We thank all reviewers for their comments. All responses and changes will be incorporated into the revision. Code will be released for full reproducibility.

Baselines (R1, R3, R4). We compared PVCNN with SSCN (sparse-conv) and O-CNN (octree) as suggested by three reviewers. PVCNN is **4.8**× faster than SSCN with superior performance (Table 1). For O-CNN, the inference is fast; however, it is more than 10× slower to preprocess the input data (*i.e.*, condense the point cloud and construct the octree) and postprocess the output (*i.e.*, refine the boundaries using the dense CRF). Also, as pointed out by R1, these two approaches are orthogonal to our method and can be combined together: replacing the volumetric convolution with the sparse/octree-based convolution.

	PVCNN	SSCN	O-CNN
Mean IoU	**86.2**	86.0	85.9
Preprocess	0 ms	0 ms	86.9 ms
Inference	50.7 ms	241.8 ms	5.3 ms
Postprocess	0 ms	0 ms	842.4 ms

Table 1: Results on ShapeNet Part.

Scale Normalization (R4). We *only* normalize the point cloud in the voxel branch while keeping the coordinates in the point branch unchanged; therefore, there is no information loss after two branches are merged together. Without instance scale normalization, the voxel grids are more than 10× sparser on average, and the volumetric convolution is no longer effective to extract features. An alternative is to drop outlier points that do not lie in any voxel grids, which will inevitably induce information loss (see Table 2). Thus, the instance scale normalization is critical in the voxel branch.

Devoxelization (R1, R4). We compared different implementations of devoxelization. From Table 2, the trilinear interpolation (w/ TI) performs better than the nearest neighbor (w/o TI), which is because the points near the voxel boundaries will introduce larger fluctuations to the gradient, making it harder to optimize, as mentioned by R4.

Ablation Studies (R4). More analyses are provided in Table 2, including different feature fusion methods, voxel resolutions (also see Table 2 in the paper), and number of convolutions per block. Our design choice is the best. We'll add them to the paper.

	mIoU
PVCNN (on S3DIS)	**54.33**
w/o scale normalization	52.62

	mIoU
PVCNN (on ShapeNet)	**86.2**
devoxelization w/o TI	85.7
feature fusion by concat	86.1
voxel resolution (0.75×)	85.7
voxel resolution (1.25×)	86.1
1 voxel-conv per block	85.6
3 voxel-convs per block	86.1

Table 2: Ablation studies.

Evaluation on S3DIS (R4). We follow exactly the same data processing and evaluation protocol as PointCNN (L252) to make sure that the improvements are entirely from our proposed PVConv rather than different evaluation protocols. We thank R4 for suggesting the ScanNet2 dataset which we will experiment on.

Evaluation on KITTI (R1, R3). We choose F-PointNet as our baseline, which is a popular open-source 3D detection model. We do not compare with other state-of-the-art (voxel/point-based) models because the *region proposal network* has a large impact on the final results; we want to remove the influence of different region proposal networks and only evaluate the effectiveness of our *convolution primitive*. Besides, we do not present results on the testing set because F-PointNet only provides the 2D region proposals on the training and validation set.

Feature Visualization (R3). We illustrate the voxel and point branch features from the final PVConv in Figure 1. Note that warmer color represents larger magnitude. It is interesting to see that the voxel branch captures large, continuous parts while the point branch captures isolated, discontinuous details (*e.g.*, table legs, lamp necks). The two branches provide complementary information and can be explained by the fact that the convolution operation extracts features with continuity and locality.

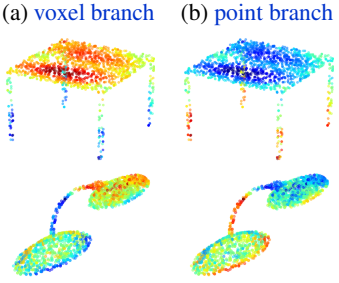

Figure 1: Feature visualization.

Statistical Significance (R1). On KITTI, our model has already been evaluated for **20 times** to reduce the variance (L266). On ShapeNet and S3DIS, our results have relatively small variances: PVCNN achieves (86.13 ± **0.04**)% in mIoU on ShapeNet, and PVCNN++ achieves (58.95 ± **0.08**)% in mIoU on S3DIS (trained for **4 times** on both datasets). We want to emphasize that our goal is to achieve high accuracy *as well as* better efficiency. Some improvements over PointCNN might be small in accuracy (0.1% in Table 1); however, the speedup is significant (**2.7**×).

Improvement Breakdown (R1). The speedup comes from our better algorithm with lower complexity rather than the implementation: PVCNN can *theoretically* save the number of incontiguous memory accesses by k times (where k is the number of neighbors) and achieve better locality (L190-194). Both baselines and our models are implemented using well-optimized deep learning libraries (PyTorch and TensorFlow). We do nothing further to optimize the speed.

Sublinear Memory Growth (R1). Asymptotically, the memory indeed grows cubically; however, the voxel resolution of PVCNN is kept very low thanks to the high-resolution point branch (L246-247). With low resolutions (L242-244), the memory consumption is dominated by the point branch, not the voxel branch. Thus, the memory grows sub-linearly.

Paper Presentation (R1, R4). We will revise our paper to make it concise and objective. We would like to clarify that the groups are assigned by the accuracy in Table 1 and 3 in the paper, and we put our PVCNN's into groups to highlight the speedup with similar accuracy. As suggested, we also compared *both* point-based and voxel-based models (see right) to make Figure 4 in the paper more comprehensive.