Supplementary Material for
KPConv: Flexible and Deformable Convolution for Point Clouds

Abstract

This supplementary document is organized as follows:

- *Sec. 1 details our network architectures, the training parameters, and compares the model sizes and speeds.*
- *Sec. 2 presents the kernel point initialization method.*
- *Sec. 3 describes how our regularization strategy tackles the "lost" kernel point phenomenon.*
- *Sec. 4 enumerates more segmentation results with class scores.*
- ***KPConv Method** video[1] illustrates KPConv principle with animated diagrams, and shows some learned kernel deformations.*
- ***KPConv Results** video[2] shows indoor and outdoor scenes segmented by KP-FCNN.*

1. Network Architectures and Parameters

As explained in the main paper, our architectures are built with convolutional blocks, designed like bottleneck ResNet blocks [6]. This is the case whether we use a normal or strided KPConv, with rigid or deformable kernels. Figure 1 describes these blocks, and Figure 2 our two network architectures made from them. In Figure 2, we show an example of point cloud from ModelNet40 dataset, subsampled at every layer. It illustrates how the convolution radius (red sphere) grows proportionally to the subsampling grid size. In all our experiments with deformable KPConv, we use deformable kernels in the last 5 convolutional blocks (2^{nd} block from layer 3, and both block from layer 4 and 5). The green number above layers in Figure 2 are the feature dimensions used in our blocks (D in Figure 1).

Our layers process point clouds of variable sizes, so we cannot stack them along a new "batch" dimension. We thus stack our point and feature tensors along their first dimension (number of points). As the neighbor and pooling indices do not point from one input cloud to another, each batch element is processed independently without any implementation trick. We only need to keep track of the batch element indices in order to define the global pooling of **KP-CNN**. Since the number of points can vary a lot, we use a variable batch size by selecting as many elements as possible until a certain number of total of batch points is reached. This limit is chosen so that the average batch size correspond to the target batch size. A very similar batch strategy has already been described by [7].

KP-CNN training. We use a Momentum gradient Descent optimizer to minimize a cross-entropy loss, with a batch size of 16, a momentum of 0.98 an initial learning rate of 10^{-3}. Our learning rate is scheduled to decrease exponentially, and we choose the exponential decay to ensure it is divided by 10 every 100 epochs. A 0.5 probability dropout is used in the final fully connected layers. The network converges in 200 epochs. In the case of deformable kernels, the regularization loss is added to the output loss with a multiplicative factor of 0.1.

KP-FCNN training. We also use a Momentum gradient Descent optimizer to minimize a point-wise cross-entropy loss, with a batch size of 10, a momentum of 0.98 an initial learning rate of 10^{-2}. The same learning rate schedule is used and no dropout is used. Among all experiments, the network needs 400 epochs at most to converge. For

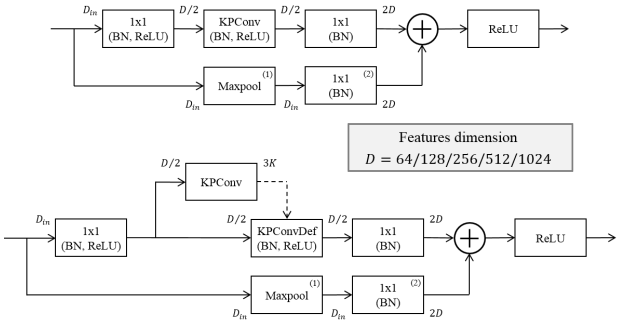

Figure 1. Convolutional blocks used in our architectures. Both rigid (top) and deformable (bottom) KPConv use resnet connections, batch normalization and leaky ReLU. Optional blocks: shortcut max pooling$^{(1)}$ is only needed for strided KPConv, and shortcut 1x1 convolution$^{(2)}$ is only needed when $D_{in} \neq 2D$.

[1] https://www.youtube.com/watch?v=uwuvp9mc_0o&t=19s
[2] https://www.youtube.com/watch?v=_cFQxJorSAI

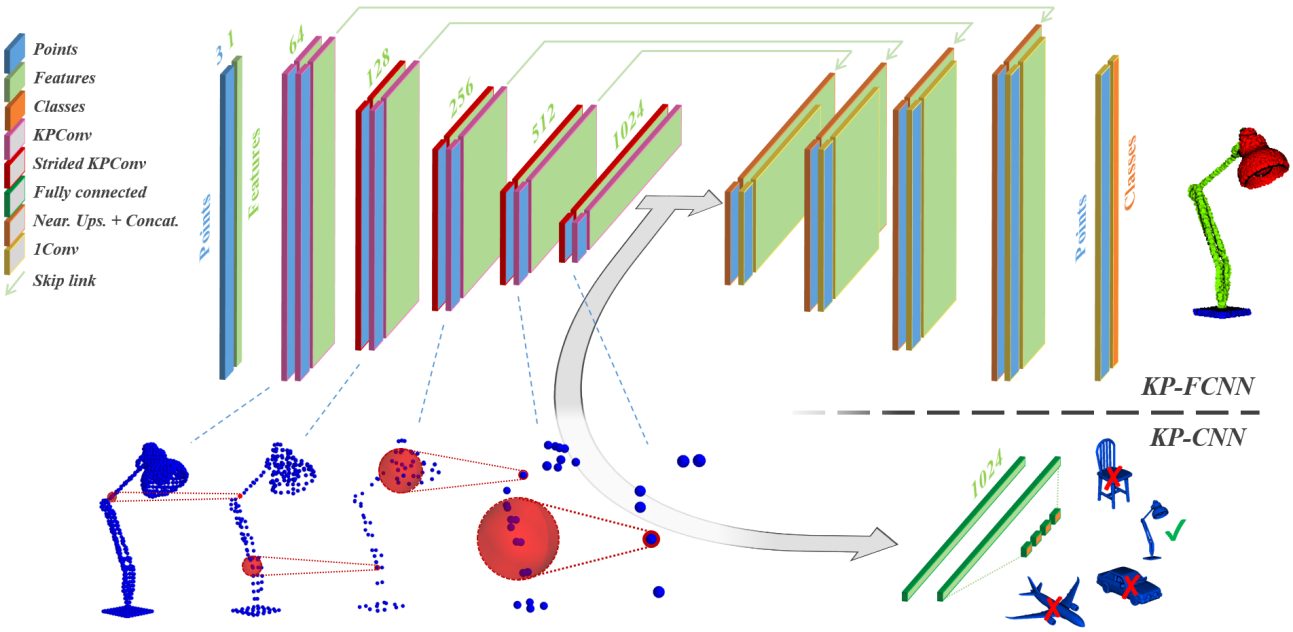

Figure 2. Illustration of our 2 network architectures for segmentation (top) and classification (bottom) of 3D point clouds. During a forward pass, features are transformed by consecutive operations (represented by edge colors) while points are fed to each layer as a support structure guiding the operations.

real scene segmentation, we can generate any number of input spheres, so we define an epoch as 500 optimizer steps, which is equivalent to 5000 spheres seen by the network. The same deformable regularization loss is used.

Model sizes and speeds. Table 1 shows the statistics of our models on different datasets. First we notice that KP-FCNN and KP-CNN have similar number of parameters, because the decoder part of KP-FCNN only involves light 1x1 convolution. We see that the running speeds are different from one dataset to another, which is not surprising. Indeed, the number of operations performed during a forward pass of our network depends on the number of points of the current batch, and the maximum number of neighbors of these points. Our models have been prototyped with a RTX 2080Ti in this experiment, which explains the slight difference with the Titan Xp used in the main paper.

		MN40	SNP	Scannet	Sem3D
Avg pts/elem		6800	2370	8950	3800
Avg pts/batch		109K	38K	90K	38K
Params	*rigid*	14.3M	14.2M	14.1M	14.1M
	deform	15.2M	15.0M	14.9M	14.9M
Training (batch/s)	*rigid*	3.5	5.5	4.3	8.8
	deform	3.1	4.3	3.9	7.1
Inference (batch/s)	*rigid*	8.7	16.7	9.3	17.5
	deform	8.0	12.2	8.1	15.0

Table 1. Model statistics on 4 datasets: ModelNet40, ShapeNetPart, Scannet, Semantic3D.

2. Kernel Points Initialization

Our KPConv operates in a ball, and requires kernel points regularly placed in this domain. There is no obvious regular disposition of points in a sphere, so we chose to solve this issue by translating it into an optimization problem. The problem is simple, we want the K points $\widetilde{x}_k$ to be as far from each other as possible inside a given sphere. We thus assign a repulsive potential to each point:

$$\forall x \in \mathbb{R}^3, \quad E_k^{rep}(x) = \frac{1}{\|x - \widetilde{x}_k\|} \tag{1}$$

And add an attractive potential to the sphere center to avoid them diverging indefinitely:

$$\forall x \in \mathbb{R}^3, \quad E^{att}(x) = \|x\|^2 \tag{2}$$

The problem then consists of minimizing the global energy:

$$E^{tot} = \sum_{k<K} \left(E^{att}(\widetilde{x}_k) + \sum_{l \neq k} E_k^{rep}(\widetilde{x}_l) \right) \tag{3}$$

The solution is found by gradient descent with the points initialized randomly and some optional constraints. In our case, we fix one of the points at the center of the sphere. For

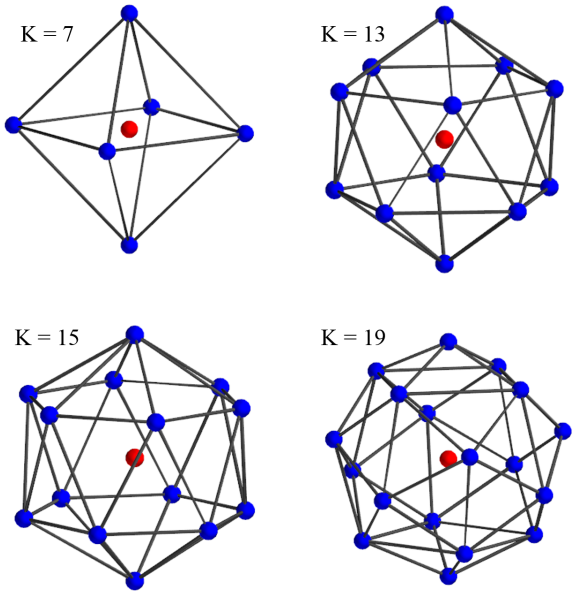

Figure 3. Illustration of the kernel points in stable dispositions.

some values of K (listed in Table 2), the points converge to a unique stable disposition. Those stable dispositions are in fact regular polyhedrons. Each polyhedron can be described by grouping points sharing a plane perpendicular to the polyhedron symmetrical axis. For a better understanding, some of these dispositions are shown in Figure 3.

In every layer of KP-CNN and KP-FCNN, the points locations are rescaled from the chosen stable disposition to the appropriate radius and randomly rotated. Note that E^{tot} can also be used as a regularization loss in KP-CNN, when the kernel point positions are trained.

K	disposition name	groups along symmetrical axis
5	Tetrahedron	
7	Octahedron	1-4-1
13	Icosahedron	1-5-5-1
15	-	1-6-6-1
18	-	1-5-5-5-1
19	-	1-4-4-4-4-1
21	-	1-6-6-6-1
25	-	4-4-4-4-4-4

Table 2. Stable dispositions of the kernel point positions when the center point is fixed. If a disposition has an axis of symmetry, we describe it by the successive groups of points sharing a plane perpendicular to this axis.

3. Effect of the Kernel Point Regularization

When we designed deformable KPConv, we first used a straightforward adaptation of image deformable convolutions, but the network had very poor performances. We investigated the kernel deformations after the network convergence and noticed that the kernel points were often pulled away from the input points. This phenomenon comes from the sparse nature of point clouds, there is empty space around the points. We remind that the shifts are predicted by the network, thus, they depend on the input shape.

For a particular input during training, if a kernel point is shifted away from the input points, then the gradient of its shift $\Delta_k(x)$ is null. It is thus "lost" by the network and remains away for similar input shapes. Because of the stochastic nature of the network optimizer, this happens for many input shapes during convergence.

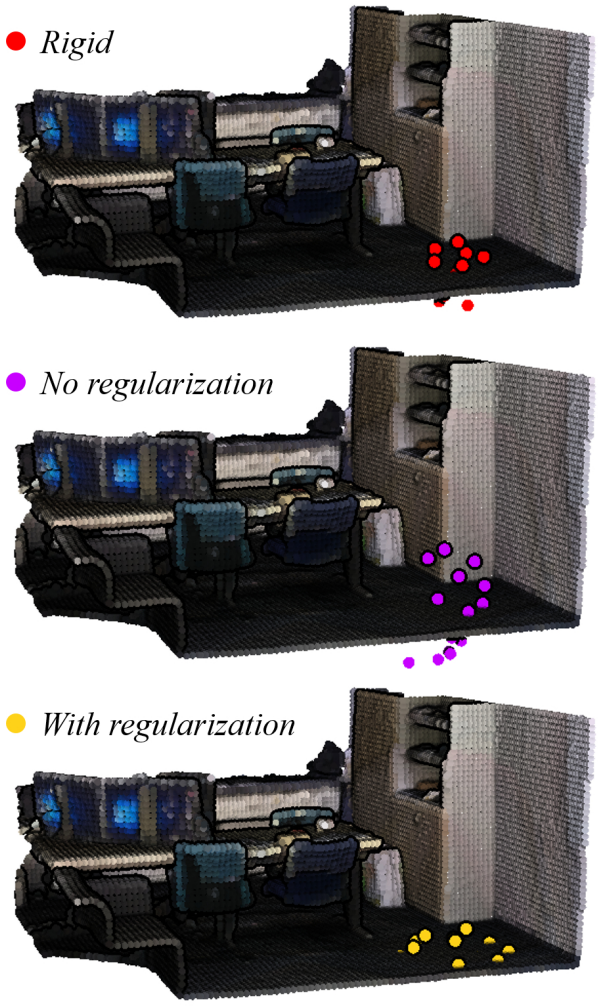

Figure 4. Illustration of the deformations learned by a KPConv network with or without regularization.

Figure 4 illustrates "lost" kernel points on the example of a room floor. First we see the rigid kernel in red, its scale gives an idea of the kernel points influence range. In the middle, the purple points depict a deformed kernel predicted by a network without any regularization loss. Most purple points are far from the floor plane and thus "lost".

Our regularization strategy, described in the main paper, prevents this phenomenon, as shown in the bottom of Figure 4. We can notice that our regularization strategy does not only prevent the "lost" kernel points. It also helps to maximize the number of active kernel points in KPConv (those with input points in range). Almost every yellow point is close to the floor plane.

4. More Segmentation Results

In this section, we provide more details on our segmentation experiments, for benchmarking purpose with future works. We give class scores for our experiments on ShapeNetPart (Table 3) and S3DIS (Tables 4 and 5) dataset. Scannet [2], Semantic3D [5] and NPM3D [15] are online benchmarks, the class scores can be found on their respective website.

References

[1] M. Atzmon, H. Maron, and Y. Lipman. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics (TOG)*, 37(4):71, 2018.

[2] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. http://kaldir.vc.in.tum.de/scannet_benchmark.

[3] B. Graham, M. Engelcke, and L. van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9224–9232, 2018.

[4] F. Groh, P. Wieschollek, and H. P. Lensch. Flex-convolution. In *Asian Conference on Computer Vision*, pages 105–122. Springer, 2018.

[5] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys. Semantic3d. net: A new large-scale point cloud classification benchmark. *arXiv preprint arXiv:1704.03847*, 2017. http://www.semantic3d.net.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[7] P. Hermosilla, T. Ristchel, P.-P. Vázquez, Á. Vinacua, and T. Ropinski. Monte carlo convolution for learning on non-uniformly sampled

Method	class avg.	inst. avg.	aero	bag	cap	car	chair	ear	guit	knif	lamp	lapt	moto	mug	pist	rock	skate	table
Kd-Net [9]	77.4	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
SO-Net [11]	81.0	84.9	82.8	77.8	88.0	77.3	90.6	73.5	90.7	83.9	82.8	94.8	69.1	94.2	80.9	53.1	72.9	83.0
PCNN by Ext [1]	81.8	85.1	82.4	80.1	85.5	79.5	90.8	73.2	91.3	86.0	85.0	95.7	73.2	94.8	83.3	51.0	75.0	81.8
PointNet++ [14]	81.9	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
SynSpecCNN [23]	82.0	84.7	81.6	81.7	81.9	75.2	90.2	74.9	**93.0**	86.1	84.7	95.6	66.7	92.7	81.6	60.6	**82.9**	82.1
DGCNN [20]	82.3	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
SpiderCNN [21]	82.4	85.3	83.5	81.0	87.2	77.5	90.7	76.8	91.1	87.3	83.3	95.8	70.2	93.5	82.7	59.7	75.8	82.8
SubSparseCNN [3]	83.3	86.0	84.1	83.0	84.0	80.8	**91.4**	78.2	91.6	**89.1**	85.0	95.8	73.7	95.2	84.0	58.5	76.0	82.7
SPLATNet [16]	83.7	85.4	83.2	84.3	89.1	80.3	90.7	75.5	92.1	87.1	83.9	96.3	75.6	**95.8**	83.8	64.0	75.5	81.8
PointCNN [12]	84.6	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
FlexConv [4]	85.0	84.7	83.6	**91.2**	**96.7**	79.5	84.7	71.7	92.0	86.5	83.2	**96.6**	71.7	95.7	86.1	**74.8**	81.4	**84.5**
KPConv *rigid*	85.0	86.2	83.8	86.1	88.2	**81.6**	91.0	**80.1**	92.1	87.8	82.2	96.2	77.9	95.7	**86.8**	65.3	81.7	83.6
KPConv *deform*	**85.1**	**86.4**	**84.6**	86.3	87.2	81.1	91.1	77.8	92.6	88.4	82.7	96.2	**78.1**	**95.8**	85.4	69.0	82.0	83.6

Table 3. Segmentation mIoUs on ShapeNetPart.

Method	mIoU	mRec	ceil.	floor	wall	beam	col.	wind.	door	chair	table	book.	sofa	board	clut.
Pointnet [13]	41.1	49.0	88.8	97.3	69.8	0.1	3.9	46.3	10.8	52.6	58.9	40.3	5.9	26.4	33.2
SegCloud [18]	48.9	57.4	90.1	96.1	69.9	0.0	18.4	38.4	23.1	75.9	70.4	58.4	40.9	13.0	41.6
Eff 3D Conv [24]	51.8	68.3	79.8	93.9	69.0	0.2	28.3	38.5	48.3	71.1	73.6	48.7	59.2	29.3	33.1
TangentConv [17]	52.6	62.2	90.5	97.7	74.0	0.0	20.7	39.0	31.3	69.4	77.5	38.5	57.3	48.8	39.8
RNN Fusion [22]	57.3	63.9	92.3	**98.2**	79.4	0.0	17.6	22.8	62.1	74.4	80.6	31.7	66.7	62.1	56.7
SPGraph [10]	58.0	66.5	89.4	96.9	78.1	0.0	**42.8**	48.9	61.6	84.7	75.4	69.8	52.6	2.1	52.2
ParamConv [19]	58.3	67.1	92.3	96.2	75.9	**0.3**	6.0	**69.5**	63.5	66.9	65.6	47.3	68.9	59.1	46.2
KPConv *rigid*	65.4	70.9	92.6	97.3	81.4	0.0	16.5	54.5	**69.5**	90.1	80.2	74.6	66.4	63.7	58.1
KPConv *deform*	**67.1**	**72.8**	**92.8**	97.3	**82.4**	0.0	23.9	58.0	69.0	**91.0**	**81.5**	**75.3**	**75.4**	**66.7**	**58.9**

Table 4. Semantic segmentation IoU scores on S3DIS *Area-5*. Additionally, we give the mean class recall, a measure that some previous works call mean class accuracy.

Method	mIoU	mRec	ceil.	floor	wall	beam	col.	wind.	door	chair	table	book.	sofa	board	clut.
Pointnet [13]	47.6	66.2	88.0	88.7	69.3	42.4	23.1	47.5	51.6	42.0	54.1	38.2	9.6	29.4	35.2
RSNet [8]	56.5	66.5	92.5	92.8	78.6	32.8	34.4	51.6	68.1	60.1	59.7	50.2	16.4	44.9	52.0
SPGraph [10]	62.1	73.0	89.9	95.1	76.4	62.8	47.1	55.3	68.4	**73.5**	**69.2**	63.2	45.9	8.7	52.9
PointCNN [12]	65.4	75.6	**94.8**	**97.3**	75.8	63.3	51.7	58.4	57.2	71.6	69.1	39.1	61.2	52.2	58.6
KPConv *rigid*	69.6	78.1	93.7	92.0	82.5	62.5	49.5	65.7	**77.3**	57.8	64.0	68.8	71.7	60.1	59.6
KPConv *deform*	**70.6**	**79.1**	93.6	92.4	**83.1**	**63.9**	**54.3**	**66.1**	76.6	57.8	64.0	**69.3**	**74.9**	**61.3**	**60.3**

Table 5. Semantic segmentation IoU scores on S3DIS *k-fold*. Additionally, we give the mean class recall, a measure that some previous works call mean class accuracy.

point clouds. *ACM Transactions on Graphics (TOG)*, 37(6):235–1, 2018.

[8] Q. Huang, W. Wang, and U. Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2626–2635, 2018.

[9] R. Klokov and V. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017.

[10] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[11] J. Li, B. M. Chen, and G. H. 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.

[12] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018.

[13] C. R. Qi, H. Su, K. Mo, and L. 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.

[14] C. R. Qi, L. Yi, H. Su, and L. 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.

[15] X. Roynard, J.-E. Deschaud, and F. Goulette. Paris-lille-3d: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. *The International Journal of Robotics Research*, 37(6):545–557, 2018. http://npm3d.fr.

[16] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.

[17] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018.

[18] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *2017 International Conference on 3D Vision (3DV)*, pages 537–547. IEEE, 2017.

[19] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018.

[20] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[21] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.

[22] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang. 3d recurrent neural networks with context fusion for point cloud semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 415–430. Springer, 2018.

[23] L. Yi, H. Su, X. Guo, and L. J. Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *CVPR*, pages 6584–6592, 2017.

[24] C. Zhang, W. Luo, and R. Urtasun. Efficient convolutions for real-time semantic segmentation of 3d point clouds. In *2018 International Conference on 3D Vision (3DV)*, pages 399–408. IEEE, 2018.