NPM3D - TP6

Aubin TCHOÏ

February 2023

All experiments are conducted on a laptop with an Intel i7 CPU (2.60GHz and 6 physical cores) and an Nvidia GTX 1660Ti GPU.

Question 1

Test accuracy	15.5%
Training time	61 s
Epochs	75
Learning rate	10^{-3}

Table 1: Performance of an MLP neural network on ModelNet10

The table above describes the results obtained using an MLP alongside the parameters used. The MLP model performs poorly and is only slightly better than random attributions.

To choose the number of epochs I initially set a higher value and noticed that past around 30 epochs the model stops learning as it reaches a loss of 0.

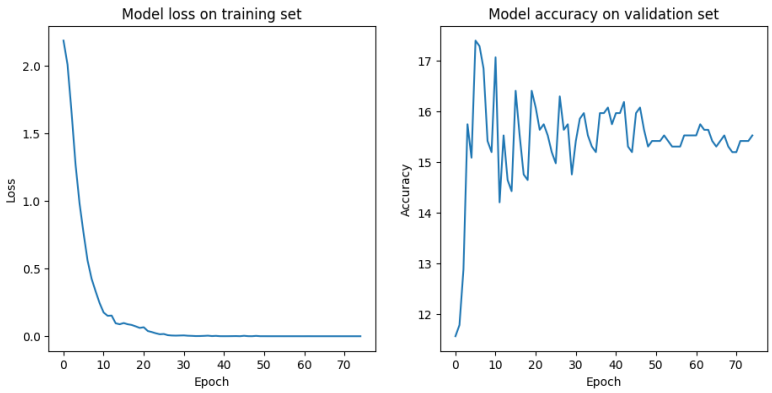

Figure 1: Evolution of the train loss and test accuracy during training on ModelNet10

In the figure above we can observe that the model quickly reaches a null loss, as it massively overfits on the training set (the training loss vanishes without any increase in the test accuracy). This is a common issue with MLP models, the model indeed contains 1707274 parameters for only 3991 training samples that each contain 1024 points and it seems that the model is able to some extent to memorize the training set.

We expect this model to perform poorly since by flattening the data it does not take into account the fact that it is made of 3D points, and it is also not invariant by shuffling, even though the order of the points in the point clouds have no signification.

Question 2

Test accuracy	90.9%
Training time	896 s
Epochs	150
Learning rate	10^{-3}

Table 2: Performance of the basic version of PointNet on ModelNet10

The basic version of PointNet reaches a plateau after 150 epochs. To be sure of that I waited a few additional rounds of scheduler (divides by two the learning rate every 20 epochs).

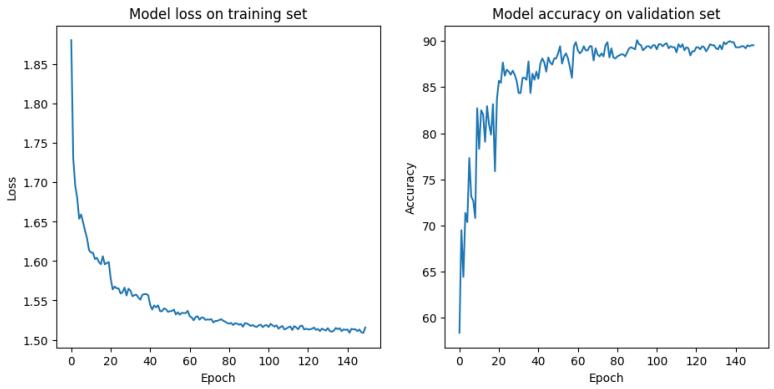

Figure 2: Evolution of the train loss and test accuracy during training on ModelNet10

This model clearly outperforms the MLP and it does not seem to overfit on the training set. This architecture is invariant by shuffling the dataset because the MLP layers before the max pool are shared. The spatial information is also conserved as the MLP layers operate on the unflattened data. My understanding of the behavior of the neural network is that the first MLP layers are trained to give a specialization to each column of the n x 1024 tensor that is fed to the max pool. This way each 3D point can convey a certain amount of information in the global feature if it stood out by having a particularly high value in one column.

I tried changing the learning rate by increasing it to 10^{-2}, and the convergence observed was slower (loss of 1.88 after 80 epochs, down from 1.53 previously). I also tried a learning rate of 10^{-4}, without any significant change in the convergence speed.

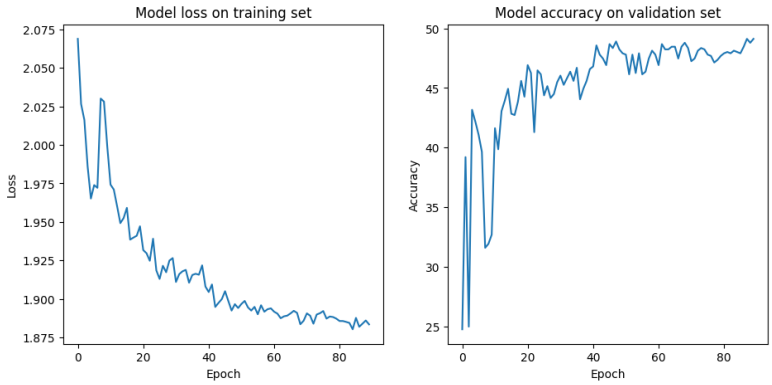

Figure 3: Evolution of the train loss and test accuracy during training on ModelNet10 for a learning rate of 10^{-2}

Question 3

Test accuracy	91.5%
Training time	1720 s
Epochs	150
Learning rate	10^{-3}

Table 3: Performance of the full version of PointNet on ModelNet10

The same remarks than before apply in the choice of the number of epochs and the learning rate. From my experiences, I found that this architecture involving an additional T-Net module achieves almost the same accuracy than the basic one but requires longer training.

This is partially due to the fact that the goal of the T-Net is to learn a transformation that will be used to align the point cloud using a matrix multiplication, and that the same thing is partially done by the existing data augmentation technique that consists in adding random rotation around the z-axis to each sample. By doing so, the model is able to learn to be rotation-invariant, and does not effectively requires the T-Net to achieves 90% of accuracy.

The longer training times observed come from the training of the T-Net and the additional operations involved (including one big matrix multiplication). The T-Net is indeed not a small architecture, the model that includes it contains 1614995 parameters, up from 811914 parameters in the basic version of PointNet.

Question 4

For the data augmentation I added random symmetries around the x-axis or the y-axis because the symmetry of a sofa is also a sofa, and the model can learn on both. I also added random rotations around the x-axis and the y-axis. In the context of the ModelNet datasets we can argue that the neural network should learn to identify the objects (chair, dresser, ..) independently from their x and y orientation, for instance chairs are sometimes put upside down on top of tables to help cleaning up the floor below. However this would not apply for outdoor data, as the z-axis plays a very specific role because of gravity.

I chose to use the architecture of PointNet without the T-Net because of the remark in the previous question.

	Without data augmentation	With data augmentation
Test accuracy	90.9%	91.7%
Training time	896 s	4711 s
Epochs	150	700
Learning rate	10^{-3}	10^{-3}

Table 4: Performance of the full version of PointNet on ModelNet10

Using data augmentation we achieve slightly better results but we need to train for many more epochs, after 150 epochs we only have an accuracy of 61%, compared to 90.9% without data augmentation. This was expected, as we add methods to randomly sample data we increase the number of training samples the model needs to learn from in order to recognize certain features in each sample. The overall gain in accuracy is not substantial, however this data augmentation part would probably be very useful in cases where we only have few samples of data and the model tends to overfit on them. This part thus probably greatly increases the model's ability to generalize to samples that are different from the ones it saw in the training dataset.