Supplementary Material
PU-Transformer: Point Cloud Upsampling Transformer

Shi Qiu[1,2], Saeed Anwar[1,2], and Nick Barnes[1]

[1] Australian National University
[2] Data61-CSIRO, Australia
{shi.qiu, saeed.anwar, nick.barnes}@anu.edu.au

1 Overview

This supplementary material includes behavior analysis, ablation studies, limitations, visualizations, and future direction of our proposed PU-Transformer for point cloud upsampling.

2 Behavior Analysis

2.1 Positional Fusion Block

Although our Positional Fusion block utilizes similar operations as the Local Context Fusion (LCF) block proposed in [1], there are three main differences between these two methods. First, our block operates on the *patches* of point clouds that have explicit borders, while the LCF extracts the local context from a *whole* point cloud where more outliers could be involved. Second, all of our blocks in PU-Transformer share the *same* geometric relations, but each LCF block requires a *distinct* geometric relation that is specified in the corresponding point cloud resolution. Last but not least, our block serves as a feature *encoding* block that helps to gradually expand the channel dimension of the point cloud feature map, while the LCF aims to *refine* the feature representations in the same embedding space of the input.

We also investigate the embedding design in the Positional Fusion block as shown in Table 1. Coupled with models $A1$-$A3$ in the main paper's Tab. 3, our embedding method (*i.e.*, Eq. 5 in the main paper) is verified to learn better local feature representations than DGCNN's approach (*i.e.*, "D_2" in Table 1).

In addition, the effects of our Positional Fusion block can be analyzed from the comparisons in Figure 1: by applying our proposed block, the generated points can better align with the contour of a point cloud object, retaining high-fidelity local detail with fewer outliers.

Table 1: Ablation study of the Positional Fusion block's embedding design. $\Delta\mathcal{P}$: Eq. 1 in the main paper; $\Delta\mathcal{F}$: Eq. 3 in the main paper (*i.e.*, "*EdgeConv*" in DGCNN [2]).

models	Embedding Design		Results ($\times 10^{-3}$)		
	$\mathcal{G}_{geo}$	$\mathcal{G}_{feat}$	**CD** $\downarrow$	**HD** $\downarrow$	**P2F** $\downarrow$
D_1	$\Delta\mathcal{P}$	None	0.524	6.129	1.961
D_2	None	$\Delta\mathcal{F}$	0.633	5.331	2.252
D_3	$\Delta\mathcal{P}$	$\Delta\mathcal{F}$	0.480	5.172	1.602
Ours	$\text{concat}\left[\underset{k}{\text{dup}}(\mathcal{P}); \Delta\mathcal{P}\right]$	$\text{concat}\left[\underset{k}{\text{dup}}(\mathcal{F}); \Delta\mathcal{F}\right]$	**0.451**	**3.843**	**1.277**

(a) PU-Transformer w/o the Positional Fusion block

(b) PU-Transformer with the Positional Fusion block

Fig. 1: Upsampling results of the PU-Transformer *with* and *without* using the Positional Fusion block.

2.2 SC-MSA Block

In Sec. 3.3 of the main paper, we state that it is easier for our SC-MSA block to integrate the information between the connected multi-head outputs compared to regular multi-head self-attention (MSA) [3]. The main reason can be explained as follows: since two consecutive heads share some input channels, both of the two heads' outputs are affected/regulated by such shared channel-wise information, leading to less varying estimations of point-wise dependencies. As *any* two consecutive heads in our SC-MSA will follow the above manner, the outputs of all connected multi-heads become less varying, benefiting the overall estimations of point-wise dependencies.

To further compare regular MSA and our SC-MSA, we test their performances given different numbers of transformer encoders:

Table 2: Performances of the PU-Transformer with different numbers of Transformer Encoder. All metric units are 10^{-3}.

# Transformers	Attn Type	# Parameters	**CD ↓**	**HD ↓**	**P2F ↓**	**Total Changes**
$L = 3$	MSA	385.4k	0.534	4.664	1.696	↓ 0.964
	SC-MSA	438.3k	0.487	4.081	1.362	
$L = 4$	MSA	482.0k	0.506	4.447	1.545	↓ 0.732
	SC-MSA	547.3k	0.472	4.010	1.284	
$L = 5$	MSA	855.5k	0.498	4.218	1.427	↓ 0.572
	SC-MSA	969.9k	0.451	3.843	1.277	

In the above Table 2, we observe that SC-MSA's gain is more significant in a lighter PU-Transformer model (*i.e.*, with fewer Transformer Encoders), while the parameter increase is affordable. Moreover, there is practical evidence to support our argument: as the evaluation curves plotted in Figure 2, we clearly observe that our SC-MSA assists faster convergence and better performance on the test set than the regular MSA method.

3 Ablation Studies

3.1 Normalization Operations

As indicated in Fig. 2 and Alg. 1 of the main paper, the Transformer Encoder incorporates two normalization operations in the fashion of transformers. In practice, NLP-related models favor layer normalization (LN) [4] while image-related methods prefer batch normalization (BN) [5]. In terms of the point cloud upsampling task, we select the type of normalization operations (*i.e.*, "Norm$_1$" and "Norm$_2$" in Table 3) in the PU-Transformer based on the practical performance. Table 3 shows the quantitative results of *five* possible options (D_1 to D_5), indicating that the two normalization operations are crucial while the effects of BN and LN are very similar. Considering the relative simplicity and effectiveness,

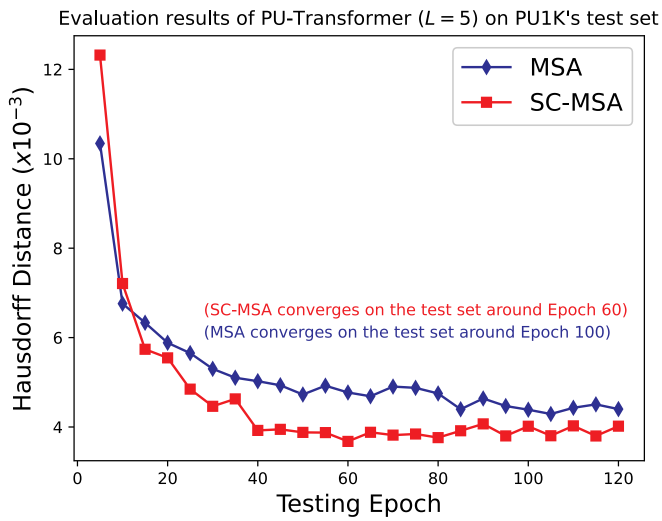


Fig. 2: The evaluation results of using SC-MSA or MSA [3] in the PU-Transformer body, respectively. Overall, compared to regular MSA method, our SC-MSA contributes to a better convergence and testing performance.

we thus adopt the LN operation for both "Norm_1" and "Norm_2" (*i.e.*, model D_5), in order to further regulate the point features encoded by our Positional Fusion and SC-MSA blocks.

3.2 PU1K and PU-GAN Datasets

Different from some works [7,8,9] testing their proposed models using their own data, we quantitatively evaluate the PU-Transformer on two public datasets: *PU1K* [6] and *PU-GAN* [10]. Particularly, we utilize the same experimental settings and results from PU-GCN [6] and Dis-PU [11], in order to have a fair comparison with state-of-the-art methods in Tab. 1 and 2 of the main paper. Moreover, we investigate the difference between the *PU1K* and *PU-GAN* datasets by *swapping* their training and testing data. According to the results (E_1&E_2, E_3&E_4) in Table 4, we find that given a small scale of training data[3], our PU-Transformer can still achieve a similar performance when using a large scale of training data[4]. In addition, as shown between E_1&E_3 or E_2&E_4, the test set of *PU1K* is more challenging than the *PU-GAN*'s, since there are 100 more testing samples in the *PU1K* dataset.

[3] 24,000 samples in the *PU-GAN* dataset
[4] 69,000 samples in the *PU1K* dataset

Table 3: PU-Transformer's quantitative results of using different normalization operations in the Transformer Encoder, tested on the *PU1K* dataset [6]. The best results are denoted in **bold**. ("Norm_1": the operation applied in step 4 of Alg. 1; "Norm_2": the operation applied in step 5 of Alg. 1; "BN": batch normalization [5]; "LN": layer normalization [4]; "**CD**": Chamfer Distance; "**HD**": Hausdorff Distance; "**P2F**": Point-to-Surface Distance.)

models	Norm_1	Norm_2	**CD**↓ ($\times 10^{-3}$)	**HD**↓ ($\times 10^{-3}$)	**P2F**↓ ($\times 10^{-3}$)
D_1	*none*	*none*	0.684	6.810	1.522
D_2	BN	BN	0.453	4.144	1.395
D_3	BN	LN	**0.441**	3.869	1.306
D_4	LN	BN	0.477	4.105	1.285
$\boldsymbol{D_5}$	LN	LN	0.451	**3.843**	**1.277**

Table 4: PU-Transformer's quantitative results when using different training and testing data from *PU1K* dataset [6] and *PU-GAN* dataset [10]. ("**CD**": Chamfer Distance; "**HD**": Hausdorff Distance; "**P2F**": Point-to-Surface Distance.)

models	training data	testing data	**CD**↓ ($\times 10^{-3}$)	**HD**↓ ($\times 10^{-3}$)	**P2F**↓ ($\times 10^{-3}$)
E_1	*PU1K*	*PU1K*	0.451	3.843	1.277
E_2	*PU-GAN*	*PU1K*	0.469	4.227	1.387
E_3	*PU1K*	*PU-GAN*	0.278	2.091	1.838
E_4	*PU-GAN*	*PU-GAN*	0.273	2.605	1.836

3.3 Testing on PU-GAN's Codebase

Our reported results in the main paper are testing on the codebase[5] of PU-GCN [6], which is also reported in Dis-PU [11] for a fair comparison. Moreover, we have tested the performance of PU-Transformer on PU-GAN's dataset and the original evaluation method[6], which are widely adopted in recent works. The upsampling results are in Table 5:

Table 5: PU-Transformer's quantitative results when using *PU-GAN*'s dataset [10] and evaluation method. ("**CD**": Chamfer Distance; "**HD**": Hausdorff Distance; "**P2F**": Point-to-Surface Distance; *N/A*: due to lack of ground truth points.)

Method	256 input points			2048 input points			4096 input points		
(10^{-3})	CD	HD	P2F	CD	HD	P2F	CD	HD	P2F
PU-GAN [10]	2.072	16.592	8.055	0.280	4.640	2.330	0.131	**1.284**	1.687
PU-EVA [12]	1.784	13.939	8.727	0.266	3.070	2.362	**0.123**	1.394	1.416
Ours	**1.506**	**12.820**	**6.903**	**0.248**	**1.791**	**1.838**	*N/A*	*N/A*	**1.249**

4 Visualizations

4.1 Upsampling Noisy Input

In Tab. 4 of the main paper, we quantitatively compare the PU-Transformer's robustness to random noise against other point cloud upsampling methods. Moreover, in Figure 3, we qualitatively visualize its upsampling results under different noise levels. Generally, our approach is robust to random noise since the upsampling results in all noisy cases retain the high-fidelity shapes. However, it is worth noting that the generated point cloud's uniformity can be affected as the noise level increases.

4.2 Upsampling Different Input Sizes

In Figure 4, we provide more examples to visualize our PU-Transformer's performance on upsampling various sizes of point cloud data. Similar to the effects shown in Fig. 5 of the main paper, given different numbers of input points, our proposed model can always generate dense output of high-quality.

[5] https://github.com/guochengqian/PU-GCN

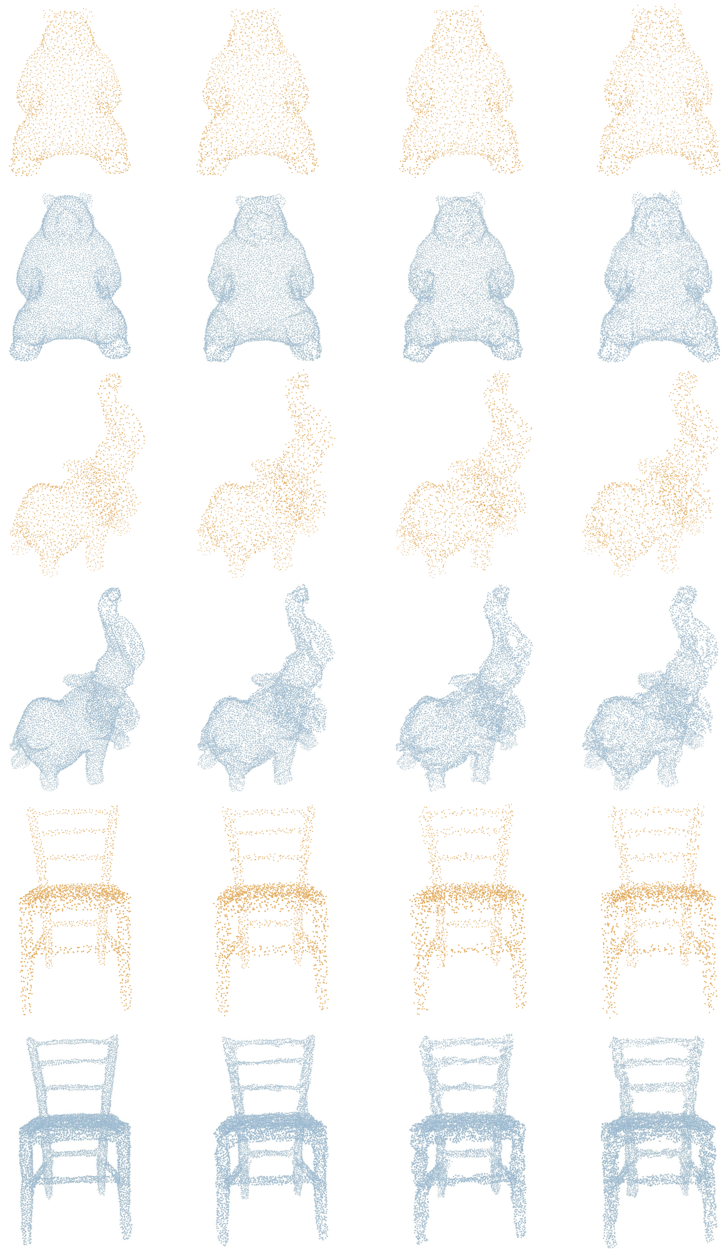
[6] https://github.com/liruihui/PU-GAN

4.3 Upsampling Real Point Clouds

We present a few examples of upsampling real point cloud data with our PU-Transformer. Particularly, Figure 5 illustrates the upsampled results of a LiDAR street [13], an indoor living room [14], a conference room [15], and some real-scanned objects [16]. In general, the overall quality of input data is significantly improved, where the generated points are well organized in a uniform distribution. For object instances (*e.g.*, "cars", and "chairs"), the representative features have been enhanced, benefiting an easier visual recognition.

5 Applications of Point Cloud Upsampling

We expect the proposed point cloud upsampling methods to better reconstruct *semantic* qualities benefiting *downstream tasks* such as classification [17,18], semantic segmentation [19,20] and object detection [21,22]. To demonstrate the feasiblity, we can make up a test by randomly selecting 256 (or 512) points from each original sample in the test set of classification benchmarks (*e.g.*, ModelNet40 [23] or ScanObjectNN [16]), apply different $4\times$ upsampling methods to generate 1024 (or 2048) points, and finally test the classification results using a same pretrained classification model (*e.g.*, DGCNN [2]).



(a) β = 0.5% (b) β = 1% (c) β = 1.5% (d) β = 2%

Fig. 3: Visualizations of PU-Transformer in upsampling noisy input point clouds, where the noise is generated from a standard normal distribution $\mathcal{N}(0, 1)$ and multiplied with a factor $\beta = 0.5\%$, 1%, 1.5%, and 2%, respectively. The input point clouds are in orange color, while the corresponding upsampled results are in blue.

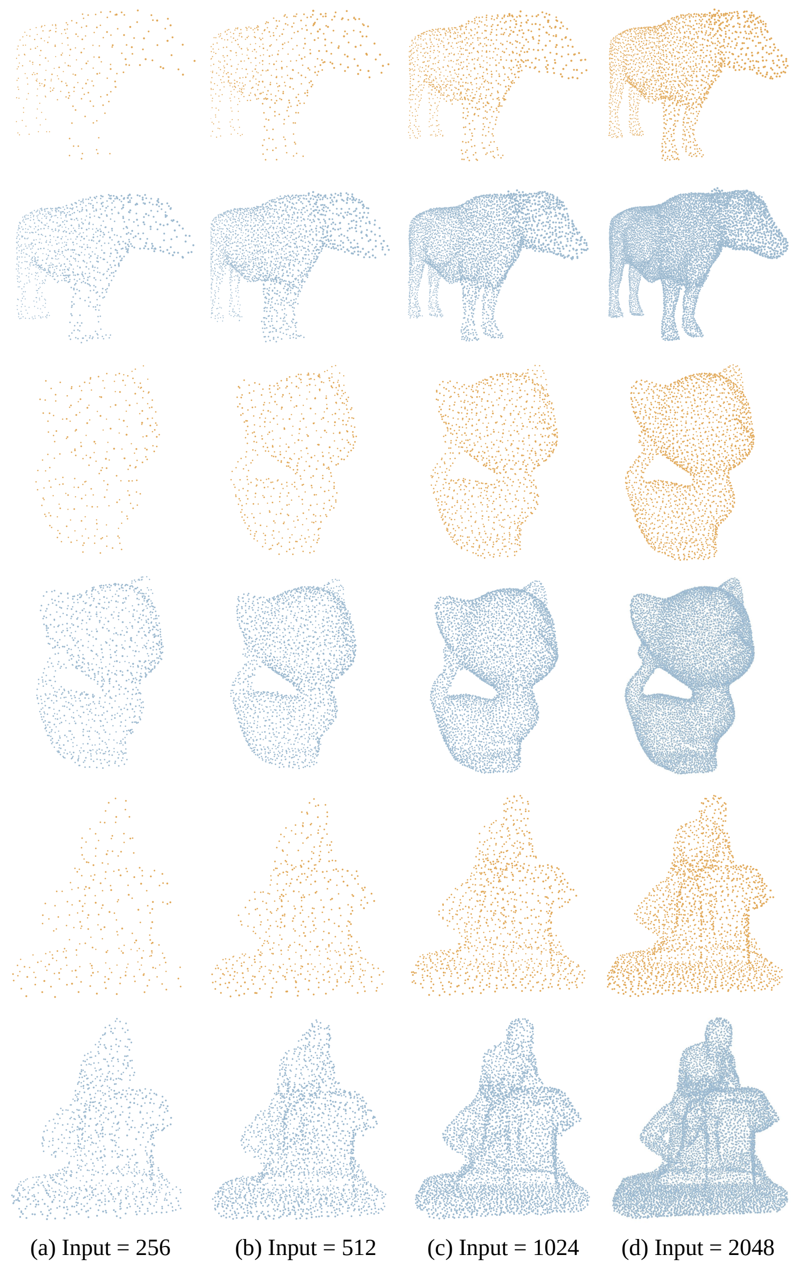


Fig. 4: Visualizations of PU-Transformer in upsampling different sizes of point clouds, where the number of input points is 256, 512, 1024, and 2048, respectively. The input point clouds are in orange color, while the corresponding upsampled results are in blue.

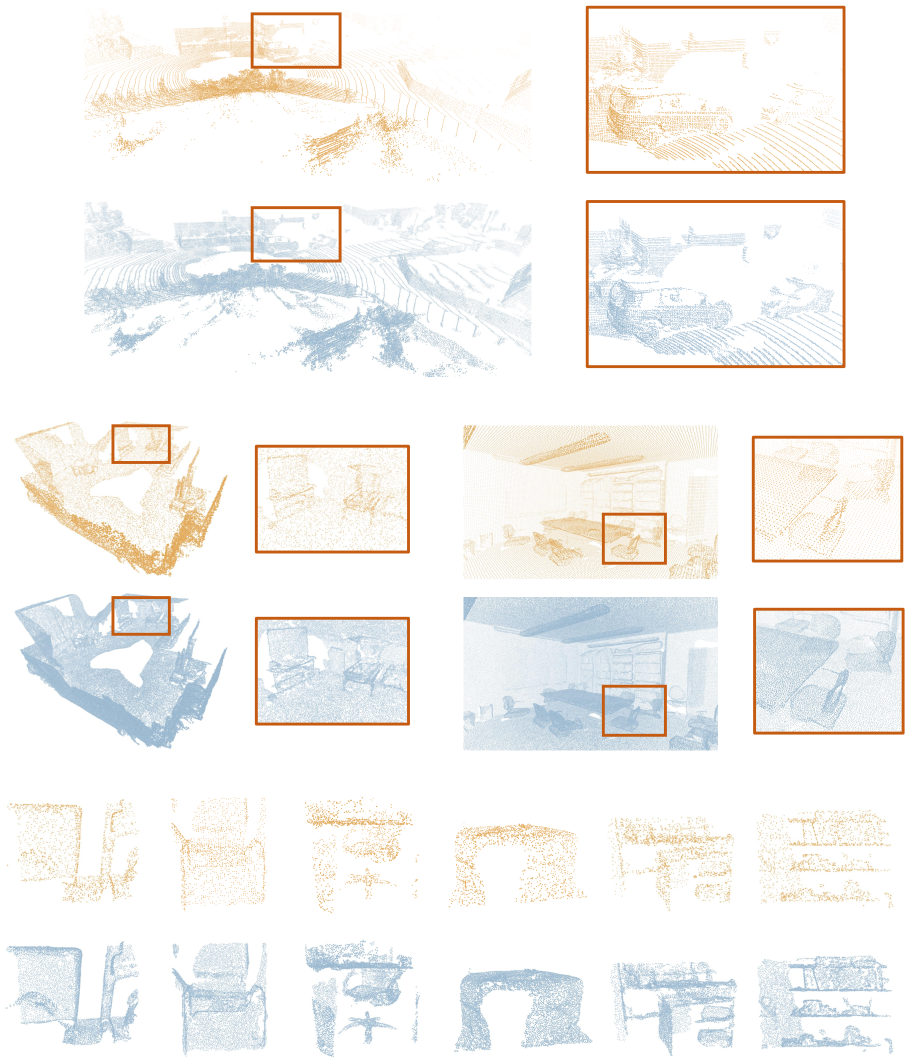


Fig. 5: Visualizations of PU-Transformer in upsampling real point clouds, including a LiDAR street (from SemanticKTTI dataset [13]), a living room (from ScanNet dataset [14]), a conference room (from S3DIS dataset [15]), as well as some real-scanned objects (from ScanObjectNN dataset [16]). The input point clouds are in orange color, while the corresponding upsampled results are in blue.

References

1. Qiu, S., Anwar, S., Barnes, N.: Pnp-3d: A plug-and-play for 3d point clouds. arXiv preprint arXiv:2108.07378 (2021)
2. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG) **38** (2019) 146
3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. (2017) 5998–6008
4. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
5. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
6. Qian, G., Abualshour, A., Li, G., Thabet, A., Ghanem, B.: Pu-gcn: Point cloud upsampling using graph convolutional networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 11683–11692
7. Yu, L., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: Pu-net: Point cloud upsampling network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 2790–2799
8. Yifan, W., Wu, S., Huang, H., Cohen-Or, D., Sorkine-Hornung, O.: Patch-based progressive 3d point set upsampling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 5958–5967
9. Qian, Y., Hou, J., Kwong, S., He, Y.: Pugeo-net: A geometry-centric network for 3d point cloud upsampling. In: European Conference on Computer Vision, Springer (2020) 752–769
10. Li, R., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: Pu-gan: a point cloud upsampling adversarial network. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 7203–7212
11. Li, R., Li, X., Heng, P.A., Fu, C.W.: Point cloud upsampling via disentangled refinement. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 344–353
12. Luo, L., Tang, L., Zhou, W., Wang, S., Yang, Z.X.: Pu-eva: An edge-vector based approximation solution for flexible-scale point cloud upsampling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2021) 16208–16217
13. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: Semantickitti: A dataset for semantic scene understanding of lidar sequences. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 9297–9307
14. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 5828–5839
15. Armeni, I., Sax, S., Zamir, A.R., Savarese, S.: Joint 2d-3d-semantic data for indoor scene understanding. arXiv preprint arXiv:1702.01105 (2017)
16. Uy, M.A., Pham, Q.H., Hua, B.S., Nguyen, T., Yeung, S.K.: Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2019) 1588–1597

17. Qiu, S., Anwar, S., Barnes, N.: Geometric back-projection network for point cloud classification. IEEE Transactions on Multimedia (2021)
18. Qiu, S., Anwar, S., Barnes, N.: Dense-resolution network for point cloud classification and segmentation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). (2021) 3813–3822
19. Qiu, S., Anwar, S., Barnes, N.: Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (2021) 1757–1767
20. Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: Randla-net: Efficient semantic segmentation of large-scale point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 11108–11117
21. Qiu, S., Wu, Y., Anwar, S., Li, C.: Investigating attention mechanism in 3d point cloud object detection. In: International Conference on 3D Vision (3DV), IEEE (2021)
22. Qi, C.R., Chen, X., Litany, O., Guibas, L.J.: Imvotenet: Boosting 3d object detection in point clouds with image votes. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. (2020) 4404–4413
23. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 1912–1920