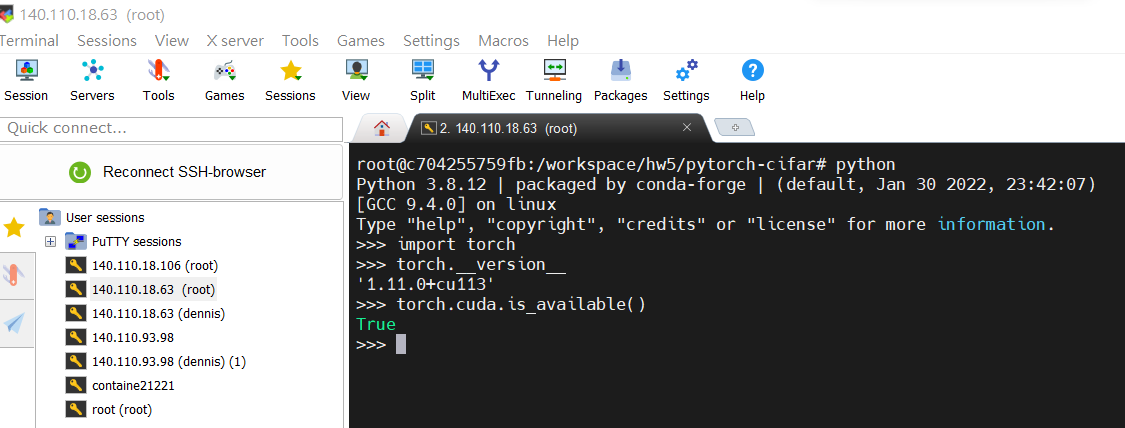
Student ID: 410551027

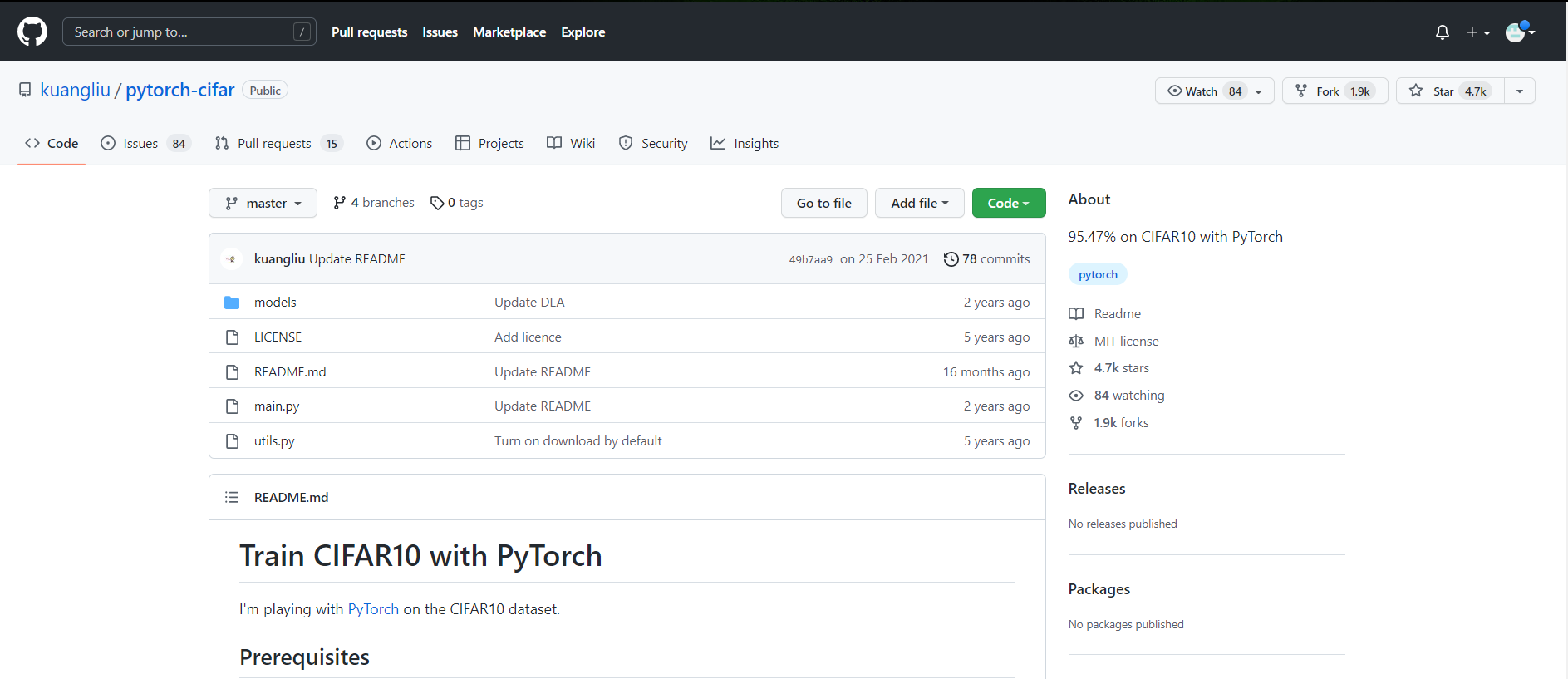
Student name: 江衍涵

1. I am using a high-speed computing computer, system Information as follows:

|  |
| --- |
| Hardware details |
| OS info: |
|  |
| CPU info: |
|  |
| GPU info: |
|  |

1. I am using pytorch-cifar(repository) on github, which is a paper called “Deep Layer Aggregation”published in the CVPR journal. I use this repository to write homework 5 assigments. It’s deep learning framework is pytorch, and version is as follows. My environment is running in Docker, and its tag is “cyh1195/cyh:1110612cs\_hw5”. The file path is in “/workspace/code”.
   * + Python 3.8.12
     + PyTorch 1.11+cu113
     + https://github.com/kuangliu/pytorch-cifar
     + docker tag: docker pull cyh1195/cyh:1110612cs\_hw5





1. **How to use the program?**

|  |
| --- |
| **# Start training with:**  **python train.py**  **# You can manually resume the training with:**  **python train.py –resume**  **# Using Model Inference:**  **python inference.py** |

1. **Model desinged:**

I use the model structure of this paper, and his model hyperparameter settings as in the Table 1. Data pre-processing is done only for normalization, and no other actions are done. This deep layer aggregation structures iteratively and hierarchically merge the feature hierarchy to make networks with better accuracy and fewer parameters. Aggregation is a decisive aspect of architecture as Fig 1, and as the number of modules multiply their connectivity is made all the more important. By relating architectures for aggregating channels, scales, and resolutions we identified the need for deeper aggregation, and addressed it by iterative deep aggregation and hierarchical deep aggregation.

Table 1. Hyperparameter settings

|  |  |
| --- | --- |
| model architecture | SimpleDLA |
| optimizer | SGD |
| learning rate | 0.0001 |
| momentum | 0.9 |
| weight\_decay | 0.0005 |
| batch size | 256 |
| epoch | 5000 |
| criterion(loss function) | CrossEntropyLoss |
| scheduler | CosineAnnealingLR |

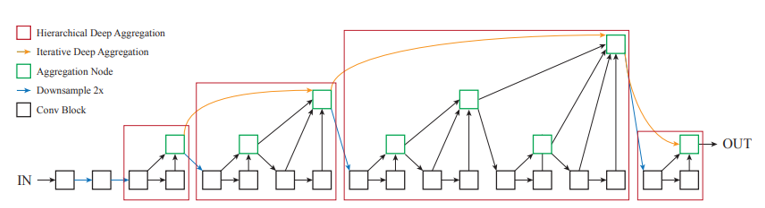


Figure 1: Deep layer aggregation

1. **Result:**

First, I trained the SimpleDLA model 195 times using training.py and the original cifar-10, the training process is shown in Figure 2. After training, the weights are saved in the “ckpt\_1000.pth” file as a pre-trained model. Then use the x\_train.npy dataset from Homework 5, and train it again with the same hyperparameter settings. Finally, the test.npy dataset is tested with inference.py, and the average accuracy of the test.npy dataset is 99%.

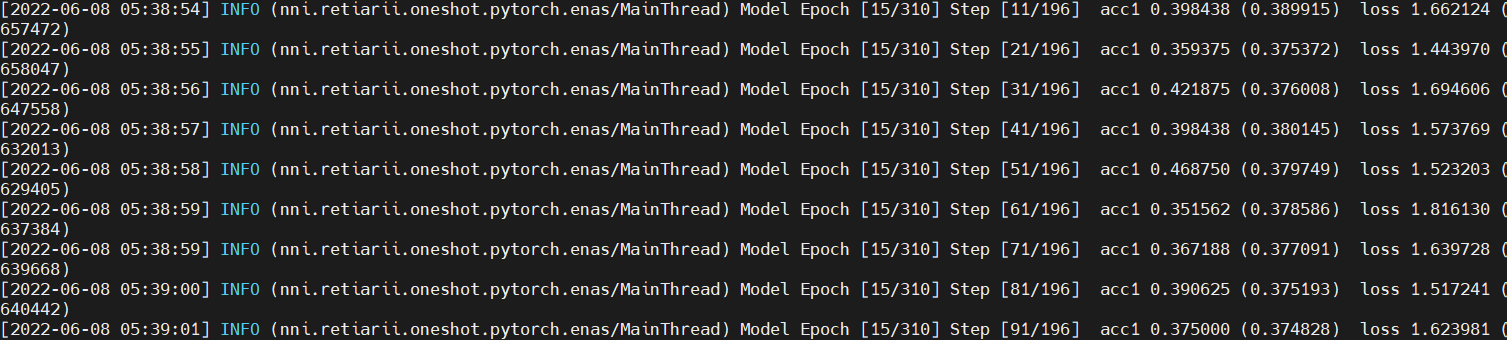
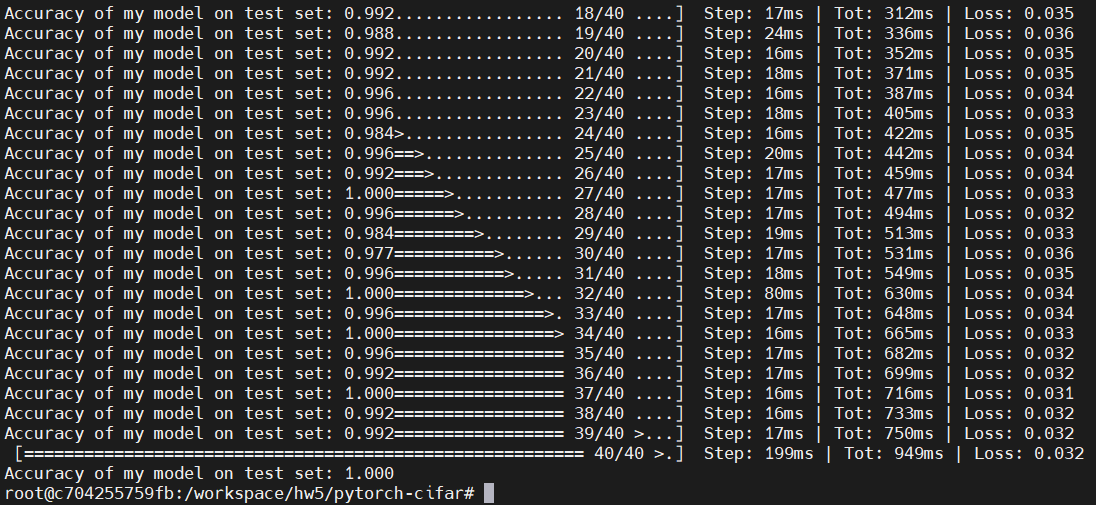


Figure 2: History of traing model

Figure 3: The inferend result from testing data is an accuracy of 99%