Supplemental Material for Adaptive Graph Convolution for Point Cloud Analysis

A. Network architecture

Part segmentation. Our segmentation network on ShapeNetPart dataset includes a spatial transformer network [1] before the convolution layers. It processes the input points and outputs a 3×3 matrix in order to apply a global transformation. We apply standard graph convolutions (64, 128, 1024), followed by a max pooling function and fully-connected layers (512, 256). The output matrix is initialized as an identity matrix. Here, it is also possible to replace these graph convolutions with AdaptConv layers, but this does not lead to a significant improvement. Since the input contains normals as additional attributes, we apply the 3×3 matrix separately to the point and normal dimensions. The STN module can be seen as a global adaptive kernel that is convolved with all input points similar as in our AdaptConv. We report the results of networks with and without STN in Tab. 1.

Kernel function for adaptive convolution. We detail the design of g_m that generates adaptive kernels in Eq.1 of the main paper. As discussed before, g_m is a two-layer shared MLP on the feature input Δf_{ij}. It is an inevitable choice to use a shared mapping as the kernel function. However, note that g_m is not the convolution kernel (fixed kernel) that is applied to points, but is to explore the different feature correspondences for different pairs of points. In the implementation, we process all g_m $(m = 1, 2, ..., M)$ together and obtain the adaptive kernels for the following convolution (see Fig. 1). The first layer is one shared MLP(d) for all g_m, and we organize the kernels as a weight matrix $(c \times M)$ which is then applied to the corresponding Δx_{ij} of dimension c by matrix multiplication. After a LeakyReLU, the edge feature h_{ij} is obtained and finally we apply Eq.3 of the main paper for the output feature of the central point. The resnet connection is an optional block which is used in our segmentation model.

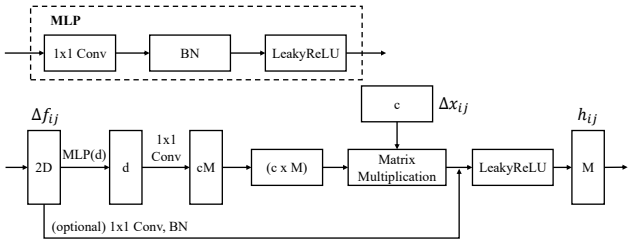


Figure 1. Kernel function used in our adaptive convolution. We apply a two-layer MLP for the adaptive weight matrix. The output edge feature is obtained by matrix multiplication between Δx_{ij} and the weight matrix. Optional resnet block: shortcut 1×1 convolution and batch normalization layer.

B. Ablation studies

In this section, we give more details for ablation networks used in Sec.4.3 of the main paper.

Attentional convolution. In order to compare our model with attentional graph convolutions, we design several ablations which replace AdaptConv layers with attentional convolution layers in the network. Following the design of [2], the output feature can be formulated as follows:

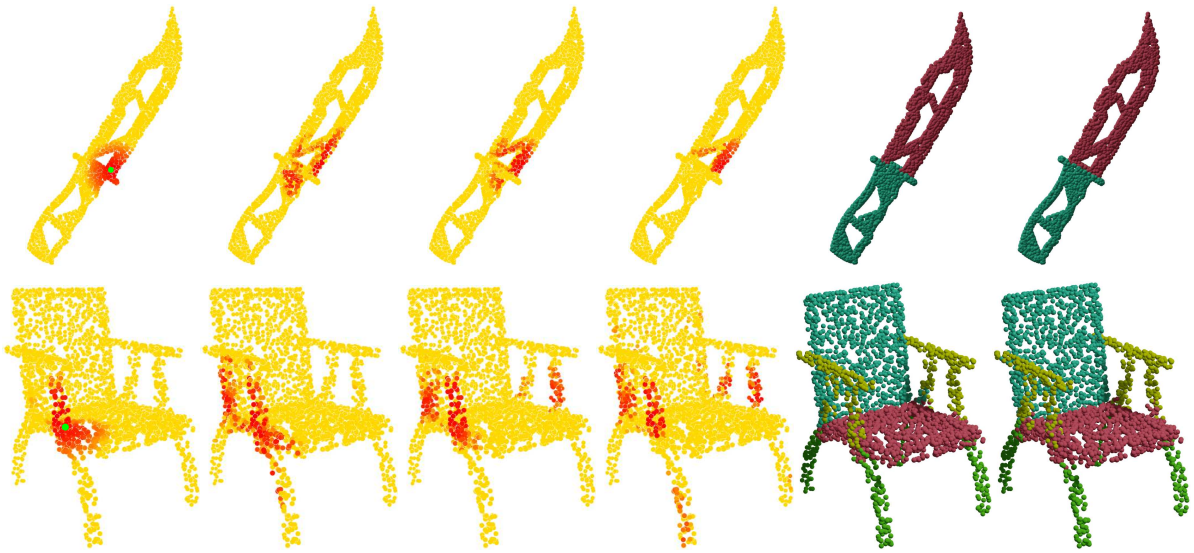
$$f_i' = \max_{j \in \mathcal{N}(i)} a_{ij} * h(f_j), \tag{1}$$

where $h : \mathbb{R}^D \rightarrow \mathbb{R}^M$ is a shared MLP and a_{ij} is the attentional weight calculated as:

$$a_{ij} = \mathbf{softmax}_j(\alpha(\Delta f_{ij})). \tag{2}$$

Here, $\alpha(\cdot)$ is a mapping function, and $\Delta f_{ij} = [h(f_i), h(f_j) - h(f_i)]$ since the attentional weights are applied to $h(f_j)$ instead of f_j. A softmax is used to make $\sum_j a_{ij} = 1, j \in \mathcal{N}(i)$. In Sec.4.3 of the main paper, the point-wise attentional weight (Attention Point) uses $a_{ij} \in \mathbb{R}$, *i.e.*, the function is $\alpha : \mathbb{R}^{2M} \rightarrow \mathbb{R}$. The channel-wise attenional weight (Attention Channel) uses $a_{ij} \in \mathbb{R}^M$ and, in this case, $*$ denotes the element-wise product.

The attentional weights a_{ij} are based on the produced features $h(f_j)$ in order to determine the different contributions of the neighboring points. However, since the applied convolution kernel $h(\cdot)$ is still a fixed/isotripic one as we discussed in the main paper, they still cannot solve the intrisic limitation of current graph convolutions. The results of these ablation networks trained on the ShapeNetPart dataset are given in Sec.4.3 of the main paper. Furthermore, we show more comparisons on ModelNet40 for classification in Tab. 2.



(a) Spatial (b) Layer1 (c) Layer2 (d) Layer4 (e) Ours (f) GT

Figure 2. Visualize the Euclidean distances between the target point (green point in (a)) and other points in the feature space. Red color denotes a closer point and yellow one is far from the target. We show the feature distances in several layers of the network, which provides a clear insight that our network is able to distinguish points belonging to different semantic parts. It can also capture non-local similar structures (see the handles in the second row).

Method	mcIoU(%)	mIoU(%)
w/o STN	83.2	86.2
STN	83.4	86.4

Table 1. Segmentation results for models using STN.

Method	mAcc(%)	OA(%)
GraphConv	88.8	92.5
Attention Point	88.5	92.1
Attention Channel	89.2	92.2
Ours	90.7	93.4

Table 2. Results of ablation networks on ModelNet40.

Method	#parameters	Time(ms)	OA(%)
Baseline (w/o AdaptConv)	1.81M	93.1	92.5
AdaptConv (Layer2)	1.85M	129.1	93.4
AdaptConv (Layer3)	1.95M	168.4	93.0
AdaptConv (Layer4)	2.35M	276.0	93.2

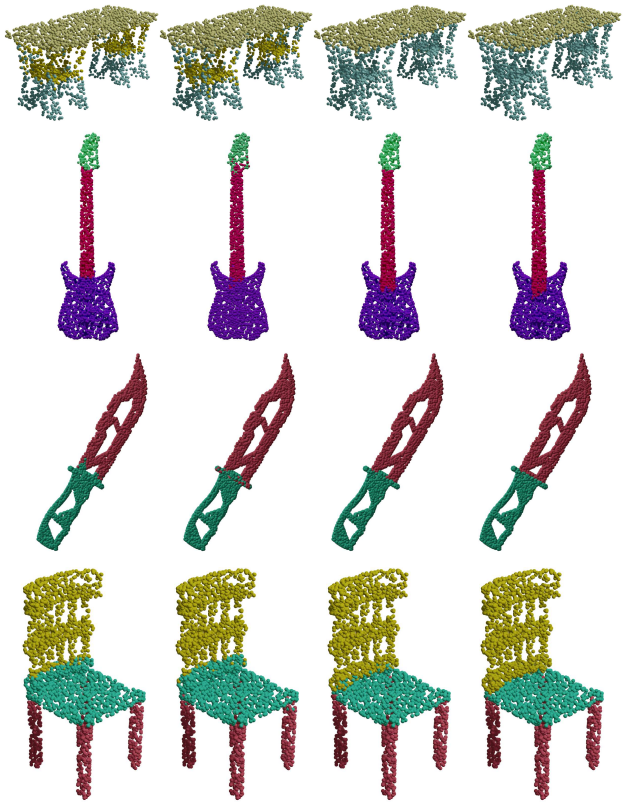
Table 3. Number of parameters, forward pass time (per batch) and overall accuracy for different models using AdaptConv.

C. Model complexity

The standard graph convolution adopted in this paper contains $2DM$ parameters ($2D$ denotes the dimension of feature input Δf_{ij}). Here, D and M denote the input and output dimensions respectively. As described in Sec. A, the kernel function uses a two-layer MLP which contains $dD + dcM$ parameters where c is the dimension of Δx_{ij} ($c = 3 \times 2$ for point coordinates $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ input). d is the dimension of hidden layer of the kernel function (see Fig. 1) and it can be adjusted to reduce the model size. In practice, we design the network architecture with two layers of AdaptConv which already achieves a pleasing performance. We further report the results and parameter numbers on ModelNet40 using different numbers of AdaptConv layers in Tab. 3. The adopted design (Layer2) significantly improves the network performance while the model size is relatively small. For the time performance evaluation, we also report the forward pass times of different models. The proposed AdaptConv layer is able to improve the performance of existing graph CNNs while being efficient.

D. More visualizations

In this section, we provide more results to further demonstrate the effectiveness of the proposed AdaptConv over fixed kernel methods. We first visualize the segmentation results on ShapeNetPart dataset in Fig. 3. In this experiment, we compare the results of DGCNN [3], attentional graph convolution (Attention Point described in Sec. B) and AdaptConv. Our results are better in challenging regions, such as part boundaries and object edges. This verifies that our method is able to capture distinguishable features for points belonging to different parts. More visualizations on the ShapeNetPart dataset are given in Fig. 4 and Fig. 5. In Fig. 5, we color the points with incorrect predicted labels in red.



(a) DGCNN (b) Attention (c) Ours (d) GT

Figure 3. Segmentation results on ShapeNet dataset. The labelled points are visualized in different colors. We compare our adaptive graph convolution with DGCNN [3] (standard graph convolution) and attentional convolution network (Attention Point). Our method produces better results especially for points close to the object boundaries and edges. Zoom in to see clearer.

We offer more insights of the adaptive mechanism from learned features. As shown in Fig. 2, a target point (green point) is picked up and we compute the distances in the feature space to other points. When the point is close to edges between different semantic parts, our network encourages it to have distinguishable features which captures better geometric information. Thus, it is separated from other parts of the objects, as shown in the first row of Fig. 2. Also, we see that in the second row of Fig. 2, points belonging to the same semantic part share similar features while they may not be spatially close. Note that, Fig. 2(a) indicates the spatial distances with regard to the central point.

References

[1] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *arXiv preprint arXiv:1506.02025*, 2015. 1

[2] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 1

[3] 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. 2, 3

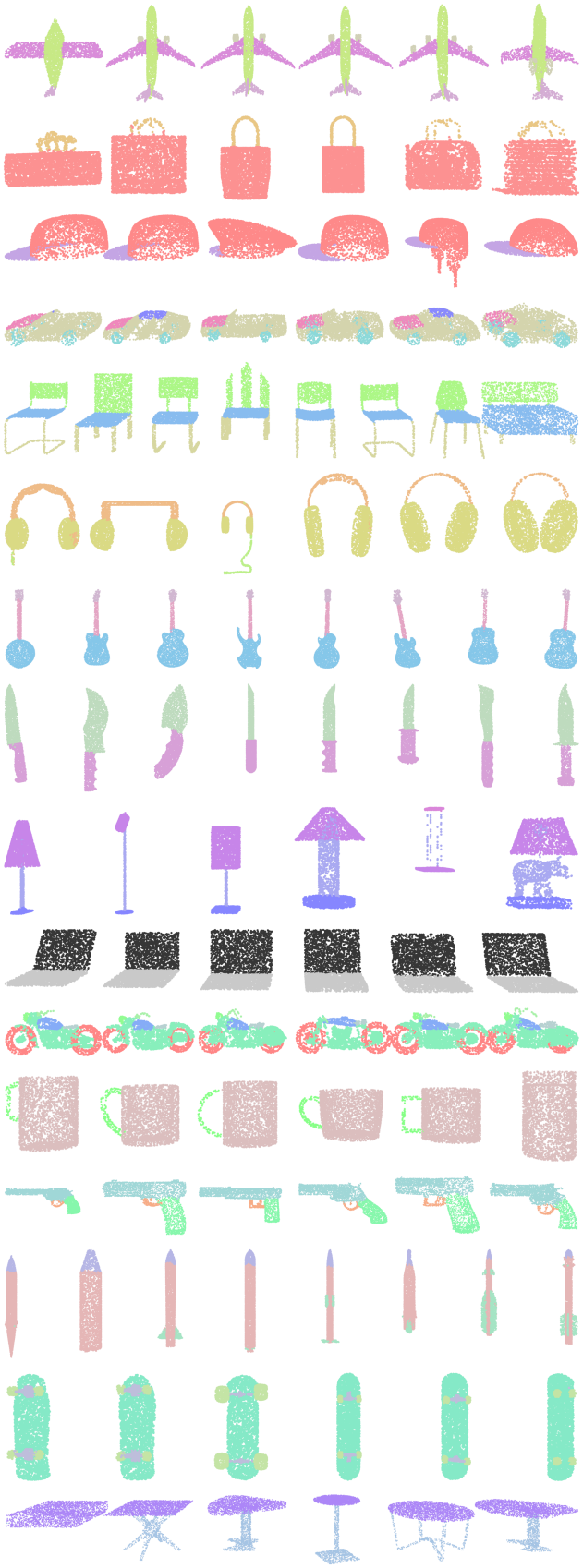


Figure 4. Part segmentation results on ShapeNet dataset.

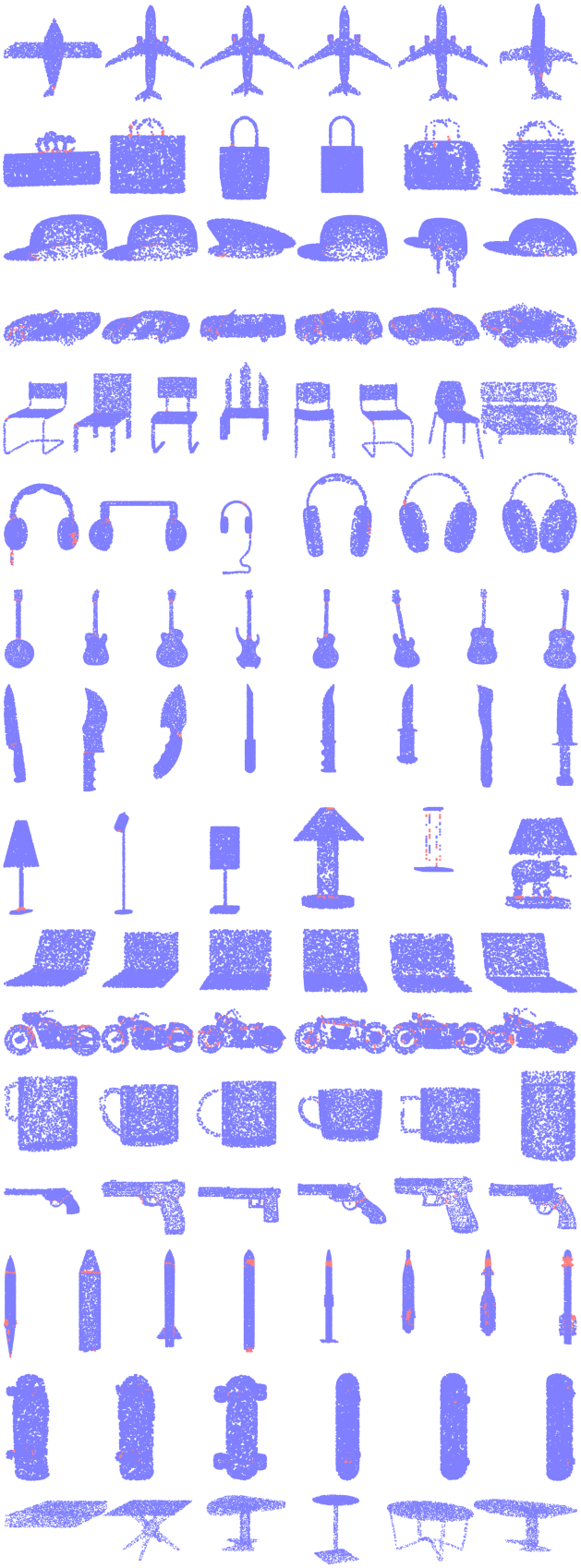


Figure 5. The error labels (red) compared with ground truth.