Answer 2:

We are using MNIST digit classification for different model architectures.

1. Steps for Model 1 architecture :
    I. Input size is 28 X 28 and 1 is the depth of the given image = 1 * 28 * 28
    II. Flatten(1*28*28)
        a. **Flatten operation that converts the feature volume to 1D feature vectors which is passed to FC layer.**
    III. Fully Connected layers with input neurons = 784 and hidden neurons = 100
    IV. And then apply Sigmoid Activation function for linear layers
    V. Lastly, FC is used as output for 10 class predictions with input neurons = 100 and output neurons = 10
    VI. Here, for Training parameters used
        - Sigmoid activation function
        - SGD is used as an optimizer
        - learning rate of 0.1
        - 60 epoch
        - mini-batch size of 10

    Prediction Accuracy got **97.92**

2. Steps for Model 2 architecture :
    I. Input size is 28 X 28 and 1 is the depth of the given image = 1 * 28 * 28
        = 1 * 28 * 28
    II. 2D convolutional layer, taking in 1 input channel (image), outputting 40 convolutional features, with a square kernel size of 5 and stride 1
        = 40*24*24
    III. Apply RELU for convolution
    IV. Max pooling of 2*2
        = 40*12*12

V. 2D convolutional layer, taking in 40 input channel, outputting 40 convolutional features, with a square kernel size of 5 and stride 1
= 40*8*8

VI. Apply RELU for convolution

VII. Max pooling of 2*2
= 40*4*4

VIII. Flatten (40*4*4)

IX. Fully Connected layers with input neurons = 640 (40*4*4) and hidden neurons = 100

X. And then apply Sigmoid Activation function for linear layers

XI. Lastly, FC is used as output for 10 class predictions with input neurons = 100 and output neurons = 10

XII. Here, for Training parameters used

- Relu activation function
- SGD is used as an optimizer
- learning rate of 0.1
- 60 epoch
- mini-batch size of 10

Prediction Accuracy got **99.45**

Observations: Accuracy got increased after applying Convolution layer

MaxPooling is used to downsize the feature map, here depending on the problem, we need to accordingly use maxpooling, because if we reduce more and more spatial resolution, it may happen, we may lose some of the small feature or object in that image.

3. Steps for Model 3 architecture :

I. Input size is 28 X 28 and 1 is the depth of the given image = 1 * 28 * 28
= 1 * 28 * 28

II. 2D convolutional layer, taking in 1 input channel (image), outputting 40 convolutional features, with a square kernel size of 5 and stride 1

= 40*24*24

III. Apply RELU

IV. Max pooling of 2*2

= 40*12*12

V. 2D convolutional layer, taking in 40 input channel, outputting 40 convolutional features, with a square kernel size of 5 and stride 1

= 40*8*8

VI. Apply RELU

VII. Max pooling of 2*2

= 40*4*4

VIII. Flatten (40*4*4)

IX. Fully Connected layers with input neurons = 640 (40*4*4) and hidden neurons = 100

X. And then apply RELU Activation function for linear layers

XI. Lastly, FC is used as output for 10 class predictions with input neurons = 100 and output neurons = 10

XII. Here, for Training parameters used

- Relu activation function
- SGD is used as an optimizer
- learning rate of 0.03
- 60 epoch
- mini-batch size of 10

Accuracy got **99.36**

Observations: Accuracy got decreased after applying RELU instead of Sigmoid to linear layer. As RELU will make everything zero, if it is less than zero, otherwise it just preserves the value.

4. Steps for Model 4 architecture :

I. Input size is 28 X 28 and 1 is the depth of the given image = 1 * 28 * 28

= 1 * 28 * 28

II. 2D convolutional layer, taking in 1 input channel (image), outputting 40 convolutional features, with a square kernel size of 5 and stride 1
    = 40*24*24
III. Apply RELU
IV. Max pooling of 2*2
    = 40*12*12
V. 2D convolutional layer, taking in 40 input channel, outputting 40 convolutional features, with a square kernel size of 5 and stride 1
    = 40*8*8
VI. Apply RELU
VII. Max pooling of 2*2
    = 40*4*4
VIII. Flatten (40*4*4)
IX. Fully Connected layers with input neurons = 640 (40*4*4) and hidden neurons = 100
X. And then apply RELU Activation function for linear layers
XI. Fully Connected layers with input neurons = 100 and hidden neurons = 100
XII. And then apply RELU Activation function for linear layers
XIII. Lastly, FC is used as output for 10 class predictions with input neurons = 100 and output neurons = 10
XIV. Here, for Training parameters used

- Relu activation function
- SGD is used as an optimizer
- learning rate of 0.03
- 60 epoch
- mini-batch size of 10

Accuracy got **99.43**

Observations: Adding one FC hidden layers improved some accuracy.

5. Steps for Model 5 architecture :
    I. Input size is 28 X 28 and 1 is the depth of the given image = 1 * 28 * 28
        = 1 * 28 * 28
    II. 2D convolutional layer, taking in 1 input channel (image), outputting 40 convolutional features, with a square kernel size of 5, stride 1 and padding 2
        = 40*28*28
    III. Apply RELU
    IV. Max pooling of 2*2
        = 40*14*14
    V. 2D convolutional layer, taking in 40 input channel, outputting 40 convolutional features, with a square kernel size of 5, stride 1 and padding 2
        = 40*14*14
    VI. Apply RELU
    VII. Max pooling of 2*2
        = 40*7*7
    VIII. Apply dropout with a rate of 0.5
    IX. Flatten (40*7*7)
    X. Fully Connected layers with input neurons = 1960 (40*7*7) and hidden neurons = 1000
    XI. And then apply RELU Activation function for linear layers
    XII. Apply dropout with a rate of 0.5
    XIII. Fully Connected layers with input neurons = 1000 and hidden neurons = 1000
    XIV. And then apply RELU Activation function for linear layers
    XV. Lastly, FC is used as output for 10 class predictions with input neurons = 1000 and output neurons = 10
    XVI. Here, for Training parameters used
        - Relu activation function
        - SGD is used as an optimizer
        - learning rate of 0.03

- 40 epoch
- mini-batch size of 10
- Dropout 0.5

Accuracy got **99.49**

Observations: All the hidden neurons are changed to 1000 and except FC output layer all FC layer are followed with dropout

Dropout acts as strong regularization and improves the generalization ability of the network.

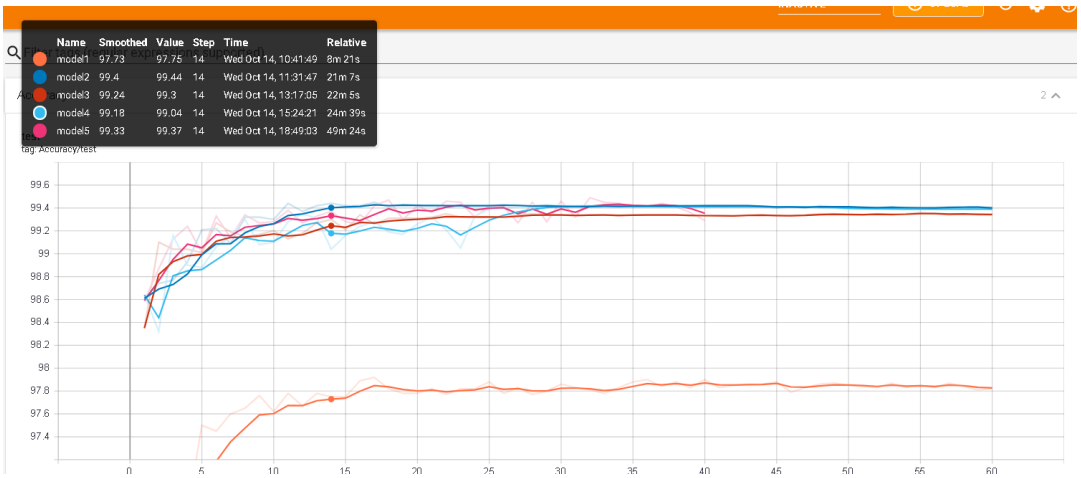
Accuracy got improved.

We use Dropout to make the network not overfit the training data.

TensorBoard Analysis:

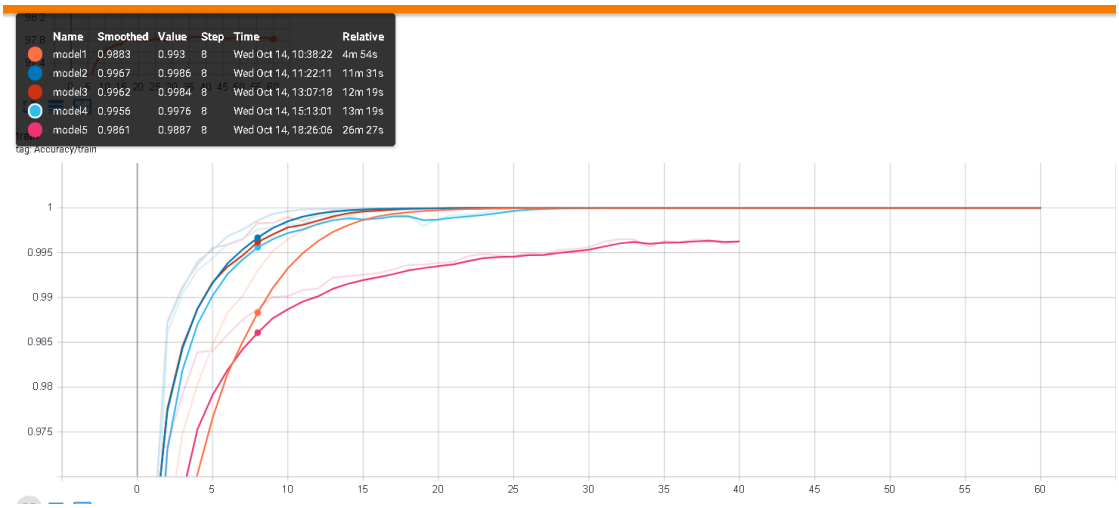
Here, we can see all the different variations of 5 model architecture that we trained. Each model is denoted by separate color.

Test Accuracy Graph:



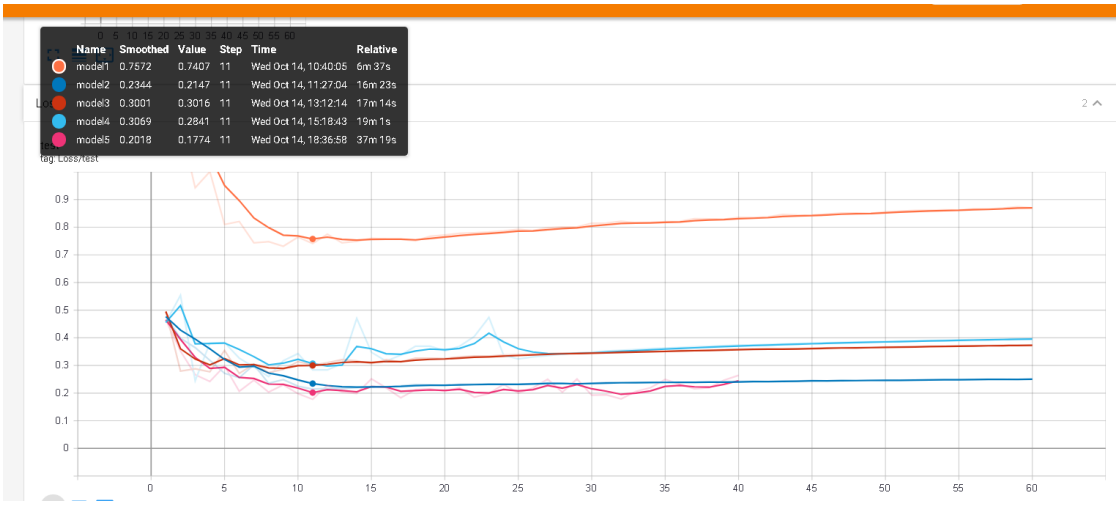
Observations: Initially, the accuracy increases but after a while it becomes constant with small variations.

Train Accuracy Graph:



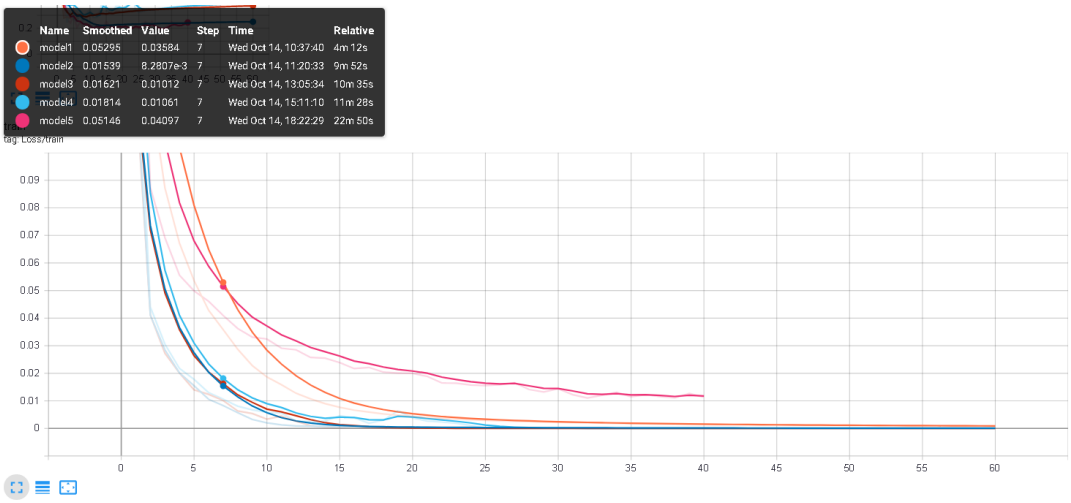
Observations: Initially, the accuracy increases but after a while it becomes constant with NO variations. And, the network starts to overfit.

Test Loss Graph:



Observations: Loss initially drops, and after a while it becomes almost constant and slowly starts to rise as the model slowly starts overfitting.

Train Loss Graph:



Observations: Loss initially drops, and after a while it becomes almost constant. And as the model slowly starts overfitting, it keeps gradually reducing the train loss further. But with very small variations.

GitHub link:

https://github.com/SunilDevlops/Programs/tree/master/ComputerVision/ProgramAssignment1/Question2