Why Discard if You can Recycle?: A Recycling Max Pooling Module for 3D Point Cloud Analysis

Supplementary Material

Jiajing Chen, Burak Kakillioglu, Huantao Ren, Senem Velipasalar
Syracuse University, Electrical Engineering and Computer Science Dept. Syracuse, NY, USA
{jchen152, bkakilli, hren11, svelipas}@syr.edu

1. Overview

This supplementary material contains additional details and results complementing the main paper. Section 2 provide additional example segmentation outputs for qualitative comparison. Section 3 shows examples for the visualization of the recycled points. Section 4 provides motivation and the reason for our approach using the refinement loss, and Section 5 investigates the distribution of the number of points, kept by different networks using the traditional max pooling, for different object classes in the ModelNet40 dataset. Section 6 provides a discussion.

2. Additional Qualitative Results

Example segmentation outputs obtained on the S3DIS dataset is provided in Fig. 1 for qualitative comparison. The figure shows the outputs of PointNet, DGCNN and DPFA without (w/o) and with (w/) incorporating our proposed RMP module. Some regions are marked by black ellipses to show the improvement provided by our proposed RMP. These results were obtained when models were trained on five areas, and tested on the the sixth area, which was set aside during training.

3. Visualization of the Recycled Points

In Fig. 2, red points are the ones utilized by the traditional max-pooling. Green color shows the points that are recycled by our proposed RMP module, which would otherwise have been discarded by the traditional max-pooling. Gray color shows the discarded points after two levels of max-pooling. For different object classes, it can be observed that PointNet has the most and PointNet++ has the least number of gray points. This is consistent with Table 1 of the main paper, since PointNet has the smallest and PointNet++ has the largest point utilization percentage after the first max-pooling.

We can also see that the recycled (green) points complement the red ones, and can provide additional useful features for refinement. For instance, in the airplane example, points are recycled around the tip of the right wing of the plane, which were originally discarded by traditional max-pooling.

4. Motivation for the Refinement Approach

After recycling some of the points discarded by the traditional max-pooling, the next step is to decide how to combine their features with those of the points, which were originally kept after the traditional max-pooling.

Summing or concatenation are two commonly used approaches to fuse features. In contrast, our approach incorporates the refinement loss to use the features of the recycled points to refine the features of the original points. To further motivate the benefit of our approach, we have performed experiments by using summing or concatenation for feature fusion. More specifically, we have used PointNet, PointNet++, DGCNN, DPFA, GDANet and CurveNet to perform point cloud classification on the ModelNet40 dataset. After obtaining $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$, either summation or concatenation was performed to obtain the final feature vector $\boldsymbol{f}$, which was then employed for final prediction. The results are summarized in Table 1. The 'Original' refers to the performance of the corresponding original network. As can be seen, both summing and concatenation of $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ drags the performance of the original network down. On the other hand, as shown in Tables 3, 4 and 5 of the main paper, our proposed approach using the refinement loss, improves the performance of all the baseline networks.

Fig. 3 shows the testing accuracy plots, after each epoch, during the whole training process for different baseline networks. Red is the plot for the original baseline network, while green and blue plots are obtained with concatenating or summing $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$, respectively. As can be seen, the original networks' performance is degraded during the whole training process.

These observations together with the Table 2 of the main paper support that, even though $\boldsymbol{F}_2$ contains valuable in-

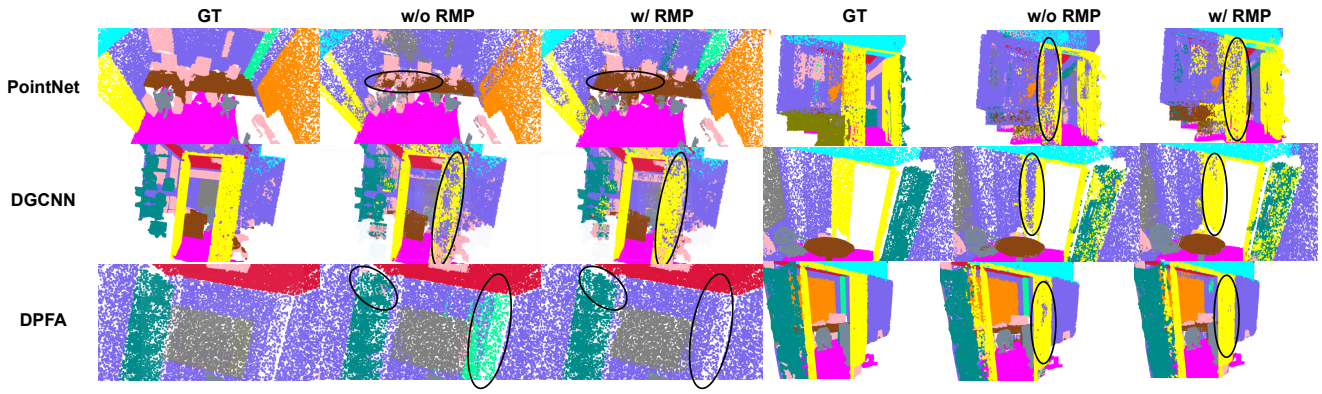

Figure 1. **Example segmentation outputs for qualitative comparison.** Outputs of PointNet, DGCNN and DPFA, without (w/o) and with (w/) incorporating our proposed RMP module, on the S3DIS dataset. Some regions are marked by black ellipses to show the improvement provided by our RMP.

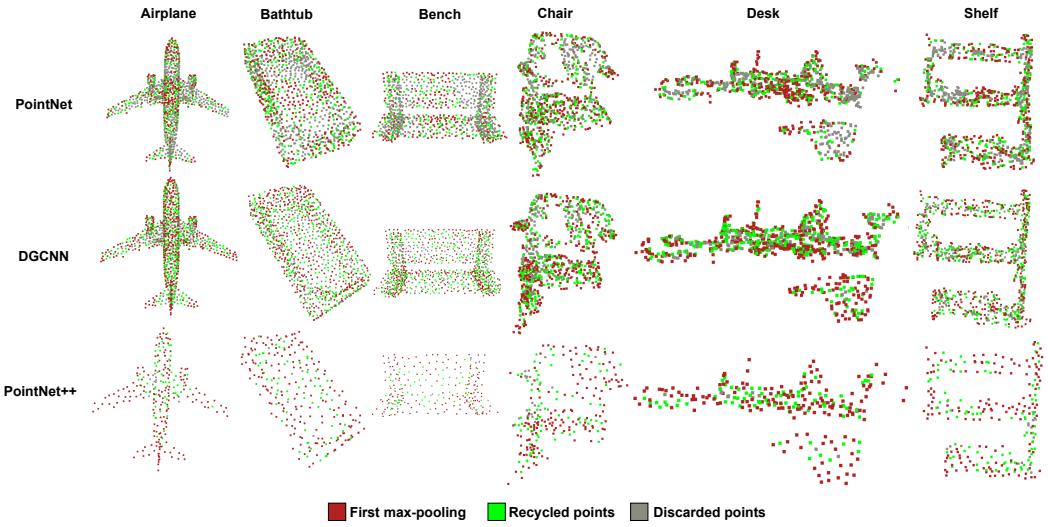

Figure 2. **Visualization of the utilized points.** Red points are kept by the traditional max-pooling. Green points are recycled by our proposed RMP module. Gray color shows the discarded points after two levels of max-pooling. First three objects are from the ModelNet40 dataset, and last 3 objects are from the ScanObjectNN dataset.

	PointNet	PointNet++	DGCNN	GDANet	DPFA	CurveNet
Original	90.12%	93.06%	92.51%	92.30%	93.10%	92.82%
Concatenation	90% (↓0.12%)	92.5% (↓0.56%)	91.49% (↓1.02%)	92.06% (↓0.24%)	91.85% (↓1.25%)	91.94% (↓0.88%)
Summation	89.88% (↓0.24%)	92.06% (↓1.00%)	92.14% (↓0.37%)	91.61% (↓0.69)	92.82% (↓0.28)	92.26% (↓0.56%)

Table 1. **Motivation for the refinement approach.** Both the summation and concatenation of $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ drag the original network's performance down.

formation for prediction, it is still not as representative or as powerful as $\boldsymbol{F}_1$. Thus, simply fusing $\boldsymbol{F}_2$ with $\boldsymbol{F}_1$, via summation or concatenation, might corrupt the original $\boldsymbol{F}_1$, which in turn causes a performance drop. To avoid this, we design the Refinement Loss to refine $\boldsymbol{F}_1$ by $\boldsymbol{F}_2$, rather than fusing them together, for prediction. Supported by the results presented in our main paper, our proposed approach provides a promising way to take advantage of these still informative features, and indeed improves the performance of all the original baseline networks.

5. Analysis of Points Kept by Max Pooling

In order to analyze the number of points kept after the traditional max-pooling operation, we performed experiments with PointNet, PointNet++ and DGCNN. These

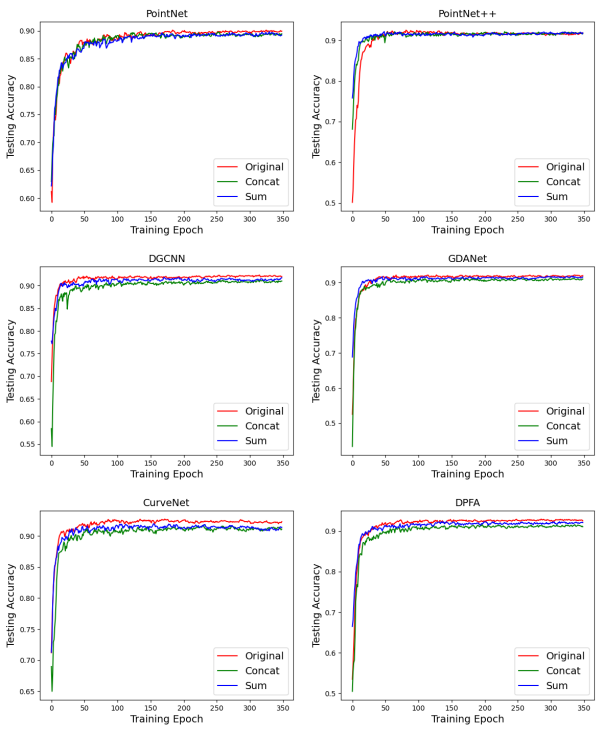

Figure 3. This figure shows the each epoch's testing accuracy during training process of models. It's could be observed that, if concatenation or summing is used to fuse $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$, the original network's performance is dragged down.

Model	Number of classes following Normal Distr.	Number of classes not following Normal Distr.
PointNet	33	7
PointNet++	32	8
DGCNN	33	7

Table 2. For all three models, the number of points kept after max-pooling follows a normal distribution for most object classes

baselines were chosen since most point-based methods have been developed based on these three networks. More specifically, we have analyzed the number of points kept after max-pooling for each of the 40 classes in the ModelNet40 dataset. After recording the number of utilized points for each sample of each class, we applied a normal distribution test on the data. The results are shown in Table 2. As can be seen, for all three models, the number of points kept after max-pooling follows a normal distribution for most object classes (32 to 33 out of 40 classes). This result combined with the observation of samples from each class, can lead to the conclusion that the number of points kept after max-pooling is related to the sample shape's complexity, i.e. for less complex shapes, fewer points are kept or vice versa.

6. Discussion

The proposed approach can be used to improve the performance of networks that use max-pooling. KPConv [2] classification model uses average pooling at the end for final prediction. Thus, our approach can be used with KPConv as well as PointCNN [1] by replacing average pooling with max-pooling, and this will be performed as future work. For 3D CNN, 4D CNN, and voxel-based segmentation methods, our method cannot be readily applied, since they do not employ max-pooling.

References

[1] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. Advances in neural information processing systems, 31:820–830, 2018. 3

[2] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 6411–6420, 2019. 3