Lab01 Report
Win Win Phyo (st122314)

AlexNet

In this part of Lab02 assignment, AlexNet model was implemented on the CIFAR-10 dataset with total 4 versions of model architecture which are are as follows:

1. AlexNet with Sequential API (Alexnet Sequential)
2. AlexNet with Sequential API with Local-Response Normalization (Alexnet Sequential with LRN)
3. AlexNet with nn.Module (AlexNet Module)
4. AlexNet with nn.Module with Local-Response Normalization (AlexNet Module with LRN)

All 4 sorts of AlexNet employed the exact same optimizer as well as the loss function which are as follows:

- criterion = nn.CrossEntropyLoss()
- optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

The models were trained for 10 epochs with the batch size of 4 and the performances at the 10th epoch of each model , the test accuracies and trainable parameters are as follows:

Models	*Train Acc*	*Val Acc*	*Test Acc*	*Parameters*
Alexnet Sequential	86.09%	78.16%	77.67%	58,322,314
Alexnet Sequential with LRN	87.35%	78.99%	79.70%	58,322,314
AlexNet Module	86.39%	79.04%	77.76%	57,044,810
AlexNet Module with LRN	85.38%	78.13%	77.27%	57,044,810

GoogLeNet

In this part of Lab02 assignment, GoogLeNet model was implemented on the CIFAR-10 dataset with total 2 versions of model architecture which are as follows:

1. GoogLeNet from scratch (GoogLeNet)
2. Pretrained GoogLeNet

Since the given architecture of GoogLeNet was not the same as what can be found on the original paper and also does not suit out problem (a 10-class classification problem), the modification of the architecture was necessary. The modification includes:

- The number of output (from 1000 to 10 classes)
- The input image size
- The addition of a convolutional layer at pre_layers
- The padding
- Replacement of BatchNorm2d to Local-Response Normalization
- The addition of the two auxiliary layers
- The losses of auxiliary layers etc.,

Two versions of GoogLeNet employed the exact same optimizer as well as the loss function which are as follows:

- criterion = nn.CrossEntropyLoss ()
- optimizer = optim.SGD (model.parameters (), lr =0.001, momentum=0.9)

The models were trained for 10 epochs with the batch size of 4 and the performances at the 10th epoch of each model , the test accuracies and trainable parameters are as follows:

Models	*Train Acc*	*Val Acc*	*Test Acc*	*Parameters*
GoogLeNet	91.58%	96.62%	86.77%	10,635,134
Pretrained GoogLeNet	97.98%	99.40%	93.60%	13,004,888

Discussion Session

The following screen shots shows that my Python classes, with one class per file and a main module that sets up my objects, runs the training process, and saves the necessary data.

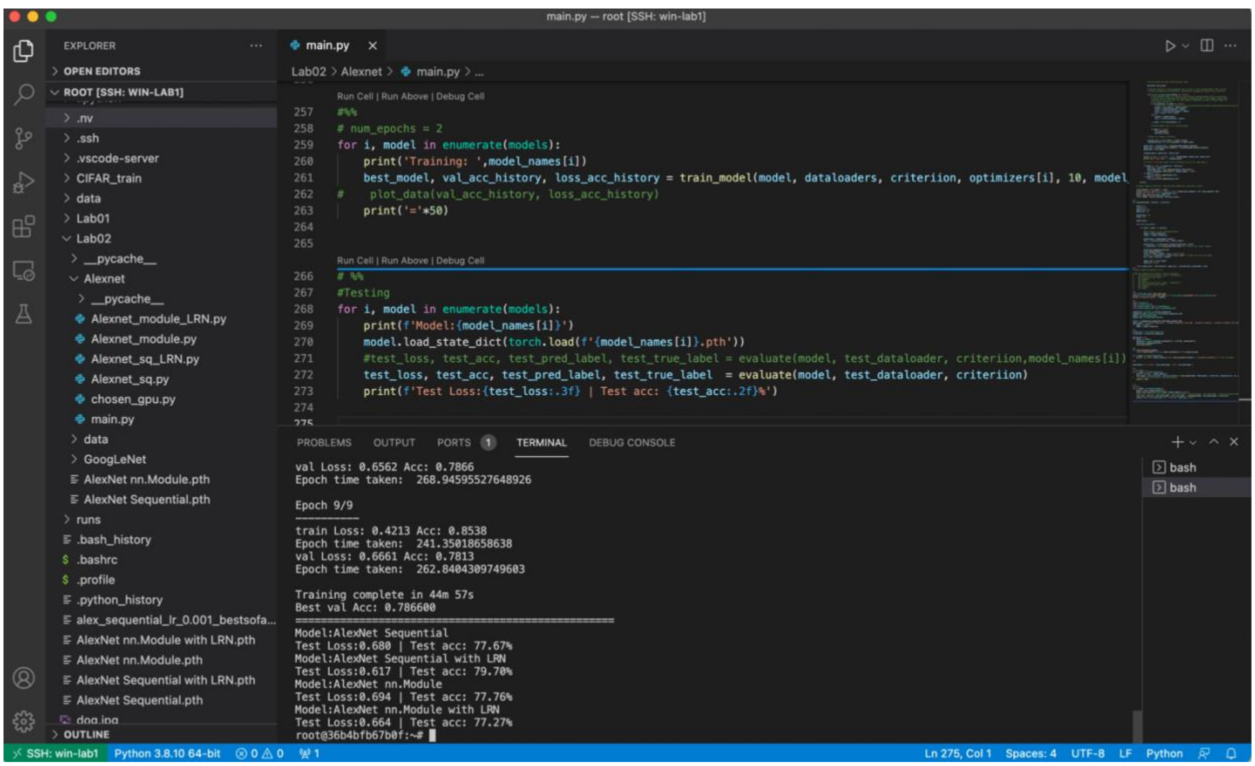

Fig 1: AlexNet Models

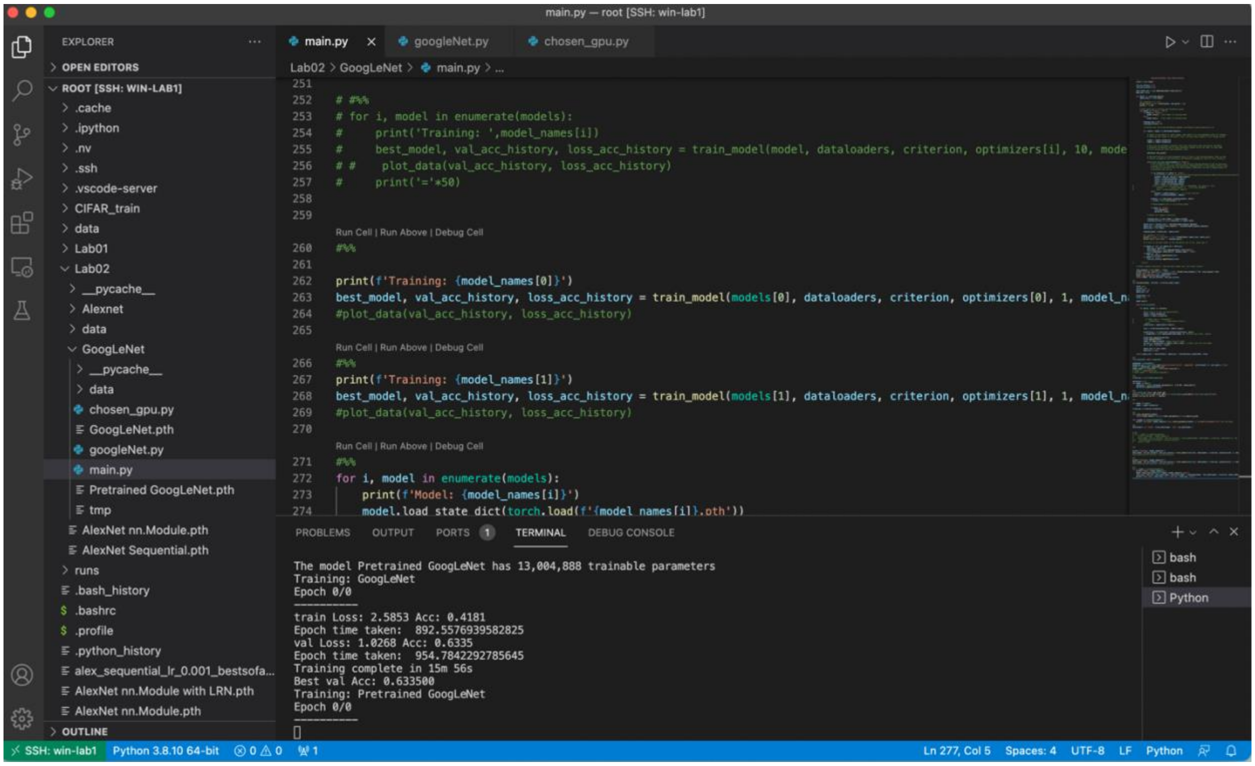

Fig 2: GoogLeNet Model

After comparing the final results of AlexNet Sequential model which is mentioned under AlexNet section and those of AlexNet Sequential with LRN model, AlexNet Sequential with LRN model (79.70%)outperforms AlexNet Sequential without LRN model (77.67%) by around 2%. In addition, the results also align to what is described in the paper. Moreover, the plots of AlexNet with LRN implemented are also smoother. Over and above that, AlexNet Module with LRN exceeds by % than simple AlexNet Module.

From the implementation under GoogLeNet section.

- AlexNet Sequential with LRN:
  Train acc = 87.35%,
  Val acc = 78.99%,
  Test acc = 79.70%,
  Number of trainable parameters = 58,322,314

Note: there are 4 different versions of AlexNet model experimented in this lab, however the best performing AlexNet model version was selected for this section.

- GoogLeNet:
  Train acc = 91.58%
  Val acc = 96.62%
  Test acc = 86.77%
  Number of trainable parameters = 10,635,134

As shown above GoogLeNet could achieve a considerably higher accuracies while having the number of trainable parameters lower. However, GoogLeNet seems to require a lot more training time and takes more time to converge when compared to AlexNet.

Comparison upon the experiment with the pretrained GoogLeNet and AlexNet Comment on what we can glean from the results about the capacity and generalization ability of these two models.

- Pretrained AlexNet:
  Train acc = 97.63%,
  Val acc = 89.22%,
  Test acc = 88.18%,
  Number of trainable parameters = 44,428,106

- Pretrained GoogLeNet:
  Train acc = 97.98%
  Val acc = 99.40%
  Test acc = 93.60%
  Number of trainable parameters = 13,004,888

The pretrained version of both models perform better than that of the from-scratch version. Both versions of GoogLeNet achieve higher accuracies on CIFAR-10 while having less the number of trainable parameters.