Ruchita Paithankar

CS-583

Homework #2 - Report

* **Linear Network**

The lower learning rate of 0.01 showed a smoother curve than a higher learning rate of 0.1 which is not a smooth curve and shows a lot of difficulty in learning, which is clear by the huge difference in test accuracy of the both the cases. I feel small values do benefit the training process for this network and a higher learning rate results in unstable training process.

Learning rate 0.01:

Test accuracy of the network: 45.51%

Train accuracy of the network: 40.67%

Learning rate 0.1:

Test accuracy of the network: 33.51%

Train accuracy of the network: 40.58%

Chart, line chart

Description automatically generated

0.01 learning rate

Chart, line chart

Description automatically generated

0.1 learning rate

* **MLP network**

The graph for training loss of the network without activation function did have a little problem learning that is why there is a difference in the training and testing accuracy graph and the testing graphs has a downfall. In the second instance the Activation function helps the training and testing accuracy to the extent that they are exactly same. Also, the training and testing accuracy increase with every epoch. So, adding the activation function and reducing the learning rate help in increasing the training accuracy.

Without Activation function and 0.001 learning rate

Test accuracy of the network: 44.45%

Train accuracy of the network: 39.19%

With Activation function and 0.1 learning rate

Test accuracy of the network: 43.54%

Train accuracy of the network: 43.54%

Chart, line chart

Description automatically generated

Without Activation function

Chart

Description automatically generated

With Activation function

I tried a higher learning rate of 0.1 and with the activation function and got a training accuracy of 85.01% and the testing accuracy of 50.17%. The graph for this is below and shows that a higher training rate is required for MLP network so that it does not get stuck in local minima.

Chart, line chart

Description automatically generated

* **Convolutional Network**

Learning rate 0.1 and kernel size = 5x5:

Chart

Description automatically generated

Test accuracy of the network: 91.60%

Train accuracy of the network: 58%

Learning rate 0.1 and kernel size = 3x3:

Trying the 3x3 kernel size threw this error. I am sure I went wrong somewhere in the math. I used the following formula to solve it : (W-F+2P)/S+1 and tried different combinations.

Text

Description automatically generated

Text

Description automatically generated

Learning rate 0.001 and kernel size = 5x5:

Increased in the learning rate give a better testing accuracy, I feel it happens because the small learning rate gets stuck in local minima and high learning rate helps jumping out of it.

Chart, line chart

Description automatically generated

Test accuracy of the network: 36.02%

Train accuracy of the network: 36.63%

This takes us to the question that how learning rate scheduling effect the network? The higher learning rate performs better for this network.

**Batch size:**

The changes made in the batch sizes did decrease the training accuracy, I feel this happens because the reduction in batch size reduced the variety of items in that batch.

Batch Size 4000 and learning rate of 0.1:

Test accuracy of the network: 65.16%

Train accuracy of the network: 73.29%

Batch Size 2000 and learning rate of 0.1:

Test accuracy of the network: 71.70%

Train accuracy of the network: 62.45%

Chart, line chart

Description automatically generated

Batch size 4000

Chart

Description automatically generated

Batch size 2000

Batch size of 4000 with reduced learning rate (0.01):

Chart

Description automatically generated

Decreasing the learning rate give a smoother curve for training loss and I believe is better at training the data because the training accuracy and the testing accuracy is nearly the same which is approximately 50%. I feel reducing the learning rate helps because smaller changes are made to the weights of each update and require more training epochs.

Data Trained with the following augmentations:

RandomHorizontalFlip()

RandomCrop(32,32)

GaussianBlur(5,0.2)

And a learning rate of 0.001

This was much slower on the command prompt going through all the Epoch, I feels because of adding the Gaussian Blur. The training and testing accuracy (31.01% and 31.24%) decreases as compared to the model trained without it. I feel this is because of the Gaussian Blur and the reduced learning rate that made the accuracy lower. But the training loss decreases in a weird way I feel that is caused the training and the testing accuracy being nearly similar.

Chart, line chart

Description automatically generated

Data Trained with the following augmentations:

RandomHorizontalFlip()

RandomCrop(32,32)

Here, I used the learning rate of 0.1 and the above augmentation and this gives the train and test accuracy as 24% and 35% respectively. The test accuracy is better than that with Gaussian blur shows that blurry images does it make it difficult to identity.

Chart, line chart

Description automatically generated

Adam with batch size of 4000 and learning rate of 0.01

Chart, line chart

Description automatically generated

Test accuracy of the network: 72.69%

Train accuracy of the network: 60.05%

SGD with batch size of 4000 and learning rate of 0.01

Chart

Description automatically generated

Test accuracy of the network: 59.16%

Train accuracy of the network: 50.59%

Adam with batch size of 2000 and learning rate of 0.01

In the previous experiment I saw that reducing the batch size gave a better training and testing accuracy so I tried it with Adam which performed better out of the two optimizers.

Chart, line chart

Description automatically generated

The testing accuracy was 59.65% so there wasn’t much of a change as compared to the 60.05% at 0.01 learning rate and batch size of 4000

Since SGD is more locally unstable. I thought that increasing the learning rate to 0.1 with a batch size same as before (4000) to see what would be the effect.

Chart

Description automatically generated

This gave approximately the same results as Adam at batch size 4000 and learning rate of 0.01. This in a way proves the fact that SGD is more locally unstable and benefits from so increasing the learning rate

Loss function affecting the result:

So I took the learning rate as 0.01 with a batch size of 4000 with Adam as the optimizer along with the loss function. This gave the worst results of all with train and test accuracy of 10%

**Chart, box and whisker chart

Description automatically generated**

So I went ahead and experimented with a learning rate of 0.1 and MSE, those were only parameters I applied and that gave a much better result. The train and test accuracy were 66% and 62% respectively. This can be compared to the very first experiment under Convolutional network and we can see that MSE performs better in the training accuracy which is 58% without MSE. I am not really sure why this happens because in Cross entropy the decision boundary in classification task is large as compared to the regression in MSE, so ideally cross entropy should perform better.

**Chart

Description automatically generated**

With MSE

How did the assignment go?

I really enjoyed the process of experimenting and making this report. It gave me a side by side comparison of how things affect the overall network and how parameters can be tweaked in order to get the desired accuracy. The one thing I struggle with is getting the math right in the convolutional network. I was honestly out of time to figure it out. Every time I changed the number of input and output channels and the kernel size the program gave me an error of mismatched sizes. So, I believe I got the numbers wrong.