CS7015: Deep learning for Computer Vision

Name: ***JOSE MOTI***

Roll No: ***CE17B118***

Date: ***10/08/19***

Programming Assignment 3

Experiment 1 - Transfer Learning

In this experiment, I would be doing transfer learning using the Resnet 18 model architecture. Experiment is done on the CIFAR-10 dataset and the training, test accuracies, as well as losses after each epoch, are also recorded.

Using 32*32 sized images

Test- 1

- For this, the 32832 images are used
- Resnet-18 model is downloaded keeping the pre-trained weights True. The final fully connected layer is changed to the required dimensions.

  model = models.resnet18(pretrained=True)

  model.fc = Linear(512, 10)

- Also, the SGD optimizer is run on only the parameters in the fc layer using Cross entropy loss. Here we can see it is similar to fine-tuning. Also, learning rates are changed manually. For the first 30 epochs lr=0.1, for the next 30 lr=0.01 and for the final 30 lr=0.001where lr is the learning rate of the SGD optimizer.

  criterion = nn.CrossEntropyLoss()

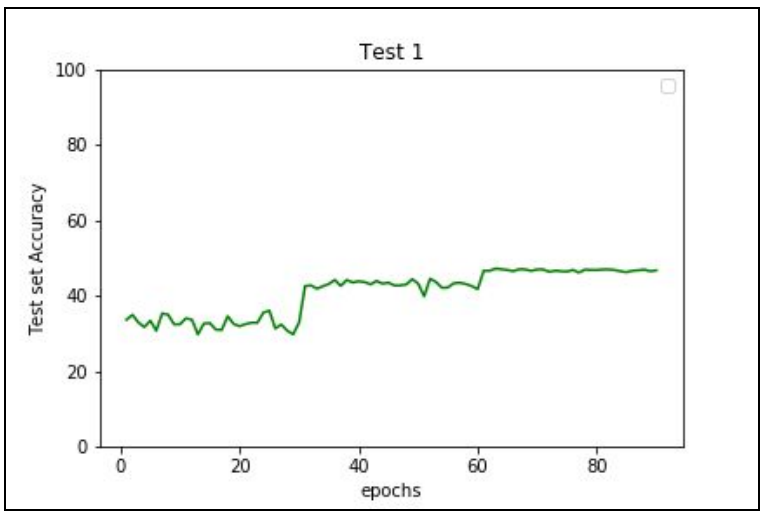
  optimizer = optim.SGD(model.fc.parameters(), lr=0.1, momentum=0.9, weight_decay=5e-4)

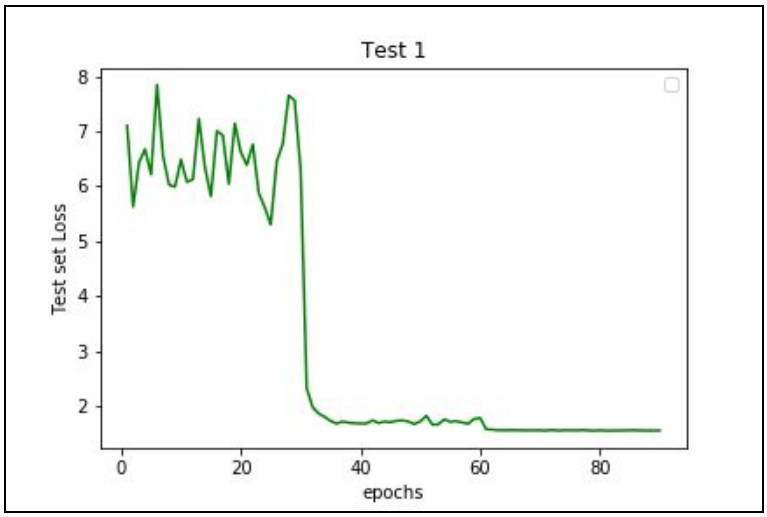
  The whole architecture can be viewed by running the code snippet from the notebook.

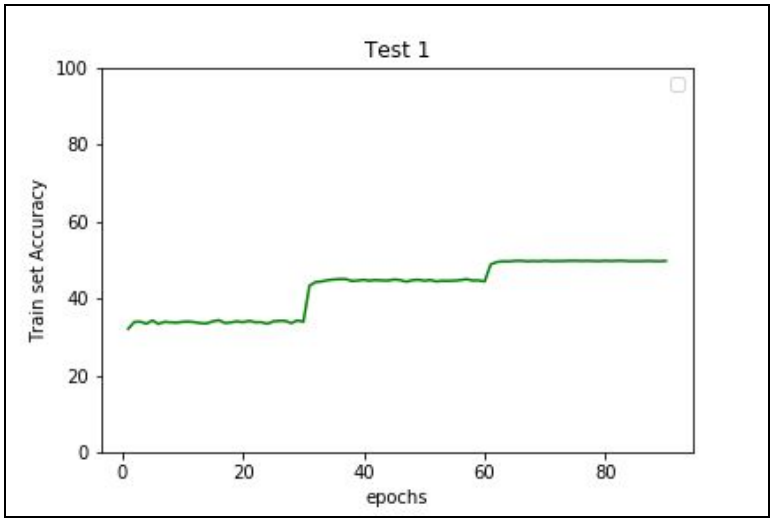
- As we can see different learning rates are used in order to maximize the accuracy and minimize the loss.
- The 3 models at 30, 60, 90 epochs are saved and will be submitted in the assignment folder.

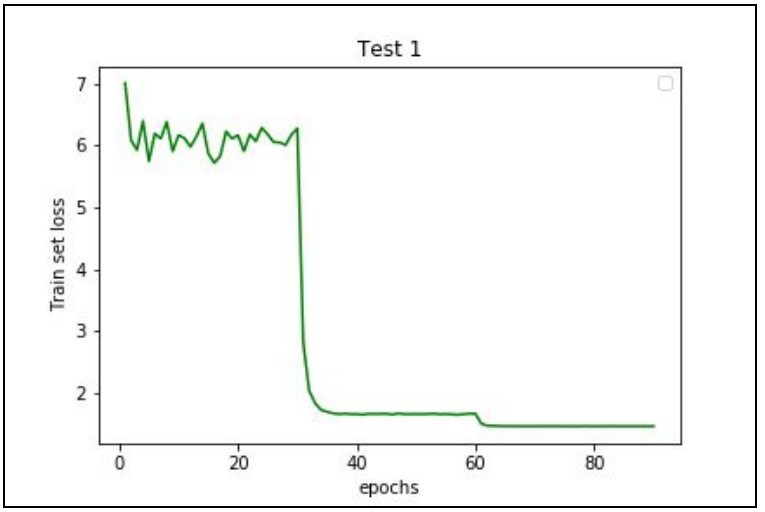
Plots

The plots of train accuracy, train loss, test accuracy, test loss vs epochs are given below.









Observation table

Model name	Test accuracy	Test loss	Train accuracy	Train loss
Test1_model-30	32.85	6.28	33.884	6.27
Test1_model-60	41.7	1.78	44.342	1.66
Test1_model-90	46.66	1.55	49.68	1.45

Inferences

- We are not able to get good accuracies similar to those from previous assignments.
- We are also getting comparatively higher losses for this test.
- The main reason for this bad performance is the size of the image we are using. We used 32*32 sized images. The initial convolution and max pool layers decrease the width and height so much that it has lesser parameters from the image itself.
- One solution for this is by resizing the 32*32 images to 224*224 images which are done in upcoming tests.
- Also in this test, the resnet 18 model is just fine-tuned in the last layer. And so it would not be able to learn a lot of things just within this last layer. This is also a reason why it shows bad results on the test as well as a train set.
- Final train accuracy, as well as test accuracy, is less than 50 percent which is considered to be bad.
- Another way to solve this problem is to change the architecture of the resnet. The resnet architecture should be changed in such a way that the initial conv, as well as the max pool layers, doesn't diminish the height and width by too much. I would try to do this as one of the tests below.

Test- 2

- 32*32 sized images are used. No resizing.
- Resnet-18 model is downloaded keeping the pretrained weights False. The final fully connected layer is changed to required dimensions.

  model = models.resnet18(pretrained=False)

  model.fc = Linear(512, 10)

- SGD , Cross Entropy Loss.
- Learning rate changed manually, 0.1 first 30 epochs. 0.01 next 30, 0.001 last 30.

  criterion = nn.CrossEntropyLoss()

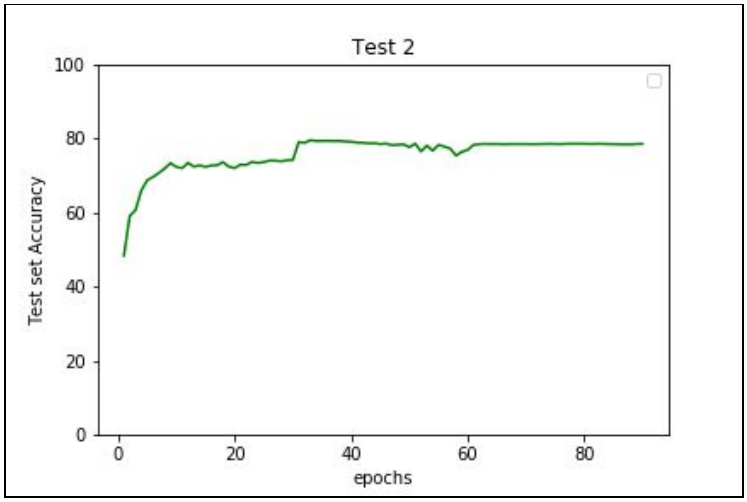
  optimizer = optim.SGD(model.parameters(), lr=0.1, momentum=0.9, weight_decay=5e-4)

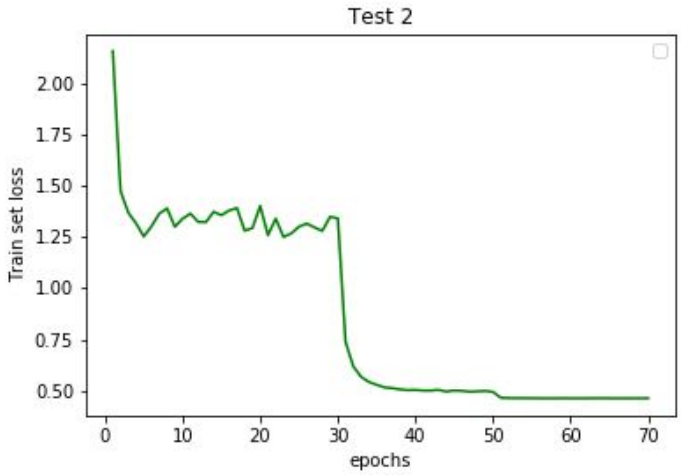
  The whole architecture can be viewed by running the code snippet from notebook.

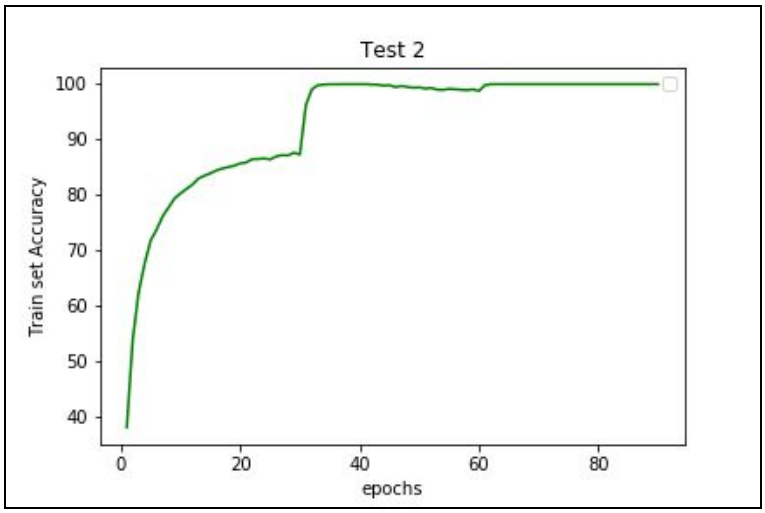
- The three saved models would be given in the assignment folder.

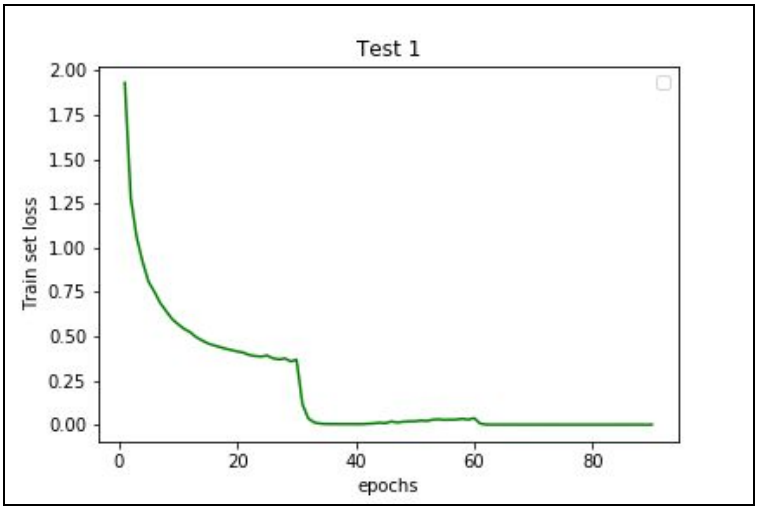
Plots

The plots of train accuracy, train loss, test accuracy, test loss vs epochs are given below.









Observation table

Model name	Test accuracy	Test loss	Train accuracy	Train loss
Test2_model-30	74.07	0.860	87.316	0.367
Test2_model-60	76.8	1.263	98.76	0.037
Test2_model-90	78.43	1.4224	100	0.0002

Inferences

- Better results compared to Test 1.
- Optimization is done on the whole model while in test 1 done for fc layers only. So more models can be trained leading to better results.
- Overfitting can be observed. Final train accuracy coming to 100% while final test accuracy remains at 78.43%.
- Even though we are using th resnet model which diminishes the width and height in the first few layers still we are getting higher accuracies than the ones we got in Assignment 2 which was about 73 percent. This is because of the increased number of parameters as well as the efficiency of resnet.

By Resizing the images

Test- 4

- Initially, the 32*32 images are resized to 224*224 sized images using the transforms command given below. This is done for both tests as well as train datasets.

  torchvision.transforms.Resize((224,224), interpolation=2)

- Resnet-18 model is downloaded keeping the pretrained weightsTrue. The final fully connected layer is changed to the required dimensions.

  model = models.resnet18(pretrained=True)

  model.fc = Linear(512, 10)

- Also, the SGD optimizer is run on the whole model parameters using Cross entropy loss.
- Also, learning rates are changed manually. First 30 epochs lr=0.1, for the next 20 lr=0.01 and for the final 20 lr=0.001where lr is the learning rate of the SGD optimizer. Total of 70 epochs used.

  criterion = nn.CrossEntropyLoss()

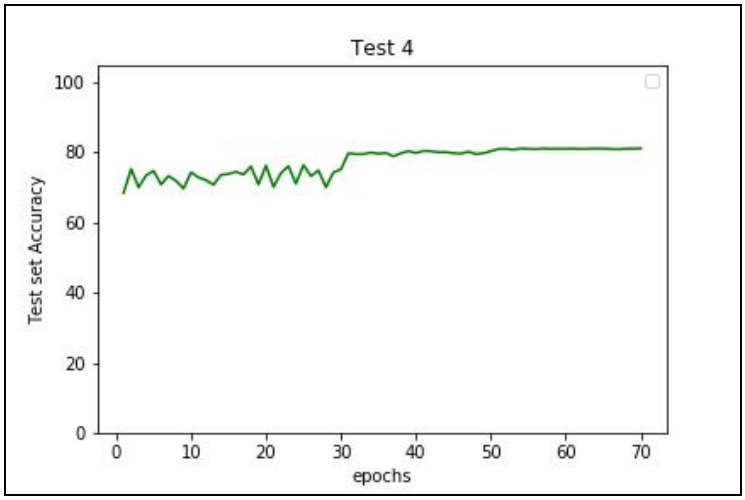
  optimizer = optim.SGD(model.fc.parameters(), lr=0.1, momentum=0.9, weight_decay=5e-4)

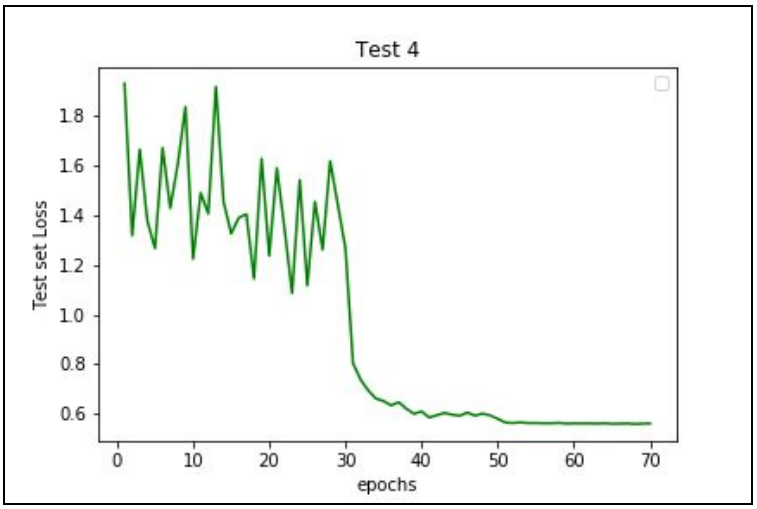
  The whole architecture can be viewed by running the code snippet from notebook.

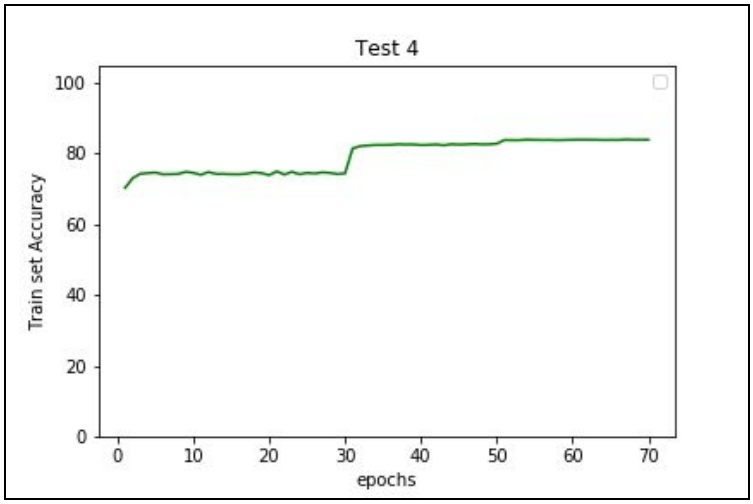
- The three models are saved and submitted.

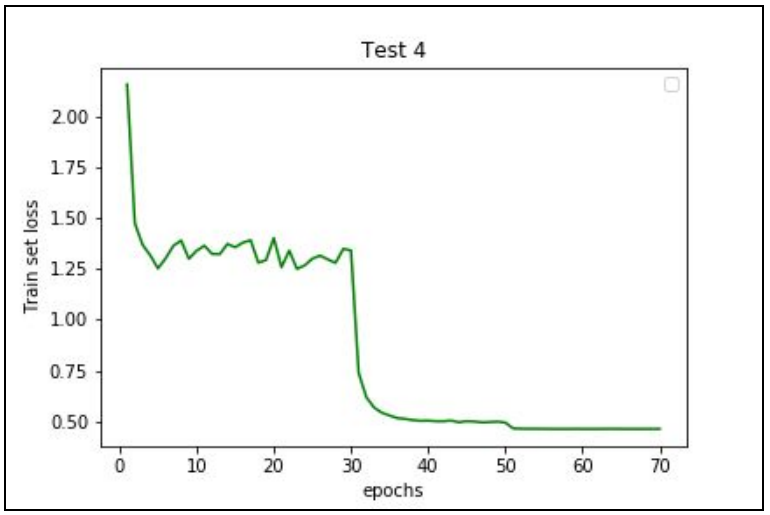
Plots

The plots of train accuracy, train loss, test accuracy, test loss vs epochs are given below.









Observation table

Model name	Test accuracy	Test loss	Train accuracy	Train loss
Test4_model-30	75.1	1.26	74.3	1.33
Test4_model-50	80.3	0.57	82.7	0.49
Test4_model-70	81.1	0.56	83.8	0.46

Inferences

- We are able to train a fairly good model with a test accuracy coming to 81 percent.
- Training accuracy here is coming close to the test accuracy. It is coming as 83.8 after epoch 70. In other words, overfitting is presented in these cases.
- The main reason for not causing overfitting is because we are optimizing the final fc layer only. Thusb the number of parameters on which the model is trained is limited.
- We can see a clear drop as different learning rates are used. This is because each learning rate provides different changes in grads leading to better accuracies.

Test- 5

- Initially, the 32*32 images are resized to 224*224 sized images using the transforms command given below. This is done for both test and train datasets.

  torchvision.transforms.Resize((224,224), interpolation=2)

- Resnet-18 model is downloaded keeping the pre-trained weights False. The final fully connected layer is changed to the required dimensions.

  model = models.resnet18(pretrained=True)

  model.fc = Linear(512, 10)

- Also, the SGD optimizer is run on the model parameters on all the layers using Cross entropy loss. Also, learning rates are changed manually. For the first 30 epochs lr=0.1, for the next 20 lr=0.01 and for the final 10 lr=0.001where lr is the learning rate of the SGD optimizer.
- Total of 60 epochs.

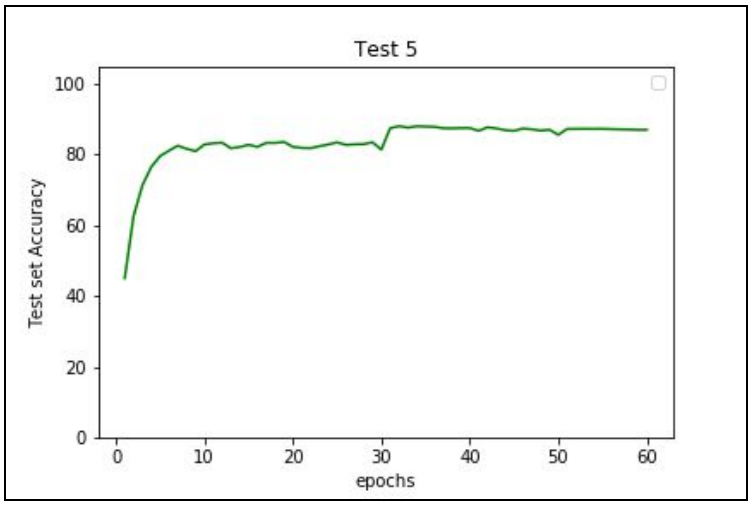
  criterion = nn.CrossEntropyLoss()

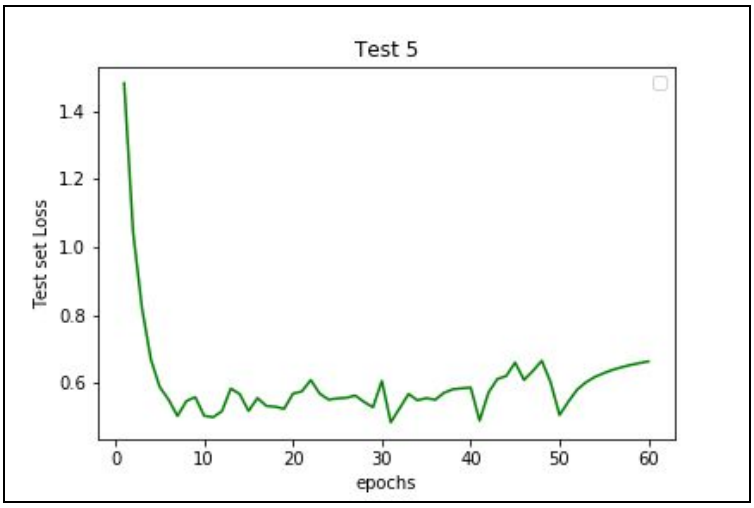
  optimizer = optim.SGD(model.parameters(), lr=0.1, momentum=0.9, weight_decay=5e-4)

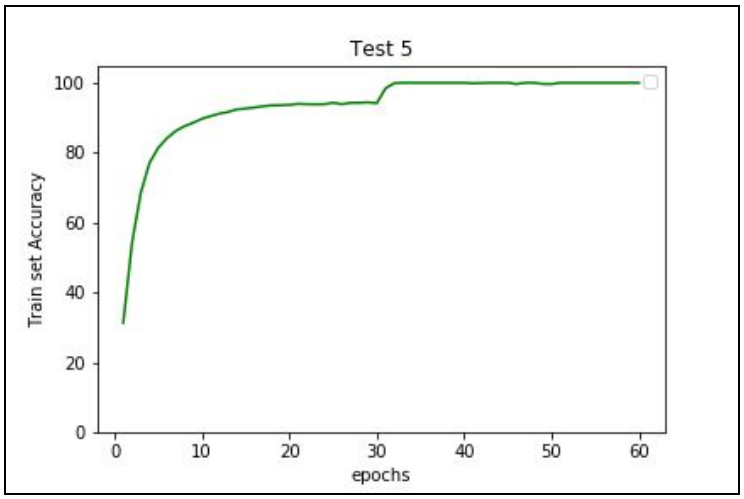
  The whole architecture can be viewed by running the code snippet from the notebook.

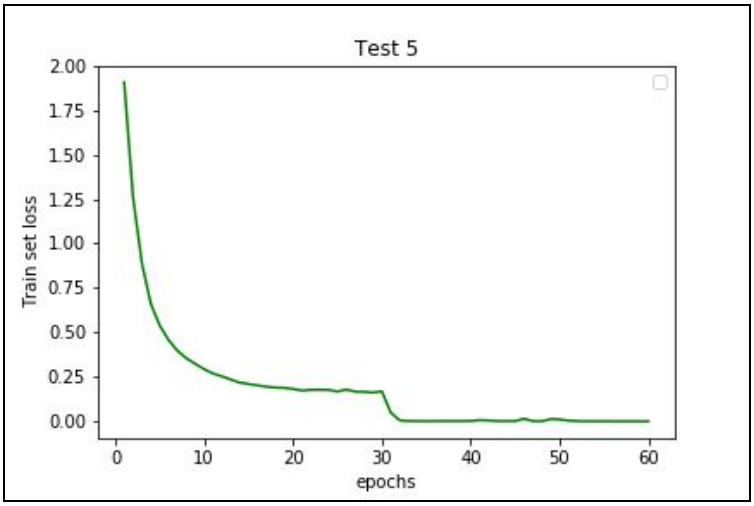
Plots

The plots of train accuracy, train loss, test accuracy, test loss vs epochs are given below.









Observation table

Model name	Test accuracy	Test loss	Train accuracy	Train loss
Test5_model-30	81.3	0.605	94.1	0.168
Test5_model-50	85.5	0.55	99.6	0.011
Test5_model-60	86.98	0.66	100	0.0005

Inferences

- Maximum test accuracy of 87 percent after 60 epochs.
- Maximum training accuracy of 100 percent after 60 epochs.
- Better training and test accuracies than Test 4.
- Found that higher epochs doesn't change much from the previous model and so implemented only 60 epochs in this model.
- Overfitting can be observed. This is mainly because of a large number of parameters in training and testing.
- Different values of hyperparameters tried out and its effect can be observed from the plots and table.

Test- 6

- 224*224 sized images used.

  torchvision.transforms.Resize((224,224), interpolation=2)

- Resnet-18 model is downloaded keeping the pretrained weights True. The final fully connected layer is changed to the required dimensions.

  model = models.resnet18(pretrained=True)

  model.fc = Linear(512, 10)

- Also, the SGD optimizer is run on the whole model parameters using Cross entropy loss. Also, the learning rates are changed manually. For the first 10 epochs lr=0.1, for the next 10 lr=0.01 and for the final 10 lr=0.001where or is the learning rate of the SGD optimizer.
- Total of 30 epochs. I found from the previous tests that even with higher epochs there is not much change because the accuracies and losses saturate quickly after 10 epochs. So I have used only 10 epochs for each learning parameter here.

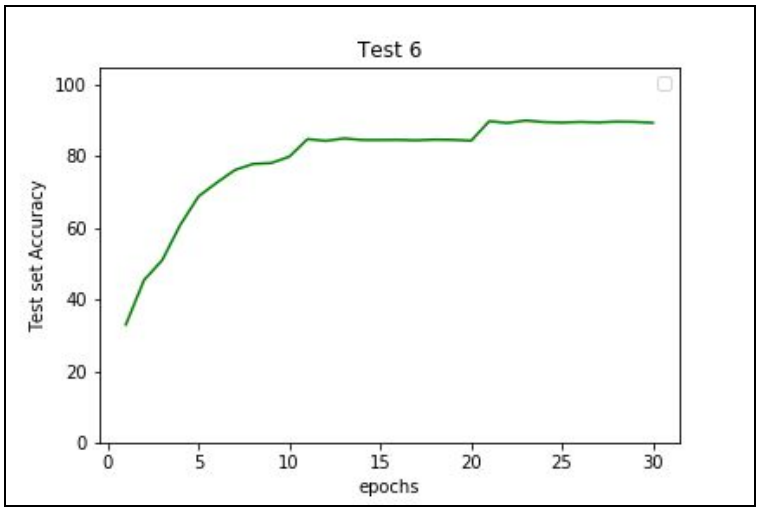
  criterion = nn.CrossEntropyLoss()

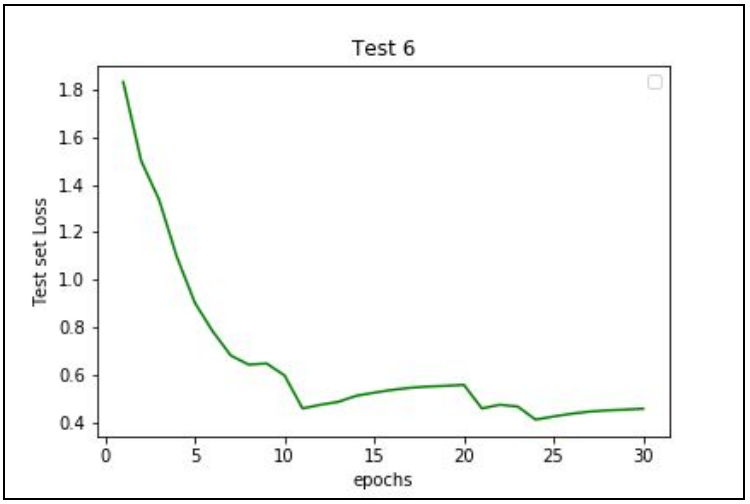
  optimizer = optim.SGD(model.parameters(), lr=0.1, momentum=0.9, weight_decay=5e-4)

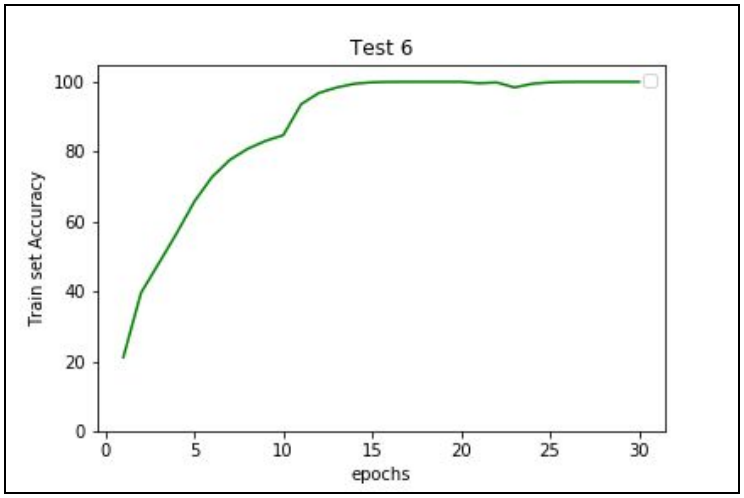
  The whole architecture can be viewed by running the code snippet from the notebook.

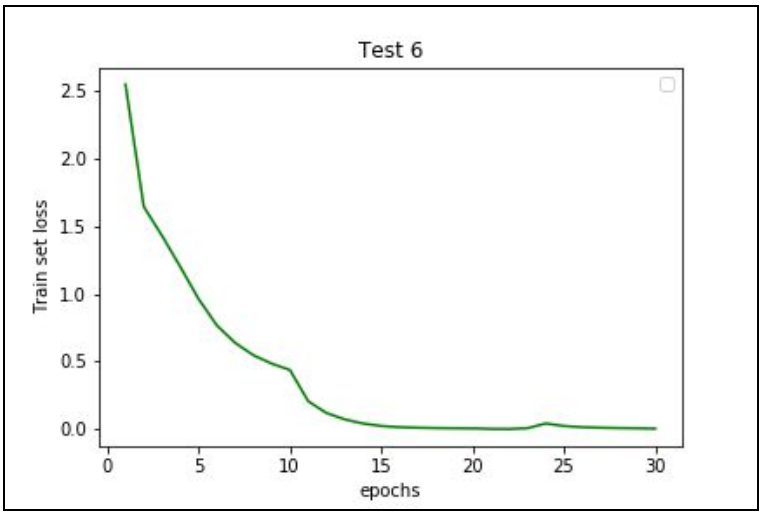
Plots

The plots of train accuracy, train loss, test accuracy, test loss vs epochs are given below.









Observation table

Model name	Test accuracy	Test loss	Train accuracy	Train loss
Test6_model-30	79.9	0.59	84.7	0.43
Test6_model-60	84.34	0.55	100	0.005
Test6_model-90	89.3	0.459	100	0.003

Inferences

- Test accuracy of 89.3.
- Train accuracy of 100 percent.
- Overfitting is observed. Because of a mainly a large number of parameters.
- Different learning rates are tried out to get the best model. We can see the effect of different learning rates on the plots.
- From the previous models, I found out that for most of the epochs the test and train accuracies are found to remain constant. They approach saturation quickly because comparatively a lower minibatch size is used. So I have decreased the number of epochs drastically, to 10 each for different learning rates.
- To further improve the accuracies I think we should be doing data augmentation techniques sinceb a clear case of overfitting is used.

Results and conclusions

Test	Test accuracy	Test loss	Train accuracy	Train loss
Test1	46.66	1.55	49.68	1.45
Test6_model-60	84.34	0.55	100	0.005
Test6_model-90	89.3	0.459	100	0.003

- I was able to understand the effect of different regularization parameters on test and train dataset.
- I was able to find out the best hyperparameter by tuning these hyperparameters.
- Also was able to understand the difference in behaviour of test/ train dataset when two different optimizers are used.

Experiment2- Augmentation

In this experiment, I would be doing augmentation. I would be using the Tiny CIFAR dataset for this problem which consists of 500 images per class in from cifar dataset while the test class remains the same.

Test- 11

- Initially from the CIFAR 10 500 images from each class are to be generated as a test set. I used the sampler given below in the data loader for this. It would generate some random images from CIFAR 10 and the number om images from each class is close to 500. (490-510) range which is suitable for this assignment.

  sampler=torch.utils.data.SubsetRandomSampler(indices)

- 224*224 sized images used.

  torchvision.transforms.Resize((224,224), interpolation=2)

- Resnet-18 model is downloaded keeping the pretrained weights True. The final fully connected layer is changed to the required dimensions.

  model = models.resnet18(pretrained=False)

  model.fc = Linear(512, 10)

- Also, the SGD optimizer is run on the whole model parameters using Cross entropy loss. Also, learning rates are changed manually. For the first 10 epochs lr=0.1, for the next 10 lr=0.01.
- Total of 30 epochs. I found from the previous tests that even with higher epochs there is not much change because the accuracies and losses saturate quickly after 10 epochs. So I have used only 10 epochs for each learning parameter here.

  criterion = nn.CrossEntropyLoss()

  optimizer = optim.SGD(model.parameters(), lr=0.1, momentum=0.9, weight_decay=5e-4)

  The whole architecture can be viewed by running the code snippet from the notebook.

- A total of 20 epochs used. After this augmentation is done for the trained dataset. The different augmentation techniques used are given below.

  torchvision.transforms.RandomSizedCrop

  torchvision.transforms.RandomVerticalFlip

  torchvision.transforms.RandomRotation

  torchvision.transforms.RandomPerspective

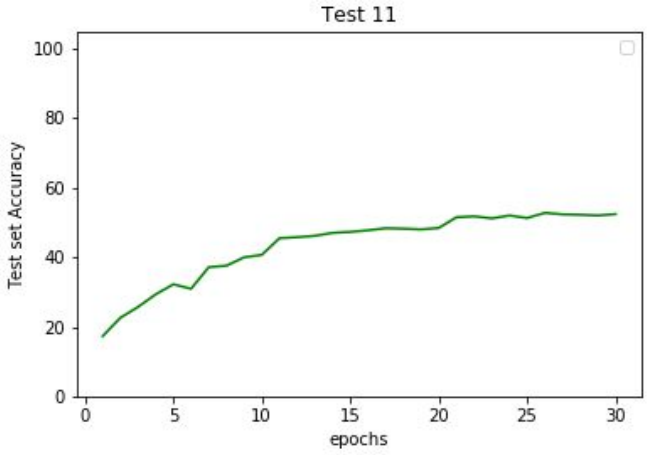
  torchvision.transforms.RandomHorizontalFlip

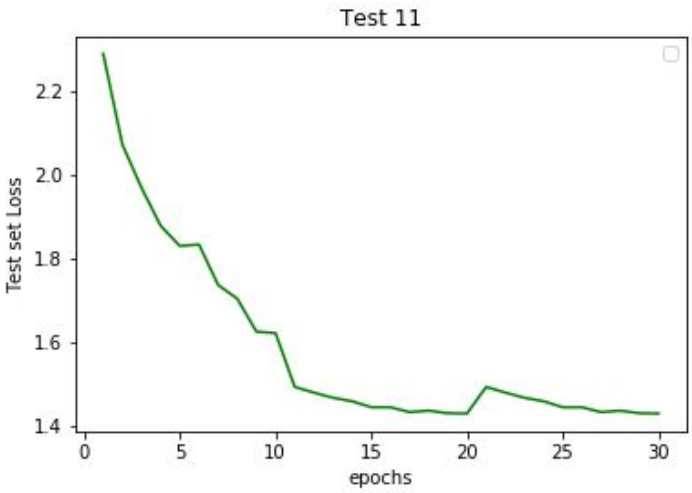
  torchvision.transforms.ColorJitter

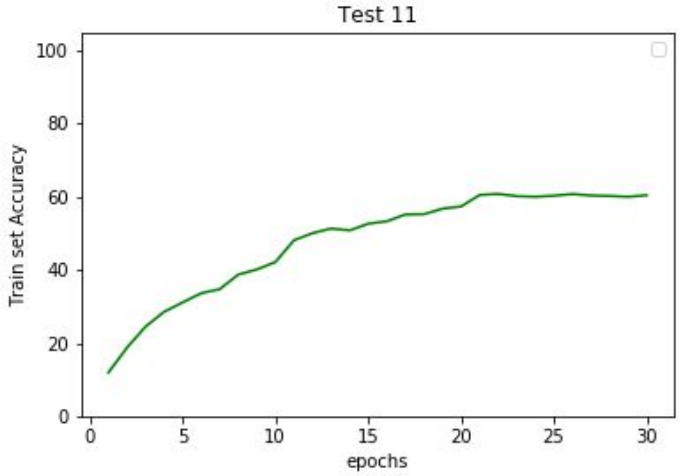
- The model is trained for 10 epochs and observations noted. From the below plot 20-30 for the training results with augmentation.

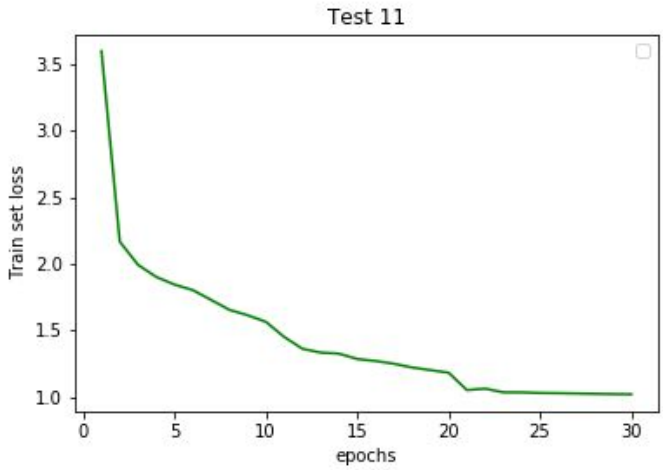
Plots

The plots of train accuracy, train loss, test accuracy, test loss vs epochs are given below.









Observation table

Test	Test accuracy	Test loss	Train accuracy	Train loss
Test12_model-10	40	1.85	42	1.65
Test12_model-20	48	1.65	57	1.46
Test12_model-30	**52**	**1.64**	**60**	**1.35**

Inferences

- Test accuracy of 52 percent.
- Train accuracy of 60 percent.
- No overfitting is observed in this case.
- The reason for all of this is the low train set size.
- With the augmentation in the epochs from 20-30, we can see a clear increase in the test as well as train accuracy. This is because augmentation can be seen as an increase in the train dataset size and thus now we have different orientations of images which helps in training better models.

Test- 12

- Initially from the CIFAR 10 500 images from each class are to be generated as a test set. I used the sampler given below in the data loader for this. It would generate some random images from CIFAR 10 and the number of om images from each class is close to 500. (490-510) the range which is suitable for this assignment.

  sampler=torch.utils.data.SubsetRandomSampler(indices)

- 224*224 sized images used.

  torchvision.transforms.Resize((224,224), interpolation=2)

- Resnet-18 model is downloaded keeping the pretrained weights True. The final fully connected layer is changed to the required dimensions.

  model = models.resnet18(pretrained=True)

  model.fc = Linear(512, 10)

- Also, the SGD optimizer is run on the whole model parameters using Cross entropy loss. Also, learning rates are changed manually. For the first 10 epochs lr=0.1, for the next 10 lr=0.01 and for the final 20 lr=0.001where or is the learning rate of the SGD optimizer.
- Total of 30 epochs. I found from the previous tests that even with higher epochs there is not much change because the accuracies and losses saturate quickly after 10 epochs. So I have used only 10 epochs for each learning parameter here.

  criterion = nn.CrossEntropyLoss()

  optimizer = optim.SGD(model.parameters(), lr=0.1, momentum=0.9, weight_decay=5e-4)

  The whole architecture can be viewed by running the code snippet from the notebook.

- A total of 40 epochs used. After this augmentation is done for the trained dataset. The different augmentation techniques used are given below.

  torchvision.transforms.RandomSizedCrop

  torchvision.transforms.RandomVerticalFlip

  torchvision.transforms.RandomRotation

torchvision.transforms.RandomPerspective

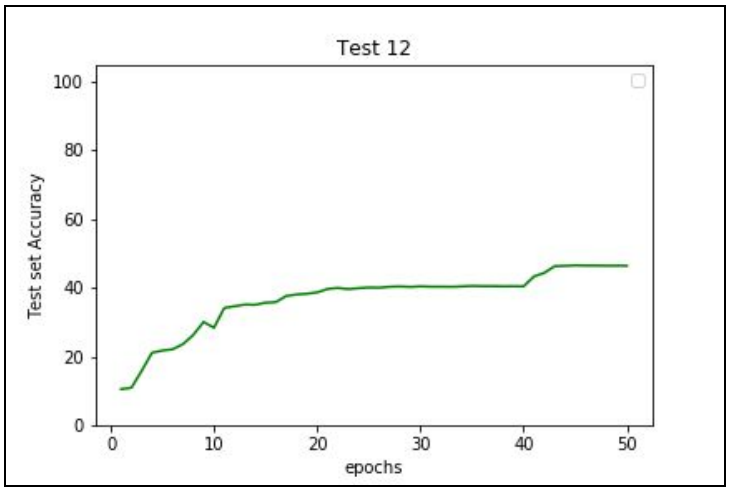
torchvision.transforms.RandomHorizontalFlip

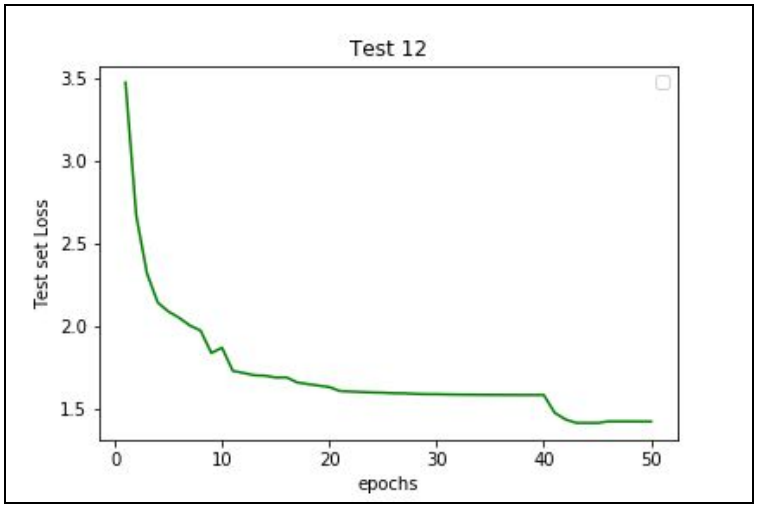
torchvision.transforms.ColorJitter

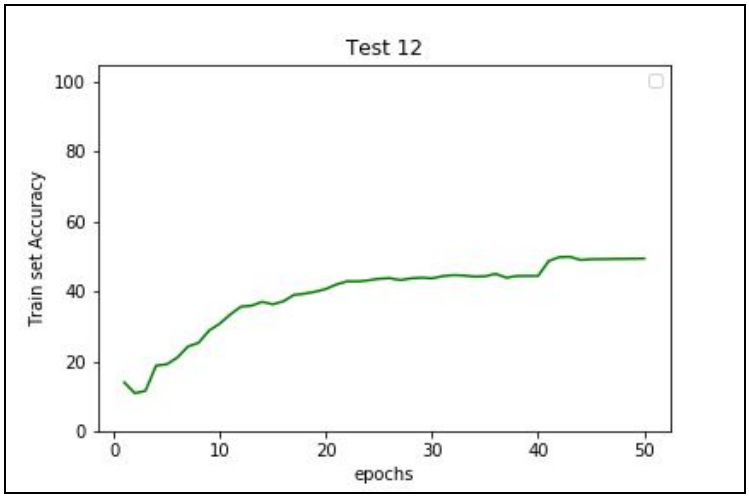
- The model is trained for 10 epochs and observations noted. From the below plot 40 -50 for the training results with augmentation.

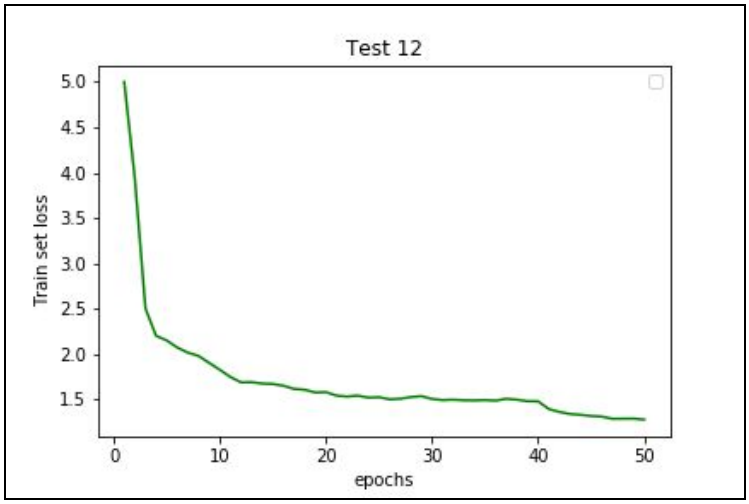
Plots

The plots of train accuracy, train loss, test accuracy, test loss vs epochs are given below.









Observation table

Test	Test accuracy	Test loss	Train accuracy	Train loss
Test12_model-10	28.55	1.85	30.83	1.65
Test12_model-20	38.6	1.65	41.2	1.46
Test12_model-30	41.3	1.64	41.5	1.35
Test12_model-40	**42.1**	**1.53**	**43.3**	**1.32**
Test12_model-50	**46.4**	**1.4**	**49.42**	**1.2**
Test12_model-60	**46.9**	**1.4**	**48.1**	**1.26**

Inferences

- Test accuracy of 46.4 percent.
- Train accuracy of 49.42 percent.
- No overfitting is observed in this case.
- The reason for all of this is the low train set size.
- With the augmentation in the epochs from 40 to 60, we can see a clear increase in the test as well as train accuracy. This is because augmentation can be seen as an increase in the train dataset size and thus now we have different orientations of images which helps in training better models.

Dropout implementation

- Further to the Test 12 dropout test is performed. In this Dropout is done along with optimization for 10 epochs and results are observed.
- As can be seen from the table given above above from epoch 50- 60 the results of dropout.
- There is a slight increase in test accuracy from 46.4 to 46.9 percent.
- There is a slight decrease in train accuracy from 49.4 to 48.1 percent.
- That is with this dropout we were successfully able to decrease the overfitting that is the difference between the train and test accuracies.

References

- Pytorch documentation
- Matplotlib documentation

Codes and models

The codes and models are given in the assignment folder. The order of the codes is given. The models mentioned in this report are also submitted along

Thank you !!!