c[0][0] bn4a_branch1[0][0]
activation_74 (Activation)	(None, 2, 2, 1024)	0	add_24[0][0]
res4b_branch2a (Conv2D)	(None, 2, 2, 256)	262400	activation_74[0][0]
bn4b_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4b_branch2a[0][0]
activation_75 (Activation)	(None, 2, 2, 256)	0	bn4b_branch2a[0][0]
res4b_branch2b (Conv2D)	(None, 2, 2, 256)	590080	activation_75[0][0]
bn4b_branch2b (BatchNormalizati	(None, 2, 2, 256)	1024	res4b_branch2b[0][0]
activation_76 (Activation)	(None, 2, 2, 256)	0	bn4b_branch2b[0][0]
res4b_branch2c (Conv2D)	(None, 2, 2, 1024)	263168	activation_76[0][0]
bn4b_branch2c (BatchNormalizati	(None, 2, 2, 1024)	4096	res4b_branch2c[0][0]
add_25 (Add)	(None, 2, 2, 1024)	0	bn4b_branch2c[0][0] activation_74[0][0]
activation_77 (Activation)	(None, 2, 2, 1024)	0	add_25[0][0]
res4c_branch2a (Conv2D)	(None, 2, 2, 256)	262400	activation_77[0][0]
bn4c_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4c_branch2a[0][0]
activation_78 (Activation)	(None, 2, 2, 256)	0	bn4c_branch2a[0][0]
res4c_branch2b (Conv2D)	(None, 2, 2, 256)	590080	activation_78[0][0]
bn4c_branch2b (BatchNormalizati	(None, 2, 2, 256)	1024	res4c_branch2b[0][0]
activation_79 (Activation)	(None, 2, 2, 256)	0	bn4c_branch2b[0][0]
res4c_branch2c (Conv2D)	(None, 2, 2, 1024)	263168	activation_79[0][0]
bn4c_branch2c (BatchNormalizati	(None, 2, 2, 1024)	4096	res4c_branch2c[0][0]
add_26 (Add)	(None, 2, 2, 1024)	0	bn4c_branch2c[0][0] activation_77[0][0]
activation_80 (Activation)	(None, 2, 2, 1024)	0	add_26[0][0]
res4d_branch2a (Conv2D)	(None, 2, 2, 256)	262400	activation_80[0][0]
bn4d_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4d_branch2a[0][0]

res4d_branch2a (Conv2D)	(None, 2, 2, 256)	262400	activation_80[0][0]
bn4d_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4d_branch2a[0][0]
activation_81 (Activation)	(None, 2, 2, 256)	0	bn4d_branch2a[0][0]
res4d_branch2b (Conv2D)	(None, 2, 2, 256)	590080	activation_81[0][0]
bn4d_branch2b (BatchNormalizati	(None, 2, 2, 256)	1024	res4d_branch2b[0][0]
activation_82 (Activation)	(None, 2, 2, 256)	0	bn4d_branch2b[0][0]
res4d_branch2c (Conv2D)	(None, 2, 2, 1024)	263168	activation_82[0][0]
bn4d_branch2c (BatchNormalizati	(None, 2, 2, 1024)	4096	res4d_branch2c[0][0]
add_27 (Add)	(None, 2, 2, 1024)	0	bn4d_branch2c[0][0] activation_80[0][0]
activation_83 (Activation)	(None, 2, 2, 1024)	0	add_27[0][0]
res4e_branch2a (Conv2D)	(None, 2, 2, 256)	262400	activation_83[0][0]
bn4e_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4e_branch2a[0][0]
activation_84 (Activation)	(None, 2, 2, 256)	0	bn4e_branch2a[0][0]
res4e_branch2b (Conv2D)	(None, 2, 2, 256)	590080	activation_84[0][0]
bn4e_branch2b (BatchNormalizati	(None, 2, 2, 256)	1024	res4e_branch2b[0][0]
activation_85 (Activation)	(None, 2, 2, 256)	0	bn4e_branch2b[0][0]
res4e_branch2c (Conv2D)	(None, 2, 2, 1024)	263168	activation_85[0][0]
bn4e_branch2c (BatchNormalizati	(None, 2, 2, 1024)	4096	res4e_branch2c[0][0]
add_28 (Add)	(None, 2, 2, 1024)	0	bn4e_branch2c[0][0] activation_83[0][0]
activation_86 (Activation)	(None, 2, 2, 1024)	0	add_28[0][0]
res4f_branch2a (Conv2D)	(None, 2, 2, 256)	262400	activation_86[0][0]
bn4f_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4f_branch2a[0][0]
activation_87 (Activation)	(None, 2, 2, 256)	0	bn4f_branch2a[0][0]
res4f_branch2b (Conv2D)	(None, 2, 2, 256)	590080	activation_87[0][0]
bn4f_branch2b (BatchNormalizati	(None, 2, 2, 256)	1024	res4f_branch2b[0][0]
activation_88 (Activation)	(None, 2, 2, 256)	0	bn4f_branch2b[0][0]
res4f_branch2c (Conv2D)	(None, 2, 2, 1024)	263168	activation_88[0][0]
bn4f_branch2c (BatchNormalizati	(None, 2, 2, 1024)	4096	res4f_branch2c[0][0]
add_29 (Add)	(None, 2, 2, 1024)	0	bn4f_branch2c[0][0] activation_86[0][0]
activation_89 (Activation)	(None, 2, 2, 1024)	0	add_29[0][0]

```
____________________________________________________________________________________________________
res5a_branch2a (Conv2D)         (None, 1, 1, 512)    524800      activation_89[0][0]
____________________________________________________________________________________________________
bn5a_branch2a (BatchNormalizati (None, 1, 1, 512)    2048        res5a_branch2a[0][0]
____________________________________________________________________________________________________
activation_90 (Activation)      (None, 1, 1, 512)    0           bn5a_branch2a[0][0]
____________________________________________________________________________________________________
res5a_branch2b (Conv2D)         (None, 1, 1, 512)    2359808     activation_90[0][0]
____________________________________________________________________________________________________
bn5a_branch2b (BatchNormalizati (None, 1, 1, 512)    2048        res5a_branch2b[0][0]
____________________________________________________________________________________________________
activation_91 (Activation)      (None, 1, 1, 512)    0           bn5a_branch2b[0][0]
____________________________________________________________________________________________________
res5a_branch2c (Conv2D)         (None, 1, 1, 2048)   1050624     activation_91[0][0]
____________________________________________________________________________________________________
res5a_branch1 (Conv2D)          (None, 1, 1, 2048)   2099200     activation_89[0][0]
____________________________________________________________________________________________________
bn5a_branch2c (BatchNormalizati (None, 1, 1, 2048)   8192        res5a_branch2c[0][0]
____________________________________________________________________________________________________
bn5a_branch1 (BatchNormalizatio (None, 1, 1, 2048)   8192        res5a_branch1[0][0]
____________________________________________________________________________________________________
add_30 (Add)                    (None, 1, 1, 2048)   0           bn5a_branch2c[0][0]
                                                                 bn5a_branch1[0][0]
____________________________________________________________________________________________________
activation_92 (Activation)      (None, 1, 1, 2048)   0           add_30[0][0]
____________________________________________________________________________________________________
res5b_branch2a (Conv2D)         (None, 1, 1, 512)    1049088     activation_92[0][0]
____________________________________________________________________________________________________
bn5b_branch2a (BatchNormalizati (None, 1, 1, 512)    2048        res5b_branch2a[0][0]
____________________________________________________________________________________________________
activation_93 (Activation)      (None, 1, 1, 512)    0           bn5b_branch2a[0][0]
____________________________________________________________________________________________________
res5b_branch2b (Conv2D)         (None, 1, 1, 512)    2359808     activation_93[0][0]
____________________________________________________________________________________________________
bn5b_branch2b (BatchNormalizati (None, 1, 1, 512)    2048        res5b_branch2b[0][0]
____________________________________________________________________________________________________
activation_94 (Activation)      (None, 1, 1, 512)    0           bn5b_branch2b[0][0]
____________________________________________________________________________________________________
res5b_branch2c (Conv2D)         (None, 1, 1, 2048)   1050624     activation_94[0][0]
____________________________________________________________________________________________________
bn5b_branch2c (BatchNormalizati (None, 1, 1, 2048)   8192        res5b_branch2c[0][0]
____________________________________________________________________________________________________
add_31 (Add)                    (None, 1, 1, 2048)   0           bn5b_branch2c[0][0]
                                                                 activation_92[0][0]
____________________________________________________________________________________________________
activation_95 (Activation)      (None, 1, 1, 2048)   0           add_31[0][0]
____________________________________________________________________________________________________
res5c_branch2a (Conv2D)         (None, 1, 1, 512)    1049088     activation_95[0][0]
____________________________________________________________________________________________________
bn5c_branch2a (BatchNormalizati (None, 1, 1, 512)    2048        res5c_branch2a[0][0]
____________________________________________________________________________________________________
activation_96 (Activation)      (None, 1, 1, 512)    0           bn5c_branch2a[0][0]
____________________________________________________________________________________________________
res5c_branch2b (Conv2D)         (None, 1, 1, 512)    2359808     activation_96[0][0]
____________________________________________________________________________________________________
bn5c_branch2b (BatchNormalizati (None, 1, 1, 512)    2048        res5c_branch2b[0][0]
____________________________________________________________________________________________________
activation_97 (Activation)      (None, 1, 1, 512)    0           bn5c_branch2b[0][0]
____________________________________________________________________________________________________
res5c_branch2c (Conv2D)         (None, 1, 1, 2048)   1050624     activation_97[0][0]
____________________________________________________________________________________________________
bn5c_branch2c (BatchNormalizati (None, 1, 1, 2048)   8192        res5c_branch2c[0][0]
____________________________________________________________________________________________________
add_32 (Add)                    (None, 1, 1, 2048)   0           bn5c_branch2c[0][0]
                                                                 activation_95[0][0]
____________________________________________________________________________________________________
activation_98 (Activation)      (None, 1, 1, 2048)   0           add_32[0][0]
____________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 2048)         0           activation_98[0][0]
____________________________________________________________________________________________________
fc10 (Dense)                    (None, 10)           20490       flatten_2[0][0]
====================================================================================================
Total params: 23,601,930
Trainable params: 23,548,810
Non-trainable params: 53,120
____________________________________________________________________________________________________
None
```

Appendix D – ResNet-50 Architecture with CIFAR-10 as input

Below is the model summary of ResNet-50 architecture built using CIFAR-10 as input dataset,

Model: "ResNet50"

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	(None, 32, 32, 3)	0	
zero_padding2d_2 (ZeroPadding2D	(None, 38, 38, 3)	0	input_2[0][0]
conv1 (Conv2D)	(None, 16, 16, 64)	9472	zero_padding2d_2[0][0]
bn_conv1 (BatchNormalization)	(None, 16, 16, 64)	256	conv1[0][0]
activation_50 (Activation)	(None, 16, 16, 64)	0	bn_conv1[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 7, 7, 64)	0	activation_50[0][0]
res2a_branch2a (Conv2D)	(None, 7, 7, 64)	4160	max_pooling2d_2[0][0]
bn2a_branch2a (BatchNormalizati	(None, 7, 7, 64)	256	res2a_branch2a[0][0]
activation_51 (Activation)	(None, 7, 7, 64)	0	bn2a_branch2a[0][0]
res2a_branch2b (Conv2D)	(None, 7, 7, 64)	36928	activation_51[0][0]
bn2a_branch2b (BatchNormalizati	(None, 7, 7, 64)	256	res2a_branch2b[0][0]
activation_52 (Activation)	(None, 7, 7, 64)	0	bn2a_branch2b[0][0]
res2a_branch2c (Conv2D)	(None, 7, 7, 256)	16640	activation_52[0][0]
res2a_branch1 (Conv2D)	(None, 7, 7, 256)	16640	max_pooling2d_2[0][0]
bn2a_branch2c (BatchNormalizati	(None, 7, 7, 256)	1024	res2a_branch2c[0][0]
bn2a_branch1 (BatchNormalizatio	(None, 7, 7, 256)	1024	res2a_branch1[0][0]
add_17 (Add)	(None, 7, 7, 256)	0	bn2a_branch2c[0][0] bn2a_branch1[0][0]
activation_53 (Activation)	(None, 7, 7, 256)	0	add_17[0][0]
res2b_branch2a (Conv2D)	(None, 7, 7, 64)	16448	activation_53[0][0]
bn2b_branch2a (BatchNormalizati	(None, 7, 7, 64)	256	res2b_branch2a[0][0]
activation_54 (Activation)	(None, 7, 7, 64)	0	bn2b_branch2a[0][0]
res2b_branch2b (Conv2D)	(None, 7, 7, 64)	36928	activation_54[0][0]
bn2b_branch2b (BatchNormalizati	(None, 7, 7, 64)	256	res2b_branch2b[0][0]
activation_55 (Activation)	(None, 7, 7, 64)	0	bn2b_branch2b[0][0]
res2b_branch2c (Conv2D)	(None, 7, 7, 256)	16640	activation_55[0][0]
bn2b_branch2c (BatchNormalizati	(None, 7, 7, 256)	1024	res2b_branch2c[0][0]
add_18 (Add)	(None, 7, 7, 256)	0	bn2b_branch2c[0][0] activation_53[0][0]
activation_56 (Activation)	(None, 7, 7, 256)	0	add_18[0][0]

res2c_branch2a (Conv2D)	(None, 7, 7, 64)	16448	activation_56[0][0]
bn2c_branch2a (BatchNormalizati	(None, 7, 7, 64)	256	res2c_branch2a[0][0]
activation_57 (Activation)	(None, 7, 7, 64)	0	bn2c_branch2a[0][0]
res2c_branch2b (Conv2D)	(None, 7, 7, 64)	36928	activation_57[0][0]
bn2c_branch2b (BatchNormalizati	(None, 7, 7, 64)	256	res2c_branch2b[0][0]
activation_58 (Activation)	(None, 7, 7, 64)	0	bn2c_branch2b[0][0]
res2c_branch2c (Conv2D)	(None, 7, 7, 256)	16640	activation_58[0][0]
bn2c_branch2c (BatchNormalizati	(None, 7, 7, 256)	1024	res2c_branch2c[0][0]
add_19 (Add)	(None, 7, 7, 256)	0	bn2c_branch2c[0][0] activation_56[0][0]
activation_59 (Activation)	(None, 7, 7, 256)	0	add_19[0][0]
res3a_branch2a (Conv2D)	(None, 4, 4, 128)	32896	activation_59[0][0]
bn3a_branch2a (BatchNormalizati	(None, 4, 4, 128)	512	res3a_branch2a[0][0]
activation_60 (Activation)	(None, 4, 4, 128)	0	bn3a_branch2a[0][0]
res3a_branch2b (Conv2D)	(None, 4, 4, 128)	147584	activation_60[0][0]
bn3a_branch2b (BatchNormalizati	(None, 4, 4, 128)	512	res3a_branch2b[0][0]
activation_61 (Activation)	(None, 4, 4, 128)	0	bn3a_branch2b[0][0]
res3a_branch2c (Conv2D)	(None, 4, 4, 512)	66048	activation_61[0][0]
res3a_branch1 (Conv2D)	(None, 4, 4, 512)	131584	activation_59[0][0]
bn3a_branch2c (BatchNormalizati	(None, 4, 4, 512)	2048	res3a_branch2c[0][0]
bn3a_branch1 (BatchNormalizatio	(None, 4, 4, 512)	2048	res3a_branch1[0][0]
add_20 (Add)	(None, 4, 4, 512)	0	bn3a_branch2c[0][0] bn3a_branch1[0][0]
activation_62 (Activation)	(None, 4, 4, 512)	0	add_20[0][0]
res3b_branch2a (Conv2D)	(None, 4, 4, 128)	65664	activation_62[0][0]
bn3b_branch2a (BatchNormalizati	(None, 4, 4, 128)	512	res3b_branch2a[0][0]
activation_63 (Activation)	(None, 4, 4, 128)	0	bn3b_branch2a[0][0]
res3b_branch2b (Conv2D)	(None, 4, 4, 128)	147584	activation_63[0][0]
bn3b_branch2b (BatchNormalizati	(None, 4, 4, 128)	512	res3b_branch2b[0][0]
activation_64 (Activation)	(None, 4, 4, 128)	0	bn3b_branch2b[0][0]
res3b_branch2c (Conv2D)	(None, 4, 4, 512)	66048	activation_64[0][0]
bn3b_branch2c (BatchNormalizati	(None, 4, 4, 512)	2048	res3b_branch2c[0][0]

add_21 (Add)	(None, 4, 4, 512)	0	bn3b_branch2c[0][0] activation_62[0][0]
activation_65 (Activation)	(None, 4, 4, 512)	0	add_21[0][0]
res3c_branch2a (Conv2D)	(None, 4, 4, 128)	65664	activation_65[0][0]
bn3c_branch2a (BatchNormalizati	(None, 4, 4, 128)	512	res3c_branch2a[0][0]
activation_66 (Activation)	(None, 4, 4, 128)	0	bn3c_branch2a[0][0]
res3c_branch2b (Conv2D)	(None, 4, 4, 128)	147584	activation_66[0][0]
bn3c_branch2b (BatchNormalizati	(None, 4, 4, 128)	512	res3c_branch2b[0][0]
activation_67 (Activation)	(None, 4, 4, 128)	0	bn3c_branch2b[0][0]
res3c_branch2c (Conv2D)	(None, 4, 4, 512)	66048	activation_67[0][0]
bn3c_branch2c (BatchNormalizati	(None, 4, 4, 512)	2048	res3c_branch2c[0][0]
add_22 (Add)	(None, 4, 4, 512)	0	bn3c_branch2c[0][0] activation_65[0][0]
activation_68 (Activation)	(None, 4, 4, 512)	0	add_22[0][0]
res3d_branch2a (Conv2D)	(None, 4, 4, 128)	65664	activation_68[0][0]
bn3d_branch2a (BatchNormalizati	(None, 4, 4, 128)	512	res3d_branch2a[0][0]
activation_69 (Activation)	(None, 4, 4, 128)	0	bn3d_branch2a[0][0]
res3d_branch2b (Conv2D)	(None, 4, 4, 128)	147584	activation_69[0][0]
bn3d_branch2b (BatchNormalizati	(None, 4, 4, 128)	512	res3d_branch2b[0][0]
activation_70 (Activation)	(None, 4, 4, 128)	0	bn3d_branch2b[0][0]
res3d_branch2c (Conv2D)	(None, 4, 4, 512)	66048	activation_70[0][0]
bn3d_branch2c (BatchNormalizati	(None, 4, 4, 512)	2048	res3d_branch2c[0][0]
add_23 (Add)	(None, 4, 4, 512)	0	bn3d_branch2c[0][0] activation_68[0][0]
activation_71 (Activation)	(None, 4, 4, 512)	0	add_23[0][0]
res4a_branch2a (Conv2D)	(None, 2, 2, 256)	131328	activation_71[0][0]
bn4a_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4a_branch2a[0][0]
activation_72 (Activation)	(None, 2, 2, 256)	0	bn4a_branch2a[0][0]
res4a_branch2b (Conv2D)	(None, 2, 2, 256)	590080	activation_72[0][0]
bn4a_branch2b (BatchNormalizati	(None, 2, 2, 256)	1024	res4a_branch2b[0][0]
activation_73 (Activation)	(None, 2, 2, 256)	0	bn4a_branch2b[0][0]
res4a_branch2c (Conv2D)	(None, 2, 2, 1024)	263168	activation_73[0][0]
res4a_branch1 (Conv2D)	(None, 2, 2, 1024)	525312	activation_71[0][0]

bn4a_branch2c (BatchNormalizati	(None, 2, 2, 1024)	4096	res4a_branch2c[0][0]
bn4a_branch1 (BatchNormalizatio	(None, 2, 2, 1024)	4096	res4a_branch1[0][0]
add_24 (Add)	(None, 2, 2, 1024)	0	bn4a_branch2c[0][0] bn4a_branch1[0][0]
activation_74 (Activation)	(None, 2, 2, 1024)	0	add_24[0][0]
res4b_branch2a (Conv2D)	(None, 2, 2, 256)	262400	activation_74[0][0]
bn4b_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4b_branch2a[0][0]
activation_75 (Activation)	(None, 2, 2, 256)	0	bn4b_branch2a[0][0]
res4b_branch2b (Conv2D)	(None, 2, 2, 256)	590080	activation_75[0][0]
bn4b_branch2b (BatchNormalizati	(None, 2, 2, 256)	1024	res4b_branch2b[0][0]
activation_76 (Activation)	(None, 2, 2, 256)	0	bn4b_branch2b[0][0]
res4b_branch2c (Conv2D)	(None, 2, 2, 1024)	263168	activation_76[0][0]
bn4b_branch2c (BatchNormalizati	(None, 2, 2, 1024)	4096	res4b_branch2c[0][0]
add_25 (Add)	(None, 2, 2, 1024)	0	bn4b_branch2c[0][0] activation_74[0][0]
activation_77 (Activation)	(None, 2, 2, 1024)	0	add_25[0][0]
res4c_branch2a (Conv2D)	(None, 2, 2, 256)	262400	activation_77[0][0]
bn4c_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4c_branch2a[0][0]
activation_78 (Activation)	(None, 2, 2, 256)	0	bn4c_branch2a[0][0]
res4c_branch2b (Conv2D)	(None, 2, 2, 256)	590080	activation_78[0][0]
bn4c_branch2b (BatchNormalizati	(None, 2, 2, 256)	1024	res4c_branch2b[0][0]
activation_79 (Activation)	(None, 2, 2, 256)	0	bn4c_branch2b[0][0]
res4c_branch2c (Conv2D)	(None, 2, 2, 1024)	263168	activation_79[0][0]
bn4c_branch2c (BatchNormalizati	(None, 2, 2, 1024)	4096	res4c_branch2c[0][0]
add_26 (Add)	(None, 2, 2, 1024)	0	bn4c_branch2c[0][0] activation_77[0][0]
activation_80 (Activation)	(None, 2, 2, 1024)	0	add_26[0][0]
res4d_branch2a (Conv2D)	(None, 2, 2, 256)	262400	activation_80[0][0]
bn4d_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4d_branch2a[0][0]
activation_81 (Activation)	(None, 2, 2, 256)	0	bn4d_branch2a[0][0]
res4d_branch2b (Conv2D)	(None, 2, 2, 256)	590080	activation_81[0][0]
bn4d_branch2b (BatchNormalizati	(None, 2, 2, 256)	1024	res4d_branch2b[0][0]
activation_82 (Activation)	(None, 2, 2, 256)	0	bn4d_branch2b[0][0]

res4d_branch2c (Conv2D)	(None, 2, 2, 1024)	263168	activation_82[0][0]
bn4d_branch2c (BatchNormalizati	(None, 2, 2, 1024)	4096	res4d_branch2c[0][0]
add_27 (Add)	(None, 2, 2, 1024)	0	bn4d_branch2c[0][0] activation_80[0][0]
activation_83 (Activation)	(None, 2, 2, 1024)	0	add_27[0][0]
res4e_branch2a (Conv2D)	(None, 2, 2, 256)	262400	activation_83[0][0]
bn4e_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4e_branch2a[0][0]
activation_84 (Activation)	(None, 2, 2, 256)	0	bn4e_branch2a[0][0]
res4e_branch2b (Conv2D)	(None, 2, 2, 256)	590080	activation_84[0][0]
bn4e_branch2b (BatchNormalizati	(None, 2, 2, 256)	1024	res4e_branch2b[0][0]
activation_85 (Activation)	(None, 2, 2, 256)	0	bn4e_branch2b[0][0]
res4e_branch2c (Conv2D)	(None, 2, 2, 1024)	263168	activation_85[0][0]
bn4e_branch2c (BatchNormalizati	(None, 2, 2, 1024)	4096	res4e_branch2c[0][0]
add_28 (Add)	(None, 2, 2, 1024)	0	bn4e_branch2c[0][0] activation_83[0][0]
activation_86 (Activation)	(None, 2, 2, 1024)	0	add_28[0][0]
res4f_branch2a (Conv2D)	(None, 2, 2, 256)	262400	activation_86[0][0]
bn4f_branch2a (BatchNormalizati	(None, 2, 2, 256)	1024	res4f_branch2a[0][0]
activation_87 (Activation)	(None, 2, 2, 256)	0	bn4f_branch2a[0][0]
res4f_branch2b (Conv2D)	(None, 2, 2, 256)	590080	activation_87[0][0]
bn4f_branch2b (BatchNormalizati	(None, 2, 2, 256)	1024	res4f_branch2b[0][0]
activation_88 (Activation)	(None, 2, 2, 256)	0	bn4f_branch2b[0][0]
res4f_branch2c (Conv2D)	(None, 2, 2, 1024)	263168	activation_88[0][0]
bn4f_branch2c (BatchNormalizati	(None, 2, 2, 1024)	4096	res4f_branch2c[0][0]
add_29 (Add)	(None, 2, 2, 1024)	0	bn4f_branch2c[0][0] activation_86[0][0]
activation_89 (Activation)	(None, 2, 2, 1024)	0	add_29[0][0]
res5a_branch2a (Conv2D)	(None, 1, 1, 512)	524800	activation_89[0][0]
bn5a_branch2a (BatchNormalizati	(None, 1, 1, 512)	2048	res5a_branch2a[0][0]
activation_90 (Activation)	(None, 1, 1, 512)	0	bn5a_branch2a[0][0]
res5a_branch2b (Conv2D)	(None, 1, 1, 512)	2359808	activation_90[0][0]
bn5a_branch2b (BatchNormalizati	(None, 1, 1, 512)	2048	res5a_branch2b[0][0]
activation_91 (Activation)	(None, 1, 1, 512)	0	bn5a_branch2b[0][0]

res5a_branch2c (Conv2D)	(None, 1, 1, 2048)	1050624	activation_91[0][0]
res5a_branch1 (Conv2D)	(None, 1, 1, 2048)	2099200	activation_89[0][0]
bn5a_branch2c (BatchNormalizati	(None, 1, 1, 2048)	8192	res5a_branch2c[0][0]
bn5a_branch1 (BatchNormalizatio	(None, 1, 1, 2048)	8192	res5a_branch1[0][0]
add_30 (Add)	(None, 1, 1, 2048)	0	bn5a_branch2c[0][0] bn5a_branch1[0][0]
activation_92 (Activation)	(None, 1, 1, 2048)	0	add_30[0][0]
res5b_branch2a (Conv2D)	(None, 1, 1, 512)	1049088	activation_92[0][0]
bn5b_branch2a (BatchNormalizati	(None, 1, 1, 512)	2048	res5b_branch2a[0][0]
activation_93 (Activation)	(None, 1, 1, 512)	0	bn5b_branch2a[0][0]
res5b_branch2b (Conv2D)	(None, 1, 1, 512)	2359808	activation_93[0][0]
bn5b_branch2b (BatchNormalizati	(None, 1, 1, 512)	2048	res5b_branch2b[0][0]
activation_94 (Activation)	(None, 1, 1, 512)	0	bn5b_branch2b[0][0]
res5b_branch2c (Conv2D)	(None, 1, 1, 2048)	1050624	activation_94[0][0]
bn5b_branch2c (BatchNormalizati	(None, 1, 1, 2048)	8192	res5b_branch2c[0][0]
add_31 (Add)	(None, 1, 1, 2048)	0	bn5b_branch2c[0][0] activation_92[0][0]
activation_95 (Activation)	(None, 1, 1, 2048)	0	add_31[0][0]
res5c_branch2a (Conv2D)	(None, 1, 1, 512)	1049088	activation_95[0][0]
bn5c_branch2a (BatchNormalizati	(None, 1, 1, 512)	2048	res5c_branch2a[0][0]
activation_96 (Activation)	(None, 1, 1, 512)	0	bn5c_branch2a[0][0]
res5c_branch2b (Conv2D)	(None, 1, 1, 512)	2359808	activation_96[0][0]
bn5c_branch2b (BatchNormalizati	(None, 1, 1, 512)	2048	res5c_branch2b[0][0]
activation_97 (Activation)	(None, 1, 1, 512)	0	bn5c_branch2b[0][0]
res5c_branch2c (Conv2D)	(None, 1, 1, 2048)	1050624	activation_97[0][0]
bn5c_branch2c (BatchNormalizati	(None, 1, 1, 2048)	8192	res5c_branch2c[0][0]
add_32 (Add)	(None, 1, 1, 2048)	0	bn5c_branch2c[0][0] activation_95[0][0]
activation_98 (Activation)	(None, 1, 1, 2048)	0	add_32[0][0]
flatten_2 (Flatten)	(None, 2048)	0	activation_98[0][0]
fc10 (Dense)	(None, 10)	20490	flatten_2[0][0]

Total params: 23,608,202
Trainable params: 23,555,082
Non-trainable params: 53,120

Appendix E – Training performance of VGG-16 using MNIST

Below are the runtime screenshots of VGG-16 model trained using MNIST dataset run over 50 epochs,

```
Train on 45000 samples, validate on 15000 samples
Epoch 1/50
45000/45000 [==============================] - 20s 453us/step - loss: 0.1132 - accuracy: 0.9623 - val_loss: 0.0922 - val_accuracy: 0.9808
Epoch 2/50
45000/45000 [==============================] - 13s 290us/step - loss: 0.0216 - accuracy: 0.9943 - val_loss: 0.0122 - val_accuracy: 0.9968
Epoch 3/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0129 - accuracy: 0.9966 - val_loss: 0.0190 - val_accuracy: 0.9950
Epoch 4/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0095 - accuracy: 0.9975 - val_loss: 0.0095 - val_accuracy: 0.9977
Epoch 5/50
45000/45000 [==============================] - 13s 293us/step - loss: 0.0075 - accuracy: 0.9982 - val_loss: 0.0087 - val_accuracy: 0.9979
Epoch 6/50
45000/45000 [==============================] - 13s 294us/step - loss: 0.0058 - accuracy: 0.9986 - val_loss: 0.0110 - val_accuracy: 0.9976
Epoch 7/50
45000/45000 [==============================] - 13s 293us/step - loss: 0.0049 - accuracy: 0.9988 - val_loss: 0.0119 - val_accuracy: 0.9977
Epoch 8/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0040 - accuracy: 0.9990 - val_loss: 0.0144 - val_accuracy: 0.9978
Epoch 9/50
45000/45000 [==============================] - 13s 293us/step - loss: 0.0045 - accuracy: 0.9990 - val_loss: 0.0195 - val_accuracy: 0.9965
Epoch 10/50
45000/45000 [==============================] - 13s 293us/step - loss: 0.0044 - accuracy: 0.9990 - val_loss: 0.0088 - val_accuracy: 0.9981
Epoch 11/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0035 - accuracy: 0.9993 - val_loss: 0.0133 - val_accuracy: 0.9983
Epoch 12/50
45000/45000 [==============================] - 13s 293us/step - loss: 0.0032 - accuracy: 0.9994 - val_loss: 0.0108 - val_accuracy: 0.9975
Epoch 13/50
45000/45000 [==============================] - 13s 293us/step - loss: 0.0033 - accuracy: 0.9994 - val_loss: 0.0091 - val_accuracy: 0.9981
Epoch 14/50
45000/45000 [==============================] - 13s 293us/step - loss: 0.0031 - accuracy: 0.9994 - val_loss: 0.0142 - val_accuracy: 0.9979
Epoch 15/50
45000/45000 [==============================] - 13s 294us/step - loss: 0.0032 - accuracy: 0.9994 - val_loss: 0.0173 - val_accuracy: 0.9983
Epoch 16/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0037 - accuracy: 0.9993 - val_loss: 0.0108 - val_accuracy: 0.9982
Epoch 17/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0037 - accuracy: 0.9993 - val_loss: 0.0165 - val_accuracy: 0.9984
Epoch 18/50
45000/45000 [==============================] - 13s 293us/step - loss: 0.0034 - accuracy: 0.9994 - val_loss: 0.0125 - val_accuracy: 0.9984
Epoch 19/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0036 - accuracy: 0.9994 - val_loss: 0.0174 - val_accuracy: 0.9983
Epoch 20/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0041 - accuracy: 0.9994 - val_loss: 0.0124 - val_accuracy: 0.9981
Epoch 21/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0042 - accuracy: 0.9993 - val_loss: 0.0157 - val_accuracy: 0.9982
Epoch 22/50
45000/45000 [==============================] - 13s 294us/step - loss: 0.0041 - accuracy: 0.9994 - val_loss: 0.0129 - val_accuracy: 0.9982
Epoch 23/50
45000/45000 [==============================] - 13s 291us/step - loss: 0.0048 - accuracy: 0.9992 - val_loss: 0.0135 - val_accuracy: 0.9979
Epoch 24/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0044 - accuracy: 0.9993 - val_loss: 0.0162 - val_accuracy: 0.9981
Epoch 25/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0058 - accuracy: 0.9992 - val_loss: 0.0153 - val_accuracy: 0.9985
Epoch 26/50
45000/45000 [==============================] - 13s 291us/step - loss: 0.0041 - accuracy: 0.9993 - val_loss: 0.0232 - val_accuracy: 0.9981
Epoch 27/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0059 - accuracy: 0.9993 - val_loss: 0.0202 - val_accuracy: 0.9985
Epoch 28/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0082 - accuracy: 0.9992 - val_loss: 0.0217 - val_accuracy: 0.9980
Epoch 29/50
45000/45000 [==============================] - 13s 291us/step - loss: 0.0057 - accuracy: 0.9993 - val_loss: 0.0147 - val_accuracy: 0.9979
Epoch 30/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0055 - accuracy: 0.9993 - val_loss: 0.0174 - val_accuracy: 0.9983
Epoch 31/50
45000/45000 [==============================] - 13s 291us/step - loss: 0.0083 - accuracy: 0.9992 - val_loss: 0.0157 - val_accuracy: 0.9978
```

```
Epoch 32/50
45000/45000 [==============================] - 13s 293us/step - loss: 0.0069 - accuracy: 0.9992 - val_loss: 0.0165 - val_accuracy: 0.9984
Epoch 33/50
45000/45000 [==============================] - 13s 291us/step - loss: 0.0146 - accuracy: 0.9987 - val_loss: 0.0275 - val_accuracy: 0.9975
Epoch 34/50
45000/45000 [==============================] - 13s 290us/step - loss: 0.0064 - accuracy: 0.9992 - val_loss: 0.0159 - val_accuracy: 0.9978
Epoch 35/50
45000/45000 [==============================] - 13s 290us/step - loss: 0.0068 - accuracy: 0.9991 - val_loss: 0.0207 - val_accuracy: 0.9982
Epoch 36/50
45000/45000 [==============================] - 13s 290us/step - loss: 0.0101 - accuracy: 0.9990 - val_loss: 0.0355 - val_accuracy: 0.9976
Epoch 37/50
45000/45000 [==============================] - 13s 289us/step - loss: 0.0101 - accuracy: 0.9990 - val_loss: 0.0260 - val_accuracy: 0.9974
Epoch 38/50
45000/45000 [==============================] - 13s 290us/step - loss: 0.0090 - accuracy: 0.9991 - val_loss: 0.0271 - val_accuracy: 0.9977
Epoch 39/50
45000/45000 [==============================] - 13s 290us/step - loss: 0.0144 - accuracy: 0.9988 - val_loss: 0.0382 - val_accuracy: 0.9974
Epoch 40/50
45000/45000 [==============================] - 13s 290us/step - loss: 0.0091 - accuracy: 0.9990 - val_loss: 0.0166 - val_accuracy: 0.9980
Epoch 41/50
45000/45000 [==============================] - 13s 290us/step - loss: 0.0187 - accuracy: 0.9985 - val_loss: 0.0105 - val_accuracy: 0.9982
Epoch 42/50
45000/45000 [==============================] - 13s 290us/step - loss: 0.0128 - accuracy: 0.9988 - val_loss: 0.0145 - val_accuracy: 0.9987
Epoch 43/50
45000/45000 [==============================] - 13s 290us/step - loss: 0.0077 - accuracy: 0.9992 - val_loss: 0.0188 - val_accuracy: 0.9980
Epoch 44/50
45000/45000 [==============================] - 13s 291us/step - loss: 0.0073 - accuracy: 0.9994 - val_loss: 0.0240 - val_accuracy: 0.9983
Epoch 45/50
45000/45000 [==============================] - 13s 290us/step - loss: 0.0085 - accuracy: 0.9991 - val_loss: 0.0191 - val_accuracy: 0.9984
Epoch 46/50
45000/45000 [==============================] - 13s 293us/step - loss: 0.0057 - accuracy: 0.9994 - val_loss: 0.0174 - val_accuracy: 0.9985
Epoch 47/50
45000/45000 [==============================] - 13s 289us/step - loss: 0.0059 - accuracy: 0.9993 - val_loss: 0.0214 - val_accuracy: 0.9983
Epoch 48/50
45000/45000 [==============================] - 13s 290us/step - loss: 0.0096 - accuracy: 0.9991 - val_loss: 0.0196 - val_accuracy: 0.9982
Epoch 49/50
45000/45000 [==============================] - 13s 292us/step - loss: 0.0086 - accuracy: 0.9990 - val_loss: 0.0244 - val_accuracy: 0.9980
Epoch 50/50
45000/45000 [==============================] - 13s 293us/step - loss: 0.0076 - accuracy: 0.9992 - val_loss: 0.0175 - val_accuracy: 0.9986
CPU times: user 6min 31s, sys: 2min 9s, total: 8min 41s
Wall time: 11min 6s
```

Appendix F – Training performance of VGG-16 using CIFAR-10

Below are the runtime screenshots of VGG-16 model trained using CIFAR-10 dataset run over 50 epochs

```
Train on 45000 samples, validate on 5000 samples
Epoch 1/50
45000/45000 [==============================] - 94s 2ms/step - loss: 2.0826 - accuracy: 0.1693 - val_loss: 2.0757 - val_accuracy: 0.1886
Epoch 2/50
45000/45000 [==============================] - 90s 2ms/step - loss: 1.9781 - accuracy: 0.1871 - val_loss: 2.0707 - val_accuracy: 0.1788
Epoch 3/50
45000/45000 [==============================] - 90s 2ms/step - loss: 1.9241 - accuracy: 0.2112 - val_loss: 1.9613 - val_accuracy: 0.2356
Epoch 4/50
45000/45000 [==============================] - 92s 2ms/step - loss: 1.8503 - accuracy: 0.2530 - val_loss: 1.9522 - val_accuracy: 0.2958
Epoch 5/50
45000/45000 [==============================] - 92s 2ms/step - loss: 1.6493 - accuracy: 0.3425 - val_loss: 1.3907 - val_accuracy: 0.4484
Epoch 6/50
45000/45000 [==============================] - 91s 2ms/step - loss: 1.4667 - accuracy: 0.4374 - val_loss: 1.4144 - val_accuracy: 0.4430
Epoch 7/50
45000/45000 [==============================] - 91s 2ms/step - loss: 1.2543 - accuracy: 0.5468 - val_loss: 1.3293 - val_accuracy: 0.5730
Epoch 8/50
45000/45000 [==============================] - 91s 2ms/step - loss: 1.0866 - accuracy: 0.6245 - val_loss: 1.6778 - val_accuracy: 0.4578
Epoch 9/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.9928 - accuracy: 0.6625 - val_loss: 0.9195 - val_accuracy: 0.6748
Epoch 10/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.8888 - accuracy: 0.7042 - val_loss: 0.8161 - val_accuracy: 0.7172
Epoch 11/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.7993 - accuracy: 0.7344 - val_loss: 0.8520 - val_accuracy: 0.7116
Epoch 12/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.7375 - accuracy: 0.7584 - val_loss: 0.7407 - val_accuracy: 0.7544
Epoch 13/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.6740 - accuracy: 0.7830 - val_loss: 0.7302 - val_accuracy: 0.7716
Epoch 14/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.6209 - accuracy: 0.8004 - val_loss: 0.7450 - val_accuracy: 0.7592
Epoch 15/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.5801 - accuracy: 0.8139 - val_loss: 0.6391 - val_accuracy: 0.7948
Epoch 16/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.5347 - accuracy: 0.8299 - val_loss: 0.6102 - val_accuracy: 0.8094
Epoch 17/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.4941 - accuracy: 0.8416 - val_loss: 0.5268 - val_accuracy: 0.8344
Epoch 18/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.4646 - accuracy: 0.8537 - val_loss: 0.5557 - val_accuracy: 0.8156
Epoch 19/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.4310 - accuracy: 0.8627 - val_loss: 0.7362 - val_accuracy: 0.7840
Epoch 20/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.4020 - accuracy: 0.8736 - val_loss: 0.5523 - val_accuracy: 0.8338
Epoch 21/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.3685 - accuracy: 0.8835 - val_loss: 0.5935 - val_accuracy: 0.8236
Epoch 22/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.3551 - accuracy: 0.8902 - val_loss: 0.5755 - val_accuracy: 0.8228
Epoch 23/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.3323 - accuracy: 0.8944 - val_loss: 0.5504 - val_accuracy: 0.8394
Epoch 24/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.3185 - accuracy: 0.9013 - val_loss: 0.5519 - val_accuracy: 0.8382
Epoch 25/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.3003 - accuracy: 0.9049 - val_loss: 0.5422 - val_accuracy: 0.8464
Epoch 26/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.2768 - accuracy: 0.9140 - val_loss: 0.5474 - val_accuracy: 0.8412
Epoch 27/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.2672 - accuracy: 0.9170 - val_loss: 0.6254 - val_accuracy: 0.8320
Epoch 28/50
45000/45000 [==============================] - 89s 2ms/step - loss: 0.2483 - accuracy: 0.9243 - val_loss: 0.5702 - val_accuracy: 0.8438
Epoch 29/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.2324 - accuracy: 0.9305 - val_loss: 0.6150 - val_accuracy: 0.8322
Epoch 30/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.2254 - accuracy: 0.9318 - val_loss: 0.5755 - val_accuracy: 0.8398
Epoch 31/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.2171 - accuracy: 0.9339 - val_loss: 0.6308 - val_accuracy: 0.8442
```

```
Epoch 32/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.2017 - accuracy: 0.9395 - val_loss: 0.6323 - val_accuracy: 0.8458
Epoch 33/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.1951 - accuracy: 0.9426 - val_loss: 0.6541 - val_accuracy: 0.8460
Epoch 34/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.1892 - accuracy: 0.9430 - val_loss: 0.6379 - val_accuracy: 0.8422
Epoch 35/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.1844 - accuracy: 0.9456 - val_loss: 0.6574 - val_accuracy: 0.8412
Epoch 36/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.1705 - accuracy: 0.9488 - val_loss: 0.7057 - val_accuracy: 0.8378
Epoch 37/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.1648 - accuracy: 0.9512 - val_loss: 0.6752 - val_accuracy: 0.8444
Epoch 38/50
45000/45000 [==============================] - 91s 2ms/step - loss: 0.1638 - accuracy: 0.9516 - val_loss: 9.9152 - val_accuracy: 0.8522
Epoch 39/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.1543 - accuracy: 0.9542 - val_loss: 6.4186 - val_accuracy: 0.8482
Epoch 40/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.1517 - accuracy: 0.9554 - val_loss: 0.6577 - val_accuracy: 0.8636
Epoch 41/50
45000/45000 [==============================] - 90s 2ms/step - loss: 0.1522 - accuracy: 0.9560 - val_loss: 0.6642 - val_accuracy: 0.8570
Epoch 42/50
45000/45000 [==============================] - 91s 2ms/step - loss: 0.1413 - accuracy: 0.9596 - val_loss: 0.7443 - val_accuracy: 0.8468
Epoch 43/50
45000/45000 [==============================] - 91s 2ms/step - loss: 0.2418 - accuracy: 0.9312 - val_loss: 0.7357 - val_accuracy: 0.8496
Epoch 44/50
45000/45000 [==============================] - 93s 2ms/step - loss: 0.1254 - accuracy: 0.9648 - val_loss: 0.6760 - val_accuracy: 0.8548
Epoch 45/50
45000/45000 [==============================] - 93s 2ms/step - loss: 0.1194 - accuracy: 0.9669 - val_loss: 0.6576 - val_accuracy: 0.8578
Epoch 46/50
45000/45000 [==============================] - 93s 2ms/step - loss: 0.1256 - accuracy: 0.9648 - val_loss: 0.6530 - val_accuracy: 0.8474
Epoch 47/50
45000/45000 [==============================] - 92s 2ms/step - loss: 0.1230 - accuracy: 0.9660 - val_loss: 1.0665 - val_accuracy: 0.8602
Epoch 48/50
45000/45000 [==============================] - 91s 2ms/step - loss: 0.1269 - accuracy: 0.9634 - val_loss: 0.6992 - val_accuracy: 0.8582
Epoch 49/50
45000/45000 [==============================] - 91s 2ms/step - loss: 0.1251 - accuracy: 0.9650 - val_loss: 9.3798 - val_accuracy: 0.8446
Epoch 50/50
45000/45000 [==============================] - 91s 2ms/step - loss: 0.1174 - accuracy: 0.9674 - val_loss: 0.7435 - val_accuracy: 0.8528
CPU times: user 48min 23s, sys: 19min 37s, total: 1h 8min 1s
Wall time: 1h 15min 27s
```

Appendix G – Training performance of ResNet-50 using MNIST

Below are the runtime screenshots of ResNet-50 model trained using MNIST dataset run over 50 epochs

```
Train on 45000 samples, validate on 15000 samples
Epoch 1/50
45000/45000 [==============================] - 54s 1ms/step - loss: 0.3489 - accuracy: 0.9047 - val_loss: 0.1352 - val_accuracy: 0.9583
Epoch 2/50
45000/45000 [==============================] - 40s 894us/step - loss: 0.0738 - accuracy: 0.9773 - val_loss: 0.1301 - val_accuracy: 0.9628
Epoch 3/50
45000/45000 [==============================] - 40s 898us/step - loss: 0.0467 - accuracy: 0.9850 - val_loss: 0.1498 - val_accuracy: 0.9568
Epoch 4/50
45000/45000 [==============================] - 40s 890us/step - loss: 0.0428 - accuracy: 0.9861 - val_loss: 0.1013 - val_accuracy: 0.9690
Epoch 5/50
45000/45000 [==============================] - 40s 891us/step - loss: 0.0291 - accuracy: 0.9910 - val_loss: 0.0960 - val_accuracy: 0.9737
Epoch 6/50
45000/45000 [==============================] - 41s 910us/step - loss: 0.0314 - accuracy: 0.9905 - val_loss: 0.1063 - val_accuracy: 0.9735
Epoch 7/50
45000/45000 [==============================] - 41s 905us/step - loss: 0.0278 - accuracy: 0.9909 - val_loss: 0.0842 - val_accuracy: 0.9782
Epoch 8/50
45000/45000 [==============================] - 40s 897us/step - loss: 0.0494 - accuracy: 0.9856 - val_loss: 0.1498 - val_accuracy: 0.9576
Epoch 9/50
45000/45000 [==============================] - 41s 904us/step - loss: 0.0347 - accuracy: 0.9889 - val_loss: 0.1389 - val_accuracy: 0.9655
Epoch 10/50
45000/45000 [==============================] - 40s 890us/step - loss: 0.0223 - accuracy: 0.9929 - val_loss: 0.0874 - val_accuracy: 0.9777
Epoch 11/50
45000/45000 [==============================] - 40s 896us/step - loss: 0.0192 - accuracy: 0.9943 - val_loss: 0.1762 - val_accuracy: 0.9557
Epoch 12/50
45000/45000 [==============================] - 41s 909us/step - loss: 0.0210 - accuracy: 0.9934 - val_loss: 0.0512 - val_accuracy: 0.9855
Epoch 13/50
45000/45000 [==============================] - 41s 906us/step - loss: 0.0246 - accuracy: 0.9922 - val_loss: 0.1106 - val_accuracy: 0.9721
Epoch 14/50
45000/45000 [==============================] - 41s 902us/step - loss: 0.2318 - accuracy: 0.9426 - val_loss: 0.1438 - val_accuracy: 0.9555
Epoch 15/50
45000/45000 [==============================] - 41s 916us/step - loss: 0.0575 - accuracy: 0.9826 - val_loss: 0.0703 - val_accuracy: 0.9787
Epoch 16/50
45000/45000 [==============================] - 41s 913us/step - loss: 0.0377 - accuracy: 0.9882 - val_loss: 0.0802 - val_accuracy: 0.9761
Epoch 17/50
45000/45000 [==============================] - 41s 913us/step - loss: 0.0309 - accuracy: 0.9900 - val_loss: 0.0637 - val_accuracy: 0.9822
Epoch 18/50
45000/45000 [==============================] - 41s 906us/step - loss: 0.0215 - accuracy: 0.9927 - val_loss: 0.0641 - val_accuracy: 0.9825
Epoch 19/50
45000/45000 [==============================] - 41s 904us/step - loss: 0.0225 - accuracy: 0.9923 - val_loss: 0.0602 - val_accuracy: 0.9843
Epoch 20/50
45000/45000 [==============================] - 40s 898us/step - loss: 0.0191 - accuracy: 0.9937 - val_loss: 0.0541 - val_accuracy: 0.9858
Epoch 21/50
45000/45000 [==============================] - 40s 882us/step - loss: 0.0177 - accuracy: 0.9944 - val_loss: 0.0572 - val_accuracy: 0.9853
Epoch 22/50
45000/45000 [==============================] - 40s 883us/step - loss: 0.0191 - accuracy: 0.9940 - val_loss: 0.0921 - val_accuracy: 0.9793
Epoch 23/50
45000/45000 [==============================] - 40s 884us/step - loss: 0.0195 - accuracy: 0.9934 - val_loss: 0.0622 - val_accuracy: 0.9827
Epoch 24/50
45000/45000 [==============================] - 40s 880us/step - loss: 0.0154 - accuracy: 0.9951 - val_loss: 0.0678 - val_accuracy: 0.9817
Epoch 25/50
45000/45000 [==============================] - 40s 886us/step - loss: 0.0176 - accuracy: 0.9948 - val_loss: 0.0725 - val_accuracy: 0.9825
Epoch 26/50
45000/45000 [==============================] - 40s 887us/step - loss: 0.0160 - accuracy: 0.9945 - val_loss: 0.0683 - val_accuracy: 0.9835
Epoch 27/50
45000/45000 [==============================] - 39s 875us/step - loss: 0.0207 - accuracy: 0.9934 - val_loss: 0.0999 - val_accuracy: 0.9749
Epoch 28/50
45000/45000 [==============================] - 39s 874us/step - loss: 0.0153 - accuracy: 0.9951 - val_loss: 0.0452 - val_accuracy: 0.9885
Epoch 29/50
45000/45000 [==============================] - 39s 874us/step - loss: 0.0162 - accuracy: 0.9948 - val_loss: 0.0473 - val_accuracy: 0.9869
Epoch 30/50
45000/45000 [==============================] - 39s 868us/step - loss: 0.0165 - accuracy: 0.9948 - val_loss: 0.0540 - val_accuracy: 0.9862
Epoch 31/50
45000/45000 [==============================] - 39s 871us/step - loss: 0.0155 - accuracy: 0.9952 - val_loss: 0.0576 - val_accuracy: 0.9847
```

```
Epoch 32/50
45000/45000 [==============================] - 39s 869us/step - loss: 0.0170 - accuracy: 0.9947 - val_loss: 0.0639 - val_accuracy: 0.9821
Epoch 33/50
45000/45000 [==============================] - 39s 869us/step - loss: 0.0117 - accuracy: 0.9965 - val_loss: 0.0521 - val_accuracy: 0.9865
Epoch 34/50
45000/45000 [==============================] - 39s 875us/step - loss: 0.0146 - accuracy: 0.9956 - val_loss: 0.0694 - val_accuracy: 0.9841
Epoch 35/50
45000/45000 [==============================] - 39s 867us/step - loss: 0.0305 - accuracy: 0.9932 - val_loss: 3.8085 - val_accuracy: 0.6384
Epoch 36/50
45000/45000 [==============================] - 39s 865us/step - loss: 0.0802 - accuracy: 0.9830 - val_loss: 0.0501 - val_accuracy: 0.9851
Epoch 37/50
45000/45000 [==============================] - 39s 868us/step - loss: 0.0217 - accuracy: 0.9937 - val_loss: 0.0397 - val_accuracy: 0.9892
Epoch 38/50
45000/45000 [==============================] - 39s 866us/step - loss: 0.0160 - accuracy: 0.9950 - val_loss: 0.0473 - val_accuracy: 0.9869
Epoch 39/50
45000/45000 [==============================] - 39s 868us/step - loss: 0.0130 - accuracy: 0.9957 - val_loss: 0.0369 - val_accuracy: 0.9904
Epoch 40/50
45000/45000 [==============================] - 39s 862us/step - loss: 0.0108 - accuracy: 0.9964 - val_loss: 0.0455 - val_accuracy: 0.9893
Epoch 41/50
45000/45000 [==============================] - 39s 866us/step - loss: 0.0137 - accuracy: 0.9958 - val_loss: 0.0427 - val_accuracy: 0.9891
Epoch 42/50
45000/45000 [==============================] - 39s 868us/step - loss: 0.0113 - accuracy: 0.9963 - val_loss: 0.0451 - val_accuracy: 0.9897
Epoch 43/50
45000/45000 [==============================] - 39s 865us/step - loss: 0.0105 - accuracy: 0.9968 - val_loss: 0.0406 - val_accuracy: 0.9891
Epoch 44/50
45000/45000 [==============================] - 39s 865us/step - loss: 0.0130 - accuracy: 0.9959 - val_loss: 0.0505 - val_accuracy: 0.9866
Epoch 45/50
45000/45000 [==============================] - 39s 863us/step - loss: 0.0120 - accuracy: 0.9958 - val_loss: 0.0640 - val_accuracy: 0.9833
Epoch 46/50
45000/45000 [==============================] - 39s 864us/step - loss: 0.0129 - accuracy: 0.9963 - val_loss: 0.0427 - val_accuracy: 0.9883
Epoch 47/50
45000/45000 [==============================] - 39s 864us/step - loss: 0.0103 - accuracy: 0.9971 - val_loss: 0.0635 - val_accuracy: 0.9833
Epoch 48/50
45000/45000 [==============================] - 39s 864us/step - loss: 0.0123 - accuracy: 0.9962 - val_loss: 0.0557 - val_accuracy: 0.9867
Epoch 49/50
45000/45000 [==============================] - 39s 863us/step - loss: 0.0103 - accuracy: 0.9968 - val_loss: 0.0505 - val_accuracy: 0.9876
Epoch 50/50
45000/45000 [==============================] - 39s 867us/step - loss: 0.0110 - accuracy: 0.9965 - val_loss: 0.0391 - val_accuracy: 0.9906
```

Appendix H – Training performance of ResNet-50 using CIFAR-10

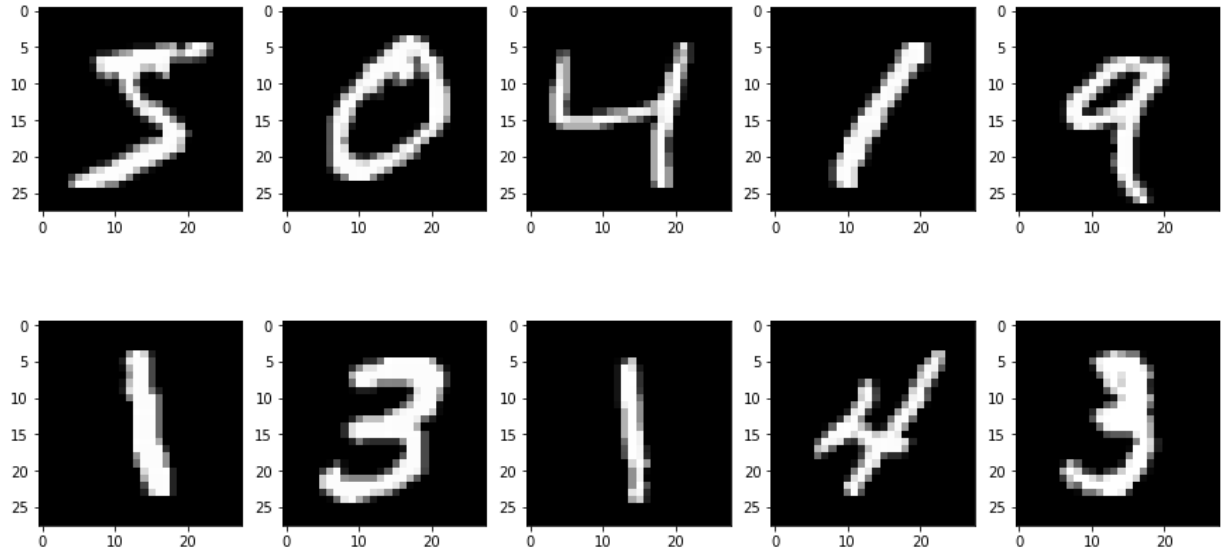
Below are the runtime screenshots of ResNet-50 model trained using CIFAR-10 dataset run over 50 epochs,

```
Train on 37500 samples, validate on 12500 samples
Epoch 1/50
37500/37500 [==============================] - 109s 3ms/step - loss: 2.1417 - accuracy: 0.3469 - val_loss: 2.4233 - val_accuracy: 0.1819
Epoch 2/50
37500/37500 [==============================] - 96s 3ms/step - loss: 2.1461 - accuracy: 0.3398 - val_loss: 9.5876 - val_accuracy: 0.3291
Epoch 3/50
37500/37500 [==============================] - 96s 3ms/step - loss: 1.9796 - accuracy: 0.3746 - val_loss: 192.6929 - val_accuracy: 0.1574
Epoch 4/50
37500/37500 [==============================] - 96s 3ms/step - loss: 1.9137 - accuracy: 0.3756 - val_loss: 2.0766 - val_accuracy: 0.4412
Epoch 5/50
37500/37500 [==============================] - 96s 3ms/step - loss: 1.7626 - accuracy: 0.4300 - val_loss: 2.6499 - val_accuracy: 0.2346
Epoch 6/50
37500/37500 [==============================] - 97s 3ms/step - loss: 1.7308 - accuracy: 0.4397 - val_loss: 1.6394 - val_accuracy: 0.4326
Epoch 7/50
37500/37500 [==============================] - 97s 3ms/step - loss: 1.7066 - accuracy: 0.4246 - val_loss: 2.3185 - val_accuracy: 0.3047
Epoch 8/50
37500/37500 [==============================] - 97s 3ms/step - loss: 1.6061 - accuracy: 0.4557 - val_loss: 2.7672 - val_accuracy: 0.2920
Epoch 9/50
37500/37500 [==============================] - 96s 3ms/step - loss: 1.6112 - accuracy: 0.4597 - val_loss: 1.4841 - val_accuracy: 0.5108
Epoch 10/50
37500/37500 [==============================] - 96s 3ms/step - loss: 1.4211 - accuracy: 0.5204 - val_loss: 1.2975 - val_accuracy: 0.5302
Epoch 11/50
37500/37500 [==============================] - 96s 3ms/step - loss: 1.3445 - accuracy: 0.5377 - val_loss: 1.3232 - val_accuracy: 0.5290
Epoch 12/50
37500/37500 [==============================] - 97s 3ms/step - loss: 1.2179 - accuracy: 0.5775 - val_loss: 1.3290 - val_accuracy: 0.5290
Epoch 13/50
37500/37500 [==============================] - 96s 3ms/step - loss: 1.1155 - accuracy: 0.6179 - val_loss: 1.0959 - val_accuracy: 0.6157
Epoch 14/50
37500/37500 [==============================] - 97s 3ms/step - loss: 1.0513 - accuracy: 0.6405 - val_loss: 1.4536 - val_accuracy: 0.4909
Epoch 15/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.9687 - accuracy: 0.6711 - val_loss: 1.0990 - val_accuracy: 0.6206
Epoch 16/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.9271 - accuracy: 0.6798 - val_loss: 1.3355 - val_accuracy: 0.5346
Epoch 17/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.8609 - accuracy: 0.7062 - val_loss: 0.9581 - val_accuracy: 0.6738
Epoch 18/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.7342 - accuracy: 0.7472 - val_loss: 1.5961 - val_accuracy: 0.5927
Epoch 19/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.7534 - accuracy: 0.7457 - val_loss: 3.4680 - val_accuracy: 0.2910
Epoch 20/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.6141 - accuracy: 0.7891 - val_loss: 0.9515 - val_accuracy: 0.6974
Epoch 21/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.5243 - accuracy: 0.8175 - val_loss: 0.9719 - val_accuracy: 0.6877
Epoch 22/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.4594 - accuracy: 0.8407 - val_loss: 1.3947 - val_accuracy: 0.6042
Epoch 23/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.4210 - accuracy: 0.8551 - val_loss: 1.0771 - val_accuracy: 0.6900
Epoch 24/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.3472 - accuracy: 0.8785 - val_loss: 1.1454 - val_accuracy: 0.6772
Epoch 25/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.2986 - accuracy: 0.8969 - val_loss: 1.1582 - val_accuracy: 0.6854
Epoch 26/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.2415 - accuracy: 0.9154 - val_loss: 1.0864 - val_accuracy: 0.7085
Epoch 27/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.2129 - accuracy: 0.9260 - val_loss: 1.1558 - val_accuracy: 0.7022
Epoch 28/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.2117 - accuracy: 0.9285 - val_loss: 1.2921 - val_accuracy: 0.6909
Epoch 29/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.1692 - accuracy: 0.9410 - val_loss: 1.2011 - val_accuracy: 0.7141
Epoch 30/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.1691 - accuracy: 0.9429 - val_loss: 1.2366 - val_accuracy: 0.7178
Epoch 31/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.1348 - accuracy: 0.9539 - val_loss: 1.3627 - val_accuracy: 0.6922
```

```
Epoch 32/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.1337 - accuracy: 0.9538 - val_loss: 1.3763 - val_accuracy: 0.7036
Epoch 33/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.1543 - accuracy: 0.9483 - val_loss: 1.3086 - val_accuracy: 0.6897
Epoch 34/50
37500/37500 [==============================] - 98s 3ms/step - loss: 0.1189 - accuracy: 0.9599 - val_loss: 1.3558 - val_accuracy: 0.7218
Epoch 35/50
37500/37500 [==============================] - 98s 3ms/step - loss: 0.0941 - accuracy: 0.9681 - val_loss: 1.5329 - val_accuracy: 0.7008
Epoch 36/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.1052 - accuracy: 0.9644 - val_loss: 1.4292 - val_accuracy: 0.7093
Epoch 37/50
37500/37500 [==============================] - 98s 3ms/step - loss: 0.1193 - accuracy: 0.9597 - val_loss: 1.3338 - val_accuracy: 0.7078
Epoch 38/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.1284 - accuracy: 0.9591 - val_loss: 1.5081 - val_accuracy: 0.7127
Epoch 39/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.0826 - accuracy: 0.9717 - val_loss: 1.7183 - val_accuracy: 0.6643
Epoch 40/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.0971 - accuracy: 0.9680 - val_loss: 1.4263 - val_accuracy: 0.7188
Epoch 41/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.1383 - accuracy: 0.9561 - val_loss: 1.3908 - val_accuracy: 0.7149
Epoch 42/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.0618 - accuracy: 0.9787 - val_loss: 1.5755 - val_accuracy: 0.7074
Epoch 43/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.0991 - accuracy: 0.9687 - val_loss: 1.4657 - val_accuracy: 0.7066
Epoch 44/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.0629 - accuracy: 0.9787 - val_loss: 1.4880 - val_accuracy: 0.7254
Epoch 45/50
37500/37500 [==============================] - 96s 3ms/step - loss: 0.0718 - accuracy: 0.9763 - val_loss: 2.0739 - val_accuracy: 0.6507
Epoch 46/50
37500/37500 [==============================] - 97s 3ms/step - loss: 0.1200 - accuracy: 0.9608 - val_loss: 1.4383 - val_accuracy: 0.7091
Epoch 47/50
37500/37500 [==============================] - 98s 3ms/step - loss: 0.0785 - accuracy: 0.9747 - val_loss: 1.4924 - val_accuracy: 0.7181
Epoch 48/50
37500/37500 [==============================] - 98s 3ms/step - loss: 0.0515 - accuracy: 0.9821 - val_loss: 1.5878 - val_accuracy: 0.7060
Epoch 49/50
37500/37500 [==============================] - 98s 3ms/step - loss: 0.0694 - accuracy: 0.9769 - val_loss: 1.6121 - val_accuracy: 0.6898
Epoch 50/50
37500/37500 [==============================] - 98s 3ms/step - loss: 0.0596 - accuracy: 0.9794 - val_loss: 1.6362 - val_accuracy: 0.7040
```

Appendix I – MNIST dataset input samples

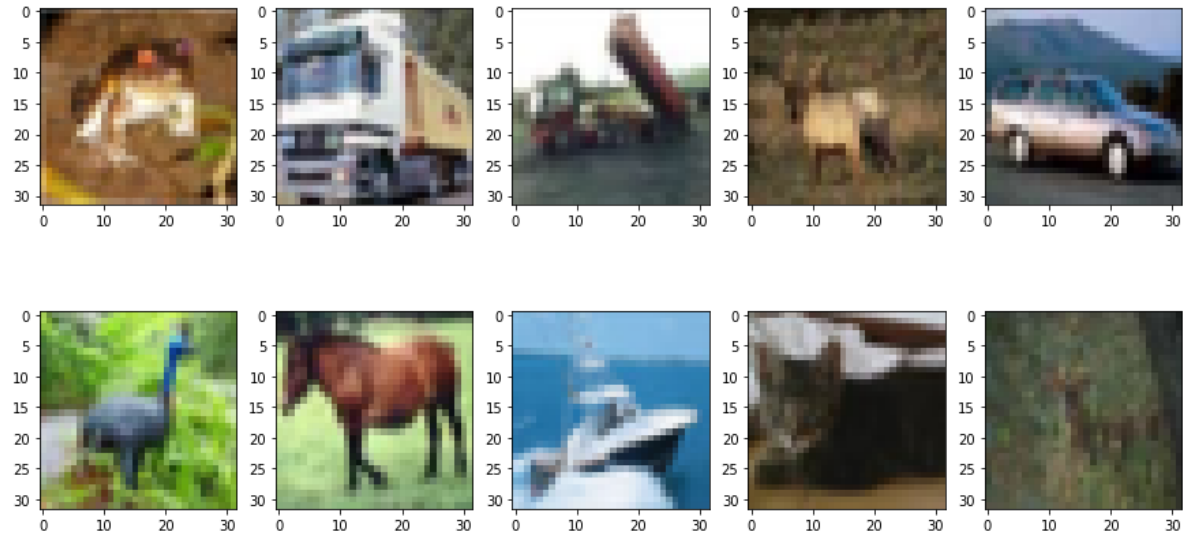
Visualization of few training samples from MNIST dataset are shown below,



```
Labels : [5 0 4 1 9]
Labels : [1 3 1 4 3]
```

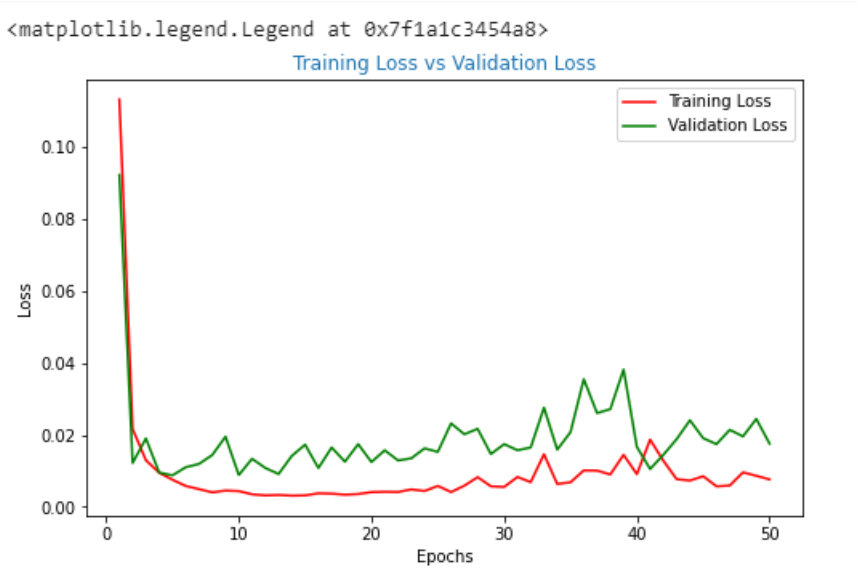
Appendix J – CIFAR-10 dataset Input samples

Visualization of few training samples from CIFAR-10 dataset are shown below,



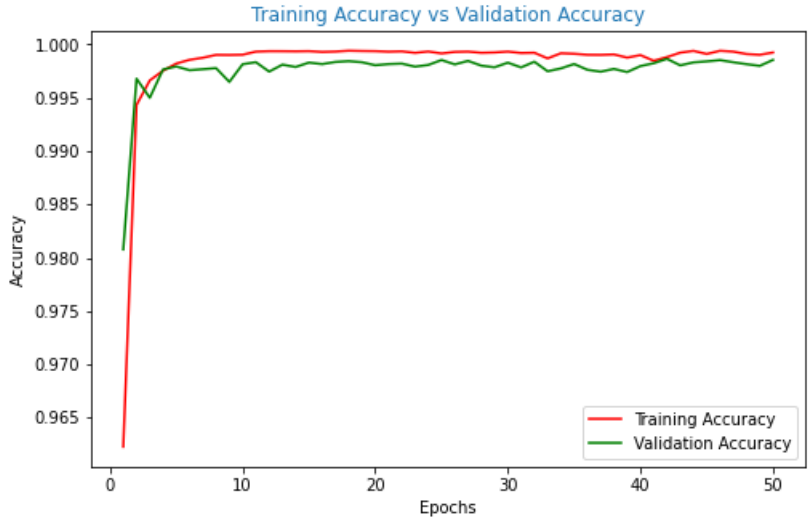
```
Labels : [[6]
 [9]
 [9]
 [4]
 [1]]
Labels : [[2]
 [7]
 [8]
 [3]
 [4]]
```

Appendix K – Graphical representation of Training Loss and Validation Loss over epoch size for VGG-16 model trained on MNIST dataset



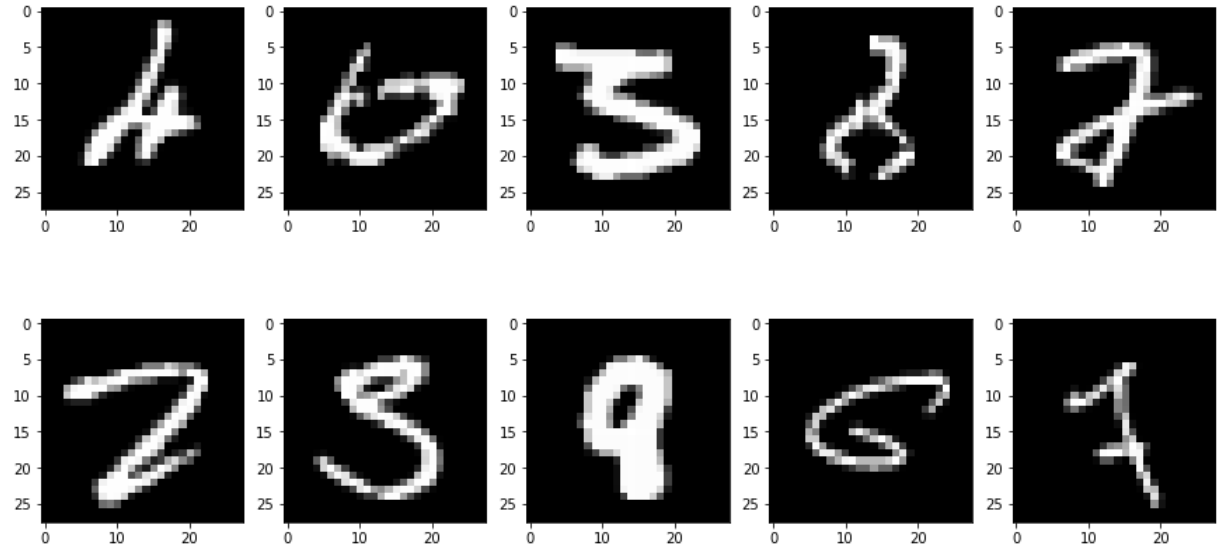
Appendix L – Graphical representation of Training Accuracy and Validation Accuracy over epoch size for VGG-16 model trained on MNIST dataset

```
<matplotlib.legend.Legend at 0x7f1a1c2ca128>
```



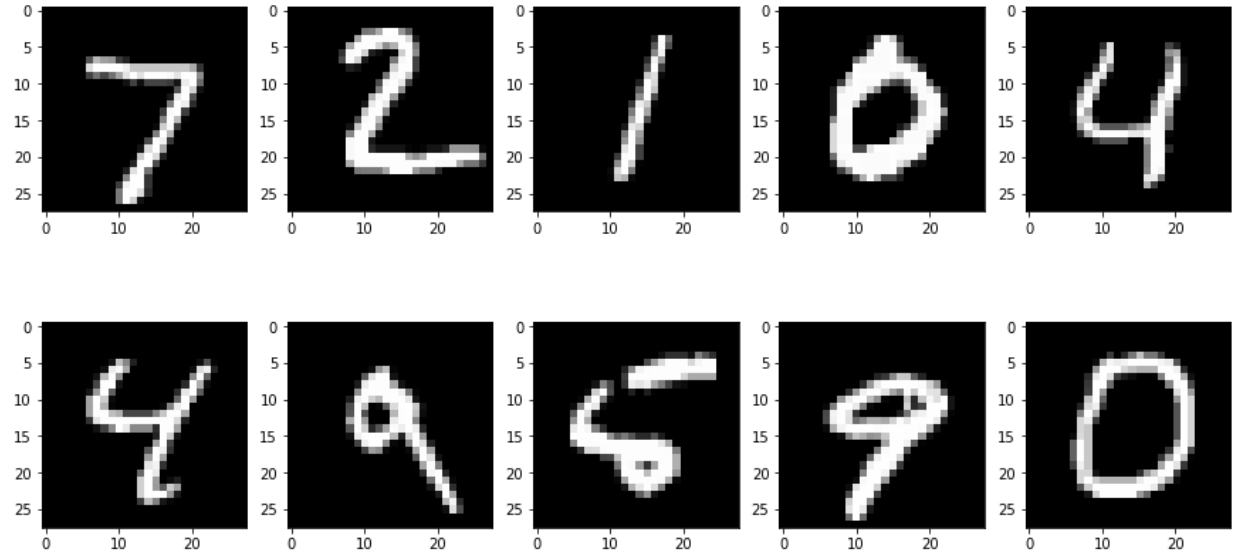
Appendix M – Test results of VGG-16 model trained on MNIST dataset

Visualization of few incorrectly classified test samples are shown below,



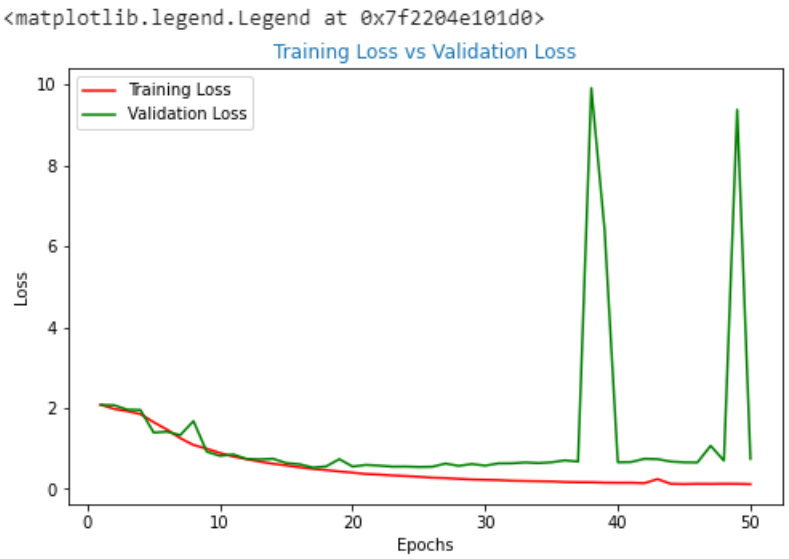
```
True:          [4 6 3 8 2]
classified as: [2 0 5 2 7]
True:          [2 3 8 6 7]
classified as: [7 5 9 5 1]
```

Visualization of few correctly classified test samples are shown below,



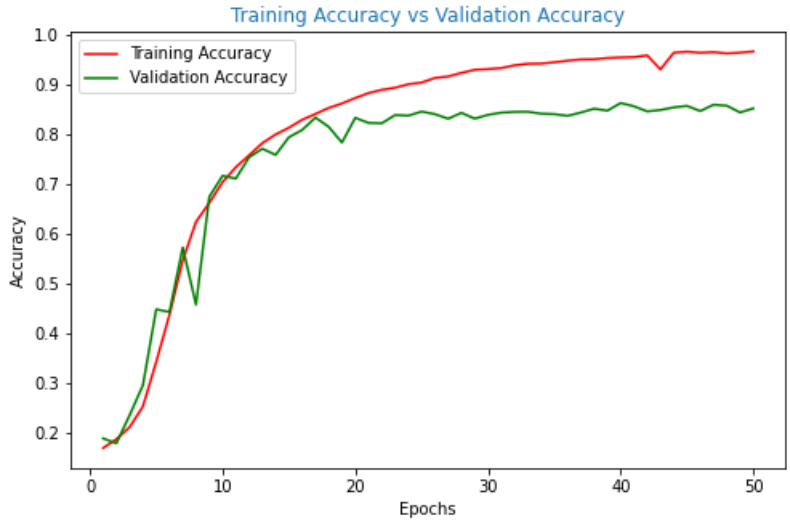
```
True:          [7 2 1 0 4]
classified as: [7 2 1 0 4]
True:          [4 9 5 9 0]
classified as: [4 9 5 9 0]
```

Appendix N - Graphical representation of Training Loss and Validation Loss over epoch size for VGG-16 model trained on CIFAR-10 dataset



Appendix O - Graphical representation of Training Accuracy and Validation Accuracy over epoch size for VGG-16 model trained on CIFAR-10 dataset

```
<matplotlib.legend.Legend at 0x7f2204d8dc18>
```

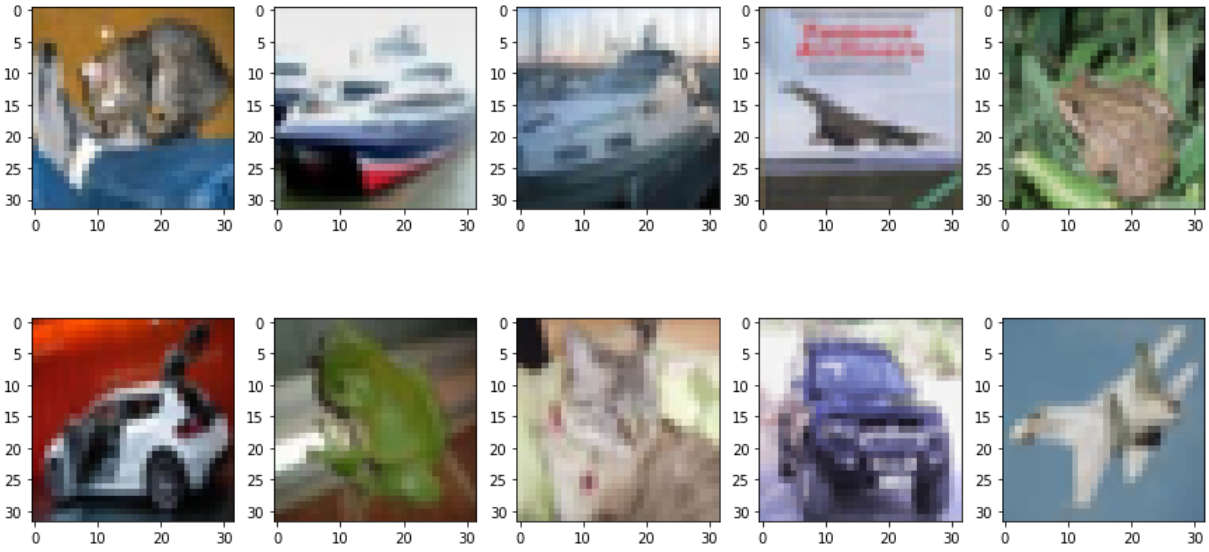


Appendix P – Test Results of VGG-16 model trained on CIFAR-10 dataset

Visualization of few incorrectly classified test samples are shown below,

```
True:           [[7]
 [5]
 [4]
 [4]
 [2]]
classified as: [9 4 3 2 8]
True:           [[7]
 [4]
 [6]
 [3]
 [9]]
classified as: [3 7 3 5 0]
```

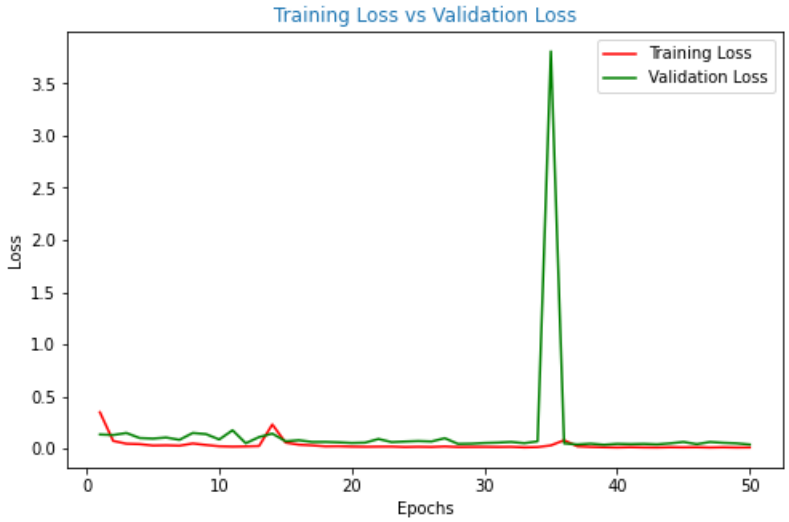
Visualization of few correctly classified test samples are shown below,



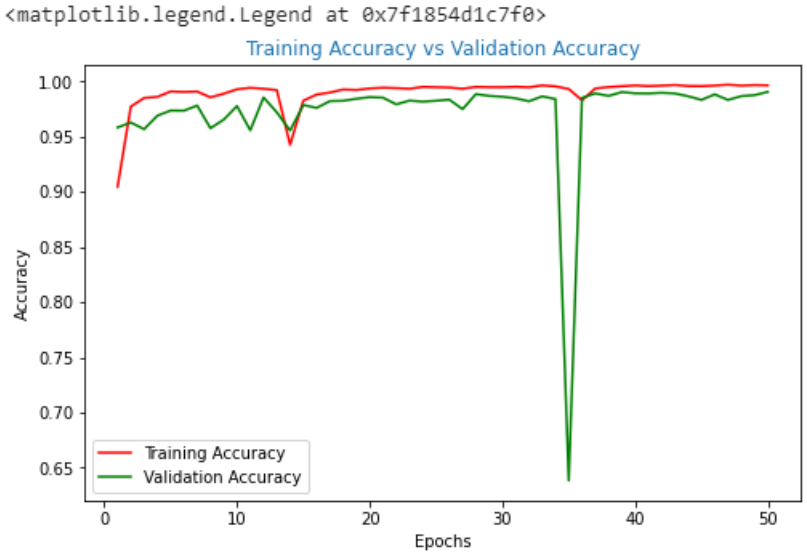
```
True:          [[3]
 [8]
 [8]
 [0]
 [6]]
classified as: [3 8 8 0 6]
True:          [[1]
 [6]
 [3]
 [1]
 [0]]
classified as: [1 6 3 1 0]
```

Appendix Q - Graphical representation of Training Loss and Validation Loss over epoch size for ResNet-50 model trained on MNIST dataset

```
<matplotlib.legend.Legend at 0x7f1855538470>
```

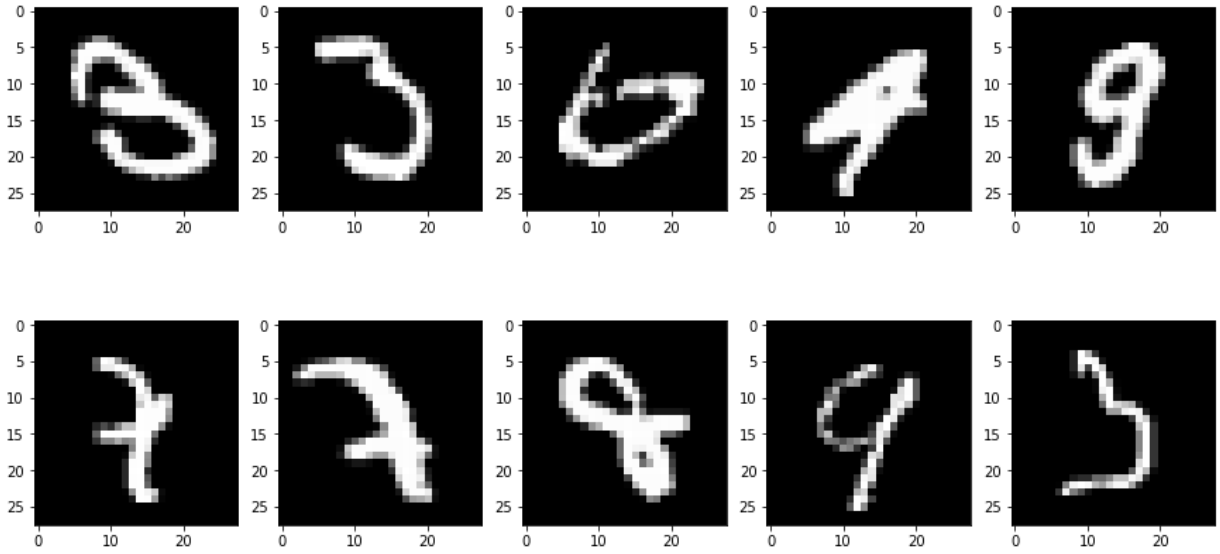


Appendix R - Graphical representation of Training Accuracy and Validation Accuracy over epoch size for ResNet-50 model trained on MNIST dataset



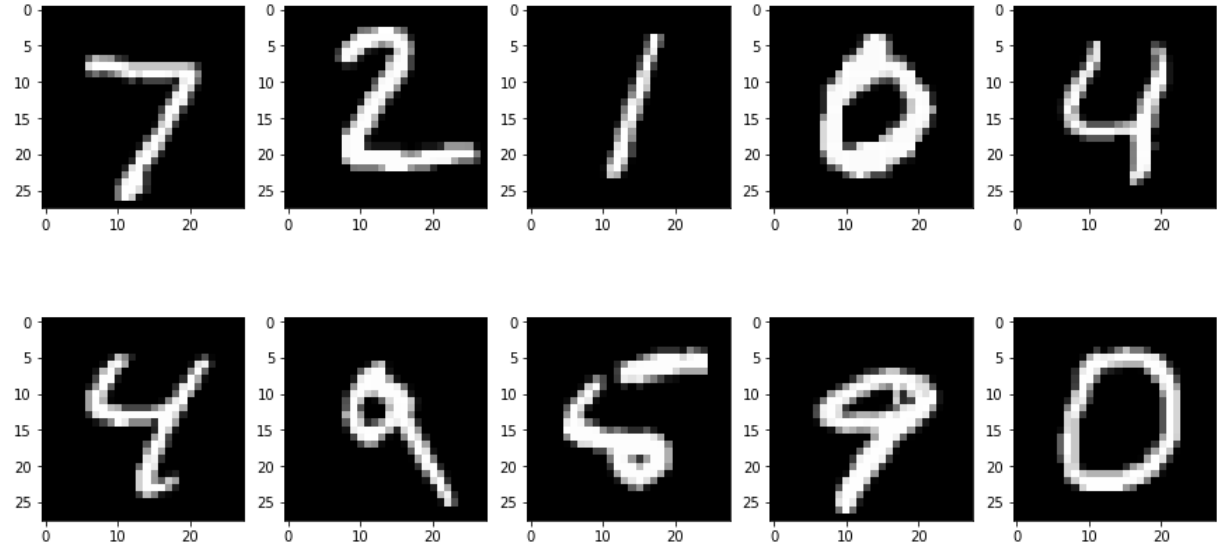
Appendix S - Test Results of ResNet-50 model trained on MNIST dataset

Visualization of few incorrectly classified test samples are shown below,



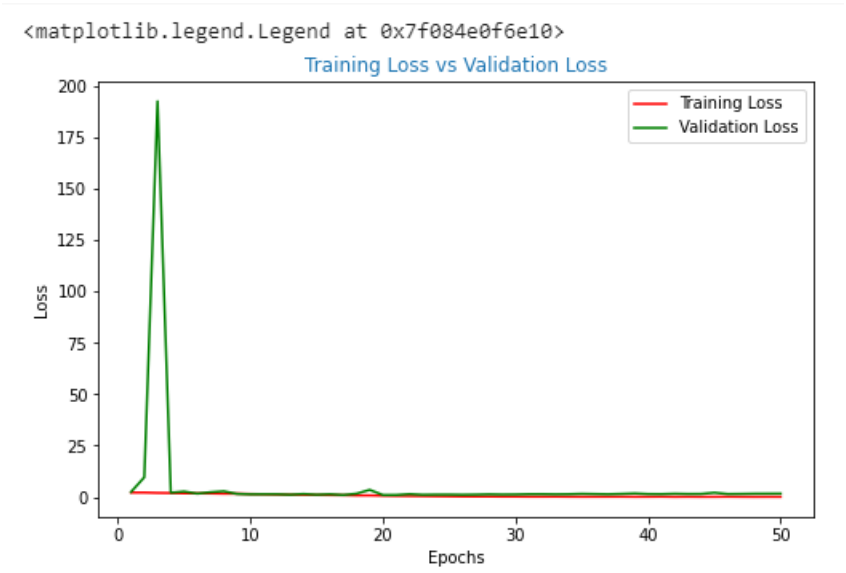
```
True:          [3 3 6 4 9]
classified as: [2 2 0 9 8]
True:          [7 7 8 9 3]
classified as: [3 2 9 7 5]
```

Visualization of few correctly classified test samples are shown below,



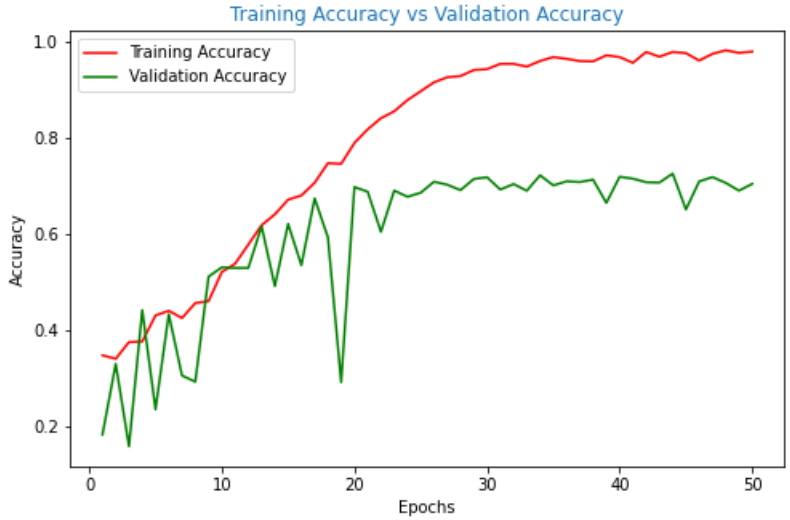
```
True:          [7 2 1 0 4]
classified as: [7 2 1 0 4]
True:          [4 9 5 9 0]
classified as: [4 9 5 9 0]
```

Appendix T - Graphical representation of Training Loss and Validation Loss over epoch size for ResNet-50 model trained on CIFAR-10 dataset



Appendix U - Graphical representation of Training Accuracy and Validation Accuracy over epoch size for ResNet-50 model trained on CIFAR-10 dataset

```
<matplotlib.legend.Legend at 0x7f084eadca58>
```

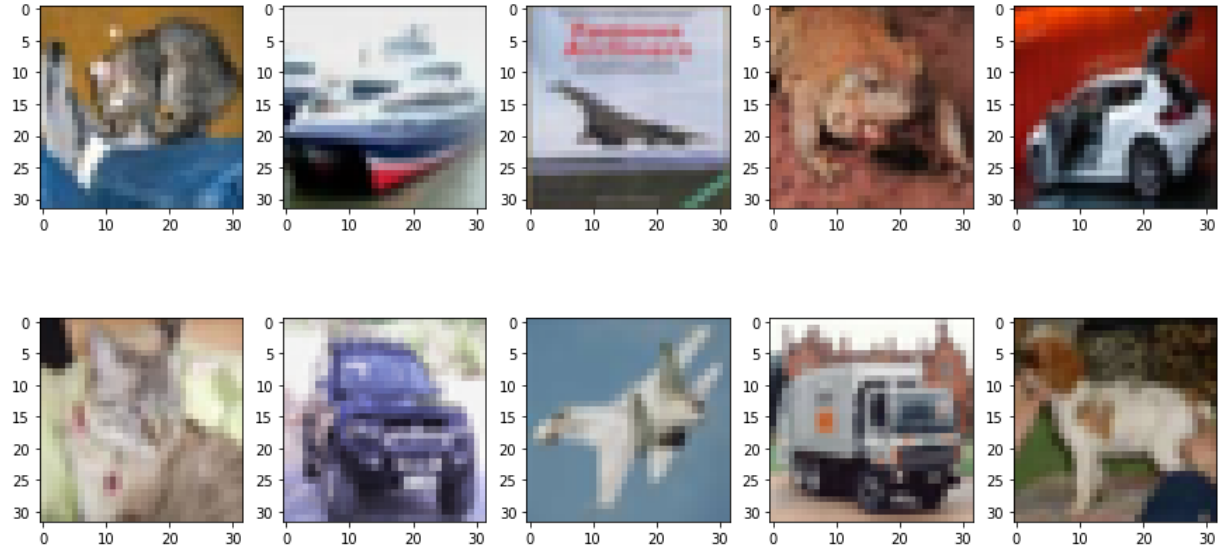


Appendix V - Test Results of ResNet-50 model trained on CIFAR-10 dataset

Visualization of few incorrectly classified test samples are shown below,

```
True:          [[8]
 [6]
 [7]
 [7]
 [4]]
classified as: [0 4 9 2 0]
True:          [[2]
 [6]
 [5]
 [4]
 [2]]
classified as: [4 3 4 7 1]
```

Visualization of few correctly classified test samples are shown below,



```
True:           [[3]
 [8]
 [0]
 [6]
 [1]]
classified as: [3 8 0 6 1]
True:           [[3]
 [1]
 [0]
 [9]
 [5]]
classified as: [3 1 0 9 5]
```