📌

CS5260 Assignment 6

ZHENG YINJIE, A0228498L

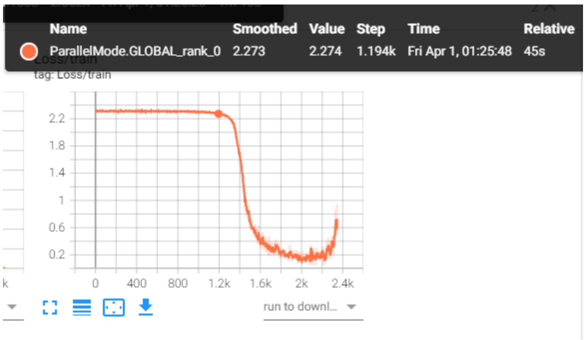
Link to the GitHub

https://github.com/Yinjie-ZHENG/YINJIE-Assignment-6

Optimizer: SGD

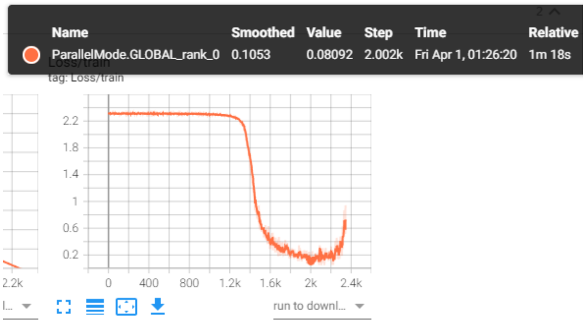
```
# optimizer
optimizer = torch.optim.SGD(model.parameters(), lr=0.06, momentum=0.9, weight_decay=5e-4)
```

Learning rate range test:



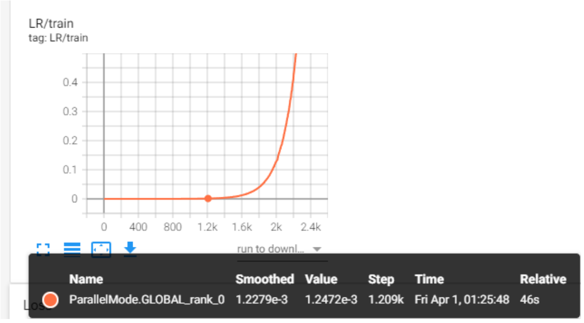
Name	Smoothed	Value	Step	Time	Relative
● ParallelMode.GLOBAL_rank_0	2.273	2.274	1.194k	Fri Apr 1, 01:25:48	45s

LR range test - Loss1



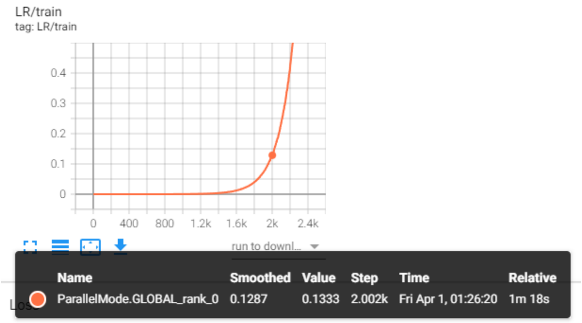
Name	Smoothed	Value	Step	Time	Relative
● ParallelMode.GLOBAL_rank_0	0.1053	0.08092	2.002k	Fri Apr 1, 01:26:20	1m 18s

LR range test - Loss2



Name	Smoothed	Value	Step	Time	Relative
● ParallelMode.GLOBAL_rank_0	1.2279e-3	1.2472e-3	1.209k	Fri Apr 1, 01:25:48	46s

LR range test - lower LR



Name	Smoothed	Value	Step	Time	Relative
● ParallelMode.GLOBAL_rank_0	0.1287	0.1333	2.002k	Fri Apr 1, 01:26:20	1m 18s

LR range test - upper LR

Here for the SGD optimizer, it can be easily found that the learning rate range should be around step 1.2K and 2K. That is, the LR range test told me the **LR should be around 1.2e-3 to 0.14**.

Learning rate scheduling method: Multistep, OneCycle

1. Multistep

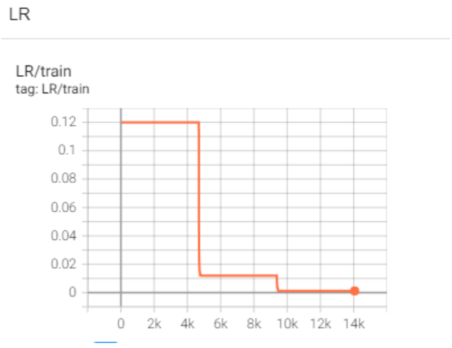
   Since the above LR range test showed us that LR should be around 1.2e-3 to 0.14, here I set the LR scheduler as:

```
lr_scheduler  = torch.optim.lr_scheduler.MultiStepLR(optimizer, milestones=[10*len(train_dataloader),20*len(train_dataloader)], ga
```

(Total epoch is set to be 30), so I set the step of changing LR when it comes to epochs 10 and 20.

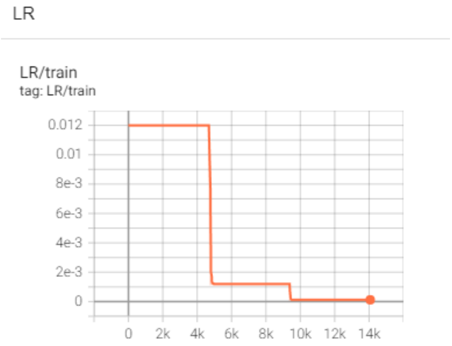
Here are detailed descriptions of my learning rate scheduling method:

a. LR=0.12 if Epoch < 10; LR=0.012 if 10≤Epoch<20; LR=1.2e-3 if Epoch≥30



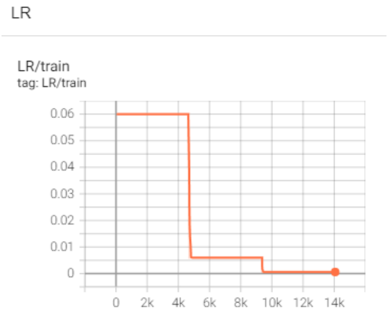
LR for Multistep, strategy a

b. LR=0.012 if Epoch < 10; LR=1.2e-3 if 10≤Epoch<20; LR=1.2e-4 if Epoch≥30



LR for Multistep, strategy b

c. LR=0.06 if Epoch < 10; LR=6e-3 if 10≤Epoch<20; LR=6e-4 if Epoch≥30

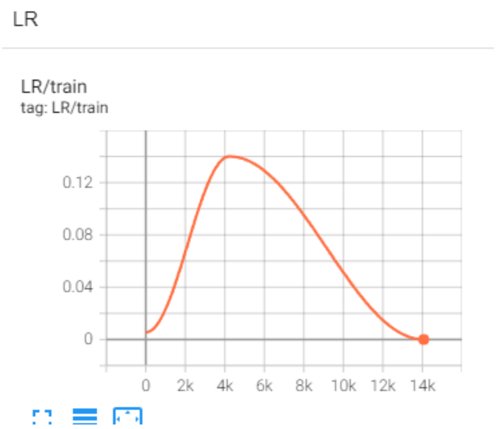


LR for Multistep, strategy c

2. OneCycle

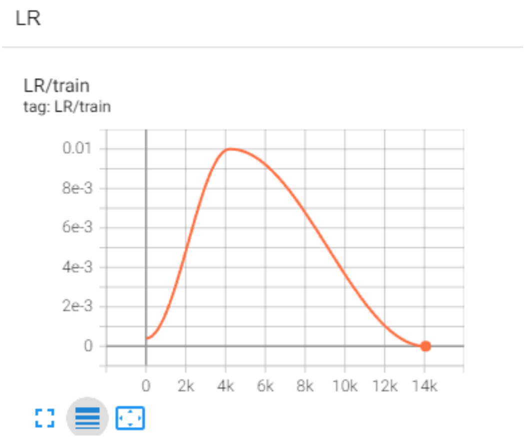
Sets the learning rate of each parameter group according to the 1cycle learning rate policy. Here are detailed descriptions of my learning rate scheduling method:

a. **max_lr = 0.14**



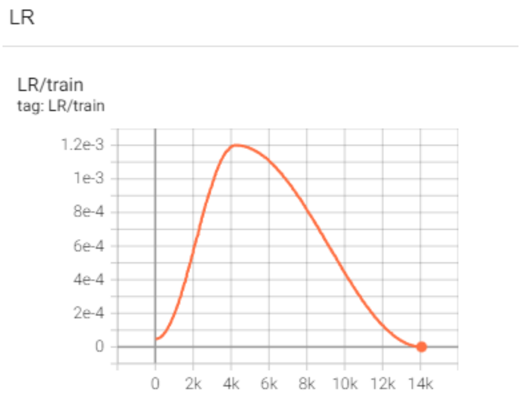
LR for OneCycle, strategy a

b. **max_lr = 0.01**



LR for OneCycle, strategy b

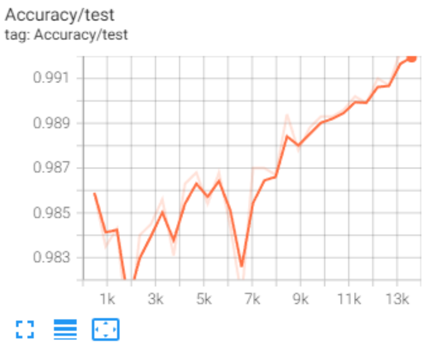
c. **max_lr = 1.2e-3**



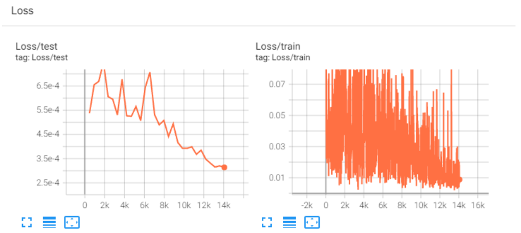
LR for OneCycle, strategy c

Result - OneCycle

LR for OneCycle, strategy a

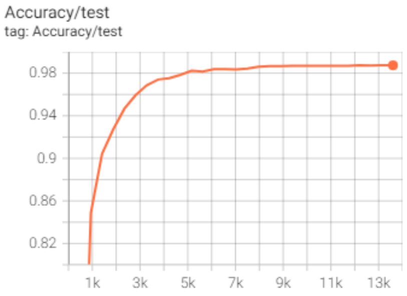


ACC for OneCycle, strategy a

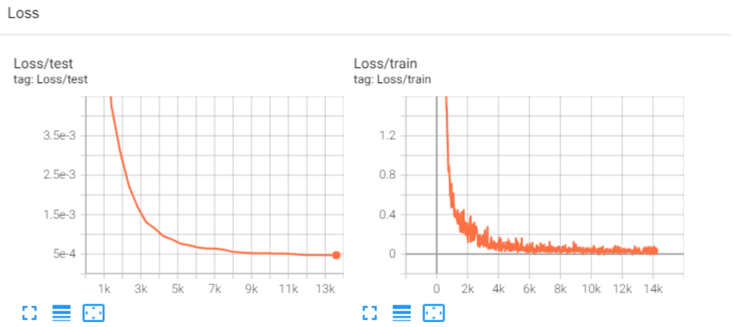


Loss for OneCycle, strategy a

LR for OneCycle, strategy b

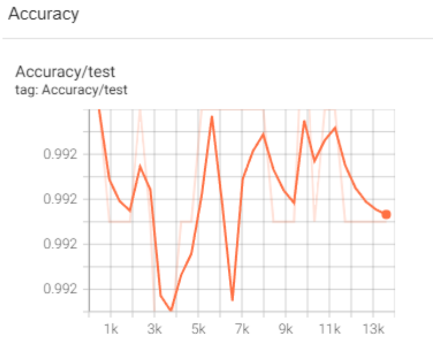


ACC for OneCycle, strategy b

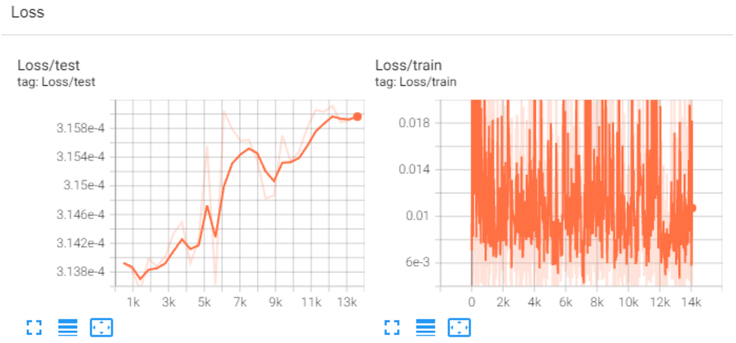


Loss for OneCycle, strategy b

LR for OneCycle, strategy c



ACC for OneCycle, strategy c



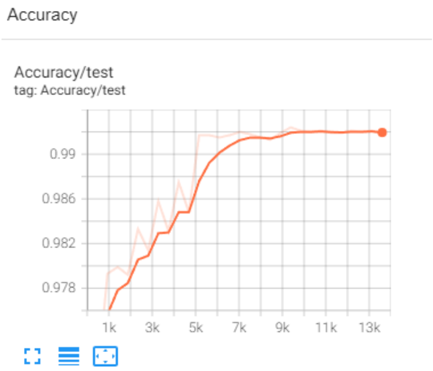
Loss for OneCycle, strategy c

Best LR- SGD+OneCycle

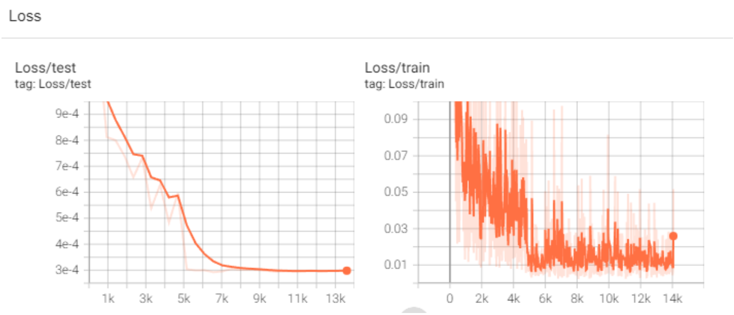
Based on the above results, it is concluded that the best (maximum) LR region should be around 0.01. Because it can be seen that strategy a(LR=0.14) cannot converge; loss of strategy c(LR=**1.2e-3**) even still grows when training.

Result - Multistep

Multistep, strategy a

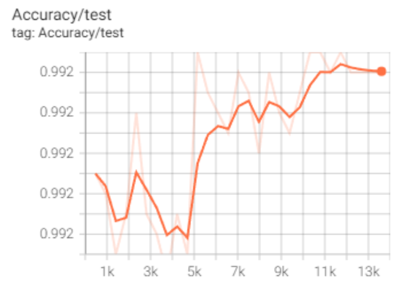


ACC for Multistep, strategy a

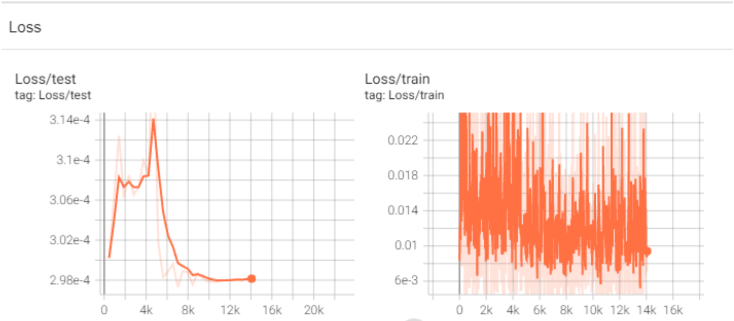


Loss for Multistep, strategy a

Multistep, strategy b

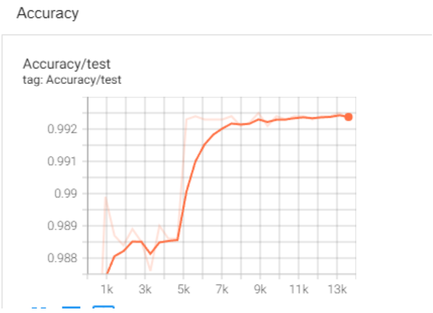


Acc for Multistep, strategy b

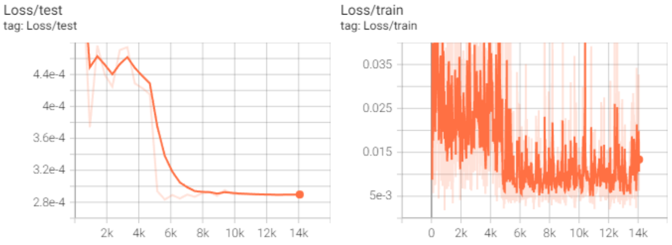


Loss for Multistep, strategy b

Multistep, strategy c



Acc for Multistep, strategy c



Loss for Multistep, strategy c

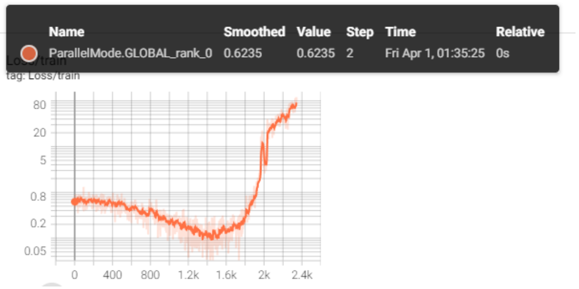
Best LR- SGD+Multistep

Based on the above results, it is concluded that the best initial LR region should be around 0.06 to 0.12. Because it can be seen that strategy b(initial LR=0.012) cannot converge, but strategies a(initial LR=0.12) and c (initial LR=0.06) can converge.

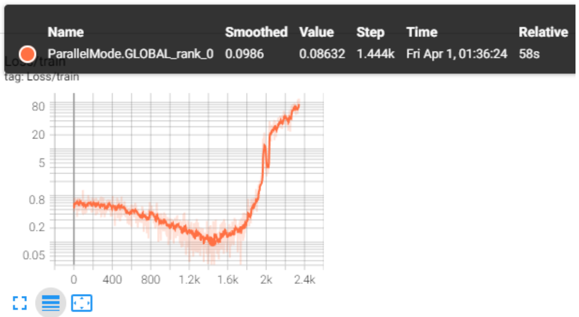
Further exploration

1. SGD+OneCycle: I tried to set the initial LR for SGD to 100 or even larger LR, but in PyTorch, the learning rate is assigned by lr_scheduler(parameter “max_lr” in OneCycleLR.

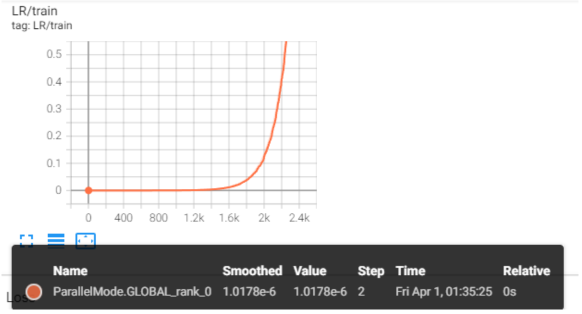
2. ADAM LR range test: The LR range test plot for ADAM is quite different from SGD. It drops steadily at first and explodes after a certain epoch(LR). Therefore, I think for the ADAM, the LR range test may just tell us a smaller range, which means maybe when real training, I can try a larger LR.



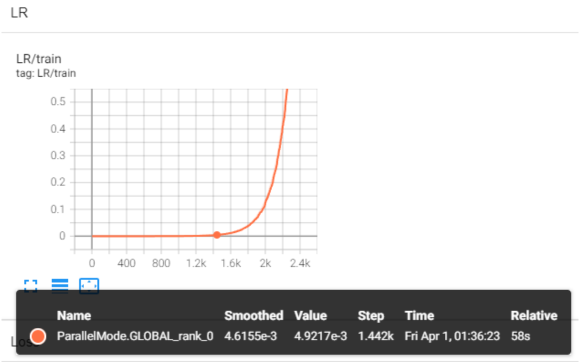
	Name	Smoothed	Value	Step	Time	Relative
●	ParallelMode.GLOBAL_rank_0	0.6235	0.6235	2	Fri Apr 1, 01:35:25	0s



	Name	Smoothed	Value	Step	Time	Relative
●	ParallelMode.GLOBAL_rank_0	0.0986	0.08632	1.444k	Fri Apr 1, 01:36:24	58s



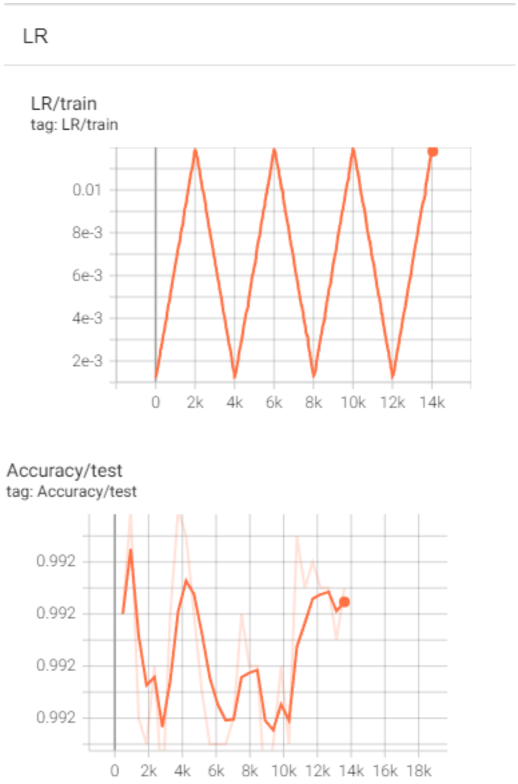
	Name	Smoothed	Value	Step	Time	Relative
●	ParallelMode.GLOBAL_rank_0	1.0178e-6	1.0178e-6	2	Fri Apr 1, 01:35:25	0s



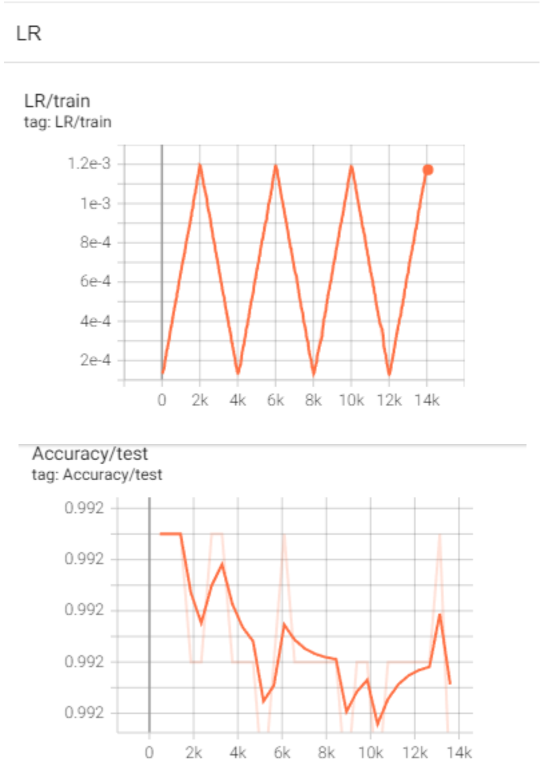
	Name	Smoothed	Value	Step	Time	Relative
●	ParallelMode.GLOBAL_rank_0	4.6155e-3	4.9217e-3	1.442k	Fri Apr 1, 01:36:23	58s

3.SGD+CYCLE: I tried to explore CYCLE scheduler but colab GPU resource are limited. Here’s my finding:

- LR range from 1.2e-3 to 1.2e-2



- LR range from 1.2e-4 to 1.2e-3


LR
LR/train
tag: LR/train
1.2e-3
1e-3
8e-4
6e-4
4e-4
2e-4
2k
4k
6k
8k
10k
12k
14k
Accuracy/test
tag: Accuracy/test
2k
4k
6k
8k
10k
12k
14k