Appendix of "A Closer Look at Local Aggregation Operators in Point Cloud Analysis"

Ze Liu[1,2⋆†], Han Hu[2⋆], Yue Cao[2], Zheng Zhang[2], and Xin Tong[2]

[1] University of Science and Technology of China
liuze@mail.ustc.edu.cn
[2] Microsoft Research Asia
{hanhu,yuecao,zhez,xtong}@microsoft.com

A1 Training/Inference Settings

In this section, we describe the training and inference settings of each dataset in detail.

ModelNet40 In training, we adopt the SGD optimizer, with initial learning rate of 0.002, which is decayed by $0.1^{1/200}$ every epoch. The momentum is 0.98 and weight decay is 0.001. The data is augmented with anisotropic random scaling (from 0.6 to 1.4), and gaussian noise of std = 0.002. We train networks for 600 epochs on 4 GPUs with 16 point clouds per GPU.

In inference, the model of the last epoch is used. We follow a common practice of voting scheme [4, 7, 2, 5], which augment each shape 100 times using the same augmentation method in training, and the predicted logits (the values before SoftMax) of these 100 augmented shapes are averaged to produce the final proabalities.

S3DIS Following [6, 1, 5], we use 3 color channels as features. In training, we adopt the SGD optimizer, with initial learning rate is 0.02, which is decayed by $0.1^{1/200}$ every epoch. The momentum is 0.98 and weight decay is 0.001. The data is augmented with anisotropic random scaling (from 0.7 to 1.3), gaussian noise of std = 0.001, random rotations around z-axis, random droping colors with 0.2 probability. The networks are trained for 600 epochs, using 4 GPUs and 8 point clouds per GPU.

In inference, the model of the last epoch is used. We divide each point cloud into regular overlaped spheres (totally 100). A point may appear in multiple spheres, and its logit (before SoftMax) is set as the average of this point's logits in different spheres.

PartNet In training, we adopt the AdamW optimizer [3] with learning rate of 0.000625. The momentum is 0.98 and the weight decay is 0.001. The data is augmented with anisotropic random scaling (from 0.8 to 1.2), and gaussian noise of std = 0.001. The networks are trained for 300 epochs on 4 GPUs with 8 point clouds per GPU.

⋆ Equal contribution. †This work is done when Ze Liu is an intern at MSRA.

In inference, the model of the last epoch is used. We adopt the 10-augment voting scheme (using the same augmentation method as in training) to compute each point's probabilities.

A2 Detailed Experimental Settings for Section 6.2

In Section 6.2 of the main paper, we evaluate different methods with varying architecture width, depth and bottleneck ratios. In this section, we provide detailed experimental settings as below.

For the experiments of varying architecture widths (see Fig. 3 left column of the main paper), we fix the depth and bottleneck ratio as $N_r + 1 = 1$ and $\gamma = 2$, respectively. For the experiments of varying architecture depths (see Fig. 3 middle column of the main paper), we fix the width and bottleneck ratio as $C = 36$ and $\gamma = 2$, respectively. For the experiments of varying architecture bottleneck ratios (see Fig. 3 right column of the main paper), we fix the width and depth as $C = 144$ and $N_r + 1 = 1$, respectively.

For different methods, the designing settings are as follows:

- PointMLP. We use $\{\Delta\mathbf{p}_{ij}, \mathbf{f}_i, \Delta\mathbf{f}_{ij}\}$ as input features, MAX pooling as reduction function and 1 FC layer.
- PseudoGrid. We use SUM as reduction function and 15 grid points.
- AdaptWeight. We use $\{\Delta\mathbf{p}_{ij}\}$ as input features, AVG pooling as reduction function and 1 FC layer.
- PosPool. We use AVG pooling as reduction function and the computation follows Eq. 8 in our main paper.
- PosPool*. We use AVG pooling as reduction function and the computation follows Eq. 9 in our main paper.

A3 More Variants of PosPool

In this section, we present more variants for PosPool, which all have no learnable weights. We first present a general formulation of these variants:

$$G(\Delta\mathbf{p}_{ij}, \mathbf{f}_j) = \text{Concat}[e^0\mathbf{f}_j^0; ...; e^{g-1}\mathbf{f}_j^{g-1}], \tag{1}$$

where $\{e^0, ..., e^{g-1}\}$ are g scalar encoding functions w.r.t. the relative position; $\mathbf{f}_j = \left[\mathbf{f}_j^0; \mathbf{f}_j^0; ...; \mathbf{f}_j^{g-1}\right]$ are an equal-sized partition of vector $\mathbf{f}_j$. In the following, we will present 7 variants by using different encoding functions $\{e^0, ..., e^{g-1}\}$.

Second Order Instead of directly using the 3-dimensional $\Delta\mathbf{p}_{ij}$ as in the standard formulation of Eq. 8 in the main paper, the second-order variant considers 6 additional encoding scalars by squares and pairwise multiplications of relative coordinates, as

$$\begin{aligned} &e^0 = \Delta x_{ij}, e^1 = \Delta y_{ij}, e^2 = \Delta z_{ij}, \\ &e^3 = \Delta x_{ij}^2, e^4 = \Delta y_{ij}^2, e^5 = \Delta z_{ij}^2, \\ &e^6 = \Delta x_{ij}\Delta y_{ij}, e^7 = \Delta x_{ij}\Delta z_{ij}, e^8 = \Delta y_{ij}\Delta z_{ij}. \end{aligned} \tag{2}$$

Third Order The third order variant uses additional third-order multiplications as encoding functions:

$$
\begin{aligned}
&e^0 = \Delta x_{ij}, e^1 = \Delta y_{ij}, e^2 = \Delta z_{ij}, e^3 = \Delta x_{ij}^2, e^4 = \Delta y_{ij}^2, e^5 = \Delta z_{ij}^2, \\
&e^6 = \Delta x_{ij}\Delta y_{ij}, e^7 = \Delta x_{ij}\Delta z_{ij}, e^8 = \Delta y_{ij}\Delta z_{ij}, \\
&e^9 = \Delta x_{ij}\Delta y_{ij}^2, e^{10} = \Delta x_{ij}\Delta z_{ij}^2, e^{11} = \Delta y_{ij}\Delta z_{ij}^2, \\
&e^{12} = \Delta x_{ij}^2\Delta y_{ij}, e^{13} = \Delta x_{ij}^2\Delta z_{ij}, e^{14} = \Delta y_{ij}^2\Delta z_{ij}, \\
&e^{15} = \Delta x_{ij}^3, e^{16} = \Delta y_{ij}^3, e^{17} = \Delta z_{ij}^3.
\end{aligned} \tag{3}
$$

Note we omit the encoding function $\Delta x_{ij}\Delta y_{ij}\Delta z_{ij}$ to ensure $g = 18$ such that $\mathbf{f}_j$'s channel number C is divisible by g. The third order encoding functions are similar as the Taylor functions in [8].

Angle and Distance In this variant, we decouple the relative position into distance $d_{ij} = \sqrt{\Delta x_{ij}^2 + \Delta y_{ij}^2 + \Delta z_{ij}^2}$ and angle $\left\{\frac{\Delta x_{ij}}{d_{ij}}, \frac{\Delta y_{ij}}{d_{ij}}, \frac{\Delta z_{ij}}{d_{ij}}\right\}$. The encoding functions are:

$$
e^0 = d_{ij}, e^1 = \frac{\Delta x_{ij}}{d_{ij}}, e^2 = \frac{\Delta y_{ij}}{d_{ij}}, e^3 = \frac{\Delta z_{ij}}{d_{ij}}. \tag{4}
$$

Angle and Gaussian Inversed Distance The above variants encourage the distant points to have larger amplitudes of encoding scalars. Here we present a variant which encourages close points to have larger amplitudes of encoding scalars, by inverse the distance by a Gaussian function:

$$
e^0 = \tilde{d}_{ij} = \exp\left(-d_{ij}^2\right), e^1 = \frac{\Delta x_{ij}}{d_{ij}}, e^2 = \frac{\Delta y_{ij}}{d_{ij}}, e^3 = \frac{\Delta z_{ij}}{d_{ij}}. \tag{5}
$$

Angle or Distance Alone We also consider variants which use angle or distance functions alone:

$$
e^0 = \frac{\Delta x_{ij}}{d_{ij}}, e^1 = \frac{\Delta y_{ij}}{d_{ij}}, e^2 = \frac{\Delta z_{ij}}{d_{ij}}. \tag{6}
$$

$$
e^0 = d_{ij}. \tag{7}
$$

$$
e^0 = \tilde{d}_{ij}. \tag{8}
$$

Results. Table 1 shows the comparison of different variants using three benchmarks. For PartNet datasets, we report the part category mean IoU on the validation set. While PosPool adopts AVG as the default reduction function, we also report the results when using other reduction function (SUM, MAX). It can be seen: 1) all variants containing full configurations of relative positions perform similarly well. They perform significantly better than the variants using angle or distance alone. 2) Whether more distant points have larger or smaller encoding amplitudes than closer points is insignificant. 3) Using AVG as the reduction function performs comparably well than those using SUM, and slightly better than those using MAX.

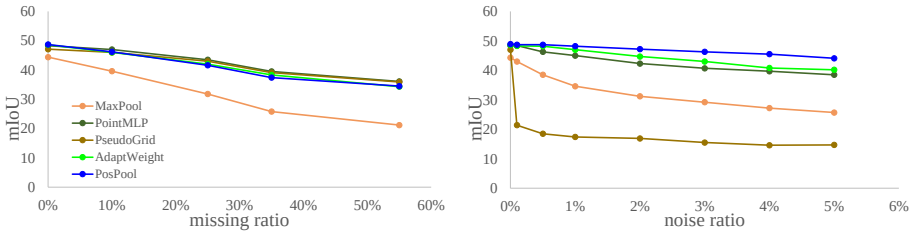

Fig. 1. The robustness test of different approaches when there are less points(left) or outlier points(right).

A4 The Robustness of Different Operators with Missing/Noisy Points

Fig 1 show the accuracy curves of AdaptWeight, PseudoGrid, MaxPool, PosPool, PointMLP for inputs with different ratios of noise or missing points. All the experiments are executed on the PartNet benchmark. The model for each curve is trained on the clean data of PartNet. Only the testing data at the inference stage includes noise and missing points.

As shown in Fig 1(left), different local aggregation operators (AdaptWeight, PseudoGrid, PointMLP, PosPool) perform similarly in robustness with varying missing point ratios, all significantly better than the MaxPool baseline. With varying noise ratios, the proposed PosPool operator performs best, slightly better than AdaptWeight and PointMLP, and significantly better than PseudoGrid and the MaxPool baseline. Fig. 2 show the activation maps of the last layer in each stage by using clean data (top row) and noisy data (bottom row, ratio 1%), respectively. While the noisy point features significantly contaminate the activations of some other regular point features in the MaxPool and PseudoGrid methods, the activations of clean points in other methods are less affected by these noisy points.

A5 More Detailed Results on PartNet

We first report the part-category mIoU for each category on PartNet. From Table 2 and Table 3 we can see that all operators show similar results on each category, which further validates our findings. Table 4 shows the number of training, validation, test samples.

We then show some qualitative results in Fig 3. All the representative methods perform similarly well on most shapes.

To understand what the networks with these operators are learnt from input, we visualize the norm of activation map before prediction by different methods (operators), suggesting that different operators tend to offer similar activations for a same input point cloud, as shown in Fig 4.

A6 Detailed Space and Time complexity Analysis

In this section, we provide detailed analysis for the space and time complexity of different aggregators presented in Section 3.

A6.1 Point-wise MLP based Methods

The detailed architecture is shown in Fig 5. For $h = 1$, a shared FC is applied on each point with K neighborhoods, and the time cost is $(d+3)dnK$ and the space cost (parameter number) is $(d+3)d$. For $h \geq 2$, the time cost is

$$(d+3)(d/2)nK + (h-2)\cdot(d/2)(d/2)nK + (d/2)dnK = \\ ((2d+3) + (h-2)d/2)\cdot d/2\cdot nK, \quad (9)$$

and the space cost is

$$(d+3)(d/2) + (h-2)\cdot(d/2)(d/2) + (d/2)d = ((2d+3) + (h-2)d/2)\cdot d/2. \quad (10)$$

A6.2 Pseudo Grid Feature based Methods

Our default settings adopt depth-wise convolution. In depth-wise convolution, a d-dim learnt weight vector is associated to each grid point. Hence, the space cost (parameter number) is $d \cdot M$ and the time cost is $ndKM$.

A6.3 Adaptive Weight based Methods

Adaptive Weight based Methods involve two computation steps. Firstly, a shared MLP is used to compute the aggregation weights for each neighboring point. This step has time cost of

$$3\cdot(d/2)nK + (h-2)\cdot(d/2)(d/2)nK + (d/2)dnK = \\ (3 + d + (h-2)d/2)\cdot d/2\cdot nK, \quad (11)$$

and space cost of

$$3\cdot(d/2) + (h-2)\cdot(d/2)(d/2) + (d/2)d = (3 + d + (h-2)d/2)\cdot d/2. \quad (12)$$

Secondly, depth-wise aggregation is conducted, where the time cost is dnK and space cost is 0.

The total time cost is $(3 + d + (h-2)d/2)\cdot d/2\cdot nK + dnK = ((h-2)d/2 + d + 5)\cdot d/2\cdot nK$ and the total space cost is $((h-2)d/2 + d + 3)\cdot d/2$.

References

1. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: Advances in Neural Information Processing Systems. pp. 820–830 (2018)
2. Liu, Y., Fan, B., Xiang, S., Pan, C.: Relation-shape convolutional neural network for point cloud analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8895–8904 (2019)
3. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019)
4. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NIPS (2017)
5. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6411–6420 (2019)
6. Wang, S., Suo, S., Ma, W.C., Pokrovsky, A., Urtasun, R.: Deep parametric continuous convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2589–2597 (2018)
7. 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**(5), 146 (2019)
8. Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y.: Spidercnn: Deep learning on point sets with parameterized convolutional filters. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 87–102 (2018)

Table 1. Evaluation of different PosPool variants on three benchmarks. For PartNet datasets, the part category mean IoU on validation set are reported

method	γ	C	N_r+1	$R(\cdot)$	ModelNet40	S3DIS	PartNet
$\Delta x_{ij}, \Delta y_{ij}, \Delta z_{ij}$	8	144	2	AVG	93.0	64.2	48.5
	4	144	2	AVG	92.8	65.1	50.0
	2	144	2	AVG	93.1	66.6	49.8
	8	144	2	SUM	92.8	64.6	48.5
	8	144	2	MAX	92.6	61.1	48.4
$\mathcal{E}^m$	8	144	2	AVG	92.7	62.2	49.3
	4	144	2	AVG	93.1	63.6	50.9
	2	144	2	AVG	93.2	64.9	50.8
	8	144	2	SUM	92.8	62.9	49.1
	8	144	2	MAX	92.4	62.8	48.9
Second Order	8	144	2	AVG	93.0	63.4	49.9
	4	144	2	AVG	93.1	64.0	49.9
	2	144	2	AVG	92.9	65.7	50.9
	8	144	2	SUM	92.9	64.0	49.9
	8	144	2	MAX	92.7	63.3	48.1
Third Order	8	144	2	AVG	93.2	63.6	49.6
	4	144	2	AVG	93.3	64.5	50.0
	2	144	2	AVG	93.4	64.7	51.8
	8	144	2	SUM	92.7	64.8	47.7
	8	144	2	MAX	92.3	62.2	49.0
$d_{ij}, \frac{\Delta x_{ij}}{d_{ij}}, \frac{\Delta y_{ij}}{d_{ij}}, \frac{\Delta z_{ij}}{d_{ij}}$	8	144	2	AVG	92.8	63.5	49.0
	4	144	2	AVG	93.2	65.3	48.3
	2	144	2	AVG	92.9	65.6	49.8
	8	144	2	SUM	92.9	64.5	49.0
	8	144	2	MAX	92.7	62.3	48.4
$\tilde{d}_{ij}, \frac{\Delta x_{ij}}{d_{ij}}, \frac{\Delta y_{ij}}{d_{ij}}, \frac{\Delta z_{ij}}{d_{ij}}$	8	144	2	AVG	93.0	64.2	48.6
	4	144	2	AVG	93.2	64.1	49.1
	2	144	2	AVG	93.0	64.8	49.3
	8	144	2	SUM	93.0	64.3	48.2
	8	144	2	MAX	92.9	62.3	49.1
$\frac{\Delta x_{ij}}{d_{ij}}, \frac{\Delta y_{ij}}{d_{ij}}, \frac{\Delta z_{ij}}{d_{ij}}$	8	144	2	AVG	92.1	62.1	46.6
	4	144	2	AVG	92.1	61.8	46.5
	2	144	2	AVG	92.2	62.6	47.6
	8	144	2	SUM	91.9	60.9	46.4
	8	144	2	MAX	92.0	61.2	45.8
d_{ij}	8	144	2	AVG	90.9	53.3	43.4
	4	144	2	AVG	91.2	53.1	43.8
	2	144	2	AVG	90.9	53.4	44.6
	8	144	2	SUM	90.7	55.4	43.6
	8	144	2	MAX	91.0	56.2	44.2
$\tilde{d}_{ij}$	8	144	2	AVG	90.6	53.7	43.7
	4	144	2	AVG	90.5	53.0	42.1
	2	144	2	AVG	90.7	53.4	43.9
	8	144	2	SUM	91.1	55.3	45.4
	8	144	2	MAX	91.7	55.5	43.4

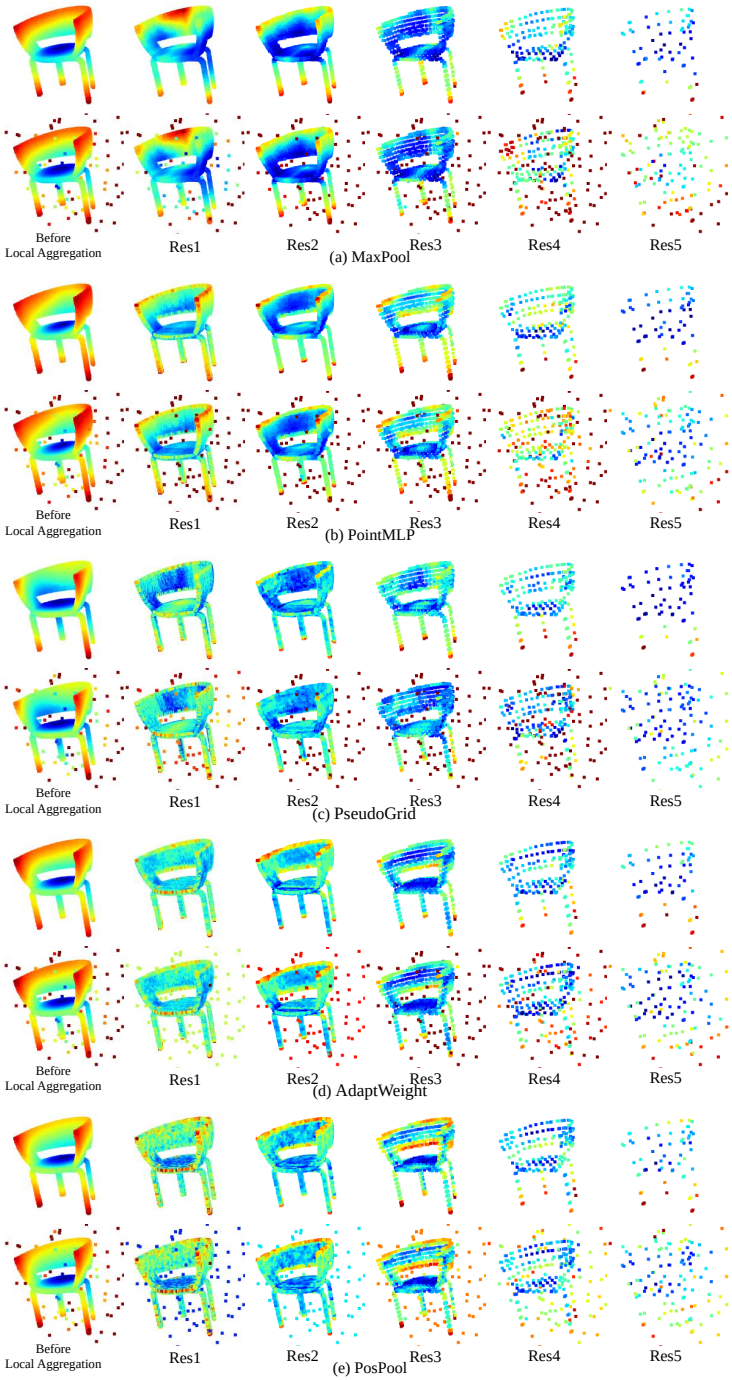

Fig. 2. The activation maps of different methods by using clean data(top) or noisy data(bottom), respectively

	Avg	Bed	Bott	Chair	Clock	Dish	Disp	Door	Ear	Fauc	Knife	Lamp	Micro	Frid	Stora	Table	Trash	Vase
PW	48.1	41.5	29.3	48.2	46.2	61.3	87.0	38.1	56.5	52.3	26.7	30.1	47.9	46.5	48.1	44.3	51.3	62.4
PG	50.8	46.8	30.0	50.5	46.1	60.4	88.5	45.4	61.4	54.2	30.5	31.1	61.8	47.9	50.6	47.2	56.8	54.6
AW	50.1	45.4	25.0	49.0	46.8	63.0	87.3	38.1	63.6	54.4	38.0	30.1	57.5	49.6	48.2	45.9	54.7	55.7
PP	50.0	46.6	28.5	49.2	47.2	60.7	86.7	39.8	55.2	54.0	41.5	31.5	58.1	48.3	48.4	45.6	57.1	51.4
PP*	50.6	47.5	29.7	49.1	47.2	65.8	88.0	46.8	58.9	54.6	31.5	28.1	60.7	47.3	50.9	45.0	54.6	55.0

Table 2. part-category mIoU% on PartNet validation sets. **PW**, **PG**, **AW**, **PP**, **PP*** refer to Pseudo Grid, Adapt Weights, PosPool, PosPool* respectively.

	Avg	Bed	Bott	Chair	Clock	Dish	Disp	Door	Ear	Fauc	Knife	Lamp	Micro	Frid	Stora	Table	Trash	Vase
PW	51.2	44.5	52.6	46.0	38.4	68.2	82.5	46.9	47.1	58.7	43.8	26.4	59.2	48.7	52.5	41.3	55.4	57.3
PG	53.0	47.5	50.9	49.2	44.8	67.0	84.2	49.1	49.9	62.7	38.3	27.0	59.4	54.3	54.1	44.5	57.4	60.7
AW	53.5	46.1	47.9	47.2	42.7	64.4	83.7	55.6	49.5	61.7	49.5	27.4	59.3	57.7	53.5	45.1	57.5	60.9
PP	53.4	45.8	46.5	48.3	40.2	66.1	84.2	49.4	51.6	63.5	48.1	27.9	62.3	56.1	53.3	43.4	58.7	62.4
PP*	53.8	49.5	49.4	48.3	49.0	65.6	84.2	56.8	53.8	62.4	39.3	24.7	61.3	55.5	54.6	44.8	56.9	58.2

Table 3. part-category mIoU% on PartNet test sets. **PW**, **PG**, **AW**, **PP**, **PP*** refer to Pseudo Grid, Adapt Weights, PosPool, PosPool* respectively.

Split	Bed	Bott	Chair	Clock	Dish	Disp	Door	Ear	Fauc	Knife	Lamp	Micro	Frid	Stora	Table	Trash	Vase
train	133	315	4489	406	111	633	149	147	435	221	1554	133	136	1588	5707	221	741
val	24	37	617	50	19	104	25	28	81	29	234	12	20	230	843	37	102
test	37	84	1217	98	51	191	51	53	132	77	419	39	31	451	1668	63	233

Table 4. The number of training, validation and test samples.

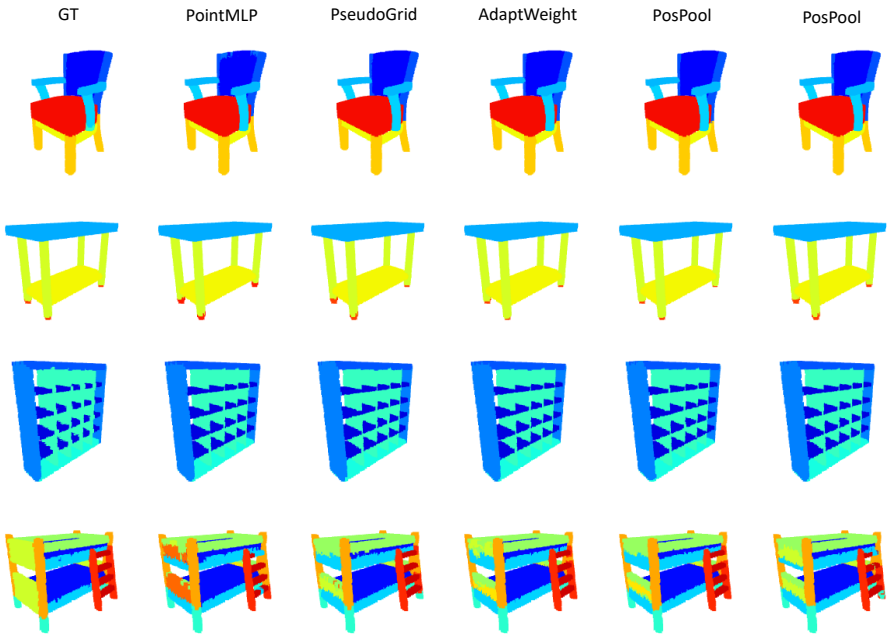

Fig. 3. Visualization of part segmentation results by different methods on the PartNet dataset.

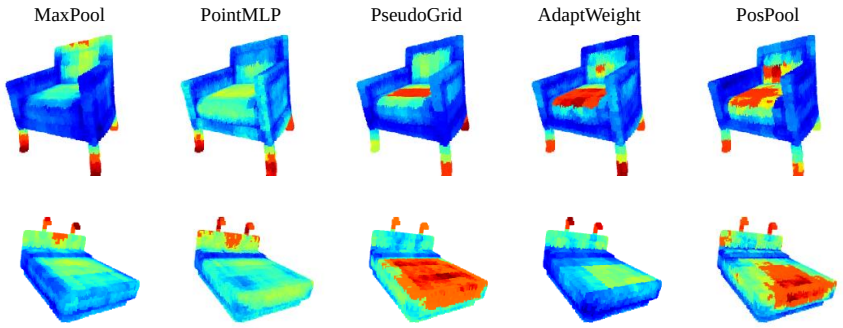

Fig. 4. Activation maps before the final prediction using different methods on PartNet validation shapes, indicating similar high energy area learnt by different methods

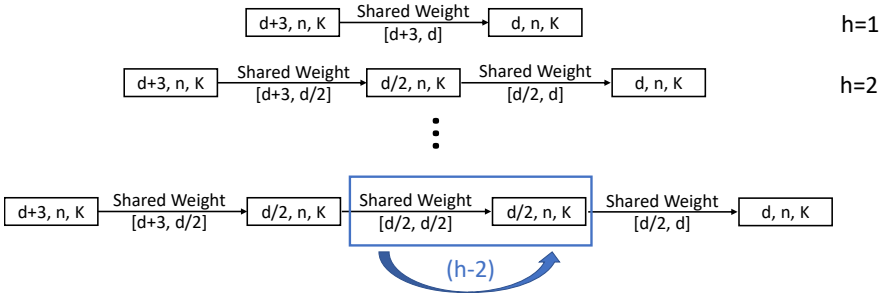

Fig. 5. The detailed architecture for Point-wise MLP based operators.