3D Point Cloud and Modeling (NPM3D)
TP 6: Deep Learning

Borachhun YOU

Question 1

The training of MLP went for 100 epochs with variable learning rate. Step scheduler was used which halved the learning rate every 20 epochs, and the initial learning rate was set to 0.001. Figure 1 shows the changes of the learning rate throughout the 100 epochs. The training took a total period of 255 seconds.

Figure 2 shows the changes of training loss as well as the test accuracy during the training period. The loss quickly reached 0 after the first 20 epochs. The accuracy, on the other hand, remained very low throughout the whole period as it peaked at 17.51% in the early stage and fluctuated around just above 15% towards the end of the training period. This poor performance of the model can be explained by the flattening of the input data, which gets rid of the information of the 3D structure, and the permutation variant nature of the model that considers the same point cloud to be different when the orders of the input points are different.

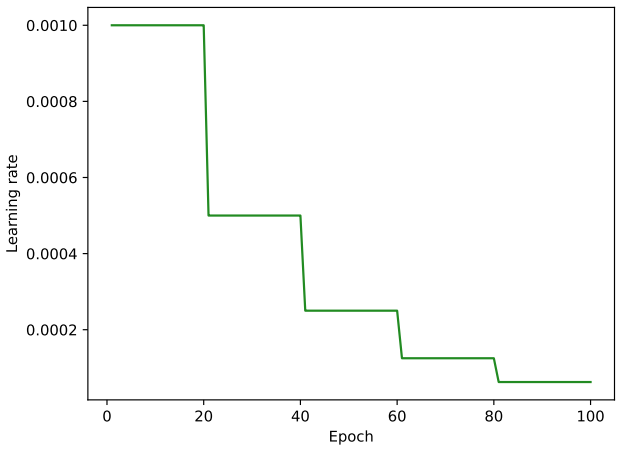

Figure 1: Learning rate

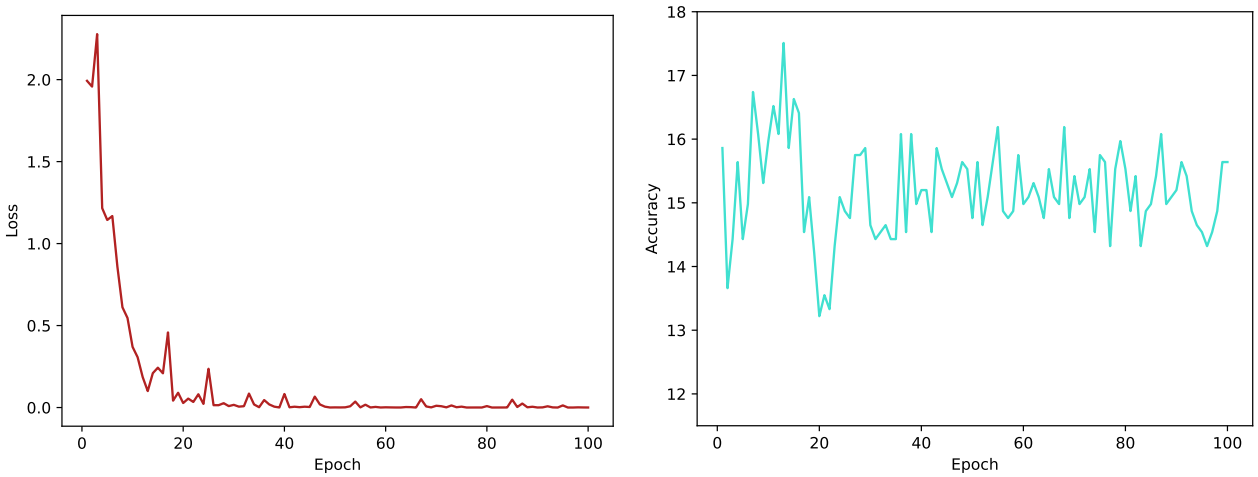

Figure 2: Training loss (left) and test accuracy (right) on ModelNet10_PLY with MLP

Question 2

The training of PointNetBasic had the same configuration as MLP, for 100 epochs and with the step learning rate scheduler. The training time was 793 seconds, which was much longer than MLP since it is a larger model.

Figure 3 shows the training loss along with the test accuracy throughout the training of the model. The loss did not converge to zero as fast as MLP, but the accuracy was significantly better compared to the previous model as it converged to around 90% and had the highest accuracy of 90.97%. This performance gain is mainly caused by the shared layers that makes the model permuation invariant.

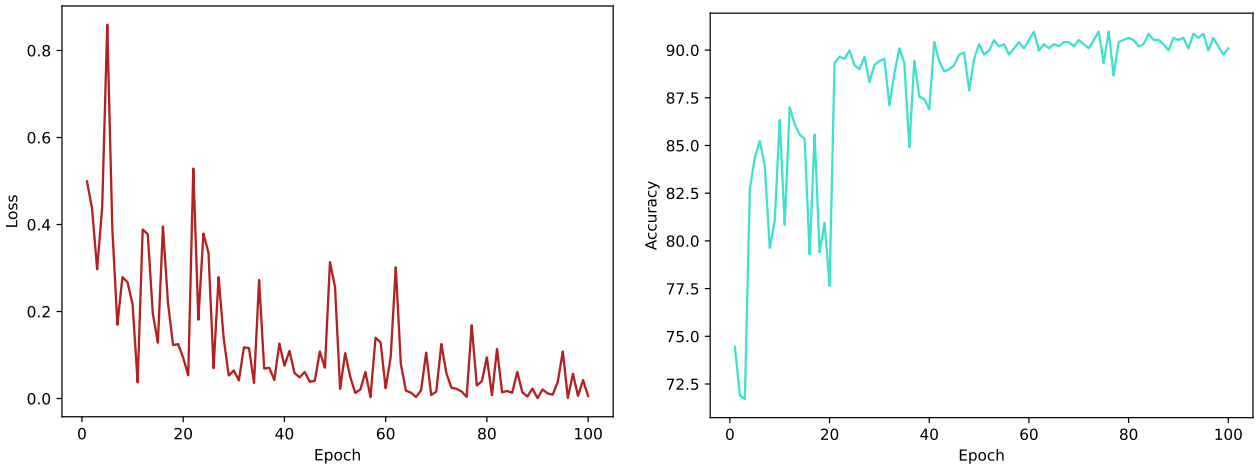

Figure 3: Training loss (left) and test accuracy (right) on ModelNet10_PLY with PointNetBasic

Question 3

Once again, the same configuration was applied to the training of PointNetFull, having trained for 100 epochs and with the step learning rate scheduler. The only difference was with the loss function, changing from `basic_loss()` that was used with the two previous models to `pointnet_full_loss()`. The training of the model took a total of 968 seconds, which was a bit more than PointNetBasic since it added a T-Net network.

Figure 4 displays the training loss and the test accuracy throughout the training period of the model. Similar to PointNetBasic, the loss did not quickly converge to zero like MLP. The accuracy was also similar to PointNetBasic with no significant improvement. It converged to around 90% and the highest accuracy was 91.30%, slightly higher than PointNetBasic.

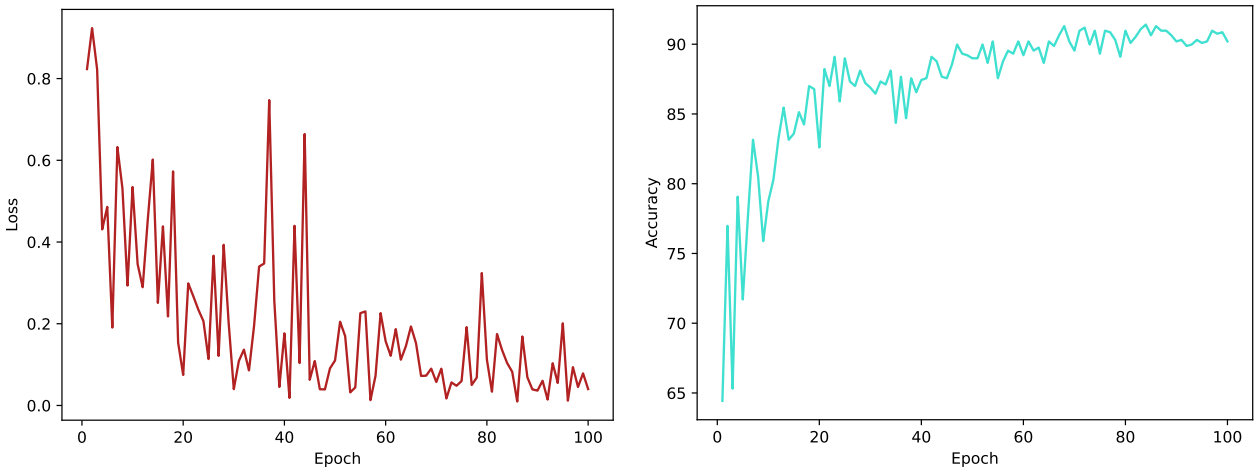

Figure 4: Training loss (left) and test accuracy (right) on ModelNet10_PLY with PointNetFull

Question 4

A new data augmentation method that was added to the existing techniques was reflection. The transformation was done along the x-axis by multiplying the x coordinate of the points with -1. PointNetFull was used for the training along with the same configuration as before.

In terms of accuracy, which is also shown in figure 5, there was not any notable improvement when including the new data augmentation method. The accuracy still converged to around 90% with the highest accuracy of 91.63%, just a bit higher compared to when the new method was not used.

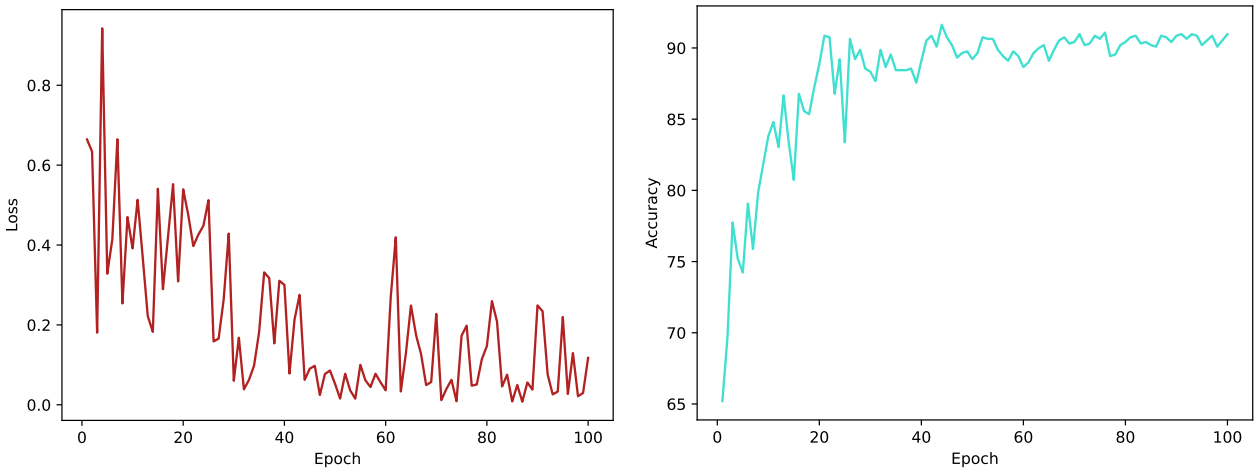

Figure 5: Training loss (left) and test accuracy (right) on ModelNet10_PLY with PointNetFull with a new data augmentation method