SuperPointNet: Point Cloud Object Recognition

Yu-Chung Hsiao

1 Introduction

Thanks to the recent demand from autonomous driving and robotics, the 3D point cloud image representation has been growing its popularity and gradually entering consumer products. This representation consists of a set of points located freely in 3D space, without being constrained to regular grids. Such irregularly-located and potentially sparse point information has posed various challenges in computer vision. Mostly notably, the standard convolution operation is not properly defined at irregular grids. One solution is to project the 3D points onto pre-defined regular grids before performing a convolution [1], or, similarly, to augment the standard convolution with an internal projection step [2]. These pre-processing steps inevitably introduce approximation errors, losing precise point locations in trade of using convenient common solutions. Other approaches involve a fusion of point cloud with other information, such as multi-view images [3]. These types of data augmentation provide rich context about point locations, at the price of largely increasing data volumes and computational complexity.

One recent breakthrough and pioneer work is PointNet [4] and its enhancement PointNet++ [5]. The central idea of PointNet is to utilize "symmetric functions" such that point cloud can be treated as an unordered set of points. This relaxes the requirement of representing point cloud in any coordinate system. Point cloud can stay in its simplest form: a list of point coordinates and be consumed directly by PointNet. PointNet++ further explored this idea and constructed a mechanism that mimics the hierarchical behavior of convolution layers. These two network architectures provide state-of-the-art accuracy for the ModelNet40 [6] benchmark and are considered outperforming in terms of its time and space complexity because of its simplicity. Another relevant and recent work that uses symmetric functions on unordered sets is Deep Sets [7], although it focuses less on the point cloud application.

In this project, I will focus on implementing PointNet [4] and attempt to propose an extension, SuperPointNet, with an aim to providing some ideas of improvement over the original design.

2 PointNet: Ideas and Implementation

The key idea of PointNet [4] is to utilize symmetric functions such that the calculation is irrespective of the input data order. A function f is called symmetric on variables $x_1, x_2, \ldots, x_n$, when the function value is unchanged upon *any* permutation of these variables. Examples of such functions include max, mean, sum,

variance, etc. Intuitively, for point cloud, the order of points x_i stored in a list is arbitrary and should not affect the type of objects it represents. PointNet uses this symmetric property to achieve the direct processing of raw point cloud data. Besides, PointNet is great for its simiplicity. In particular, PointNet only adopts two basic neural network layers in its architecture: fully-connected layers and max-pooling layers. Since fully-connected layers are not symmetric in general, they are only applied to the coordinates or the features of an individual point, without interfering with other points. On the other hand, max-pooling layers are the key "blender" that treats a set of points as a whole and extracts joint information among the points.

2.1 Transformation Net

PointNet introduces a building block "Transformation Net" that adds some flavor of traditional 3D geometry. The transformation net $T(x_1, \ldots, x_n)$ is a data-dependent linear transformation matrix. For a 3D point, this transformation behaves like an affine transformation (in the sense of 3D geometry) without translation, i.e., a 3×3 matrix. For feature space of K features, this transformation becomes a $K \times K$ matrix. Intuitively, it is data-depedent such that it can provide different transformation to maximize certain coordinate or feature component that the max-pooling layer can pick up. Mathematically, it is defined as

$$T_{K\times K} = F^{K\times K} \circ F^{256}_{RB} \circ F^{512}_{RB} \circ \mathrm{Max}_i^{1024} \circ \{F^{1024}_{RB} \circ F^{128}_{RB} \circ F^{64}_{RB} \circ x_i\}_{i=1,\ldots,n}, \quad (1)$$

where the superscript is the output dimension, F stands for a fully-connected layer, Max is a max-pooling layer across point features, and curly braces enclose a set. The subscript R denotes a Relu output and B for batch normalization. In this case, the max-pooling layer picks the maximum value across n points for each of the 1024 features. Note that the outermost fully-connected layer does not use Relu and batch normalization.

2.2 Classification Network

Using a similar notation, the overall classification network can be written as

$$\text{score} = F^{(k,256,512)} \circ \mathrm{Max}_i^{1024} \circ \{F^{(1024,128,64)}_{RB} \circ T_{64\times 64}(F^{(64,64)}_{RB} \circ (T_{3\times 3} x_i))\}_{i=1,\ldots,n}, \quad (2)$$

where the small k represents the number of classes. Here I further condense the notation by grouping similar fully connected layers with a superscript of tuples, for instance, $F^{(64,64)}$ denotes two layers of fully-connected layers with output sizes 64 and 64, respectively.

The output scores are then fed to a softmax cross entropy loss function compared against training labels as in a regular multi-way classification setup.

3 SuperPointNet

PointNet topped the accuracy at the time of its being published and it is also reported from the paper that its runtime and memory requirement is very lightweight, which is intriguing for practical applications. However, despite its good performance, PointNet does not directly utilize the information of point proximity, which is the key attribute that convolution neural net is based on. As described earlier, PointNet takes a list of raw point coordinates and performs operations as in (1) an (2). It is possible that the model learns to recognize the distance between points and take advantage of neighboring relationship. But this is never clear how this information is learned or how effective this proximity relationship is captured. I therefore propose SuperPointNet to explicitly inject the proximity information along with each raw point coordinates.

The idea of SuperPointNet is to generate "super points" that encapsulate some information of the neighboring points. Simply put, super points is a cluster of points that are generated through classification algorithm. The intuition is that those clusters should represent the main frame of an object. And with the information of part of the main frame, a ball can be better recognized where it is located relative to the whole object.

I use the Gaussian mixture model to generate the cluster assignments, and compute the mean of all points as the representative of a cluster. From (2), the input point has three coordinate x, y, and z. To encapsulate the super point information, I choose to augment these three coordinates with the coordinates of the center of its cluster, i.e.,

$$x, y, z, x_c, y_c, z_c, w_c, \tag{3}$$

where x_c is the augmented x-coordinate, for instance, using the center of a cluster, and w_c is the weight of the cluster. Various weights can be chosen, e.g., the size of a cluster, the inverse size of a cluster, and whether each of which is normalized by its mean and standard deviation. The augmented coordinates can also be of other choices, such as a random point within the same cluster, the farthest point in the same cluster, etc.

Figure 1 shows a multi-angle view of a chair in point cloud of 2048 points. By specifying 512 clusters, the Gaussian mixture model generates the super points in red. These super points properly capture the main structure of the chair, particularly at the leg parts.

After data augmentation from $\mathbb{R}^3$ to $\mathbb{R}^7$, the size of input layer for both (1) and (2) needs to be enlarged as well. Note that this only impacts the first layer, from 3 to 7 of the input size. Therefore, the runtime cost is at a similar level, both in terms of space and time.

4 Data Exploration

The benchmark ModelNet40 [6] is a common dataset for 3D point cloud recognition problem. Although many refer it as a set of point cloud objects, it is not

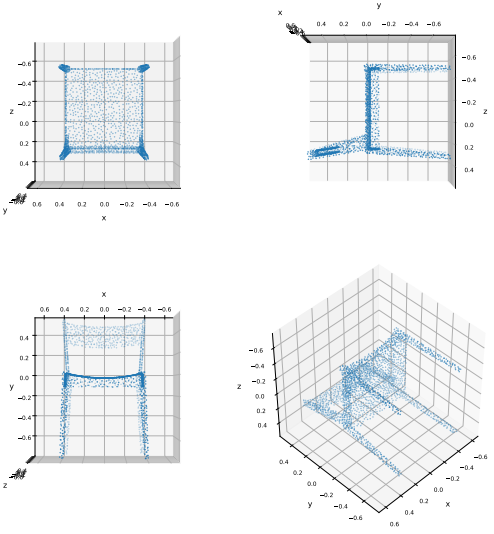

Fig. 1. Original point cloud of 2048 points for a chair.

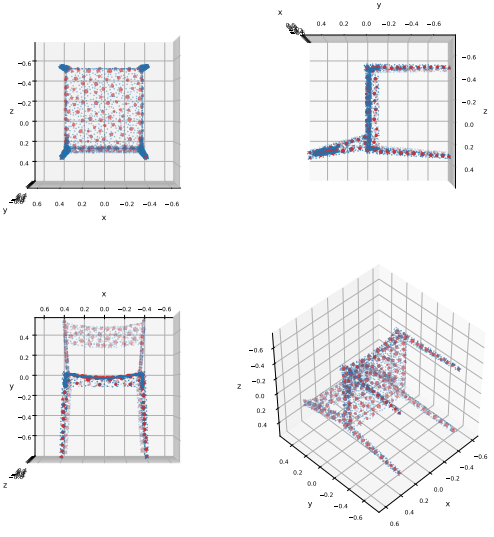

Fig. 2. The chair in Figure 1 is augmented by 512 super points in red. It can be seen that red dots capture the main frame of the chair, especially for the leg parts.

generated through sensor data of real-world objects. Rather, ModelNet40 is a set of human labeled and cleaned 3D CAD models. The process of labeling and cleaning are vaguely described in[6]: "*... manually checked each 3D model and removed irrelevant objects from each CAD model (e.g, floor, thumbnail image, person standing next to the object, etc) so that each mesh model contains only one object belonging to the labeled category.*"

More concretely, ModelNet40 contains an official split of a training and a test set, each of which comprises 9840 and 2468 images in 40 object classes. The origin of the coordinate system is shifted to the center of the object. All objects are purposefully oriented such that the gravity is always in the negative y direction. The class labels and distributions for the training and the test data are shown in Figure 5.

5 Experiment

5.1 PointNet

Based on the understanding of PointNet described in Section 2, the implementation reaches 88.2% overall accuracy, while PointNet paper reports 89.2%. Another approach for point cloud object recognition, Subvolume [8], also reports 89.2% overall accuracy. This demonstrates two things: PointNet predicts well. Second, there are still some missing details that make the implementation less performant. Sources of discrepancy may be 1) different initialization, 2) not very clear about batch normalization decay, and 3) other hyperparameter tuning.

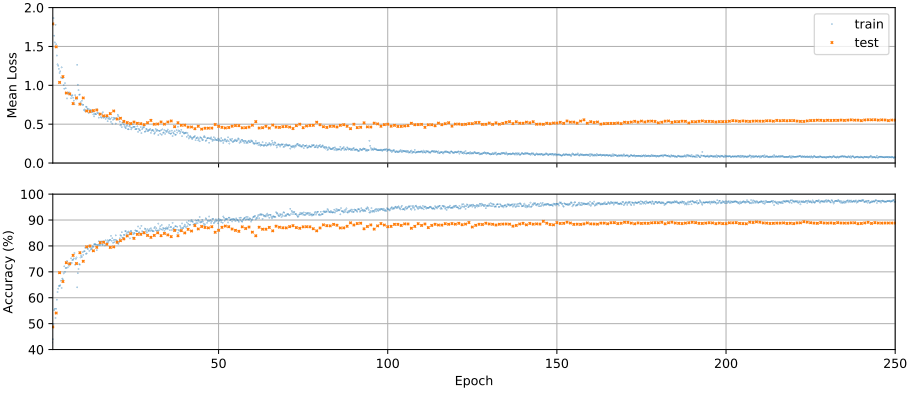

Fig. 3. Training curves for the PointNet prototype. The best test accuracy is about 88%. The best value in the original paper is 89.2%.

Noticeably, the training curve I got in Figure 3 exhibits overfitting. Although the evaluation against the test set is reasonably good, the model is apparently

overfit and the generalization may be an issue. Several methods, such as adding l_2 loss and using more aggressive dropout, are attempted but the overall trend stays similarly. It can also be seen that the mean loss starts to slightly increase, while the test accuracy keeps improving. This usually occurs when some data are shared in both the training and the test sets. In other words, it is possible that some objects in training and in testing are too similar in the ModelNet40 dataset.

5.2 SuperPointNet

The implementation of SuperPointNet is straightforward: preprocess point cloud by augmenting three coordinates into seven coordinates, as in (3) and increase the input size accordingly. In particular, I tested different types of augmentation along with different weights

- cluster center, cluster size normalized by mean and standard deviation (MS)
- cluster center, cluster size normalized by standard deviation (S)
- cluster center, inverse cluster size normalized by mean and standard deviation (IMS)
- cluster center, inverse cluster size normalized by mean and standard deviation (IS)
- random point in same cluster, inverse standard deviation (IS)

Due to the limited computation resources, I only tested the above proposals on 256 points with 64 super points as opposed to the 1024 points on the paper. The performance is summarized in Table 1

Table 1. Accuracy for various augmentations

Model	Augmented Points	Aug. Weight	Points	Accuracy
PointNet (paper)			1024	89.2%
PointNet (impl.)			1024	88.2%
PointNet (impl.)			256	**87.2%**
SuperPointNet	cluster ctr.	MS	256	84.5%
SuperPointNet	cluster ctr.	S	256	86.3%
SuperPointNet	cluster ctr.	IMS	256	85.0%
SuperPointNet	cluster ctr.	IS	256	88.1%
SuperPointNet	rand. pnt. same cluster	S	256	**88.9%**

It can be observed that

- Normalization by dividing standard deviation is better than shifting mean and do division. It can be understood as shifting by mean causes negative values in weights.

– Inverse cluster size outperforms cluster size. It is possible to interpret that large clusters are already represented by a lot of points, whereas small clusters are the ones that actually need more emphasis through weights.

– Random point in the same cluster outperforms the augmentation using cluster centers. From the pure data distribution perspective, this is reasonable since cluster centers repeat N times within the dataset if the cluster is of size N. This decreases the data variance and effectiveness of increasing the dimension of inputs.

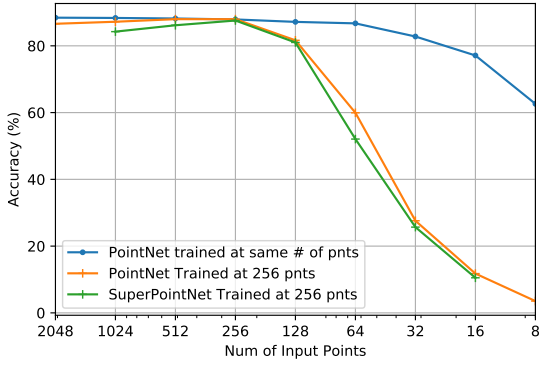

Fig. 4. Accuracy by feeding models various number of points per object.

The performance of SuperPointNet can be studied from a different angle. Since the number of input points is rarely a fixed number in the real world, it is of interest to know whether a model is robust against varying number of points per object. Figure 4 demonstrates such trends. A model performs best when it is evaluated at the same number of input points as the training condition. This applies both to PointNet and SuperPointNet. One can see from the figure that when models are trained at 256 points, the accuracy tops at 256 points as well. Comparatively, SuperPointNet is less robust than PointNet in terms of being used for a wide range of input point numbers. This is possibly due to the overfitting behavior as seen in Figure 3. As SuperPointNet uses slightly more parameters, it may experience more serious generalization issues. Another explanation is that, in this particular example, SuperPointNet is trained with augmented data by cluster centers. Since cluster centers are not real points, when there are few points, these fictitious points tend to be noise than useful information. However, this explanation does not explain why having more points encounters the same issue.

6 Conclusion

In this project, I implemented PointNet and achieved similar accuracy as the paper reports. In addition, lack of utilizing point proximity of PointNet sparkled the idea of SuperPointNet, which embeds proximity through data augmentation. About 1.7% points of improvement on overall accuracy is observed. However, SuperPointNet is less robust when varying number of input points.

7 Future Work

PointNet++ is created, potentially, from a similar observation that the proximity information was not fully utlized in PointNet. Hence, the authors designed a grouping and sampling mechanism that can generate hierarchical structure just as what convolution neural net does. However, PointNet++ requires repetitive calls of such grouping and sampling functions, making the runtime 8-16 times slower than the original PointNet. Since PointNet is superior to other approaches for its lightweight and simplicity, involving such complex computation for additional 2.6% improvement seems less attractive.

A possible extension of SuperPointNet is to precompute the hierarchical grouping beforehand. Since super points are a type of grouping, doing it recursively may effectively create the needed hierarchy. Similar to convolution neural nets, where pixels and voxels are fixed and one can always expect the range of signal propagation through layers, it is possible to build the grouping hierarchy upon given a set of points, without involving the computation of deep neural nets. This way can potentially alleviate the heavy computation incurred in PointNet++.

References

1. Maturana, D., Scherer, S.: VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (September 2015) 922–928
2. Vialatte, J.C., Gripon, V., Mercier, G.: Generalizing the Convolution Operator to extend CNNs to Irregular Domains. (June 2016)
3. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view Convolutional Neural Networks for 3D Shape Recognition. arXiv:1505.00880 [cs] (May 2015)
4. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. arXiv:1612.00593 [cs] (December 2016)
5. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. arXiv:1706.02413 [cs] (June 2017)
6. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2015) 1912–1920
7. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., Smola, A.: Deep Sets. arXiv:1703.06114 [cs, stat] (March 2017)
8. Qi, C.R., Su, H., Niessner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and Multi-View CNNs for Object Classification on 3D Data. arXiv:1604.03265 [cs] (April 2016)

Appendix

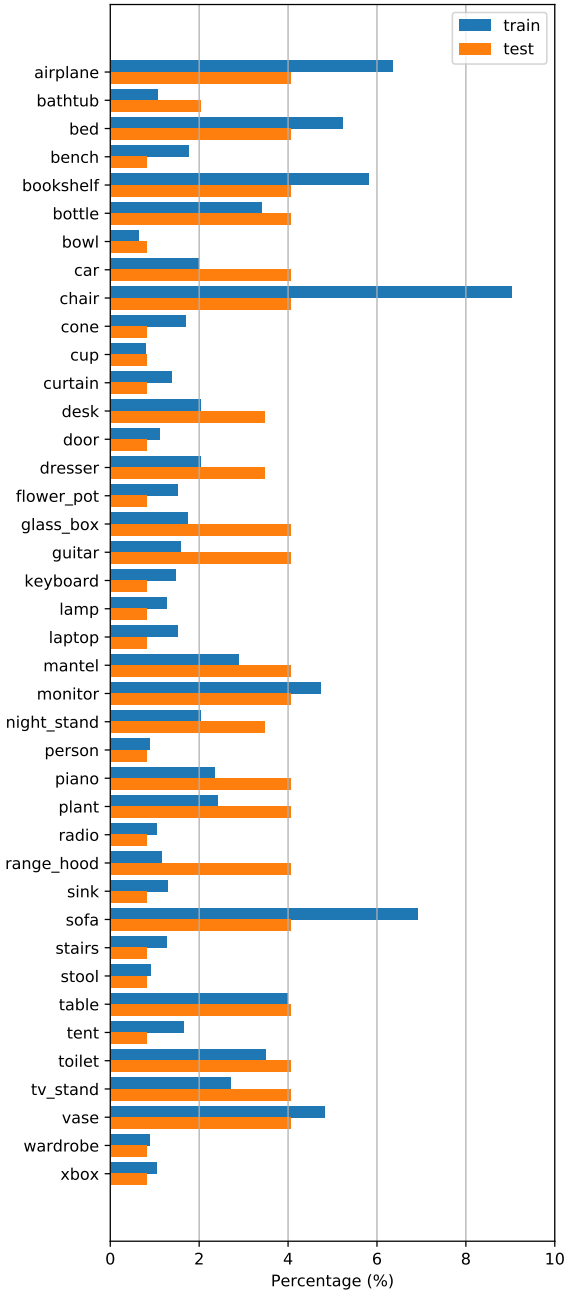

Fig. 5. ModelNet40 data labels and distributions