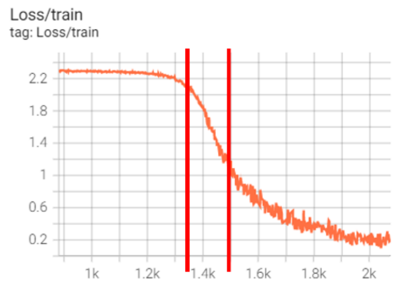
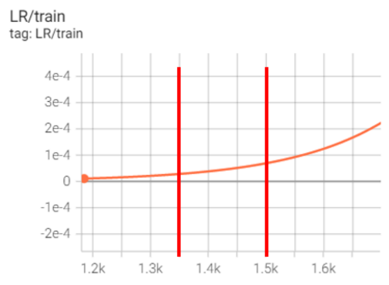
CS 5260 Assignment 6

Jianxin Bi, A0219018J

https://github.com/bijianxin292430887/cs5260_assignment6

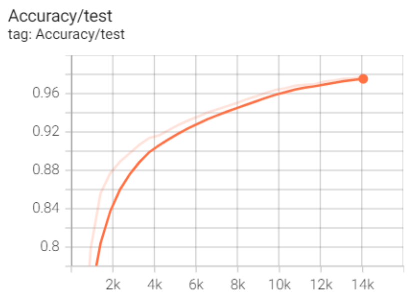
1. Optimizer: Adam(lr=0.001, betas = (0.9,0.999), eps=1e-08, weight_decay=0)
2. Lr_scheduler:
    a. LambdaLR
    b. ExponentialLR
3. LR range test for Adam optimizer: lr should take within [2.1e-5,6.8e-5], where the loss decrease the fastest. Test plots with LR region highlighted with red lines are shown below:

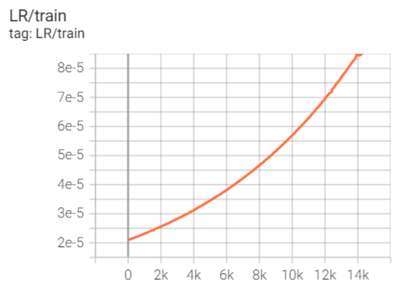


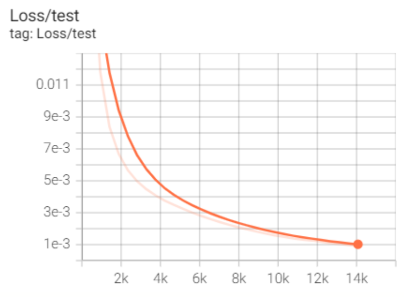


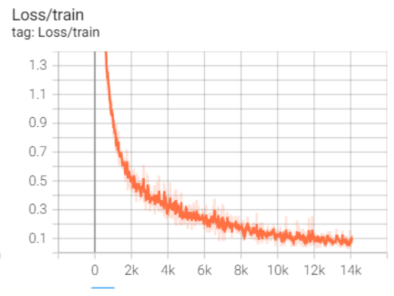
Training with Adam + ExponentialLR:

```
1. lr_scheduler = torch.optim.lr_scheduler.ExponentialLR(optimi
   zer, gamma=1.0001)
2. optimizer = torch.optim.Adam(model.parameters(), lr=2.1e-
   5, betas=(0.9, 0.999), eps=1e-08, weight_decay=0)
3. Achieve 97.7% accuracy with default LeNet5 hyperparameters.
   Batchsize = 128, Epochs = 30.
```



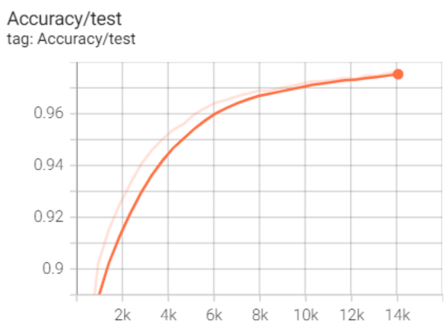


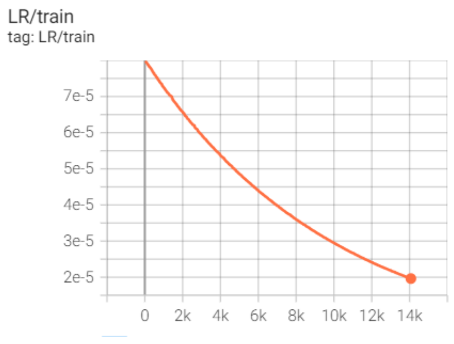




```
   If we make the learning rate decrease along training, where
1. lr_scheduler = torch.optim.lr_scheduler.ExponentialLR(optimi
   zer, gamma=0.9999)
```

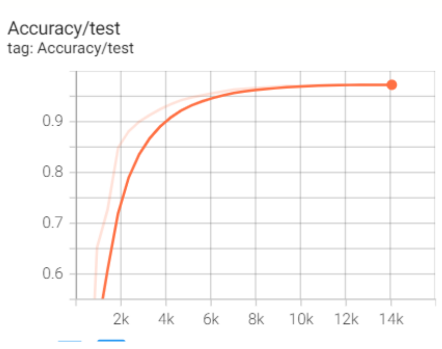
2. optimizer = torch.optim.Adam(model.parameters(), lr=8e-5, betas=(0.9, 0.999), eps=1e-08, weight_decay=0)
3. We obtain test accuracy **97.6%** surprisingly.

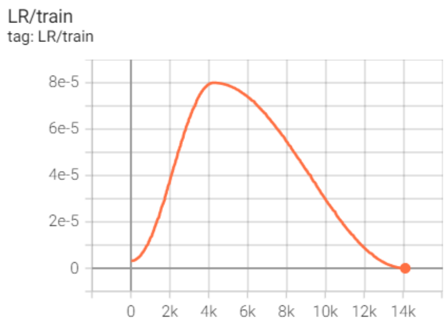


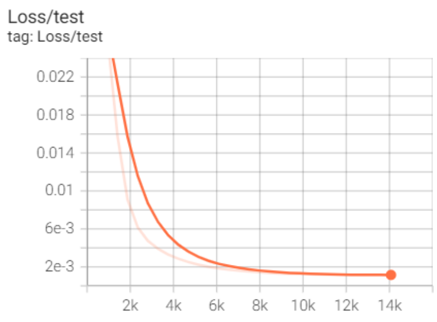


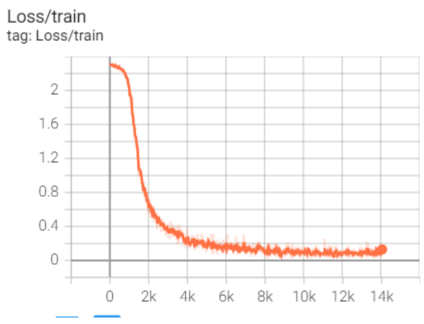
Training with Adam + OneCycleLR:

1. optimizer = torch.optim.Adam(model.parameters(), lr=8e-5, betas=(0.9, 0.999), eps=1e-08, weight_decay=0)
2. lr_scheduler = torch.optim.lr_scheduler.OneCycleLR(optimizer, max_lr=8e-5, steps_per_epoch = len(train_dataloader), epochs =30)
3. Achieve **97.26%** accuracy with default LeNet5 hyperparameters. Batchsize = 128, Epochs = 30.


Accuracy/test
tag: Accuracy/test
2k
4k
6k
8k
10k
12k
14k


LR/train
tag: LR/train
8e-5
6e-5
4e-5
2e-5
2k
4k
6k
8k
10k
12k
14k


Loss/test
tag: Loss/test
6e-3
2e-3
2k
4k
6k
8k
10k
12k
14k


Loss/train
tag: Loss/train
2k
4k
6k
8k
10k
12k
14k