Walk in the Cloud: Learning Curves for Point Clouds Shape Analysis Supplementary Materials

Tiange Xiang
University of Sydney
txia7609@uni.sydney.edu.au

Chaoyi Zhang
University of Sydney
chaoyi.zhang@sydney.edu.au

Yang Song
University of New South Wales
yang.song1@unsw.edu.au

Jianhui Yu
University of Sydney
jianhui.yu@sydney.edu.au

Weidong Cai
University of Sydney
tom.cai@sydney.edu.au

1. CurveNet Architecture Details

Local Point-Feature Aggregation (LPFA). Local feature aggregation is a basic feature propagation operator in our CurveNet. As suggested by LocSE block [4], we first encode the relative point coordinates as $[\mathbf{p}, \mathbf{p}^k, \mathbf{p}^k - \mathbf{p}]$, where $\mathbf{p}^k$ is one of the KNN neighbors of $\mathbf{p}$. We then linearly transform the relative encodings to adapt the point features. Building upon [4], we adopt an extra MLP to better propagate the element-wise integrated features. An average pooling layer is placed at the end of the block to summarize local descriptors for each neighborhood efficiently.

We employ a simplified LPFA block at the start of our CurveNet to learn an initial point-wise local spatial encodings. Such design provides subsequent network with better local descriptions than the raw coordinates.

Curve Intervention Convolution (CIC). We embed the proposed Curve Grouping operator (CG) and Curve Aggregation operator (CA) in the CIC block to enable a better point cloud feature propagation. Our CIC block follows a bottleneck design that uses a MLP at the start of the block to project the incoming features into a lower dimension for computational efficiency. We then apply the KNN algorithm to fetch and group the local neighbors based on their euclidean distance. As discussed in main paper Sec 3.1, long-ranged dependency features need to be fused to the same local neighbors, hence the key point of the CIC block design lies at using the KNN grouped index for both local aggregation and curve aggregation. To this end, we choose to apply KNN on point coordinates rather than point features. After computing the KNN index, we use the top-k selection method [2] to get the starting point set which is later used in the subsequent CG module for curve grouping. Curve feature aggregation is followed by a LPFA block that learns further local features on the same KNN indexed neighbors. At the end of a CIC block, we reproject the low dimension features back to a high dimensional space through a paired MLP.

We adopt Farthest Point Sampling (FPS) at the very start of the CIC block when a downsampling of the point cloud is required. Instead of a direct use of the FPS indices for points sifting, we apply ball query on the FPS indices [8] to find the most representative points with maximum feature values within each ball.

Table 1. Network complexity and effectiveness.

Methods	#Params	FLOPs	M40 Acc
PointNet [10]	3.47 M	0.45 G	89.2%
PointNet++ [11]	1.48 M	1.68 G	90.7%
DGCNN [13]	1.81 M	2.43 G	92.9%
RS-CNN [7]	1.41 M	0.30 G	92.9%
PCT [3]	2.88 M	2.17 G	93.2%
CurveNet (Ours)	2.14 M	0.66 G	93.8%

Task-Specified Head (TSH). The final predictions of our CurveNet are obtained through different heads regarding of multiple tasks. For classification tasks, we first use a point-wise MLP to project the incoming features to a higher dimension. We then concatenate the max pooled and average pooled tensors and pass them into two fully-connected layers with a dropout layer to derive the classification scores.

For segmentation and normal estimation tasks, we follow the attention U-Net [9] design that encoded features are skipped to the upsampling layer [10] at each level. Additional MLPs and a dropout layer are used at the end of the last upsampling layer to give the point-wise predictions.

CurveNet Structure. For all object analysis tasks, we place a simplified LPFA block at the start of our CurveNet for better relative local spatial encodings. We stack four building groups each with two CIC blocks as the feature extractor backbone for classification tasks (Table 2). Additional CIC blocks are used in the task-specified head in segmentation and normal estimation tasks, as presented in Table 3 and Table 4, respectively.

Table 2. CurveNet structure details for the classification tasks.

Block	#in	#out	#points	w/ curves
LPFA	3	32	1024	
CIC	32	64	1024	✓
CIC	64	64	1024	✓
CIC	64	128	512	✓
CIC	128	128	512	✓
CIC	128	256	256	
CIC	256	256	256	
CIC	256	512	64	
CIC	512	512	64	
TSH	512	#class	-	

Table 3. CurveNet structure details for the segmentation task.

Block	#in	#out	#points	w/ curves
LPFA	3	32	2048	
CIC	32	64	2048	✓
CIC	64	64	2048	✓
CIC	64	128	512	✓
CIC	128	128	512	✓
CIC	128	256	128	
CIC	256	256	128	
CIC	256	512	32	
CIC	512	512	32	
CIC	512	1024	8	
CIC	1024	1024	8	
TSH	1024	#class	2048	

Table 4. CurveNet structure details for the normal estimation task.

Block	#in	#out	#points	w/ curves
LPFA	3	64	1024	
CIC	64	256	1024	✓
CIC	256	256	1024	✓
CIC	256	512	256	✓
CIC	512	512	256	✓
CIC	512	1024	64	
CIC	1024	1024	64	
CIC	1024	2048	16	
CIC	2048	2048	16	
TSH	2048	3	1024	

2. Additional Ablation Studies

Complexity analysis. Computational complexity is essential to point cloud analysis. Here we report the network complexity of our CurveNet and competing methods collected from other literatures in Table 1. Compared to the most recent state-of-the-art method [3], our CurveNet achieves superior ModelNet40 classification accuracy with significantly lower parameter amounts and FLOPs.

Table 5. Extensive ablation studies.

Methods	Acc (%)	latency (ms)
CurveNet w/o curves	93.3	37.5 (150)
CurveNet w/ random walk	93.0	42.0 (155)
CurveNet w/ point-wise non-local only	93.1	40.3 (158)
CurveNet w/ learnable tolerance $\bar{\theta}$	93.3	48.3 (158)
CurveNet w/ curves	93.8	45.2 (146)

Figure 1. **Applying curve aggregation at different groups.** The y-axis stands for the classification accuracy (%), x-axis stands for the curve length, black dotted line stands for CurveNet **w/o** curves. Note that downsampling is enabled at group 3&4, and we adjust the curve parameters accordingly.

Shallow layer vs deep layer. The results of aggregating curves with various quantities and lengths on different groups of our CurveNet are shown in Figure 1. Aggregating curves at shallow layers (group 1/2) yield better results than at deep layers (group 3/4). By aggregating 100 curves with length 5 at both building group 1 and 2, our CurveNet reaches 93.84% classification accuracy on the ModelNet40 benchmark.

Curve quantity vs curve length. Figure 1 bottom right shows the results on fixed total point number in curves. We found as expected that the curves of relatively longer length could capture more geometric traits to achieve greater results, however, they are also empirically found more likely to be trapped at local regions in some cases (Figure 2).

Guided walk vs random walk. As discussed in main paper Sec. 3.2, guided walk with a deterministic π can be

easily trapped at local positions. However, random walk with a stochastic π will merely be constrained and is able to explore wide range relations without extra computations. To validate the effectiveness of such stochastic policy, we simply modify our method by determining the next transition state *randomly* at each moment, instead of the state with the *highest* score. We report the result with random walk in Table 5. Although using stochastic π infers slightly faster, we reckon that the object shapes cannot be described with randomly formed curves and hence leads to an obvious performance drop.

Curve aggregation vs point-wise non-local. Similar to the proposed curve aggregation operator, non-local aggregation [12] also integrates long-range (global) information to the current local feature space. However, the all-to-all mapping does not consider continuous segments of object shapes and hence, cannot cooperate with our CurveNet well. The ablation result of replacing the proposed CG&CA in our CurveNet with the point-wise feature non-local module is presented in Table 5. In our experiment, point-wise non-local aggregation results in poorer classification accuracy and even longer network inference latency.

3. Additional Results

3.1. Additional Results on ModelNet40

We visualize more grouped curves on ModelNet40 validation objects in Figure 2. As aforementioned in main paper Sec 4.3, we observed that long curves may lack adequate guidance and are more easily to be trapped at local regions than the short ones (red arrows in Figure 2). We attribute such glitch on long curves to the usage of a simple MLP (Eq. 3, main paper) in $\phi(\cdot)$. We believe that with a better learning protocol (e.g. more advanced message passing layer instead of a simple MLP) and a better starting point proposal strategy (instead of the simple Top-K selection [2]), the long curves might get better guidances and more scattered starting points to explore longer range information more effectively.

3.2. Additional Results on ShapeNetPart

The category-wise mIoU scores are presented in Table 6, and more qualitative segmentation results along with the grouped curves are shown in Figure 3. Our CurveNet achieves state-of-the-art results in terms of the overall performances and 9 specific objects.

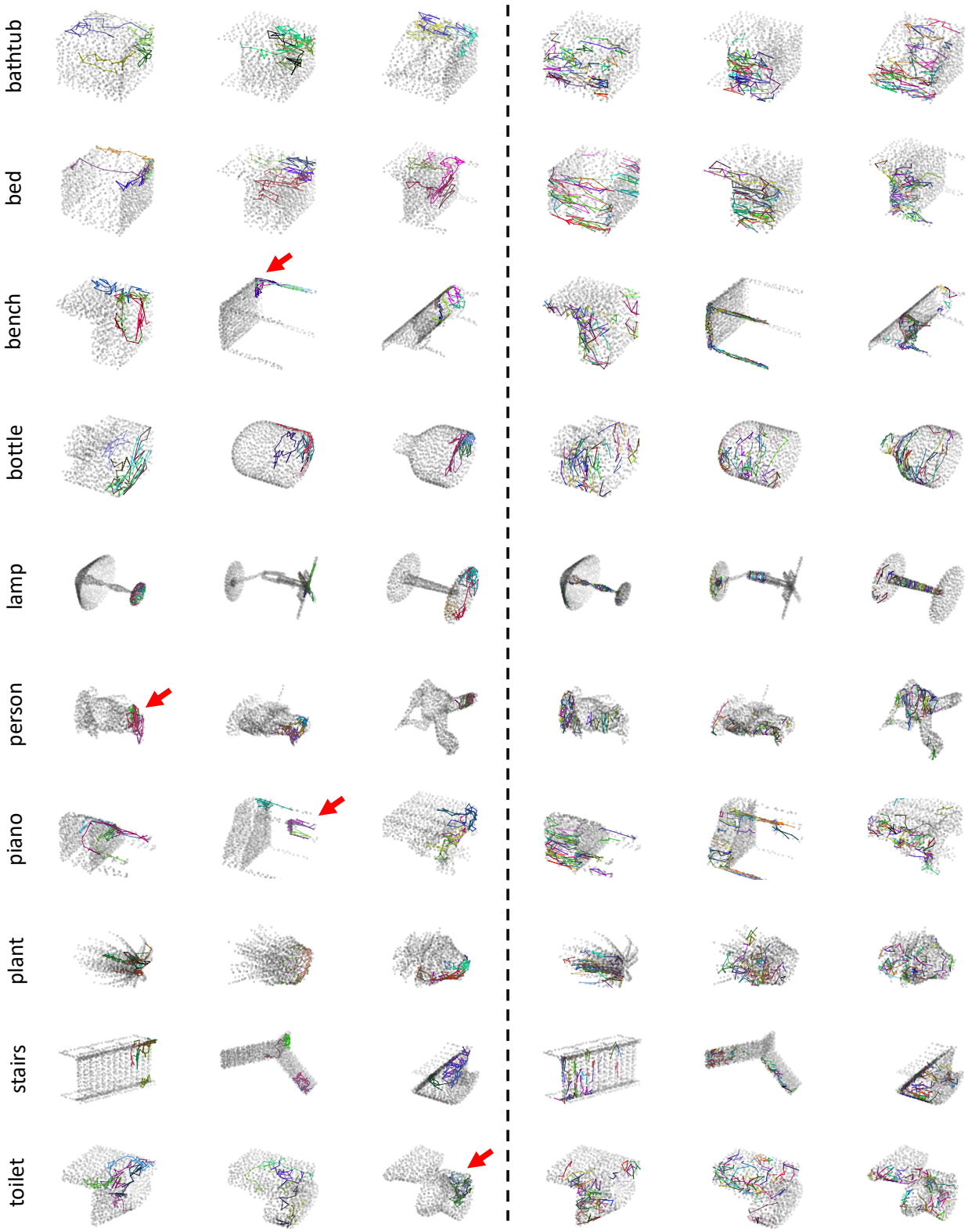

Figure 2. **Curves on the ModelNet40 validation dataset.** Left shows 5 randomly picked long-curve ($l = 30$) cases on three different instances in 10 classes. Right shows 50 randomly picked short-curve ($l = 5$) cases on the same objects. Red arrows point to the area where long curves are trapped in local regions. Curves are plotted in random colors. Better zoomed in for details.

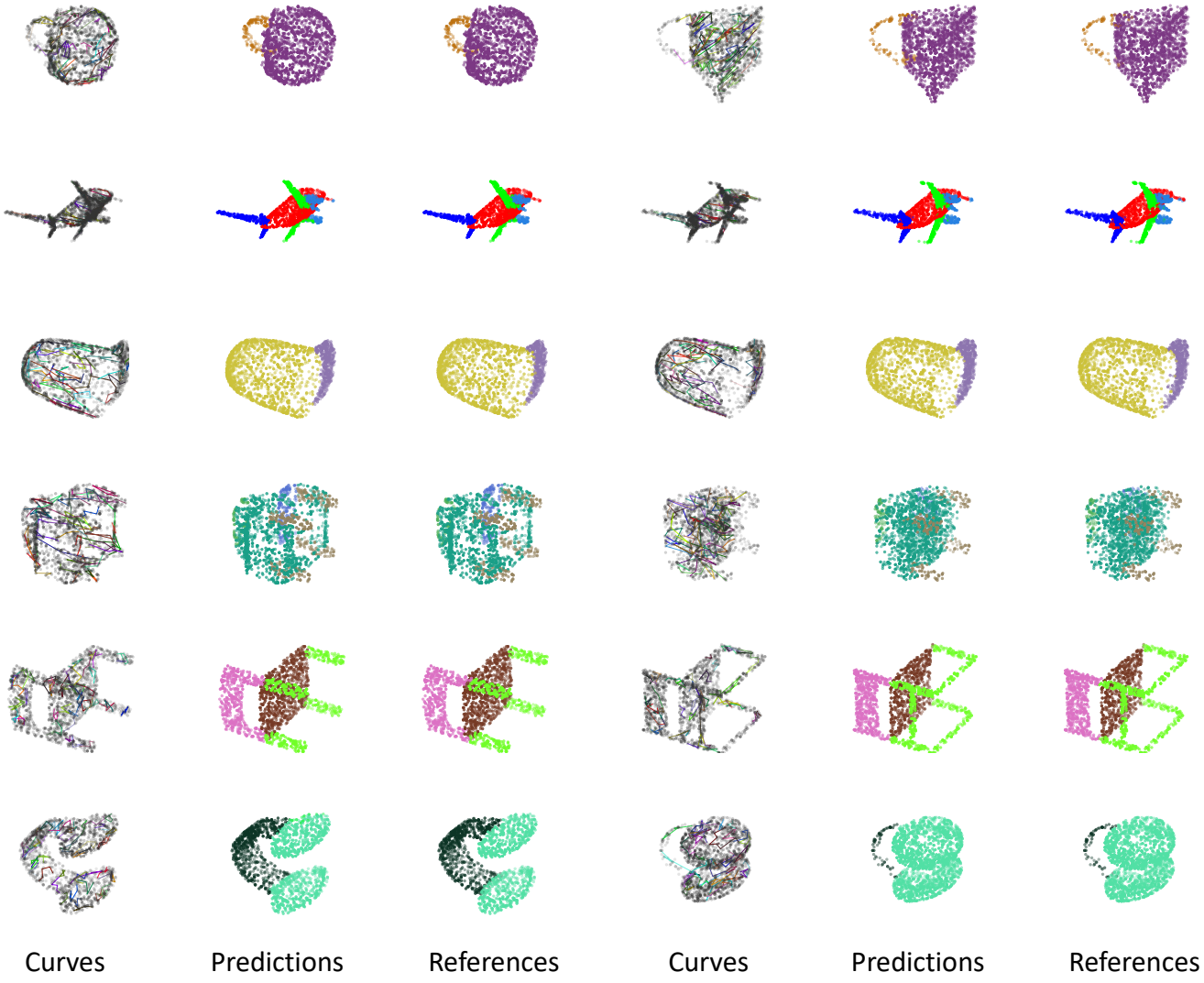

Figure 3. **Curves and segmentation results on the ShapeNetPart validation dataset.** We show 50 randomly picked curves ($l = 5$), our CurveNet predictions, and ground truth references on two instances of each of the 5 classes. Curves are plotted in random colors. Better zoomed in for details.

Table 6. ShapeNetPart validation results in mean intersection of union (%). '*' denotes methods evaluated with voting strategy [7].

Methods	mIoU	air plane	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor bike	mug	pistol	rocket	skate board	table
PointNet [10]	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
DGCNN [13]	85.1	84.2	83.7	84.4	77.1	90.9	78.5	91.5	87.3	82.9	96.0	67.8	93.3	82.6	59.7	75.5	82.0
PointCNN [6] *	86.1	84.1	86.5	86.0	80.8	90.6	79.7	**92.3**	88.4	85.3	96.1	**77.2**	95.3	**84.2**	**64.2**	**80.0**	83.0
RS-CNN [7] *	86.2	83.5	84.8	**88.8**	79.6	91.2	81.1	91.6	88.4	86.0	96.0	73.7	94.1	83.4	60.5	77.7	83.6
LocAL-Net [1] *	86.2	84.1	**88.8**	86.7	78.8	91.2	**83.3**	91.9	88.6	84.9	95.7	72.5	94.8	83.6	60.0	77.1	84.0
PCT [3] *	86.4	85.0	82.4	89.0	81.2	91.9	71.5	91.3	88.1	86.3	95.8	64.6	**95.8**	83.6	62.2	77.6	83.7
PointNet++ [11]	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
SO-Net [5]	84.6	81.9	83.5	84.8	781	90.8	72.2	90.1	83.6	82.3	95.2	69.3	94.2	80.0	51.6	72.1	82.6
CurveNet (ours)	86.6	84.8	83.8	87.6	80.2	91.8	76.1	91.5	88.6	**86.4**	96.1	72.6	95.3	82.9	59.2	76.2	83.9
CurveNet (ours) *	**86.8**	**85.1**	84.1	**89.4**	**80.8**	**91.9**	75.2	91.8	**88.7**	86.3	**96.3**	72.8	95.4	82.7	59.8	78.5	**84.1**

References

[1] Qendrim Bytyqi, Nicola Wolpert, and Elmar Schömer. Local-area-learning network: Meaningful local areas for efficient point cloud analysis. *arXiv preprint arXiv:2006.07226*, 2020. 5

[2] Hongyang Gao and Shuiwang Ji. Graph U-Nets. *arXiv preprint arXiv:1905.05178*, 2019. 1, 3

[3] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *arXiv preprint arXiv:2012.09688*, 2020. 1, 2, 5

[4] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020. 1

[5] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-Net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018. 5

[6] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on X-transformed points. In *Advances in neural information processing systems*, pages 820–830, 2018. 5

[7] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019. 1, 5

[8] Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong. A closer look at local aggregation operators in point cloud analysis. *arXiv preprint arXiv:2007.01294*, 2020. 1

[9] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention U-Net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018. 1

[10] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 5

[11] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 1, 5

[12] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 3

[13] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions On Graphics (TOG)*, 38(5):1–12, 2019. 1, 5