Deeper Networks for Image Classification

Author: Tampu Ravi Kumar

1. Introduction

Image Classification is a process of taking an input and outputting a class or a probability that the input belongs to a particular class. Deep neural network is a neural network with more than two layers. To analyse visual imagery and to perform image classification tasks, a class of deep neural networks called convolutional neural networks (CNNs) is used in deep learning. Other exciting application areas of CNN include Image Segmentation, Object Detection, Video Processing, Natural Language Processing, and Speech Recognition. CNNs have an input layer, an output layer and hidden layers. The input to a CNN is a multi-channelled image (3 channelled if RGB image), unlike neural networks, where the input is a vector. The hidden layers usually consist of convolution layers, ReLU layers, pooling layers and fully connected layers [1]. The main building block of a convolutional neural network is the convolution layer. The convolution layer comprises of a set of independent filters and each filter is independently convolved with the image [2]. ReLU stands for rectified linear unit, and is a type of activation function which is the most commonly used activation function in neural networks, especially in CNNs. ReLU is the max function(x,0) with input x which is a matrix from a convolved image. Another building block of CNN is pooling layer which operates on each feature map independently. Pooling layer progressively reduces the spatial size of the representation to reduce the number of parameters and thus reduces computation in the network [2]. The most commonly used pooling approach is max cooling. The last few layers in the CNN form Fully Connected Layers which are feed forward neural networks. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

2. Critical Analysis / Related Work

Convolutional Neural Networks are preferred over other neural networks for complex image classification tasks because they are able to extract features from an image. The various improvements in learning methodology and architecture of CNNs made it possible for it to handle large, heterogeneous, multiclass problems. Innovations in CNNs include different aspects such as modification of processing units, parameter and hyper-parameter optimization strategies, design patterns and connectivity of layers [3]. To improve the understanding of feature extraction stages, the concept of layer-wise visualization of CNN shifted the trend towards extraction of features at low spatial resolution in deep architecture as performed in VGG [3]. Simonyan & Zisserman [4] investigated the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting and found out that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. VGG architecture is the first to use very small (3×3) filters in each convolutional layer. VGG showed good results both for image classification and localization problems. The main limitation associated with VGG was the use of 138 million parameters, which make it computationally expensive and difficult to deploy it on low resource systems [4].

To achieve high accuracy with a reduced computational cost, GoogLeNet architecture was developed which was the winner of the 2014-ILSVRC competition and is also known as Inception-V1 [5]. A new concept of Inception block was introduced to CNN in GoogLeNet where it incorporated multi-scale convolutional transformations using split, transform and merge idea. The Inception module is basically the parallel combination of 1×1, 3×3, and 5×5 convolutional filters. To reduce the number of features before the expensive parallel blocks it uses 1×1 convolutional block (Network in Network), this regulates the computation. It used sparse connections where all the output feature-maps are not connected to all the input feature-maps, to overcome the problem of redundant information and reduced cost by omitting feature-maps that were not relevant [4]. Instead of a fully connected layer, global average pooling was used at the last layer thus reducing connection's density. Tuning of these parameters caused a significant decrease in the number of parameters from 138 million to 4 million parameters [3]. The main drawback of GoogLeNet was its heterogeneous topology that needs to be customized from module to module. Another limitation of GoogLeNet was a representation bottleneck that drastically reduces the feature space in the next layer and thus sometimes may lead to loss of useful information [3].

Inception and VGG, which showed the best performance in 2014-ILSVRC competition, which further proved the idea that the depth is an essential dimension in regulating learning capacity of the networks. However, the main problem with deep networks is slow training and convergence speed. Srivastava et al. in 2015, proposed a deep CNN, named as Highway Networks exploited depth for learning enriched feature representation and introducing a new cross-layer connectivity mechanism for the successful training of the deep networks [6]. The performance of a plain network decreases after adding hidden units beyond 10 layers whereas Highway Networks, on the other hand, converge significantly faster, even with the depth of 900 layers [3]. In Highway Networks, two gating units are imparted within a layer which enables the unimpeded flow of information across layers. ResNet, an architecture proposed by He et al., exploited the idea of bypass pathways used in Highway Networks that skips one or more layers [7]. The distinct feature of ResNet is reference based residual learning framework. ResNet suggested that residual functions are easy to optimize and can gain accuracy for considerably increased depth [3]. The bypass connections within layers are data-independent and parameter-free in comparison to the gates of Highway Networks. When a gated shortcut is closed in Highway Networks, the layers represent non-residual functions whereas in ResNet, residual information is always passed. Residual links (shortcut connections) speed up the convergence of deep networks, thus giving ResNet the ability to avoid gradient diminishing problems [3].

Similar to Highway Networks and ResNet, DenseNet was proposed to solve the vanishing gradient problem [8]. The main problem in ResNet was that many layers contribute little or no information as it performed additive identity transformation and preserved the information. This problem was addressed by DenseNet which connected each layer to every other layer in a feed-forward fashion. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers [8]. Thus, the network may gain the ability to explicitly differentiate between information that is preserved and information that is added to the network. Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output [8].

3. Method / Model Description

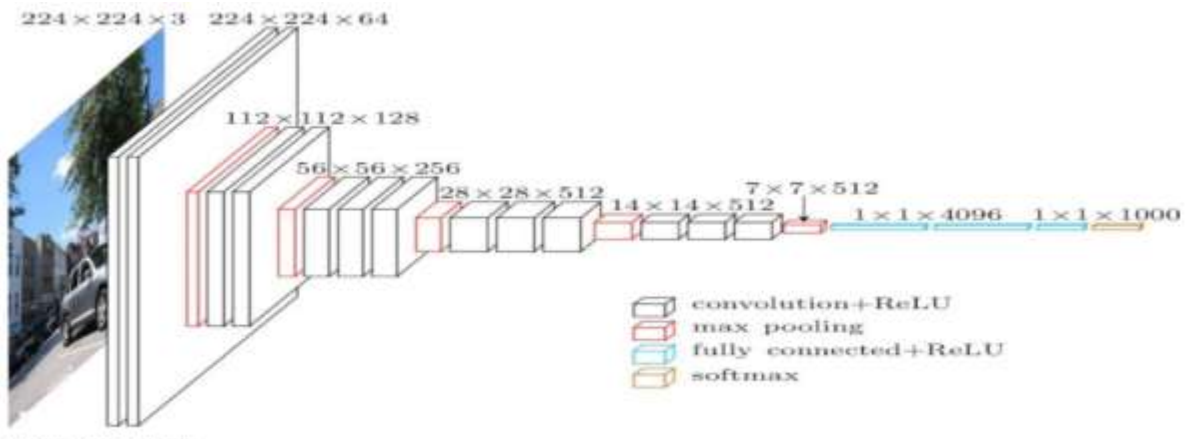
In this paper, I use various deeper networks for evaluating the effectiveness of deeper CNN models for image classification on MNIST and CIFAR-10 datasets.

3.1 Model Architecture

(I) VGG-16

- One of the most preferred architectures in the recent past. Developed by Simonyan & Zisserman in 2014.
- It has 16 convolutional layers – Complexity has been increased compared to initial versions of CNN architectures like LeNET.
- It has 138 million parameters.

The VGG16 architecture is depicted below.



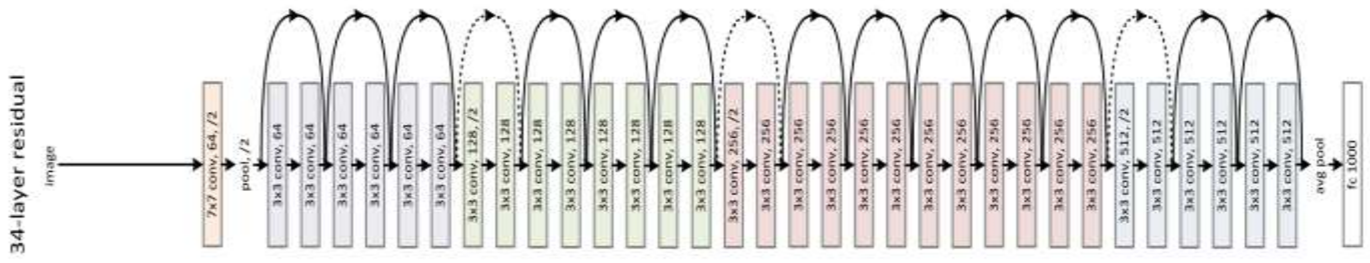
VGG16 Architecture

Input is fixed size 224 x 224 RGB image. The 13 convolutional layers are described below.

No	Convolution	Output Dimension	Pooling	Output Dimension
layer 1&2	convolution layer of 64 channel of 3x3 kernel with padding 1, stride 1	224x224x64	Max pool stride=2, size 2x2	112x112x64
layer3&4	convolution layer of 128 channel of 3x3 kernel	112x112x128	Max pool stride=2, size 2x2	56x56x128
layer5,6,7	convolution layer of 256 channel of 3x3 kernel	56x56x256	Max pool stride=2, size 2x2	28x28x256
layer8,9,10	Convolution layer of 512 channel of 3x3 kernel	28x28x512	Max pool stride=2, size 2x2	14x14x512
layer11,12,13	Convolution layer of 512 channel of 3x3 kernel	14x14x512	Max pool stride=2, size 2x2	7x7x512

Three Fully-Connected (FC) layers follow a stack of convolutional layers: first two layers have 4096 channels each and the last layer has 1000 channels. The 14th layer is Flatten layer, 15th layer is Dense layer and 16th layer contains Output classes. Max-pooling is performed over a (2×2) pixel window, with stride 2. All hidden layers are equipped with the rectification (ReLU) non-linearity. The final layer is the soft-max layer.

(II) ResNet



The ResNet architecture (shown above) can have many layers and the one described in [7] this paper has 34 layers.

- It consists of one convolutional and pooling step followed by 4 layers of similar behaviour
- Each of the layers follow the same pattern. They perform 3x3 convolution with a fixed feature map dimension (F) [64, 128, 256, 512] respectively, bypassing the input every 2 convolutions
- The width (W) and height (H) dimensions remain constant during the entire layer
- The dotted line represents that there has been a change in the dimension of the input volume. This reduction in size is achieved by an increase on the stride, from 1 to 2, at the first convolution of each layer.
- Global average pooling layer and a 1000-way fully-connected layer with Softmax are present in the end

4. Experiments

4.1 Datasets

Two datasets were used to test the performance of VGG16 and ResNet models.

a) MNIST database which is available in this page [9] is collection of handwritten digits and is a subset of NIST. It contains 60,000 training images and 10,000 testing images. The digits have been size-normalised and centred in a fixed size image. Figure 1 (in Appendix A) shows a sample of MNIST database.

b) CIFAR-10 dataset is a collection of images which contains 60,000 32x32 color images in 10 different classes. It is a labeled subset of the 80 million tiny images dataset. Figure 2 (in Appendix A) shows a sample of CIFAR-10 dataset. It is available in this page [9] for download in keras.

4.2 Testing Results

1) VGG16 model results

The number of epochs used were 50 with a batch size of 128.

a) The input image has size `(60000, 28, 28)` and since MNIST has gray-scale images, it has 1 channel. This was fed to the VGG16 model. The model summary is shown in Figure 3 (in Appendix B).

The test accuracy obtained, `Test accuracy: 98.92 %`

b) The input image has size `(50000, 32, 32, 3)` and since CIFAR-10 has RGB images, it has 3 channels. This was fed to the VGG16 model.

The test accuracy obtained, `Test accuracy: 73.71 %`

The graphs of validation loss, training loss, validation accuracy and training accuracy changes for each epoch are shown in Figure 4 and 5 (in Appendix B) for MNIST and Figure 6 and 7 (in Appendix B) for CIFAR-10.

2) ResNet Model results

The number of epochs used were 30 with batch size 32.

a) For MNIST dataset, test accuracy obtained: `Test Accuracy =0.9937999844551086`

b) For CIFAR-10 dataset, test accuracy obtained: `Test Accuracy = 0.753000020980835`

(Model summary is too long to be pasted in the appendix hence can be found in the Jupyter notebook attached.)

4.3 Further Evaluation

Further evaluation can be carried out by fine tuning parameters or adding deeper networks to improve the accuracy of VGG16 and ResNet models on CIFAR-10 dataset.

5. Conclusion

- Both VGG16 and ResNet models gave more than 98% test accuracy for MNIST dataset whereas they performed poorly on CIFAR-10 dataset with test accuracy of less than 80%
- VGG16 takes longer time to train compared to ResNet for same number of epochs
- The test accuracy increases from 0 to 15 epochs and then saturates in both the models
- The test loss shows a reverse trend, it decreases from 0 to 15 epochs and then saturates

REFERENCES

[1] Anne Bonner. “The Complete Beginner’s Guide to Deep Learning: Convolutional Neural Networks and Image Classification”. Towards Data Science, February 2019

[2] Harsh Pokharna. “The best explanation of Convolutional Neural Networks on the Internet!”. Medium, July 2016

[3] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. “A Survey of the Recent Architectures of Deep Convolutional Neural Networks”. Accepted in Artificial Intelligence Review (Springer Nature), DOI: 10.1007/s10462-020-09825-6, March 2020

[4] Karen Simonyan & Andrew Zisserman. “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION”. Published as a conference paper at ICLR 2015, April 2015

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich.“Going deeper with convolutions”. Published in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015

[6] Rupesh Kumar Srivastava, Klaus Greff, Jürgen Schmidhuber. “Highway Networks”. Presented at ICML 2015 Deep Learning workshop, November 2015

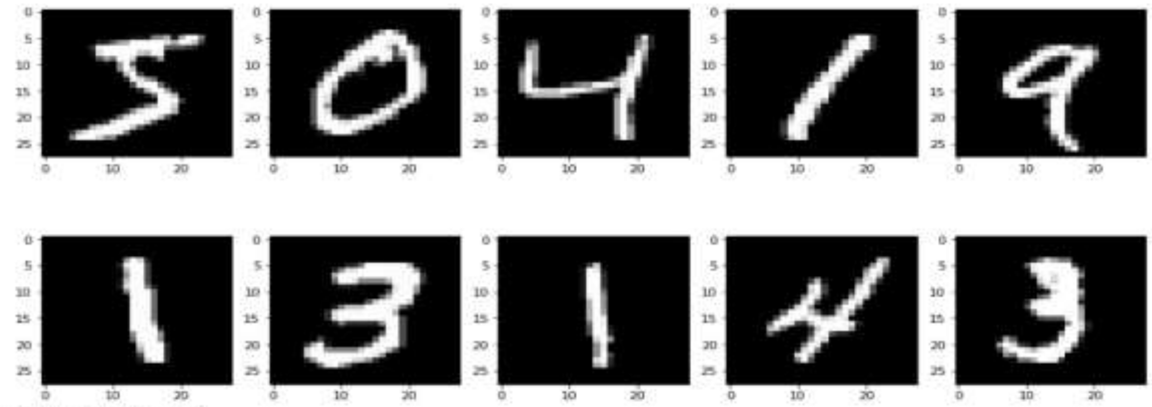
[7] Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun. “Deep Residual Learning for Image Recognition”. Published in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016

[8] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243.

[9] https://keras.io/api/datasets/

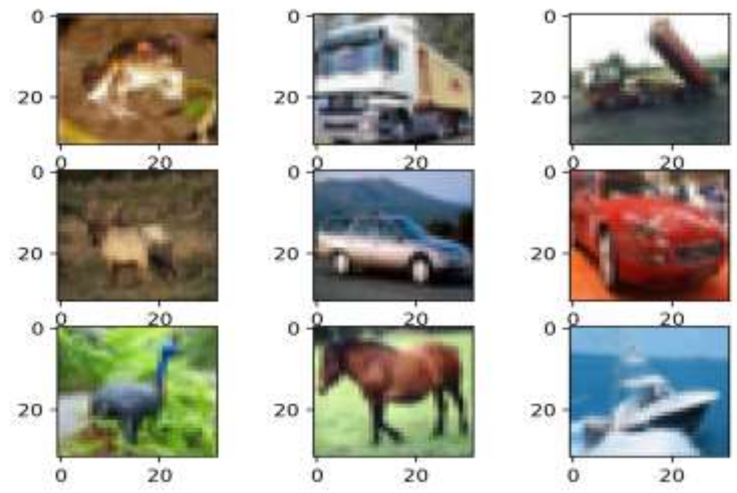
APPENDIX A - Databases

1) Figure 1: MNIST database sample



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

2) Figure 2: CIFAR10 database sample

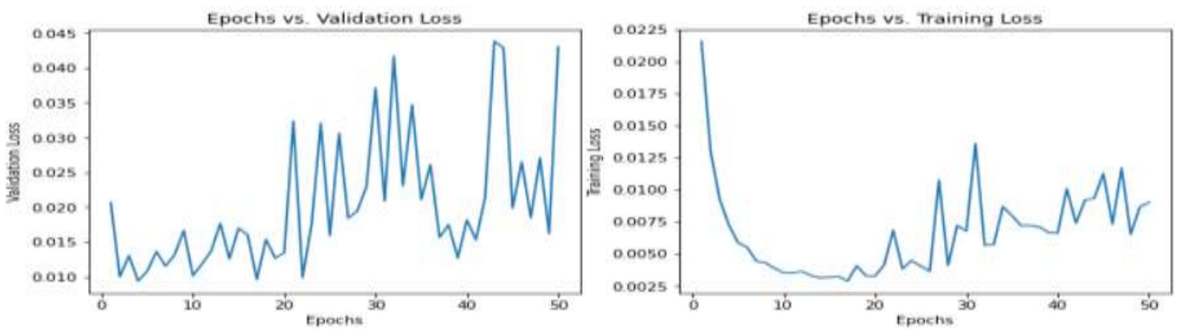


APPENDIX B – Runtime Screenshots

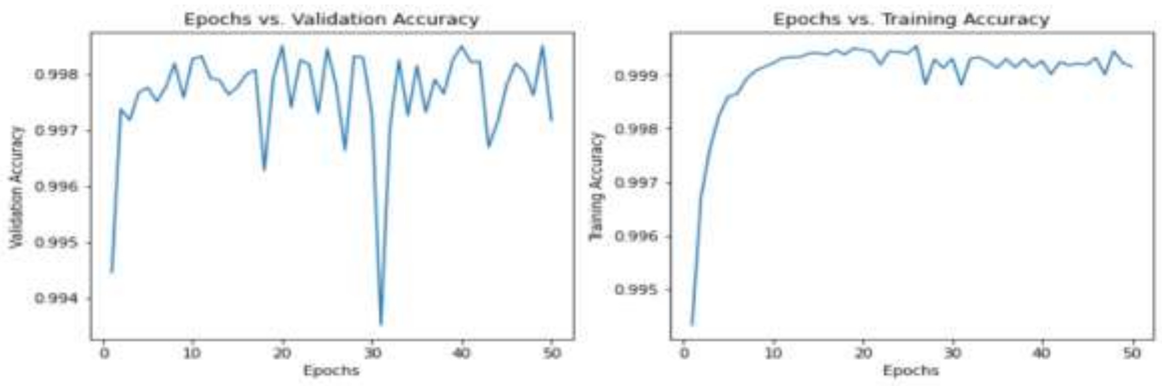
1) Figure 3: VGG16 model summary

```
_________________________________________________________________
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
CPU times: user 6min 28s, sys: 2min 14s, total: 8min 42s
Wall time: 11min 4s
```

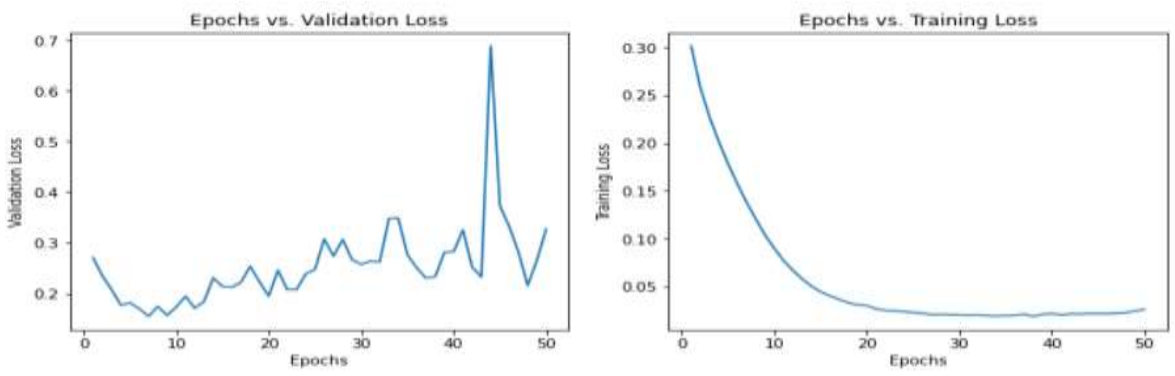
2) Figure 4: Epochs vs Validation Loss graph and Epochs vs Training Loss graph for MNSIT



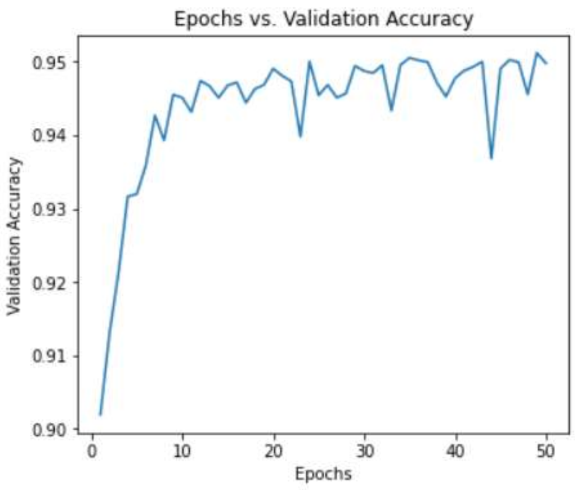
3) Figure 5: Epochs vs Validation Accuracy graph and Epochs vs Training Accuracy graph for MNIST

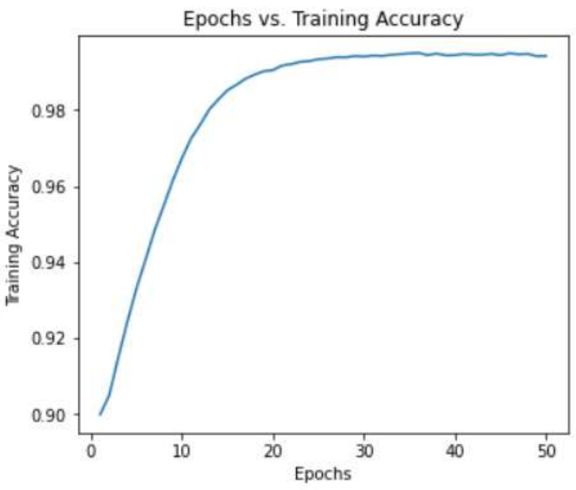


4) Figure 6: Epochs vs Validation Loss graph and Epochs vs Training Loss graph for CIFAR-10



5) Figure 7: Epochs vs Validation Accuracy graph and Epochs vs Training Accuracy graph for CIFAR-10


Epochs vs. Validation Accuracy
Validation Accuracy
Epochs
0.90
0.91
0.92
0.93
0.94
0.95
0
10
20
30
40
50


Epochs vs. Training Accuracy
Training Accuracy
Epochs
0.90
0.92
0.94
0.96
0.98
0
10
20
30
40
50