# Enhancing Local Feature Learning for 3D Point Cloud Processing using Unary-Pairwise Attention

Haoyi Xiu[1,2]
xiu.h.aa@m.titech.ac.jp
Xin Liu[2,†]
xin.liu@aist.go.jp
Weimin Wang[3,†]
wangweimin@dlut.edu.cn
Kyoung-Sook Kim[2]
ks.kim@aist.go.jp
Takayuki Shinohara[1]
shinohara.t.af@m.titech.ac.jp
Qiong Chang[4]
q.chang@c.titech.ac.jp
Masashi Matsuoka[1]
matsuoka.m.ab@m.titech.ac.jp

[1] Department of Architecture and Building Engineering, Tokyo Institute of Technology, Tokyo, Japan
[2] Artificial Intelligence Research Center, AIST, Tokyo, Japan
[3] DUT-RU International School of Information Science and Engineering, Dalian University of Technology, Dalian, China
[4] Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan

The supplementary material is organized as follows.

- Sec. A reports additional experimental results as well as visualization results.
- Sec. B elaborates on the implementation details regarding experiments in the main paper.
- Sec. C shows results of complexity analysis.

# A Additional Experimental Results

## A.1 Ablation Study

We show more results of ablation experiments regarding design choices. Like the one in the main paper, all ablation experiments are conducted on the ShapeNet dataset. We use mIoU [3] as the performance metric.

**Number of attention head** $h$. As shown in Table A.1, the network achieves the best performance with a single attention head. Nevertheless, the scores are similar among all the variants with the different number of heads; therefore, we choose 1 head as the default

† Corresponding author(s)

| (a) #head $h$ | | (b) Reduction rate | | (c) Non-linear trans. | | (d) Fusion | |
|---|---|---|---|---|---|---|---|
| Baseline | 85.1 | Baseline | 85.1 | Baseline | 85.1 | Baseline | 85.1 |
| 1 | 86.1 | 2 | 86.1 | Shared | 85.9 | Concat. | 85.6 |
| 2 | 86.0 | 4 | 86.0 | Not shared | 86.1 | Add. | 86.1 |
| 4 | 86.1 | 8 | 85.8 | - | - | - | - |
| 8 | 86.0 | - | - | - | - | - | - |

Table A.1: More results of the ablation study.

setting in the experiments. We suspect that adding more heads increases model complexity which makes the models difficult to train.

**Reduction rate**. According to Table A.1, increasing the reduction rate to 4 slightly degrades the performance while the scores start to drop significantly when the reduction rate is set to 8. Thus, we choose 2 as the default throughout the main paper.

**Non-linear transformation**. We examine the impact of using shared non-linear transformations instead of independent ones (our default choice) after the attention operations. As shown in Table A.1, *Not shared* outperforms *Shared*, which shows that it is beneficial to individually transform the outputs from each attention component. We conjecture that individual transformation for each output has the effect of enhancing/suppressing information obtained from attentions, making them more adaptive to inputs.

**Fusion method**. Several operations of combining output from the attention layer at the residual connection are compared. As shown in the Table A.1, *Add.* yields a significantly better result than *Concat.*. Therefore, we adopt addition as the default.

## A.2 Visualization Results

Here we provide additional qualitative analysis of visualization results.

**Attention Maps**. Fig. A.1 shows more examples of visualized attention maps. As shown in Fig. A.1, the attention maps of very different positions are nearly identical. Therefore, query-dependent information is less learned by SA. On the other hand, unary and pairwise attentions successfully encode the different types of information by exploiting both query-dependent and query-independent information.

**Part Segmentation**. Fig. A.2 shows more examples of part segmentation results. Qualitative results show that UPA in which query-dependent information (e.g., smoothness) is modeled produces significantly smoother predictions compared with the baseline. Furthermore, UPA is more boundary-aware than baseline presumably thanks to the unary component, which models point-wise salient boundaries.

**Scene Segmentation**. As shown in Fig. A.3, UPA successfully produces smoother predictions. Such characteristic is beneficial in scene segmentation as man-made objects often consist of simple planes. Notably, in a very challenging scene, UPA detects objects that the baseline completely fails to detect according to the difference map in the bottom row, showing its sensitivity to discern semantically similar objects.

# B Additional Details of Experiments

In this section, we elaborate on various implementation details of our experiments presented in the main paper.

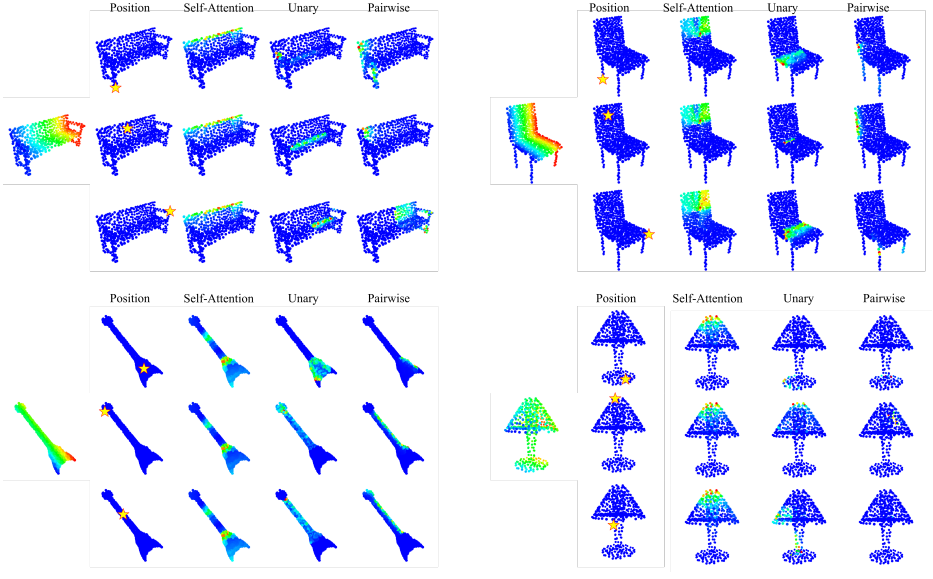


Figure A.1: More examples of attention maps.

## B.1 Implementation Details

All experiments are implemented using PyTorch and conducted on a machine with an NVIDIA Tesla V100 GPU. We use the Adam [2] optimizer for classification and the SGD optimizers for other tasks. The momentum and weight decay are set to 0.9 and 0.0001, respectively. The total training epoch is set to 200. For classification and part segmentation, the initial learning rate is set to 0.001 and reduced when validation metrics (OA and mIoU) plateau. For scene segmentation, the initial learning rate is set to 0.1 and multiplied by 0.1 every 25 epochs. The batch size is set to 32 for all tasks.

## B.2 Experimental Settings of Classification

For the classification task, we uniformly sample 1,024 points from 2,048 points by furthest point sampling [2] before feeding into the network. After sampling, each input point cloud is scaled into a unit ball, in which the longest edge becomes 1. Random anisotropic scaling (in the range of: [0.66, 1.5]) and random translation (in the range of: [-0.2, 0.2]) are applied to input point clouds before a forward pass for data augmentations.

## B.3 Experimental Settings of Part Segmentation

We randomly take 2,048 points from the original shape. The input is also scaled into the unit ball. The augmentation strategy is identical to the one we used in classification. After training, we obtain stable final results by voting.

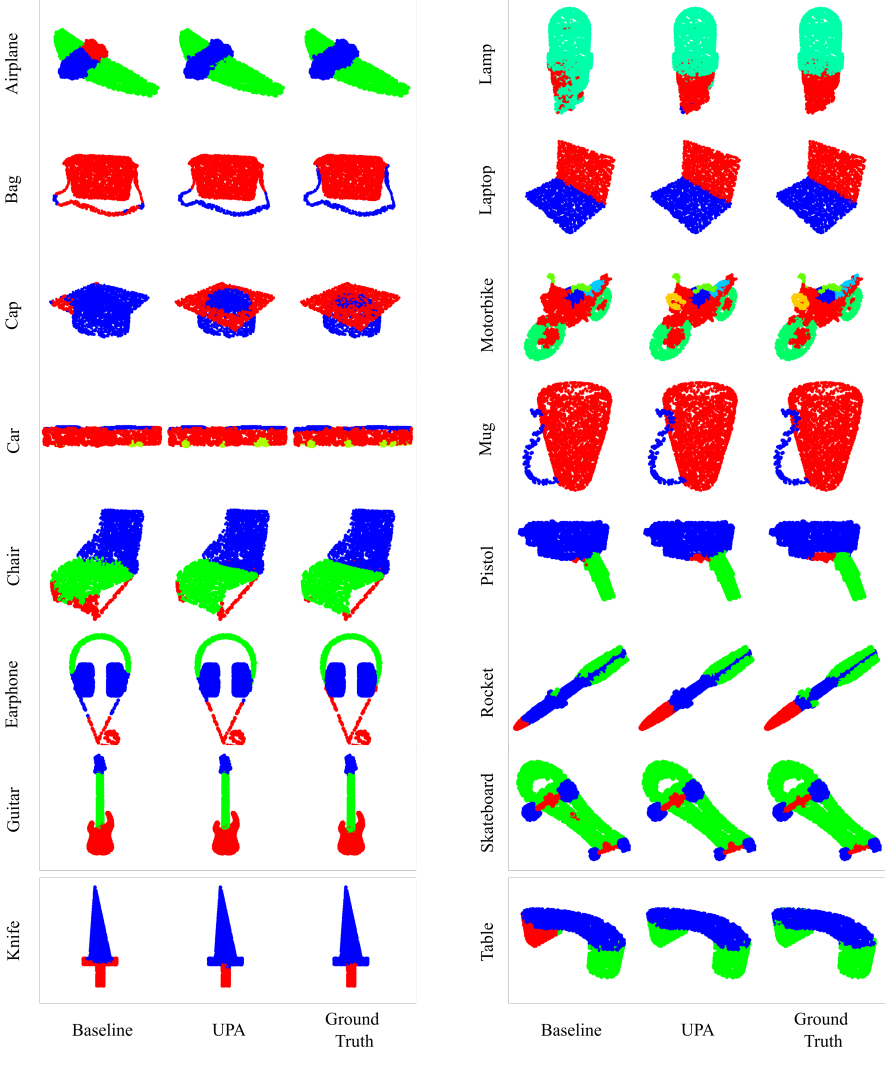


Figure A.2: More part segmentation results

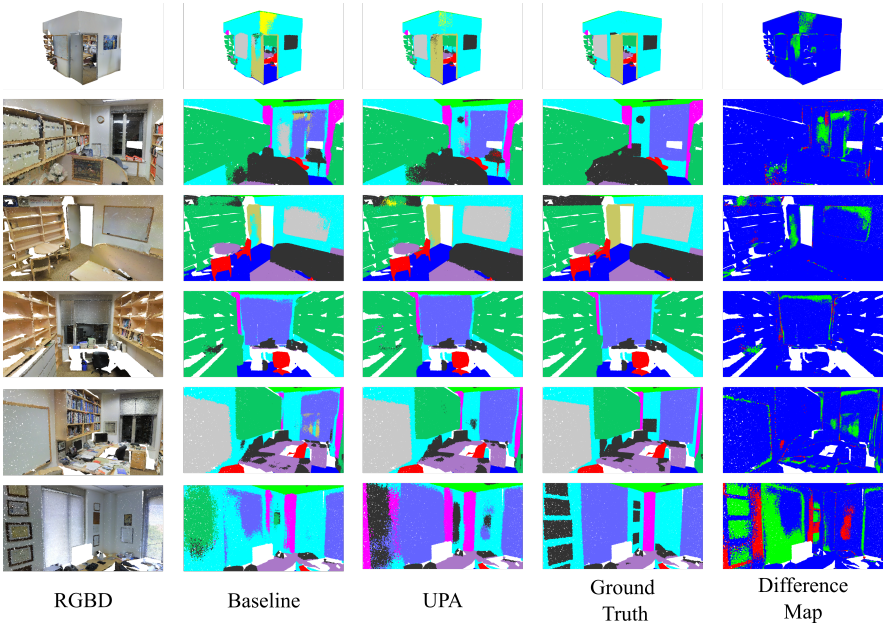


Figure A.3: More scene segmentation results.

## B.4 Experimental Settings of Scene Semantic Segmentation

We follow the data preparation procedure of PointNet [3] to split each room into $1m \times 1m$ blocks. 4,096 points inside each block are randomly sampled on the fly during the training. At the test time, we test on all points. Specifically, we sample a room with a 1m block regularly with a step size of 0.5m to ensure coverage. The number of points sampled in a room is set to 4,096. Points in a box are split into several samples when the number of points in it is larger than 4096. When the number of points is smaller than 4,096, points are randomly resampled to reach 4,096 (similar to the padding in the image processing). In such a manner, all input points are tested at least once. Finally, multiple predictions are summed up and the class with the greatest accumulated probabilities are treated as predictions. No augmentation is used during the training.

## B.5 Network Parameters

We describe the full network architecture designs for classification, part segmentation, and scene segmentation. As explained in the main paper, our architectures directly inherit from PointNet++ [4], which consists of several Set Abstraction Layers and Feature Propagation Layers. Table B.1, Table B.2, and Table B.3 present the parameter settings in each network. We use the following notations. $p$ denotes the number of sampled representative points; $r$ is the radius used for grouping the neighborhood points around each representative point; $n$ represents the number of points chosen randomly from the grouped points; $l$ denotes the operations conducted for each group of points, where [...] indicates a series of shared Multi-layer perceptrons (MLPs); $d$ indicates the fusion MLP applied to the concatenated output of

| $p$ | $r$ | $n$ | $l$ | $d$ | $k$ | $h$ |
|---|---|---|---|---|---|---|
| 512 | 0.1 | 16 | [32,32,64], Max | [128] | 20 | 8 |
| | 0.2 | 32 | [32,32,64], Max | | | |
| | 0.4 | 64 | [32,32,64], Max | | | |
| 128 | 0.2 | 16 | [128,128,256], Max | [512] | 20 | 8 |
| | 0.4 | 32 | [128,128,256], Max | | | |
| | 0.8 | 64 | [128,128,256], Max | | | |
| 1 | - | - | [512], Max | 1,024 | - | - |
| [512], dp, [256], dp, [$c$] | | | | | | |

Table B.1: Parameters of the classification network.

| $p$ | $r$ | $n$ | $l$ | $d$ | $k$ | $h$ |
|---|---|---|---|---|---|---|
| 512 | 0.1 | 16 | [32,32,64] , Max | [128] | 16 | 1 |
| | 0.2 | 32 | [32,32,64], Max | | | |
| | 0.4 | 64 | [32,32,64], Max | | | |
| 128 | 0.2 | 16 | [128,128,256], Max | [256] | 16 | 1 |
| | 0.4 | 32 | [128,128,256], Max | | | |
| | 0.6 | 64 | [128,128,256], Max | | | |
| 32 | 0.4 | 16 | [256,256,512], Max | [1024] | 16 | 1 |
| | 0.6 | 32 | [256,256,512], Max | | | |
| | 0.8 | 64 | [256,256,512], Max | | | |
| 128 | - | - | - | [512] | 16 | 1 |
| 512 | - | - | - | [256] | 16 | 1 |
| 2,048 | - | - | - | [128] | 16 | 1 |
| [128], dp, [$c$] | | | | | | |

Table B.2: Parameters of the part segmentation network.

all $l$ in this layer; $k$ denotes the neighborhood size used in UPA; $h$ is the number of head used in UPA; $dp$ is a dropout layer; $c$ is the number of class. Each MLP consists of a linear projection, a batch normalization [11] followed by a ReLU activation. The dropout probabilities are set to 0.7 for classification and 0.5 for other tasks. For the classification model, empty values of $r$ and $n$ mean all the points are grouped. On the other hand, for the part/scene segmentation model, those empty values indicate that the layer is an FP level. For all cases, empty entries for $k$ and $h$ mean that the UPA block is not applied.

# C Complexity Analysis

In this section, the space and time complexity of considered attention variants are reported. Furthermore, the number of parameters and inference speeds of baseline, UPA, SA, and local-SA are also shown.

## C.1 Complexity of Attentions

Results are shown in Table C.1. By operating locally, Local-SA and UPA successfully reduce the time and space complexity from quadratic to linear to input cardinality. As shown in the

| $p$ | $r$ | $n$ | $l$ | $d$ | $k$ | $h$ |
|---|---|---|---|---|---|---|
| 1,024 | 0.05 | 16 | [32,32,64], Max | [64] | 16 | 1 |
| | 0.1 | 32 | [32,32,64], Max | | | |
| | 0.2 | 64 | [32,32,64], Max | | | |
| 256 | 0.1 | 16 | [64,64,128], Max | [128] | 16 | 1 |
| | 0.2 | 32 | [64,64,128], Max | | | |
| | 0.4 | 64 | [64,64,128], Max | | | |
| 64 | 0.2 | 16 | [128,128,256], Max | [256] | 16 | 1 |
| | 0.4 | 32 | [128,128,256], Max | | | |
| | 0.6 | 64 | [128,128,256], Max | | | |
| 32 | 0.4 | 8 | [256,256,512], Max | [1024] | 16 | 1 |
| | 0.6 | 16 | [256,256,512], Max | | | |
| | 0.8 | 32 | [256,256,512], Max | | | |
| 64 | - | - | - | [512] | 16 | 1 |
| 256 | - | - | - | [256] | 16 | 1 |
| 1,024 | - | - | - | [128] | 16 | 1 |
| 4,096 | - | - | - | [128] | - | - |
| [128], dp, [$c$] | | | | | | |

Table B.3: Parameters of the scene segmentation network.

| Variant | Operating scope | Time Complexity | Space Complexity |
|---|---|---|---|
| Self-Attention | Global | $\mathcal{O}(n^2 \cdot d)$ | $\mathcal{O}(n^2)$ |
| Local Self-Attention | Local | $\mathcal{O}(n \cdot k \cdot d)$ | $\mathcal{O}(n \cdot k \cdot d)$ |
| Unary Attention | Local | $\mathcal{O}(n \cdot k \cdot d)$ | $\mathcal{O}(n \cdot k \cdot d)$ |
| Pairwise Attention | Local | $\mathcal{O}(n \cdot k \cdot d)$ | $\mathcal{O}(n \cdot k \cdot d)$ |

Table C.1: Properties of attention variants.

paper, the local operating scope is more effective than the global one. We conjecture that translation invariance, which acts as a useful inductive bias, eases the training.

## C.2 Model Complexity

The result is shown in Table C.2. Although UPA (*ours*) consumes more resources, it achieves significantly better results compared to the baseline (PointNet++) and other networks. As is expected, SA has the same amount of parameters as Local-SA because only the operating scope is changed. However, local-SA takes a longer time for a forward pass because of some additional operations (e.g., KNN). In practice, standard SA block cannot be applied to

| Model | ModelNet | | ShapeNet | | S3DIS | |
|---|---|---|---|---|---|---|
| | #params (M) | s/batch | #params (M) | s/batch | #params (M) | s/batch |
| Baseline | 1.83 | 0.03 | 3.29 | 0.05 | 3.79 | 0.07 |
| Ours | 3.23 | 0.05 | 6.50 | 0.13 | 6.76 | 0.12 |
| SA | 2.94 | 0.03 | 5.53 | 0.06 | 6.38 | 0.07 |
| Local-SA | 2.94 | 0.04 | 5.53 | 0.08 | 6.38 | 0.09 |

Table C.2: Model statistics.

a high-resolution input (e.g., the last layer of segmentation network) where both UPA and local-SA can be applied, revealing the benefit of the local operating scope.

# References

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[3] 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.

[4] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.