HOMEWORK 4 REPORT

Constructing a LeNet-5 style CNN network using PyTorch

Nikhit Mago, CSCI 677 - Advanced Computer Vision

1. Introduction

Network:

I have constructed a CNN architecture with 2 convolutional layers with filter size of 5, followed by max pooling layers of filter size of 2 and stride of 2. There are also 2 fully connected layers of sizes 120 and 84 respectively. I have used ReLU activation function after all the layers in the network. I've experimented with size of fully connected layers and batch normalization.

Loss function and Optimizer:

The loss function used for this network is Cross Entropy Loss and optimizer is ADAM with a learning rate of 0.001.

Data:

The data includes images of 10 classes and 3 sets: training, validation, and test. I have used the Dataloader class to read the data in mini batches of 64. I have used **two data augmentation techniques:**

a) Random Horizontal Flip
b) Random Resized Crop

Network settings:

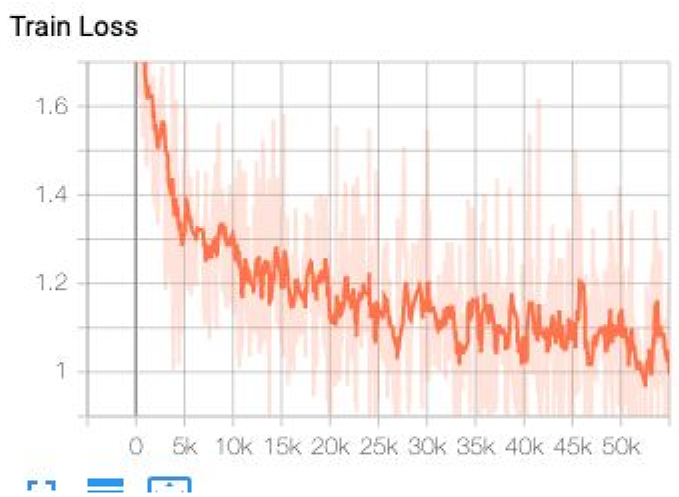
a) The number of epochs = 20
b) Train loss: reported after every batch (can have many oscillations)
c) Validation loss: reported after every epoch (average of all batches)

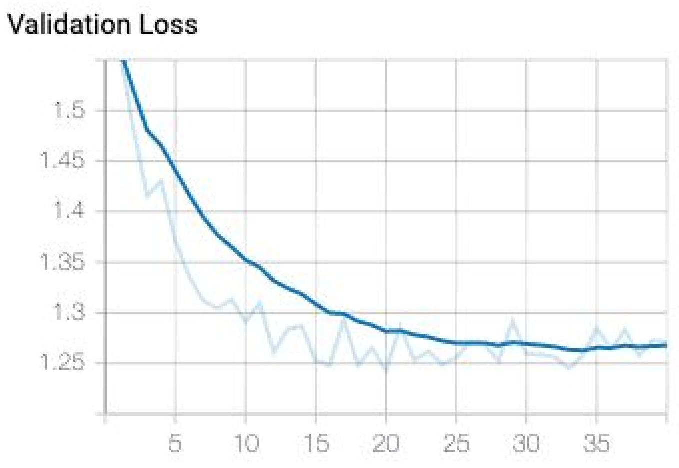
System Variations:

a) Increasing size of fully connected layers
b) Batch normalization

Baseline network:

This network includes no system variations and follows all the parameter settings mentioned above. We can observe that both train loss and validation loss decrease along the epochs which means no overfitting.

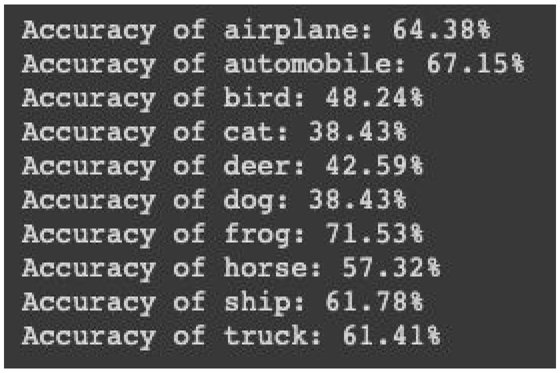




Final validation loss: 1.26

Test accuracy: 54.7%

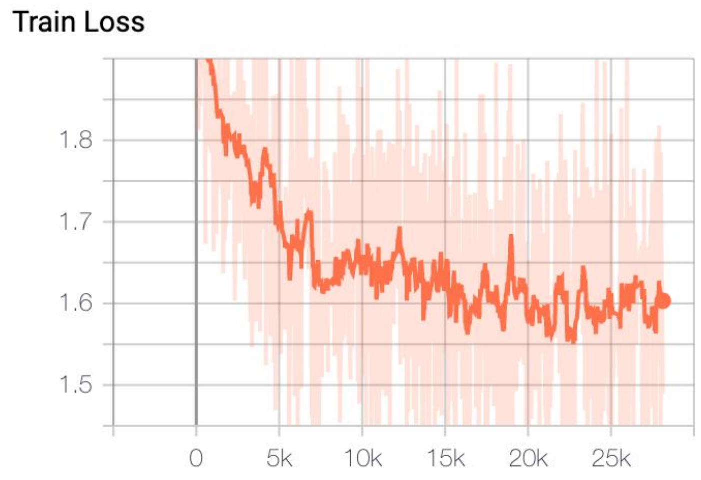
Per class test accuracy:

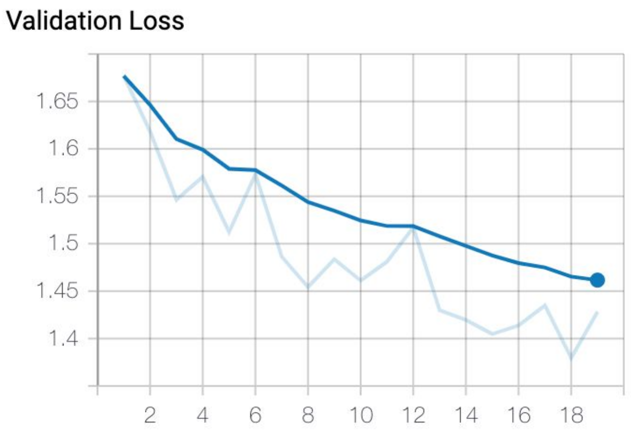


```
Accuracy of airplane: 64.38%
Accuracy of automobile: 67.15%
Accuracy of bird: 48.24%
Accuracy of cat: 38.43%
Accuracy of deer: 42.59%
Accuracy of dog: 38.43%
Accuracy of frog: 71.53%
Accuracy of horse: 57.32%
Accuracy of ship: 61.78%
Accuracy of truck: 61.41%
```

System Variation 1

Increasing the size of first fully connected layer from 120 to 200 and second layer from 84 to 128. Although, the validation loss doesn't increase along the epochs, but train loss, validation loss increases as compared to the baseline model, and test accuracy decreases.



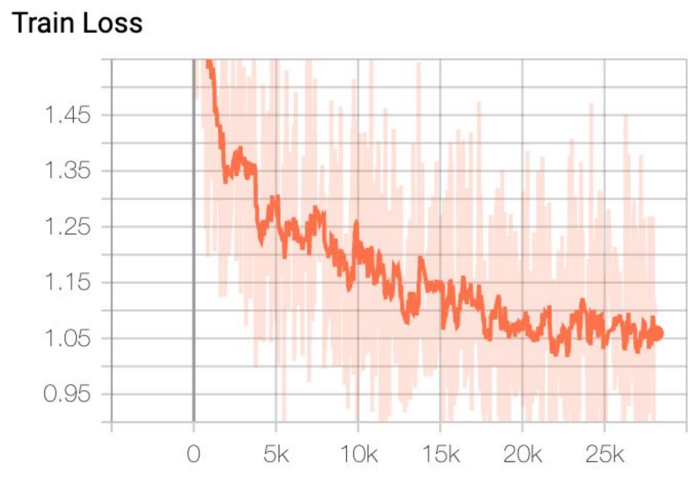


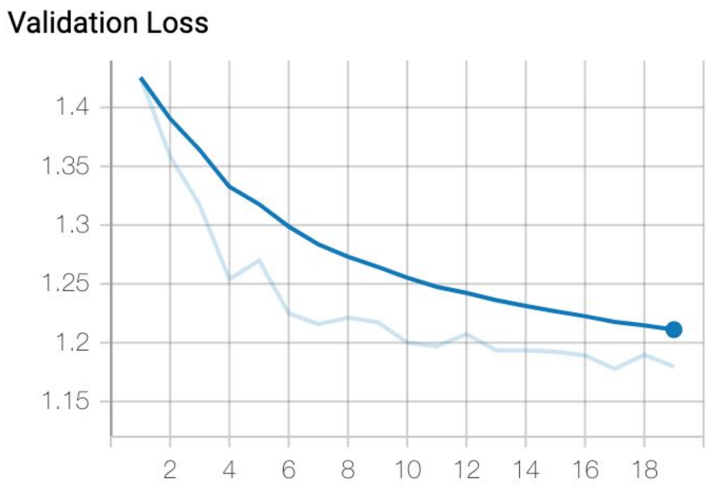
Final validation loss: 1.46

Test accuracy: 51.2%

System Variation 2

Introducing batch normalization 2d layers after the 2 convolution layers and batch normalization 1d layers after the 2 fully connected layers. This method seems to improve all 3, train loss, validation loss, and test set accuracy.

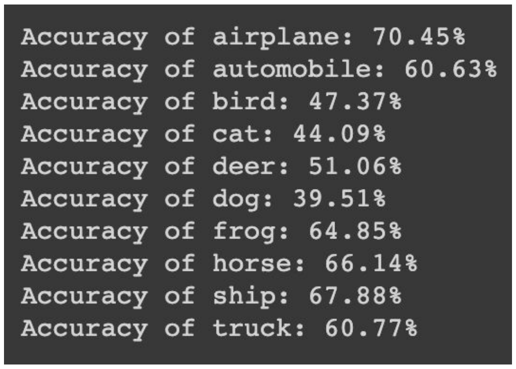




Final validation loss: 1.18

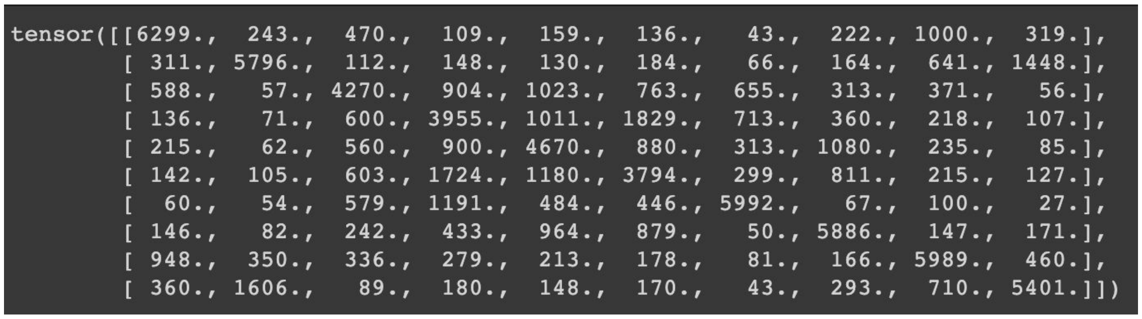
Test accuracy: 57.8%

Per class test accuracy:



```
Accuracy of airplane: 70.45%
Accuracy of automobile: 60.63%
Accuracy of bird: 47.37%
Accuracy of cat: 44.09%
Accuracy of deer: 51.06%
Accuracy of dog: 39.51%
Accuracy of frog: 64.85%
Accuracy of horse: 66.14%
Accuracy of ship: 67.88%
Accuracy of truck: 60.77%
```

Confusion Matrix:
(Order is: ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'])



```
tensor([[6299.,  243.,  470.,  109.,  159.,  136.,   43.,  222., 1000.,  319.],
        [ 311., 5796.,  112.,  148.,  130.,  184.,   66.,  164.,  641., 1448.],
        [ 588.,   57., 4270.,  904., 1023.,  763.,  655.,  313.,  371.,   56.],
        [ 136.,   71.,  600., 3955., 1011., 1829.,  713.,  360.,  218.,  107.],
        [ 215.,   62.,  560.,  900., 4670.,  880.,  313., 1080.,  235.,   85.],
        [ 142.,  105.,  603., 1724., 1180., 3794.,  299.,  811.,  215.,  127.],
        [  60.,   54.,  579., 1191.,  484.,  446., 5992.,   67.,  100.,   27.],
        [ 146.,   82.,  242.,  433.,  964.,  879.,   50., 5886.,  147.,  171.],
        [ 948.,  350.,  336.,  279.,  213.,  178.,   81.,  166., 5989.,  460.],
        [ 360., 1606.,   89.,  180.,  148.,  170.,   43.,  293.,  710., 5401.]])
```

CONCLUSION

Method	**Parameters**	**Final Val Loss**	**Test accuracy**
Baseline		1.26	54.7%
Increase size of FC layers	120 -> 200 84 -> 128	1.46	51.2%
Batch normalization		1.18	57.8%

It can be summarized that increasing the size of fully connected layers is a bad idea as it adds unnecessary model complexity which reduces the generalization ability of the network. However in the case of batch normalization, this method normalizes the outputs of the layers using mean and standard deviation,the model not only converges faster but finds global minima faster since it damped oscillations of the loss function and adds a small regularization effect.