CS5260 Assignment 6 Report

1. Introduction

In this report, we described the findings from our experiments with the Colossal AI package to train LeNet5 on MNIST. The corresponding repository for our codes is at our Github Page.

We experimented with various combinations architectures as follows:

- Optimizers: SGD, AdamW
- Schedulers: Lambda, MultiStep

2. Methodology

We adapted the Colossal AI notebook provided by the TA for each of the following task:

A. LR Range Test

See 'Colossalai LR Range Tests.ipynb'

First, we conducted the learning rate (LR) range test for each scheduler. We observed performances over 10 epochs, with batch size at 128. which over our dataset, gives us approximately 4.7K training steps. For each optimizer x scheduler, we then used the proposed LR and two controls (LR/10 and LR*10) to observe performance differences.

For each scheduler, we amended the code such that it returns increasing LRs over epochs:

- Lambda: Custom function to exponentially increase learning rate from low to high
- MultiStep: Per half epoch, LR increases by gamma=1.5

Refer to Figure 1(i) and Figure 2(i) to observe how the LRs change across training steps for each setup.

B. Final Training

See 'run.py'

The model is trained for 30 epochs as recommended by the assignment description. Batch size was fixed at 128. Learning rate was set based on LR Range Tests (See Results Section A later for outcomes). No scheduler was used, i.e. LR was fixed throughout training.

3. Results

A. LR Range Test

To conduct the LR range test, we have to compare the learning rate versus test loss. The TensorBoard allows us to do these comparisons with two plots. Subsequently, we will choose the learning rate one order lower than the learning rate where loss is minimum as the optimal learning rate value.

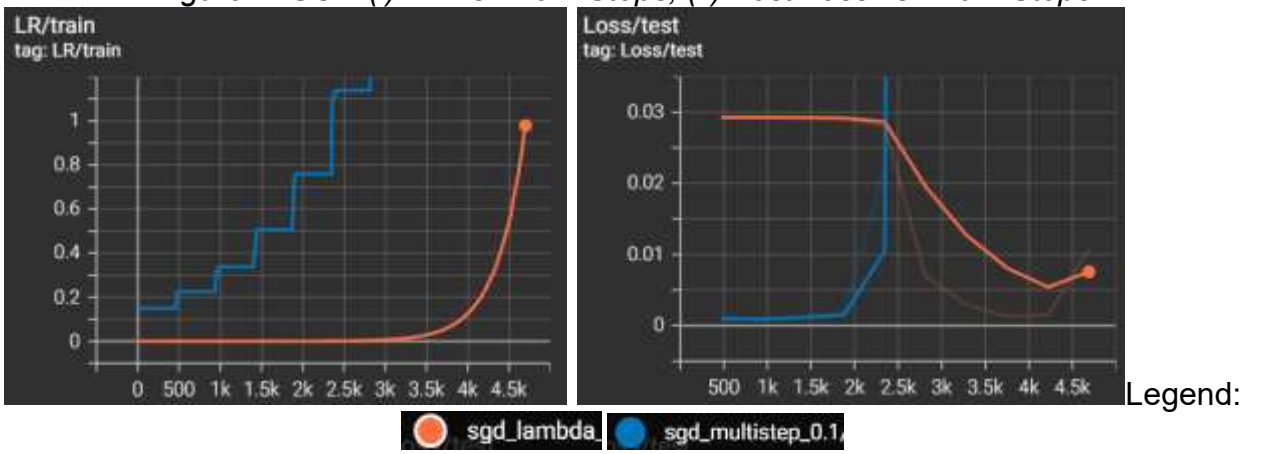
Table 1. Optimal LR per Setup based on LR Range Test

Optimizer	Scheduler	Optimal LR
SGD	Lambda	0.025
SGD	MultiStep	0.022
AdamW	Lambda	0.001
AdamW	MultiStep	0.015*

* Indicates no optimal LR was found since no true minimum loss point was detected. See below in (ii) for further details.

i. SGD:

Figure 1. SGD (i) LR vs. Train Steps; (ii) Test Loss vs. Train Steps

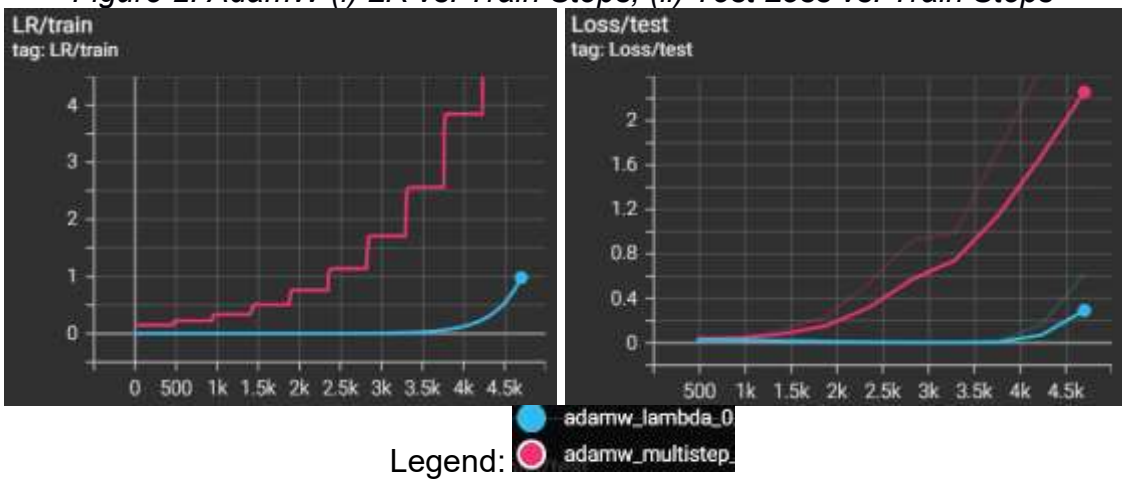


For the following schedulers, we observed minimum loss at the following points:

- Lambda: Min Loss @ 4.22K step, equivalent to around LR=0.248
- MultiStep: Min Loss @ 1K step, equivalent to around LR=0.225

ii. AdamW:

Figure 2. AdamW (i) LR vs. Train Steps; (ii) Test Loss vs. Train Steps



For the following schedulers, we observed minimum loss at the following points:

- Lambda: Min Loss @ 3.283K step, equivalent to around LR=0.01642
- MultiStep: Min Loss @ 0K step, equivalent to around LR=0.15

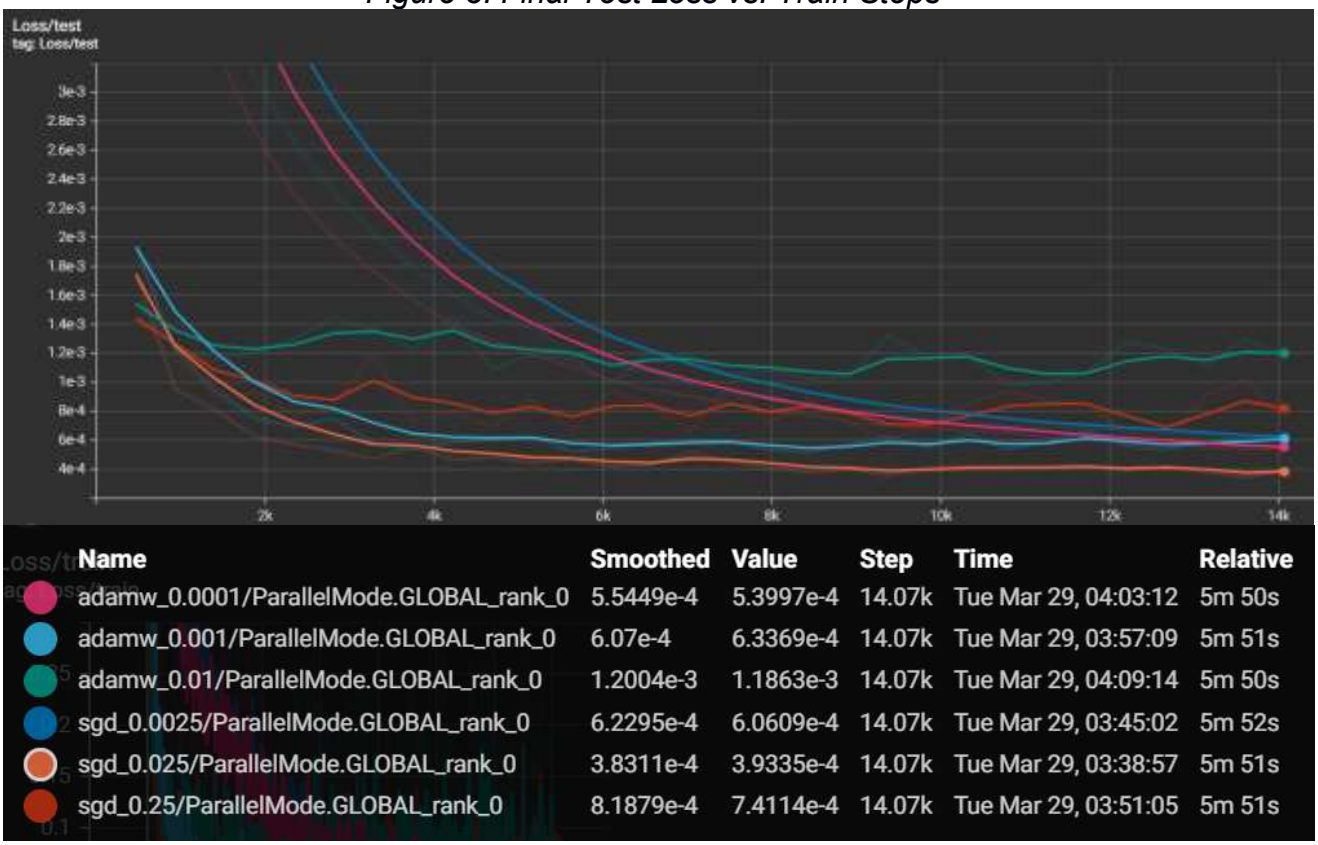
MultiStep does not have a true minimum point. On hindsight, this could be because the starting LR was too high in the first place. In fact, the minimum point for Lambda is also very gradual. We should have tried lower initial LRs for this Test.

B. Final Training

We use the LR recommended by Lambda scheduler, namely for SGD to be 0.025, and AdamW to be 0.001. We used

i. Overall Findings

Figure 3. Final Test Loss vs. Train Steps



Name	Smoothed	Value	Step	Time	Relative
adamw_0.0001/ParallelMode.GLOBAL_rank_0	5.5449e-4	5.3997e-4	14.07k	Tue Mar 29, 04:03:12	5m 50s
adamw_0.001/ParallelMode.GLOBAL_rank_0	6.07e-4	6.3369e-4	14.07k	Tue Mar 29, 03:57:09	5m 51s
adamw_0.01/ParallelMode.GLOBAL_rank_0	1.2004e-3	1.1863e-3	14.07k	Tue Mar 29, 04:09:14	5m 50s
sgd_0.0025/ParallelMode.GLOBAL_rank_0	6.2295e-4	6.0609e-4	14.07k	Tue Mar 29, 03:45:02	5m 52s
sgd_0.025/ParallelMode.GLOBAL_rank_0	3.8311e-4	3.9335e-4	14.07k	Tue Mar 29, 03:38:57	5m 51s
sgd_0.25/ParallelMode.GLOBAL_rank_0	8.1879e-4	7.4114e-4	14.07k	Tue Mar 29, 03:51:05	5m 51s

At the final step of training, we note that the best model with lowest loss is the SGD x LR=0.025 combination with loss value of 3.9335e-4.

ii. LR Check

We experimented with three learning rates of varying magnitudes, LR/10<LR<LR*10. As shown in Figure 4, we noticed that the LR Range Test was reliable for SGD, but not for AdamW optimizer. For AdamW, the lower the learning rate, the better the performance (lower the loss). Again, this could be because we did not explore low enough LRs in LR Range Test, discussed earlier in Section 3.A.ii

Figure 4. Comparing Learning Rates

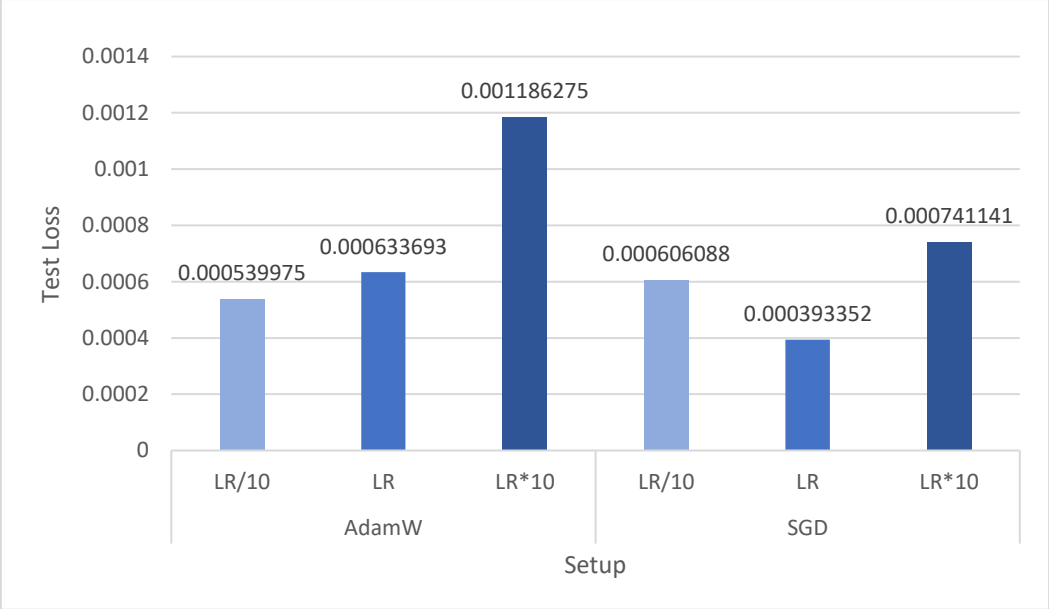


Figure 5. Final Test Loss vs. Train Steps (i) SGD; (ii) AdamW

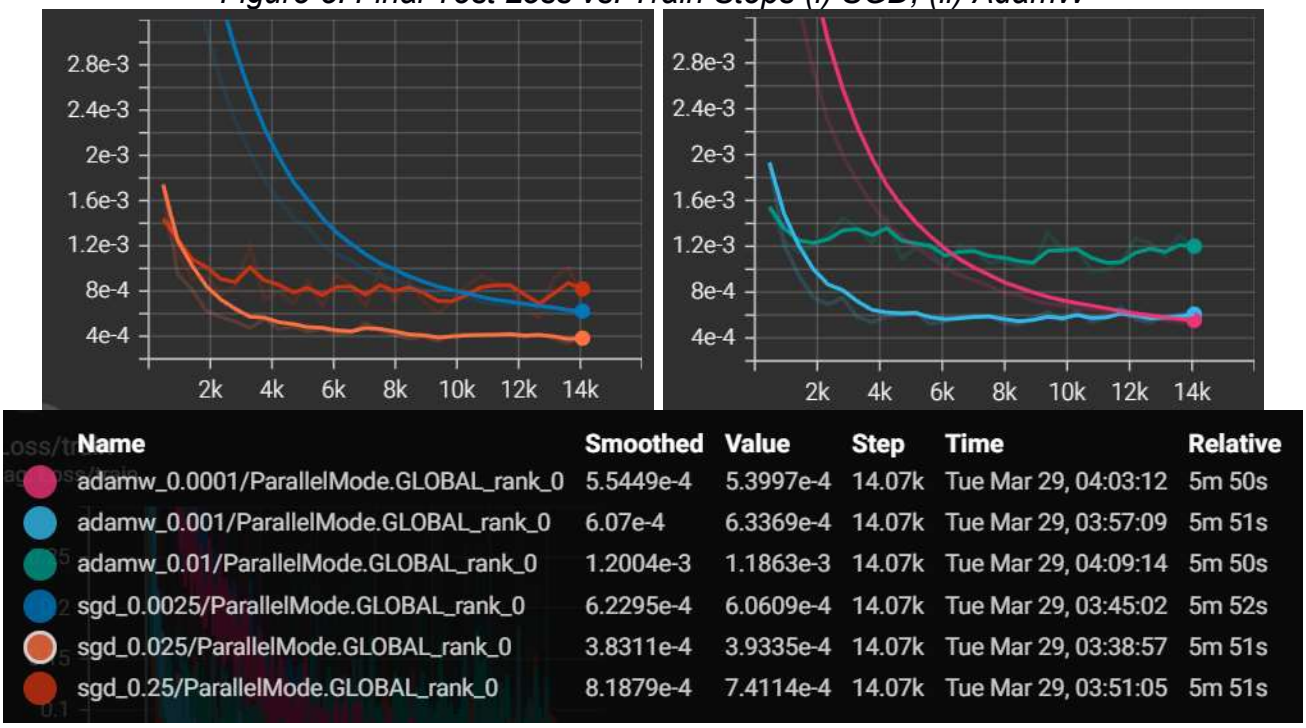


Figure 5 is equivalent to Figure 3, but converted into two plots for SGD and AdamW optimizer respectively for easier comparisons.

We noticed that for both SGD and AdamW, the lowest LR (LR/10) leads to a smoother, steeper decline in loss. This makes sense because the LR is small, so for earlier steps, learning is slow (i.e. loss is higher).

When LR is the highest (LR*10), in the initial steps, it has comparatively low loss, but by the end of training at 14K th step, the decrease in loss is marginal. This makes sense because LR is high so divergent behaviour occurs around suboptimal minima.

Conclusions

To pick a LR, the LR Range Test is reliable if a clear minimum can be found in the test loss plot as per our SGD case. Sometimes, we might need to explore a low enough initial LR, as we have found in our AdamW case. Our best LeNet model used an SGD optimizer with LR=0.025, which after 30 epochs, obtained test loss value of 3.9335e-4.

Notes

All raw findings can be found in ‘results.xlsx’ files in the repository. They are the consolidated CSV files downloaded from Tensorboard.