COL-341 Report (Part – a)

Here are my public test accuracies for 8 epochs:

Epoch 1: 0.9102173913043479

Epoch 2: 0.9478260869565217

Epoch 3: 0.9606521739130435

Epoch 4: 0.9671739130434782

Epoch 5: 0.9734782608695652

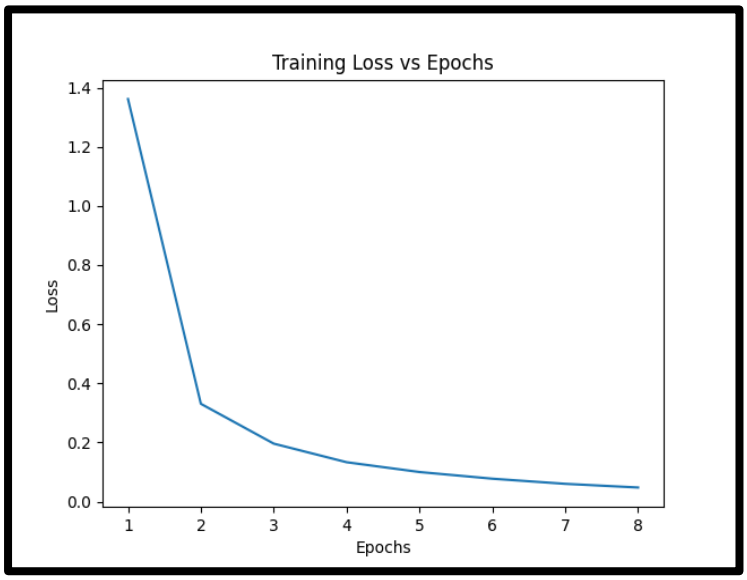
Epoch 6: 0.9752173913043478

Epoch 7: 0.9760869565217392

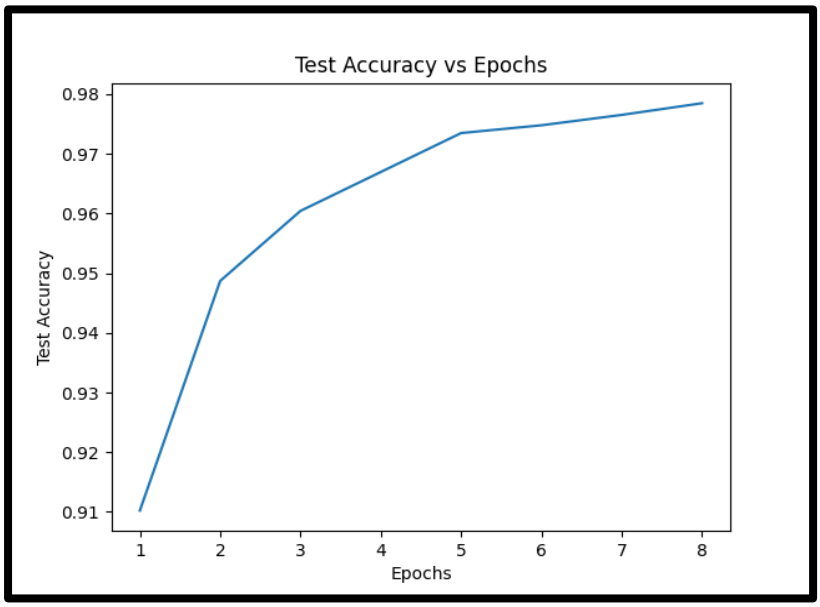
Epoch 8: 0.9789130434782609

So, accuracy after 8 epochs is 0.979.

Here, is the plot of Training Loss v/s epoch:



Here, is the plot of Test accuracy v/s epoch:



The best accuracy, I got in assignment 2.1(using Neural Network) was 0.9482 and it took 5 minutes to achieve this efficiency. The efficiency, I got using CNN for same data set is 0.979 and it took 2.75 minutes to achieve this efficiency. So, CNN is better than neural network as we are able to achieve better efficiency in CNN than neural network and also in less time.

Name: Raunak Jain

Entry Number: 2019MT10719

COL-341 Report (Part – b)

Here are my public test accuracies for 5 epochs:

Epoch 1: 0.5975

Epoch 2: 0.66625

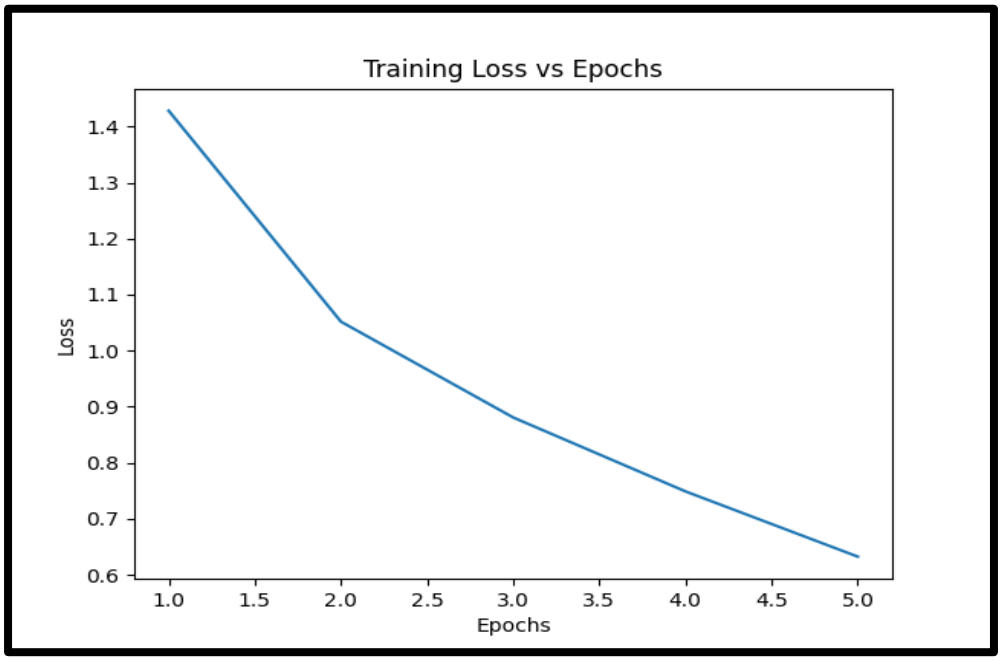
Epoch 3: 0.68325

Epoch 4: 0.70625

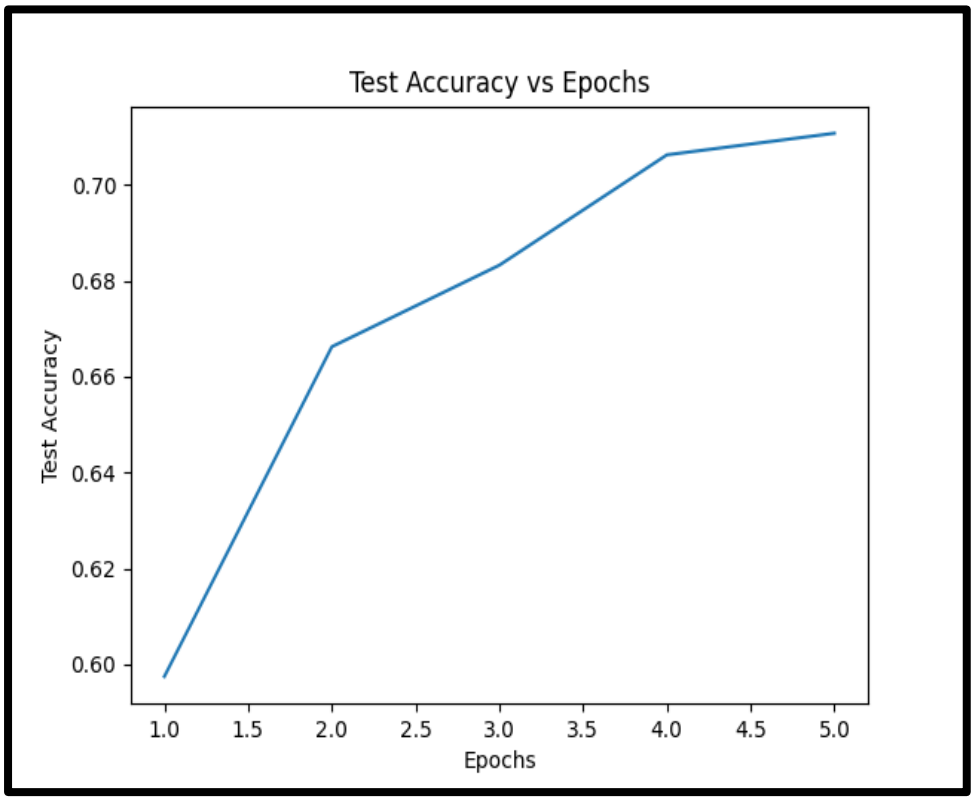
Epoch 5: 0.71075

So, accuracy after 5 epochs is 0.71.

Here, is the plot of Training Loss v/s epoch:



Here, is the plot of Test accuracy v/s epoch:



Name: Raunak Jain

Entry Number: 2019MT10719

COL-341 Report (Part c)

I have not used any feature engineering techniques as it was making my model bad. Here, is my final architecture:

conv1 (2D convolution layer): in_channels = 3, out_channels = 32, kernel_size = (3,3) and padding = same.

elu: elu Non-Linear

bn1: 2D Batch Normalisation Layer

conv12 (2D convolution layer): in_channels = 32, out_channels = 32, kernel_size = (3,3) and padding = same.

elu: elu Non-Linear

bn12: 2D Batch Normalisation Layer

conv13 (2D convolution layer): in_channels = 32, out_channels = 32, kernel_size = (3,3) and padding = same.

elu: elu Non-Linear

bn13: 2D Batch Normalisation Layer

pool1 (Max Pool Layer): kernel_size = (2,2), stride = 2

drp1 (Dropout Layer): p=0.2

conv2 (2D convolution layer): in_channels = 32, out_channels = 64, kernel_size = (3,3) and padding = same.

elu: elu Non-Linaer

bn2: 2D Batch Normalisation Layer

conv22 (2D convolution layer): in_channels = 64, out_channels = 64, kernel_size = (3,3) and padding = same.

elu: elu Non-Linear

bn22: 2D Batch Normalisation Layer

conv23 (2D convolution layer): in_channels = 64, out_channels = 64, kernel_size = (3,3) and padding = same.

elu: elu Non-Linear

bn23: 2D Batch Normalisation Layer

pool2 (Max Pool Layer): kernel_size = (2,2), stride = 2

drp2 (Dropout Layer): p=0.3

conv3 (2D convolution layer): in_channels = 64, out_channels = 128, kernel_size = (3,3) and padding = same.

elu: elu Non-Linaer

bn3: 2D Batch Normalisation Layer

conv32 (2D convolution layer): in_channels = 128, out_channels = 128, kernel_size = (3,3) and padding = same.

elu: elu Non-Linear

bn32: 2D Batch Normalisation Layer

conv33 (2D convolution layer): in_channels = 128, out_channels = 128, kernel_size = (3,3) and padding = same.

elu: elu Non-Linear

bn33: 2D Batch Normalisation Layer

pool3 (Max Pool Layer): kernel_size = (2,2), stride = 2

drp3 (Dropout Layer): p=0.4

fc1 (Fully Connected Layer): output = 512

elu: elu Non-Linear

bnfc1 (1D Batch Normalisation Layer)

drpf4 (Dropout Layer with p = 0.2)

fc2 (Fully Connected Layer): output = 256

elu: elu Non-Linear

bnfc2 (1D Batch Normalisation Layer)

drpf5 (Dropout Layer with p = 0.3)

fc3 (Fully Connected Layer): output = 10

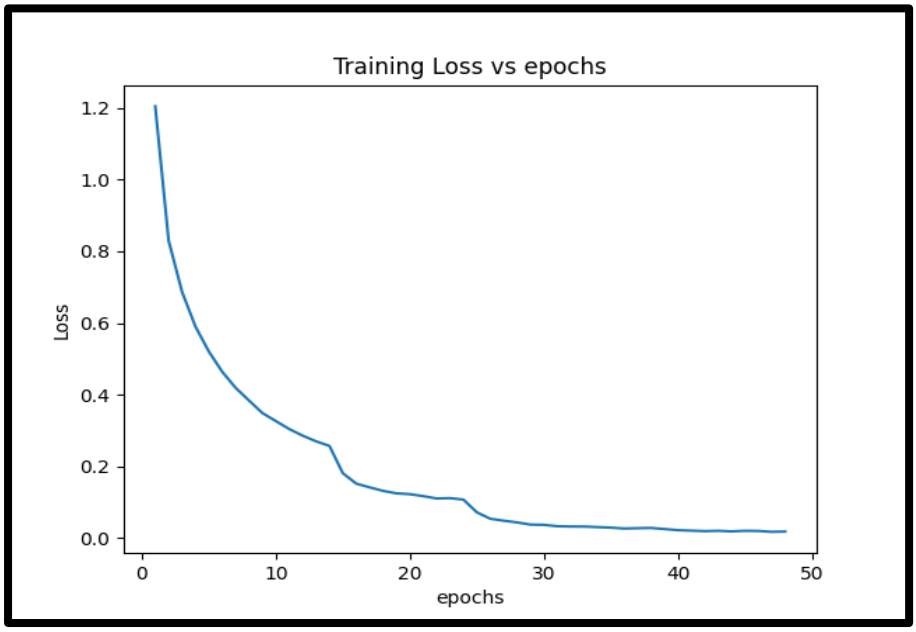
I tried various versions of architecture given in part b but was only able to achieve accuracy of about 0.765. I also tried various architecture given in paper like desnet and resnet by keeping the parameters under 3M. I was able to achieve accuracy of 0.84 in desnet architecture and 0.79 in resnet. They were not giving better efficiency due to overfitting. I saw that loss was decreasing and train accuracy was increasing but test accuracy remains constant.

Then, I tried this architecture: (https://appliedmachinelearning.blog/2018/03/24/achieving-90-accuracy-in-object-recognition-task-on-cifar-10-dataset-with-keras-convolutional-neural-networks/) and was able to achieve accuracy of 0.82. Then, I modified the architecture by increasing the fully connected layer with dropout and batch normalisation and also, adding an extra convolution layer. I used Adam as optimiser as it was giving good accuracy. I saw that the accuracy of architecture was becoming constant after 20 epochs and it was around 0.85. So, I reduce the learning rate of Adam after 15,25 and 40 epochs and due to this my accuracy increases to 0.895. I used 48 epochs as after 48 epochs train loss was increasing and so, it will affect the accuracy. I used batch size = 64 as it was giving better result. I also tried with different learning rates

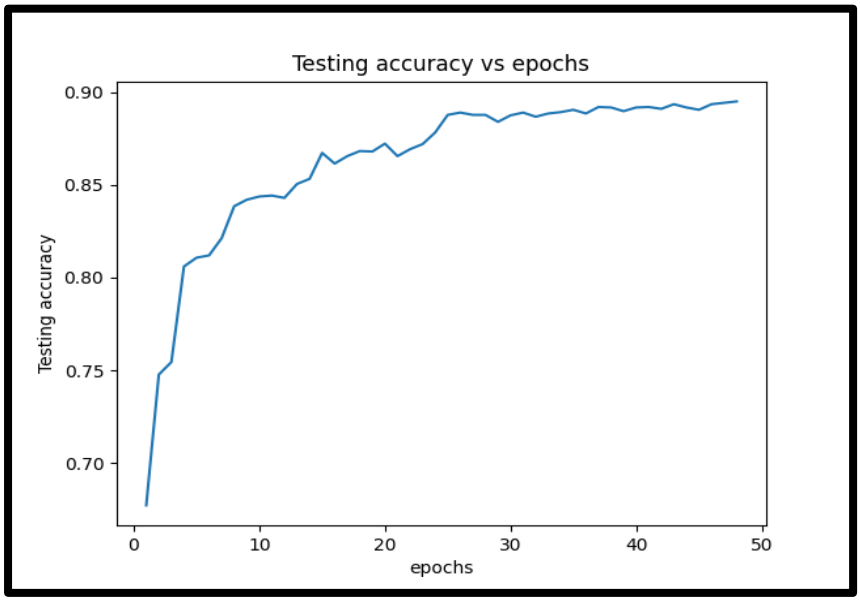
and taking learning rate = 0.001 initially was giving better result.

I have used dropout and batch normalisation layer to avoid overfitting.

Here, is my loss v/s epochs for this architecture:



Here, is my public test accuracy v/s epochs:



I was able to achieve loss = 0.018 and accuracy = 0.895 after 48 epochs and after that loss start increasing.

Number of parameters for this architecture is: 1666634 i.e. 1.6M.

Name: Raunak Jain

Entry Number: 2019MT10719