CS5260
Assignment 6

Prateek Manocha
A0228523H

GitHub: https://github.com/prateek-manocha/CS5260_Assignment6

Introduction

This report describes the experiments and the findings obtained from running experiments with The Colossal AI package to train LeNet5 on MNIST dataset.

First Learning Rate test is conducted which helps identify a suitable learning rate for training the network, given the optimizer we intend to use. Using the proposed learning rate, total 12 models were trained with different set of initial learning rate and learning rate scheduler. All the experiments were trained on same dataset and for 30 epochs, using Adam optimizer.

Summary of all models trained:

Optimizer	Learning Rate Scheduler	Learning Rate 1	Learning Rate 2	Learning Rate 3
ADAM	None	0.06	6e-5	6e-6
	Step LR	0.6	6e-4	6e-6
	Multi Step LR	0.6	6e-4	6e-6

Learning Rate Range Test

After running the learning rate test for 5 epochs, following graphs were obtained as shown in figure 1.

From the training loss curve, we can see that the loss remains fixed for first 600-650 steps as expected, after which it continuous to decrease for next 350-400 steps, and then finally it explodes. As suggested by the assignment guide, as well as the reference paper, the descending speed of train loss is maximum at around 700 steps, so we can use the corresponding learning rate of 6.05e-5 as our proposed learning rate. The very small value of proposed learning rate also hints that the Adam optimizer is the correct choice and will help model converge faster. Figure 2 shows the order of magnitude for the proposed learning rate to be used with Adam optimizer.

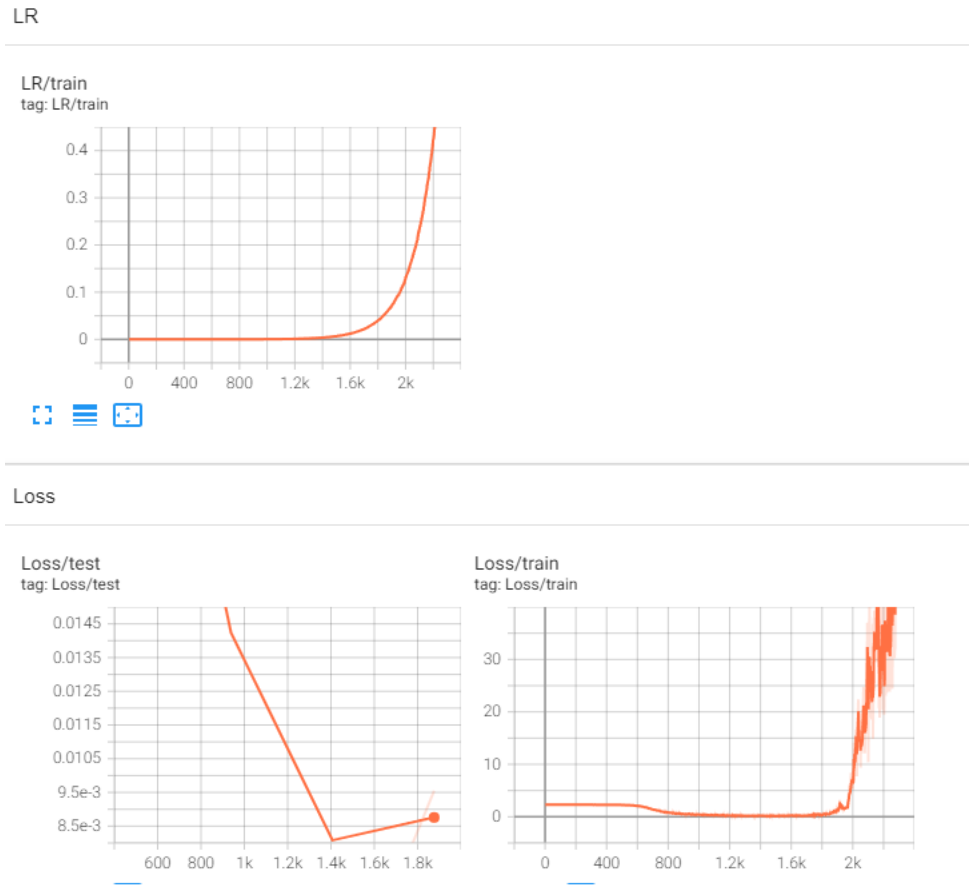


Figure 1: Learning rate vs train step (top) and Loss vs train step (bottom right).

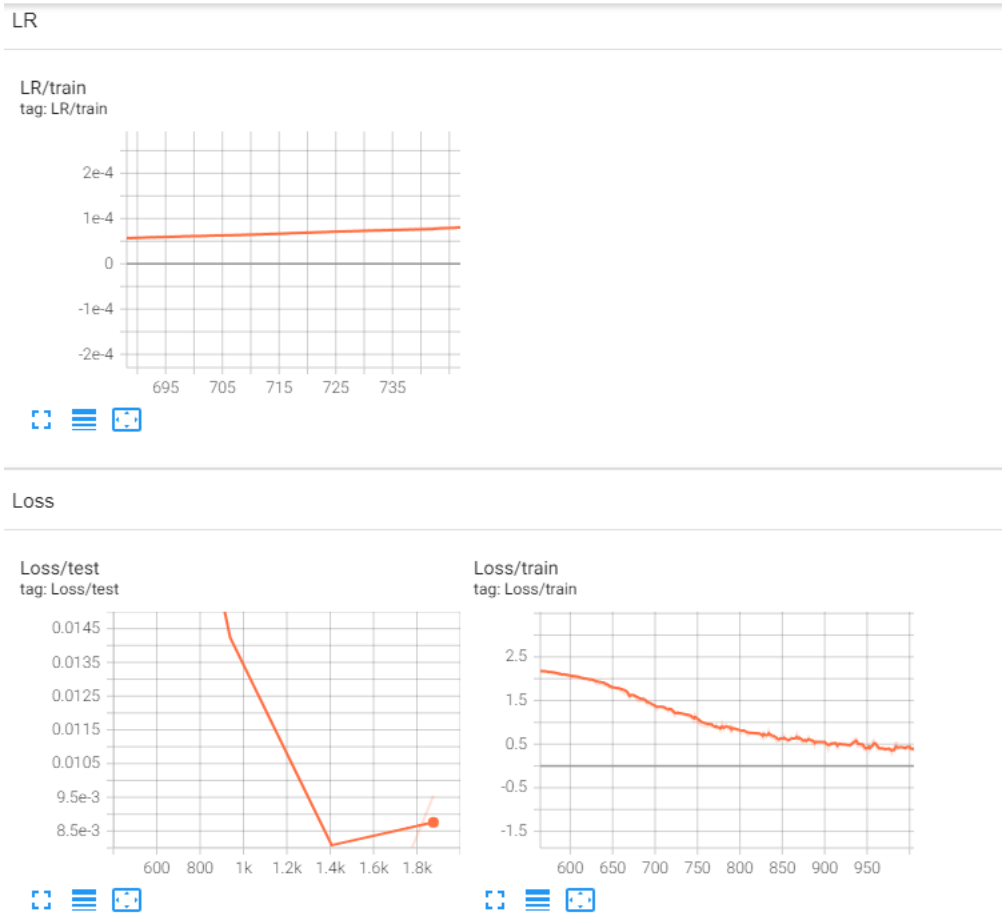


Figure 2: Zoomed in view for Learning rate vs train step (top) and Loss vs train step (bottom right) to get more accurate reading for train step and range of learning rate.

Note: The train loss decreases for a range of values, hence for ease we take only one single value for convenience and experiment with much larger and smaller learning rate to cross verify that the recommended range/order of magnitude of learning rate is good and not too small for our use.

For all the following experiments, custom training functions were written from scratch to generate loss and accuracy curve easily and experiment with multiple parameters easily.

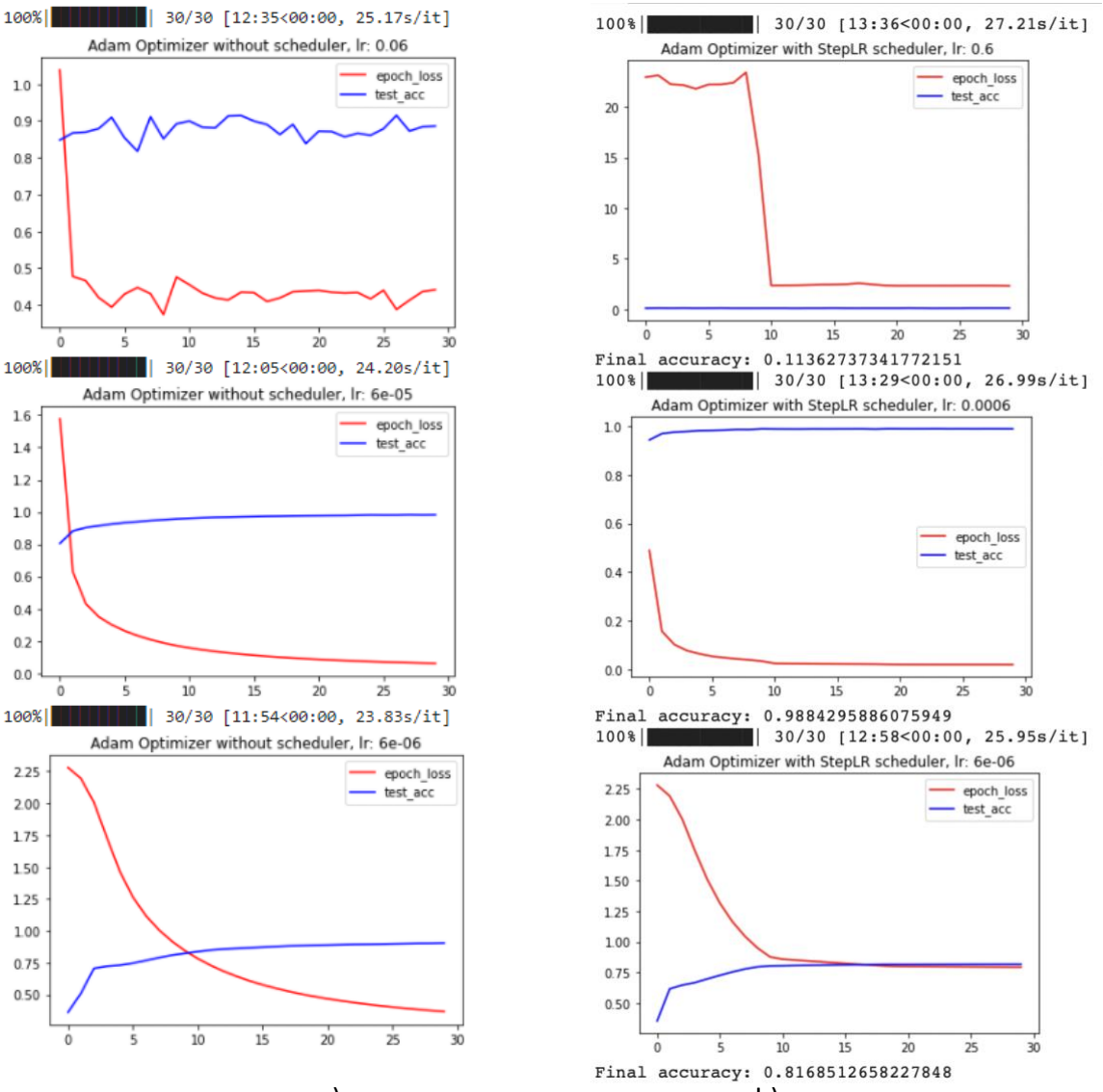
```
def train(model, optimizer, lr_scheduler, title="Results"):
    epoch_losses=[]
    epoch_test_acc=[]
    criterion = nn.CrossEntropyLoss()
    epochs = 30
    for epoch in tqdm(range(epochs)):
        all_loss = []
        model.train()
        for data in train_dataloader:
            x, labels = data
            optimizer.zero_grad()
            prediction = model(x)
            loss = criterion(prediction, labels)
            loss.backward()
            optimizer.step()
            all_loss.append(loss.item())
            if lr_scheduler:
                lr_scheduler.step()
        loss = sum(all_loss)/len(all_loss)
        epoch_losses.append(loss)

        model.eval()
        test_acc = []
        for data in test_dataloader:
            x_test, labels_test = data
            prediction = model(x_test)
            y_pred = torch.argmax(prediction, dim=1)
            accuracy = accuracy_score(labels_test.cpu().detach().numpy(), y_pred.cpu().detach().numpy())
            test_acc.append(accuracy)
        accuracy = sum(test_acc)/len(test_acc)
        epoch_test_acc.append(accuracy)
    plt.plot(epoch_losses, color='red', label='epoch_loss')
    plt.plot(epoch_test_acc, color='blue', label='test_acc')
    plt.legend(loc='best')
    plt.title(title)
    plt.show()
```

Figure 3: Train code written from scratch to evaluate the passes optimizer and learning rate scheduler.

Training using Adam Optimizer and without learning rate scheduler

Using the proposed learning rate value of 6.05e-5, we train the model again with three different learning rates: 0.06, 0.00006 and 0.000006. The expected result should be fastest convergence at learning rate with value of similar magnitude as proposed learning rate, with model struggling to converge at higher learning rate, and slower convergence for lower learning rates.
The accuracy and loss curve for each of the trained model is as shown in figure 4 a).

To further validate if the proposed order of magnitude of learning rate is valid, we re-conduct the experiment using three more learning rate schedulers, starting with relatively higher learning rates, and slowly decrementing them. The expected result is faster convergence once the learning rate gets its order of magnitude like the proposed learning rate.



a) b)

Figure 4: Model loss and test accuracy for Adam optimizer for different starting learning rate using a) constant learning rate. b) StepLR. learning rate scheduler.

Training using Adam Optimizer and with StepLR learning rate scheduler

Learning rate decreased by a factor of 10 after roughly every 5 epochs
Test accuracy and train loss for the above experiments are shown in figure 4 b).

As expected, in experiment 1 due to high initial learning rate, the model doesn't converge. Reducing learning rate helps the model decrease the loss slightly, but it still struggles due to high order of magnitude and doesn't converge, giving only 11% accuracy. However, for experiment 2 and 3, the model converges and each time the learning rate is decreased, it helps the model converge faster. For experiment 2, the learning rate is of same order as proposed by range test and model converges with just 10 epochs, giving 98.88% accuracy. With lower learning rate as in experiment 3, model is on the verge to converge but would require more epochs due to small step size in each iteration.

Training using Adam Optimizer and with MultiStepLR learning rate scheduler

Learning rate decreased by a factor of 10 after roughly 5, 15 and 20 epochs.
Test accuracy and train loss for the above experiments are shown in figure 5.
The results obtained are like StepLR and follow the same observation, i.e., as learning rate decreases to a smaller order, it helps the model converge much faster.

With large learning rate, experiment 1, loss starts to decrease only when learning rate becomes small but it stil doesn't converge. For experiment 2, with similar order magnitude learning rate as proposed rate, it converges easily and gives 98.79% accuracy. For very small learning rate, experiment 3, model needs many more epochs before it can finally converge.

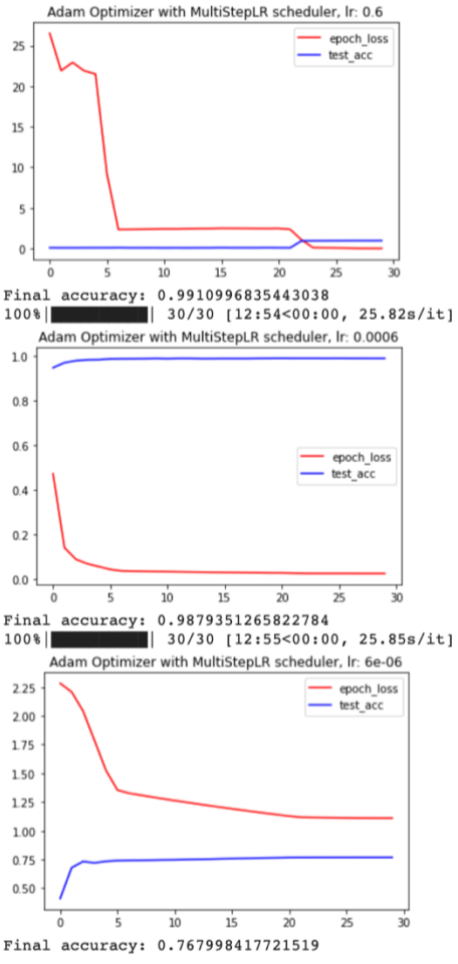


Figure 5: Model loss and test accuracy for Adam optimizer with MultiStepLR scheduler

Training using Adam Optimizer and with OneCycleLR, exponential learning rate scheduler

Test accuracy and train loss for the above experiments for OneCycleLR and explonentialLR scheduler also follow the above observed pattern, thereby validating our findings.

Conclusion

The findings/proposed learning rate obtained from learning rate range test is useful as it helps us get the rough range/order of magnitude around which the useful learning rate lies, without which we will have to train multiple models to even get a rough idea of it or spend computation training on many more epochs than required with smaller learning rate.
From all the experiments done, it is safe to say that to train LeNet5 model architecture on MNIST dataset using Adam optimizer, we can use learning rate of order of magnitude 1e-5 for faster convergence, with any larger learning rate struggling to converge and any learning rate smaller requiring more epochs for convergence.