Assignment 2

Ngawang (Choenden) Kyirong

301312227

1 late day used

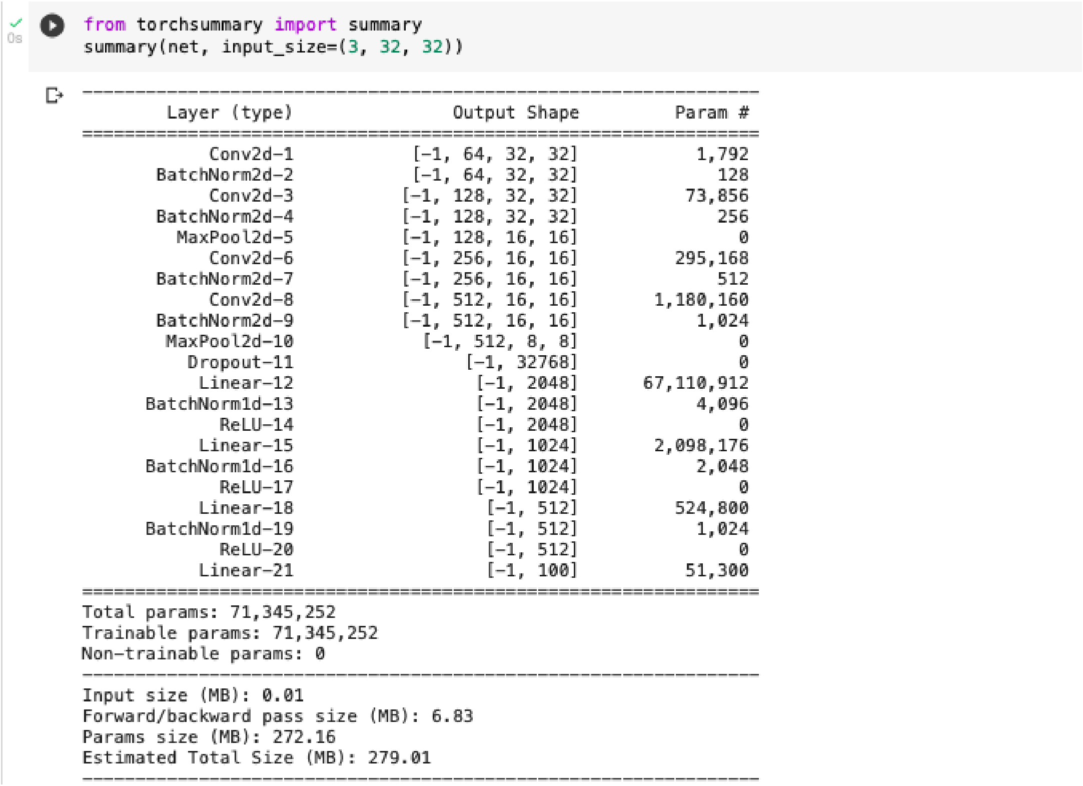
Part 1.

For this part 1 the best score i got on kaggle is: 0.63700. The model architecture is shown below. This network is a combination of convolutional layers and fully connected layers. Each convolutional step is followed by a batch normalization and each fully connected layer is followed by a batch normalization as well. Moreover, THere are only 2 instances of max pooling. Also, the activation function used was ReLU.

In terms of preprocessing, I calculated the sample mean and sample standard deviation of the whole training dataset and applied normalization on both the train, test, and validation. Moreover, data augmentation was done by horizontally flipping and random cropping on the training set only.

The change that led to my biggest jump to the low 60s in validation was the implementation of data augmentation and data transformation. No particular study was used to reach this conclusion but i used this resource below to understand and use transformations which ultimately increased my score drastically.

https://pytorch.org/tutorials/beginner/data_loading_tutorial.html#transforms

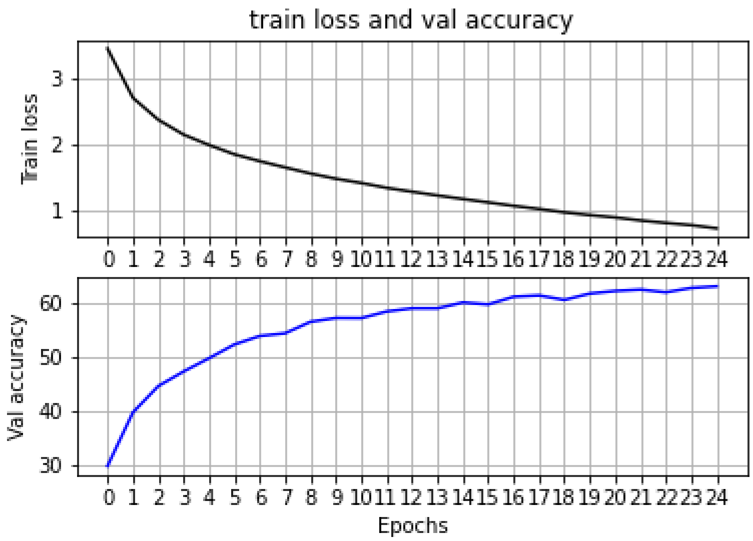


```python
from torchsummary import summary
summary(net, input_size=(3, 32, 32))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,792
       BatchNorm2d-2           [-1, 64, 32, 32]             128
            Conv2d-3          [-1, 128, 32, 32]          73,856
       BatchNorm2d-4          [-1, 128, 32, 32]             256
         MaxPool2d-5          [-1, 128, 16, 16]               0
            Conv2d-6          [-1, 256, 16, 16]         295,168
       BatchNorm2d-7          [-1, 256, 16, 16]             512
            Conv2d-8          [-1, 512, 16, 16]       1,180,160
       BatchNorm2d-9          [-1, 512, 16, 16]           1,024
        MaxPool2d-10            [-1, 512, 8, 8]               0
          Dropout-11                [-1, 32768]               0
           Linear-12                 [-1, 2048]      67,110,912
      BatchNorm1d-13                 [-1, 2048]           4,096
             ReLU-14                 [-1, 2048]               0
           Linear-15                 [-1, 1024]       2,098,176
      BatchNorm1d-16                 [-1, 1024]           2,048
             ReLU-17                 [-1, 1024]               0
           Linear-18                  [-1, 512]         524,800
      BatchNorm1d-19                  [-1, 512]           1,024
             ReLU-20                  [-1, 512]               0
           Linear-21                  [-1, 100]          51,300
================================================================
Total params: 71,345,252
Trainable params: 71,345,252
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 6.83
Params size (MB): 272.16
Estimated Total Size (MB): 279.01
----------------------------------------------------------------
```

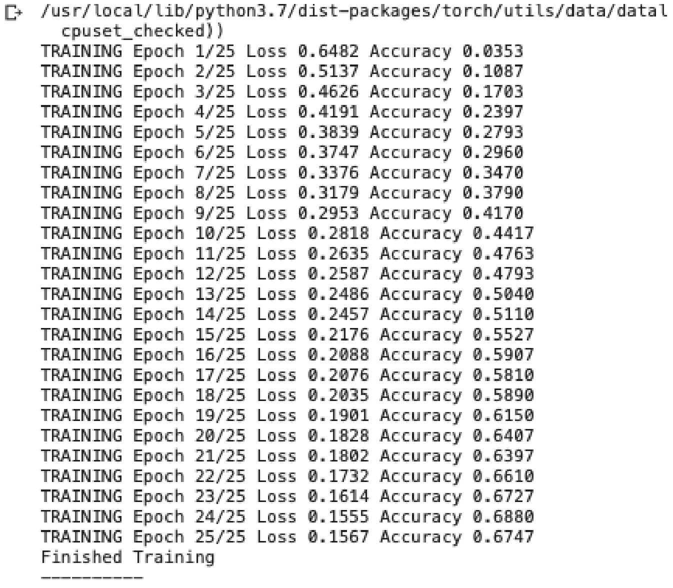
The model was trained using SGD with a learning rate of 0.005, momentum 0.9, and weight decay of 1e-4. I used 25 epochs and batch size of 32

Moreover, here we can see the 2 graphs describing the train loss and val accuracy as a function of the epoch number it was on.

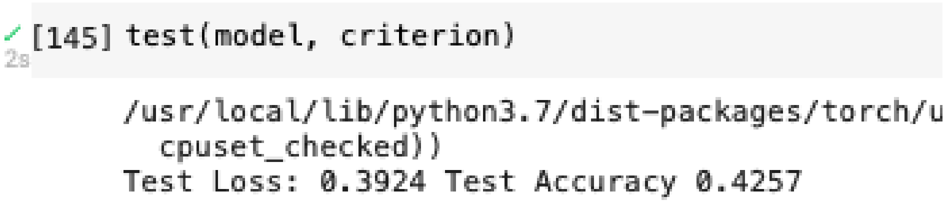


Part 2.

I have changed the hyperparameters by increasing the learning rate to 0.005 and increasing epochs to 25. Also, I normalized the data set to mean (0,0,0) and sd (1,1,1). Also, I performed data augmentation via transformations that include: resizing, random resize crop, and random horizontal flips. As shown below, here is the accuracy for using resnet18 as a feature extractor:



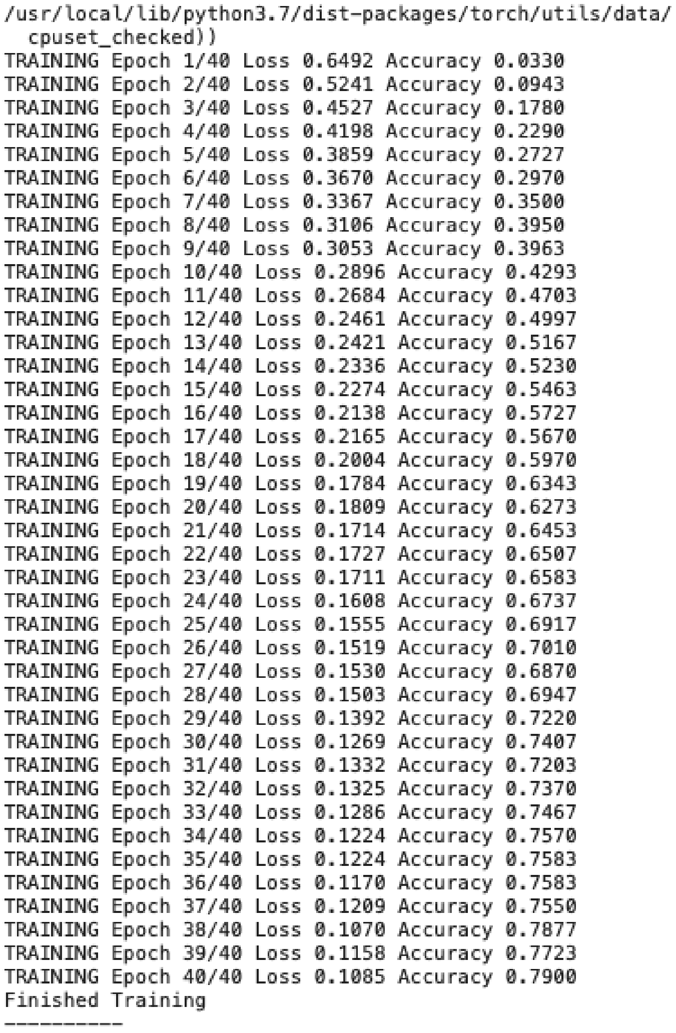
```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/datal
  cpuset_checked))
TRAINING Epoch 1/25 Loss 0.6482 Accuracy 0.0353
TRAINING Epoch 2/25 Loss 0.5137 Accuracy 0.1087
TRAINING Epoch 3/25 Loss 0.4626 Accuracy 0.1703
TRAINING Epoch 4/25 Loss 0.4191 Accuracy 0.2397
TRAINING Epoch 5/25 Loss 0.3839 Accuracy 0.2793
TRAINING Epoch 6/25 Loss 0.3747 Accuracy 0.2960
TRAINING Epoch 7/25 Loss 0.3376 Accuracy 0.3470
TRAINING Epoch 8/25 Loss 0.3179 Accuracy 0.3790
TRAINING Epoch 9/25 Loss 0.2953 Accuracy 0.4170
TRAINING Epoch 10/25 Loss 0.2818 Accuracy 0.4417
TRAINING Epoch 11/25 Loss 0.2635 Accuracy 0.4763
TRAINING Epoch 12/25 Loss 0.2587 Accuracy 0.4793
TRAINING Epoch 13/25 Loss 0.2486 Accuracy 0.5040
TRAINING Epoch 14/25 Loss 0.2457 Accuracy 0.5110
TRAINING Epoch 15/25 Loss 0.2176 Accuracy 0.5527
TRAINING Epoch 16/25 Loss 0.2088 Accuracy 0.5907
TRAINING Epoch 17/25 Loss 0.2076 Accuracy 0.5810
TRAINING Epoch 18/25 Loss 0.2035 Accuracy 0.5890
TRAINING Epoch 19/25 Loss 0.1901 Accuracy 0.6150
TRAINING Epoch 20/25 Loss 0.1828 Accuracy 0.6407
TRAINING Epoch 21/25 Loss 0.1802 Accuracy 0.6397
TRAINING Epoch 22/25 Loss 0.1732 Accuracy 0.6610
TRAINING Epoch 23/25 Loss 0.1614 Accuracy 0.6727
TRAINING Epoch 24/25 Loss 0.1555 Accuracy 0.6880
TRAINING Epoch 25/25 Loss 0.1567 Accuracy 0.6747
Finished Training
----------
```



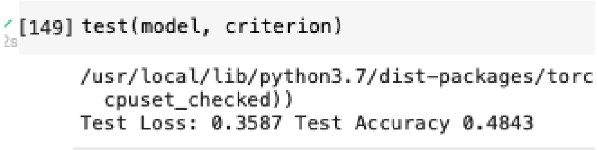
```
[145] test(model, criterion)

      /usr/local/lib/python3.7/dist-packages/torch/u
        cpuset_checked))
      Test Loss: 0.3924 Test Accuracy 0.4257
```

After changing RESNET_LAST_ONLY to false to allow the network to be fine tuned and increasing the number or epochs to 40, here are the reporting accuracies in comparison to the one above:



```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/
  cpuset_checked))
TRAINING Epoch 1/40 Loss 0.6492 Accuracy 0.0330
TRAINING Epoch 2/40 Loss 0.5241 Accuracy 0.0943
TRAINING Epoch 3/40 Loss 0.4527 Accuracy 0.1780
TRAINING Epoch 4/40 Loss 0.4198 Accuracy 0.2290
TRAINING Epoch 5/40 Loss 0.3859 Accuracy 0.2727
TRAINING Epoch 6/40 Loss 0.3670 Accuracy 0.2970
TRAINING Epoch 7/40 Loss 0.3367 Accuracy 0.3500
TRAINING Epoch 8/40 Loss 0.3106 Accuracy 0.3950
TRAINING Epoch 9/40 Loss 0.3053 Accuracy 0.3963
TRAINING Epoch 10/40 Loss 0.2896 Accuracy 0.4293
TRAINING Epoch 11/40 Loss 0.2684 Accuracy 0.4703
TRAINING Epoch 12/40 Loss 0.2461 Accuracy 0.4997
TRAINING Epoch 13/40 Loss 0.2421 Accuracy 0.5167
TRAINING Epoch 14/40 Loss 0.2336 Accuracy 0.5230
TRAINING Epoch 15/40 Loss 0.2274 Accuracy 0.5463
TRAINING Epoch 16/40 Loss 0.2138 Accuracy 0.5727
TRAINING Epoch 17/40 Loss 0.2165 Accuracy 0.5670
TRAINING Epoch 18/40 Loss 0.2004 Accuracy 0.5970
TRAINING Epoch 19/40 Loss 0.1784 Accuracy 0.6343
TRAINING Epoch 20/40 Loss 0.1809 Accuracy 0.6273
TRAINING Epoch 21/40 Loss 0.1714 Accuracy 0.6453
TRAINING Epoch 22/40 Loss 0.1727 Accuracy 0.6507
TRAINING Epoch 23/40 Loss 0.1711 Accuracy 0.6583
TRAINING Epoch 24/40 Loss 0.1608 Accuracy 0.6737
TRAINING Epoch 25/40 Loss 0.1555 Accuracy 0.6917
TRAINING Epoch 26/40 Loss 0.1519 Accuracy 0.7010
TRAINING Epoch 27/40 Loss 0.1530 Accuracy 0.6870
TRAINING Epoch 28/40 Loss 0.1503 Accuracy 0.6947
TRAINING Epoch 29/40 Loss 0.1392 Accuracy 0.7220
TRAINING Epoch 30/40 Loss 0.1269 Accuracy 0.7407
TRAINING Epoch 31/40 Loss 0.1332 Accuracy 0.7203
TRAINING Epoch 32/40 Loss 0.1325 Accuracy 0.7370
TRAINING Epoch 33/40 Loss 0.1286 Accuracy 0.7467
TRAINING Epoch 34/40 Loss 0.1224 Accuracy 0.7570
TRAINING Epoch 35/40 Loss 0.1224 Accuracy 0.7583
TRAINING Epoch 36/40 Loss 0.1170 Accuracy 0.7583
TRAINING Epoch 37/40 Loss 0.1209 Accuracy 0.7550
TRAINING Epoch 38/40 Loss 0.1070 Accuracy 0.7877
TRAINING Epoch 39/40 Loss 0.1158 Accuracy 0.7723
TRAINING Epoch 40/40 Loss 0.1085 Accuracy 0.7900
Finished Training
----------
```



```
[149] test(model, criterion)

      /usr/local/lib/python3.7/dist-packages/torc
        cpuset_checked))
      Test Loss: 0.3587 Test Accuracy 0.4843
```