Answer1:

Here we are using CIFAR-10 dataset to develop a CNN classification network to recognize RGB color images.

For the 1st task: mode = "small" -- 3 CONV layers

Network Architecture:

1. Input size is 3 x 128 x 128 (Transformations is resized to 128 x 128)
2. First 2D convolutional layer, taking in 3 input channel (image), outputting 32 convolutional features, with a square kernel size of 3, stride 1 and padding 1
   = 128 x 128 x 32
3. Apply RELU for convolution
4. Max pooling of 2*2
   = 64 x 64 x 32
5. Second 2D convolutional layer, taking in 32 input channel (image), outputting 64 convolutional features, with a square kernel size of 3, stride 1 and padding 1
   = 64 x 64 x 64
6. Apply RELU for convolution
7. Max pooling of 2*2
   = 32 x 32 x 64
8. Third 2D convolutional layer, taking in 64 input channel (image), outputting 128 convolutional features, with a square kernel size of 3, stride 1 and padding 1
   = 32 x 32 x 128
9. Apply RELU for convolution
10. Max pooling of 2*2
    = 16 x 16 x 128
11. Using AdaptiveAveragePool to specify the recommended output size to 4 x 4
    = 4 x 4 x 128
12. Flatten (4 x 4 x 128)
13. Fully Connected 1st layers with input neurons = 2048 (4 x 4 x 128)and hidden neurons = 2048
14. And then apply RELU Activation function for linear layers
15. Apply dropout with a rate of 0.5
16. Fully Connected 2nd layers with input neurons = 2048 and hidden neurons = 512
17. And then apply RELU Activation function for linear layers
18. Apply dropout with a rate of 0.5

19. Lastly, FC is used as output for 10 class predictions with input neurons = 512 and output neurons = 10
20. Training parameters used
    a. RELU activation function
    b. SGD is used as an optimizer
    c. Learning rate of 0.001
    d. 30 epochs
    e. Mini-batch size of 10
    f. Dropout 0.5
    g. Cross-entropy loss

Results:

Accuracy got of **71.07**

For the 2nd task: mode = “large” -- 5 CONV layers

Network Architecture:

1. Input size is 3 x 128 x 128 (Transformations is resized to 128 x 128)
2. First 2D convolutional layer, taking in 3 input channel (image), outputting 32 convolutional features, with a square kernel size of 3, stride 1 and padding 1
   = 128 x 128 x 32
3. Apply RELU for convolution
4. Max pooling of 2*2
   = 64 x 64 x 32
5. Second 2D convolutional layer, taking in 32 input channel (image), outputting 64 convolutional features, with a square kernel size of 3, stride 1 and padding 1
   = 64 x 64 x 64
6. Apply RELU for convolution
7. Max pooling of 2*2
   = 32 x 32 x 64
8. Third 2D convolutional layer, taking in 64 input channel (image), outputting 128 convolutional features, with a square kernel size of 3, stride 1 and padding 1
   = 32 x 32 x 128
9. Apply RELU for convolution
10. Max pooling of 2*2
    = 16 x 16 x 128
11. Fourth 2D convolutional layer, taking in 128 input channel (image), outputting 256 convolutional features, with a square kernel size of 3, stride 1 and padding 1
    = 16 x 16 x 256

12. Apply RELU for convolution
13. Max pooling of 2*2

    = 8 x 8 x 256

14. Fifth 2D convolutional layer, taking in 256 input channel (image), outputting 256 convolutional features, with a square kernel size of 3, stride 1 and padding 1

    = 8 x 8 x 256

15. Apply RELU for convolution
16. Max pooling of 2*2

    = 4 x 4 x 256

17. Using AdaptiveAveragePool to specify the recommended output size to 4 x 4

    = 4 x 4 x 256

18. Flatten (4 x 4 x 256)
19. Fully Connected 1st layers with input neurons = 4096 (4 x 4 x 256)and hidden neurons = 2048
20. And then apply RELU Activation function for linear layers
21. Apply dropout with a rate of 0.5
22. Fully Connected 2nd layers with input neurons = 2048 and hidden neurons = 512
23. And then apply RELU Activation function for linear layers
24. Apply dropout with a rate of 0.5
25. Lastly, FC is used as output for 10 class predictions with input neurons = 512 and output neurons = 10
26. Training parameters used
    a. RELU activation function
    b. SGD is used as an optimizer
    c. Learning rate of 0.001
    d. 30 epochs
    e. Mini-batch size of 10
    f. Dropout 0.5
    g. Cross-entropy loss

Results:

Accuracy got of **79.15**

Observations:

Accuracy got increased by 8.08 in 2nd task when we increased the number of convolution layers by 2.

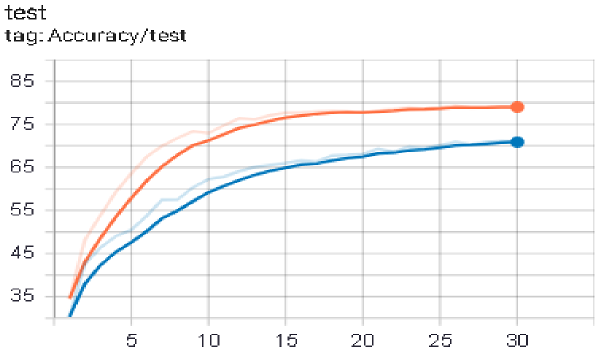
CIFAR10 is a very large dataset.

So, when we increase more layers, it helps in extracting more features from our input CIFAR10 dataset which results in increase of accuracy.

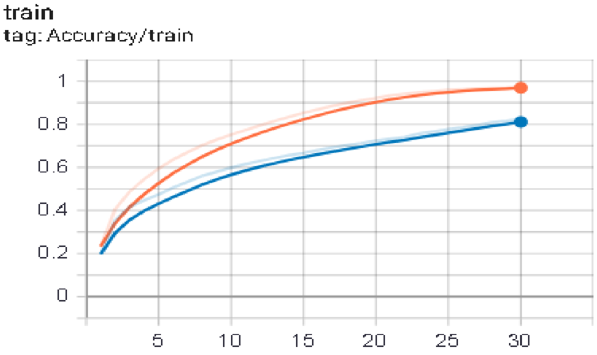
TensorBoard Analysis:

Here, we can see all the different variations of both the model (small and large) architecture that we trained. Each model is denoted by separate color. **Orange color is for large** and **Blue color is for small**

Test Accuracy Graph:

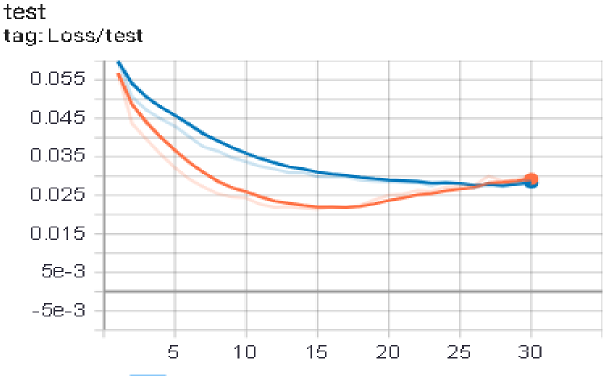


Train Accuracy Graph:

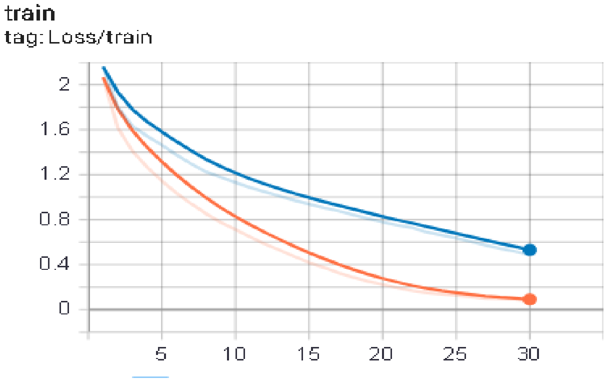


Observations: Initially, the accuracy increases but after a while it becomes constant with small variations.

Test Loss Graph:



Train 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.

GitHub link:

https://github.com/SunilDevlops/Programs/tree/master/ComputerVision/ProgramAssignment3/Question1