CSE 291 I00: Machine Learning for 3D Data Homework 3

Qiaojun Feng
A53235331
Department of Electrical and Computer Engineering
University of California, San Diego
qjfeng@ucsd.edu

For this assignment we don't have so much to explain and analysis. We just simply introduce some basic ideas of the paper, go through the network architecture and report the results we got.

1. PointNet (vanilla)

The key idea of manipulating unordered pointcloud is to design a symmetry function for unordered input. Luckily max-pooling is such a function. And also it was proved that with continuous function and symmetric function(max function) any continuous function can be approximated.

The implementation is also easy. Suppose the batch size is B, the number of point in one object is P, each point is in $\mathbb{R}^3$.

- input: Size $B \times P \times 3$
- MLP1(conv2d): Size $B \times P \times 64$
- MLP2(conv2d): Size $B \times P \times 64$
- MLP3(conv2d): Size $B \times P \times 64$
- MLP4(conv2d): Size $B \times P \times 128$
- MLP5(conv2d): Size $B \times P \times 1024$
- Maxpooling: Size $B \times 1024$
- MLP6(fc): Size $B \times P \times 512$
- MLP7(fc): Size $B \times P \times 256$
- Dropout: Size $B \times P \times 256$
- output = MLP8(fc): Size $B \times P \times 40$

Notice that the first 5 MLP(multi-layer perceptron BUT actually one layer) are implemented by convolution. For the MLP1 the kernel is [1,3] while the other are all [1,1]. And the other MLP are implemented by fully-connected layers. And between MLP7 and MLP8 there is a dropout layer with keep_prob=0.7. Each MLP has batch-normalization and non-linear ReLU activation afterward.

2. PointNet

While for 2D classification we want it to be robust, to be invariant to 2D transformation. In the 3D space it become certain geometric transformations, such as rigid transformation. It is expected that the learnt representation on the unordered pointcloud is invariant to these transformations.

Then the paper introduced some sub-network doing alignment work. T-Net1 transforms the input(3D) and is between input and MLP1. T-Net2 transform the middle feature(64D) and is between MLP2 nd MLP3. Also a new term of loss is introduced to restrict the feature transform matrix to be close to orthogonal matrix.

- input: Size $B \times P \times 3$
- T-Net1(input transform): a 3×3 matrix
- MLP1(conv2d): Size $B \times P \times 64$
- MLP2(conv2d): Size $B \times P \times 64$
- T-Net2(feature transform): a 64×64 matrix
- MLP3(conv2d): Size $B \times P \times 64$
- MLP4(conv2d): Size $B \times P \times 128$
- MLP5(conv2d): Size $B \times P \times 1024$
- Maxpooling: Size $B \times 1024$
- MLP6(fc): Size $B \times P \times 512$
- MLP7(fc): Size $B \times P \times 256$
- Dropout: Size $B \times P \times 256$
- output = MLP8(fc): Size $B \times P \times 40$

Besides, they also do some data augmentation by rotating around y axis and adding some gaussian noise. The training samples' order is shuffled for each epoch.

We trained with GTX 1080Ti. For vanilla it took around 1 hour to go 250 epochs. The full version took around 3 hours for 250 epochs. Our results is 85.2/88.0, 1% lower than the paper's reported result(86.2/89.2). We can see that for the full version, though the training loss is going down, the training accuracy is going up for 80% of the training time, the test loss and accuracy doesn't change a lot after 100 epochs. It infers that there might be some overfitting problems. We may want to feed in more robust data but we couldn't reach such high accuracy once we made the preprocessing of input data more complex. See TABLE I and Fig. 1 for more details.

REFERENCES

[1] Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proc. Computer Vision and Pattern Recognition (CVPR), IEEE 1.2 (2017): 4.
[2] GitHub - charlesq34/pointnet: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

TABLE I: sample and node number and route distance on input_maze

Model		Valid(50)	Valid(100)	Valid(150)	Valid(200)	Valid(250)	Rotation	Rotation&Jittering
Pointnet vanilla	avg.class	0.8272	0.8297	0.8366	0.8485	0.8444	0.8461	0.8399
	overall	0.8558	0.8618	0.8655	0.8691	0.8679	0.8694	0.8649
Pointnet	avg.class	0.8353	0.8425	0.8545	0.8604	0.8568	0.8610	0.8518
	overall	0.8651	0.8687	0.8772	0.8849	0.8857	0.8857	0.8801

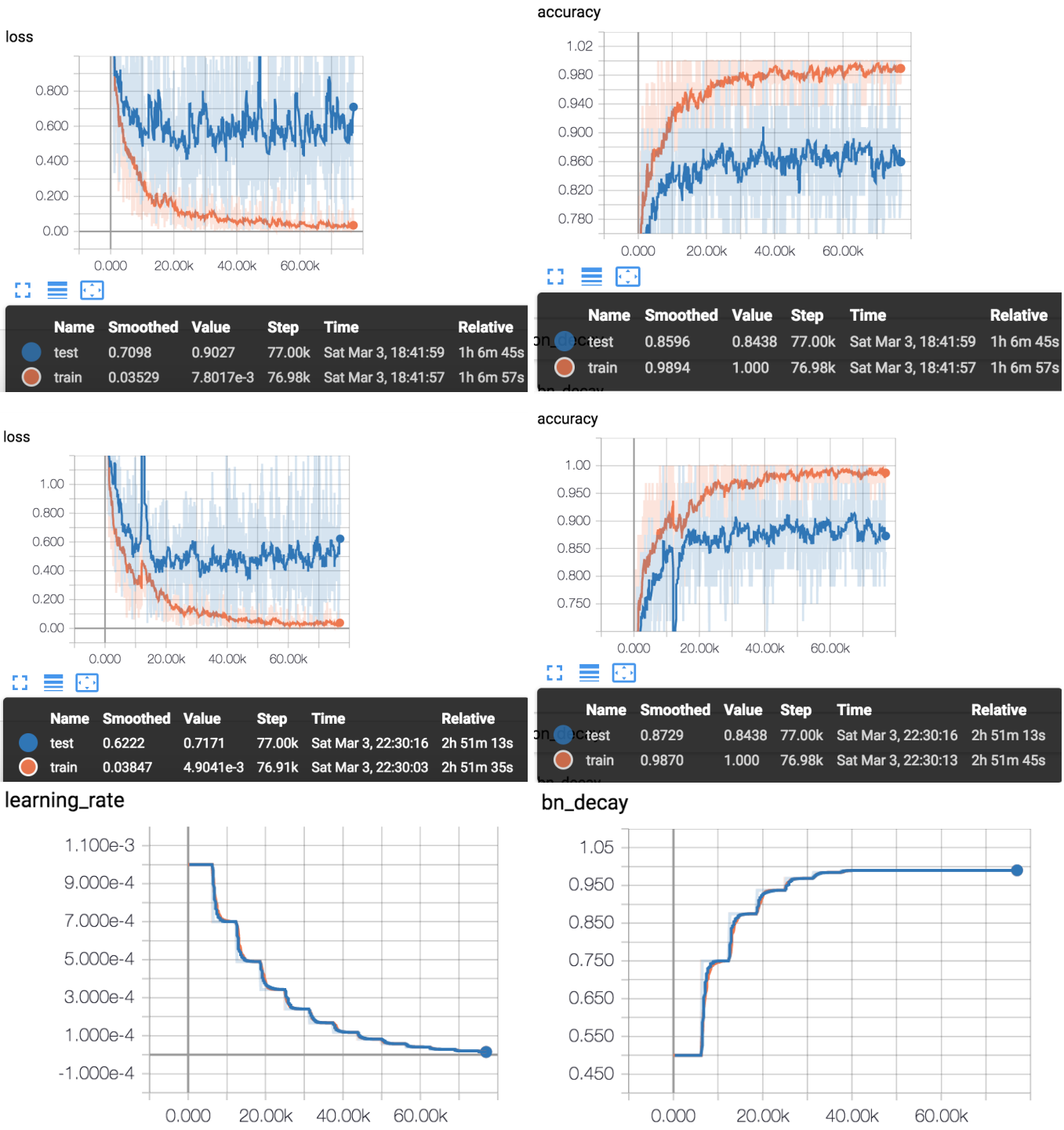

Fig. 1: Pointnet(vanilla) & Pointnet