Supplementary Material for
SPLATNet: Sparse Lattice Networks for Point Cloud Processing

Hang Su, UMass Amherst
Varun Jampani, NVIDIA
Deqing Sun, NVIDIA
Subhransu Maji, UMass Amherst
Evangelos Kalogerakis, UMass Amherst
Ming-Hsuan Yang, UC Merced
Jan Kautz, NVIDIA

In this supplementary material, we provide additional details and explanations to help readers gain a better understanding of our technique.

1. Point Cloud Density Normalization

BCL has a normalization scheme to deal with uneven point density, or more specifically, the fact that some lattice vertices are supported by more data points than others. Input signals are filtered directly with the learnable filter kernels, and are also filtered in a separate second round with their values replaced by 1s with a Gaussian kernel. The filter responses in the second round are then used for normalizing responses from the first round. This is similar to using homogeneous coordinates, which are widely adopted in bilateral filtering implementations such as [1].

2. RueMonge2014 Facade Segmentation

Network architecture of SPLATNet$_{3D}$. We use 5 BCLs ($T = 5$) followed by 2 1×1 CONV layers in SPLATNet$_{3D}$ for the facade segmentation task. We omit the initial 1×1 CONV layer since we find it has no effect on the overall performance. The number of output channels in each layer are: `B64-B128-B128-B128-B64-C64-C7`. Note that although written as a linear structure, the network has skip connections from all BCLs (layers start with '`B`') to the penultimate 1×1 CONV layer. We use an initial scale $\Lambda_0 = 32I_3$ for scaling lattice features XYZ, and divide the scale in half after each BCL: $(32I_3, 16I_3, 8I_3, 4I_3, 2I_3)$. The unit of raw input features XYZ is meter, with Y (aligned with gravity axis) having a range of 7.1 meters. For all the BCLs, we use filters operating on one-ring neighborhoods on the lattice.

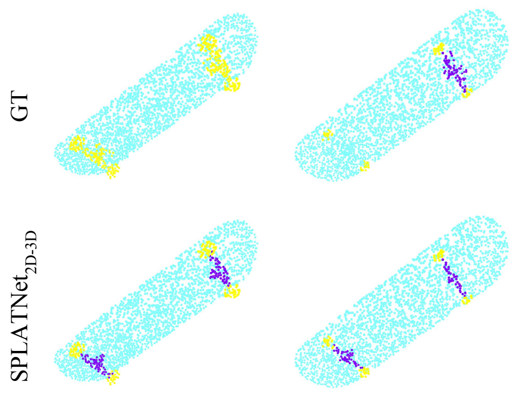
Network architecture of SPLATNet$_{2D\text{-}3D}$. We use SPLATNet$_{3D}$ as described above as the 3D component of our 2D-3D joint model. The '2D-3D Fusion' component has 2 1×1 CONV layers: `C64-C7`. DeepLab [2] segmentation architecture is used as CNN$_1$. CNN$_2$ is a small network with 2 CONV layers: `C32-C7`, where the first layer has 3×3 filters and 32 output channels, and the second one has 1×1 filters and 7 output channels. We use $\Lambda_a = 64$ and $\Lambda_b = 1000$ for 2D$\leftrightarrow$3D projections with BCLs. Note that the dataset provides one-to-many mappings from 3D points to pixels. By using a very large scale (*i.e.*, $\Lambda_b = 1000$), 3D unaries are directly mapped to the corresponding 2D pixel locations without any interpolation.

Training. We randomly sample facade segments of 60k points and use a batch size of 4 when training SPLATNet$_{3D}$. CNN$_1$ is initialized with Pascal VOC [3] pre-trained weights and fine-tuned for 2D facade segmentation. Adam optimizer [4] with an initial learning rate of 0.0001 is used for training both SPLATNet$_{3D}$ and SPLATNet$_{2D\text{-}3D}$. Since the training data is small, we augment point cloud training data with random rotations, translations, and small color perturbations. We also augment 2D image data with small color perturbations during training.

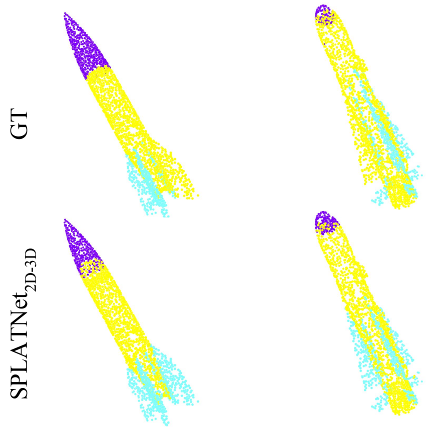
3. ShapeNet Part Segmentation

Network architecture of SPLATNet$_{3D}$. We use a 1×1 CONV layer in the beginning, followed by 5 BCLs ($T = 5$), and then 2 1×1 CONV layers in SPLATNet$_{3D}$ for the ShapeNet part segmentation task. The number of output channels in each layer are: `C32-B64-B128-B256-B256-B256-C128-Cx`. 'x' in the last CONV layer denotes the number of part categories, and ranges from 2-6 for different object categories. We use an initial scale $\Lambda_0 = 64I_3$ for scaling lattice features XYZ, and divide the scale in half after each BCL: $(64I_3, 32I_3, 16I_3, 8I_3, 4I_3)$.

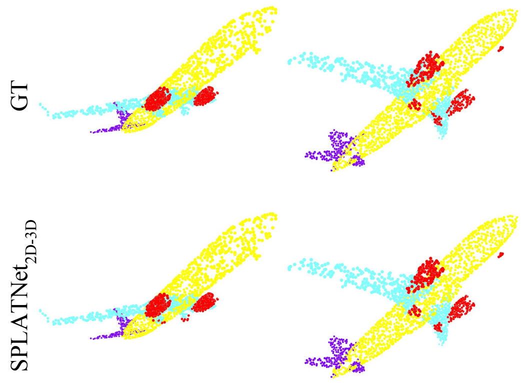
Network architecture of SPLATNet$_{2D\text{-}3D}$. We use SPLATNet$_{3D}$ as described above as the 3D component of the joint model. The '2D-3D Fusion' component has 2 1×1 CONV layers: `C128-Cx`. The same DeepLab architecture is used for CNN$_1$. We use $\Lambda_a = 32$ in BCL$_{2D\rightarrow 3D}$. Since 2D prediction is not needed, CNN$_2$ and BCL$_{3D\rightarrow 2D}$ are omitted.



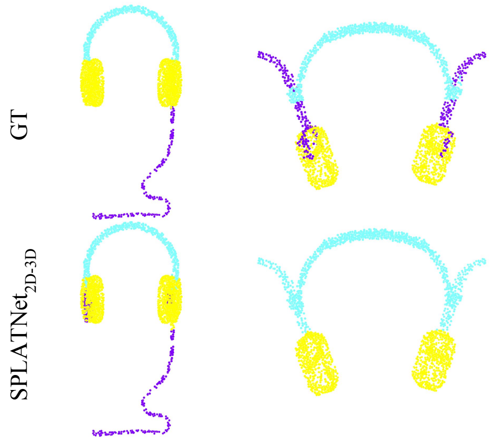
(a) Incorrect labels



(b) Incomplete labels



(c) Inconsistent labels



(d) Confusing labels

Figure 1: **Labeling issues in the ShapeNet Part dataset.** Four types of labeling issues are shown here. Two examples from the test set are given for each type, where the first row shows the ground-truth labels and the second row shows our predictions with SPLATNet$_{2D\text{-}3D}$. Our predictions appear to be more accurate than the ground truth in some cases (see the skateboard axles in 1a and the rocket fins in 1b).

Training. We train separate models for each object category. CNN$_1$ is initialized the same way as in the facade experiment. Adam optimizer with an initial learning rate of 0.0001 is used. We augment point cloud data with random rotations, translations, and scalings during training.

We train our networks until validation loss plateaus. Training SPLATNet$_{3D}$ and SPLATNet$_{2D\text{-}3D}$ take about 2.5 and 3 days respectively. With default settings, training PointNet++ takes 3.5 days on the same hardware.

Dataset labeling issues. We observed a few types of labeling issues in the ShapeNet Part dataset:

- Some object part categories are frequently labeled incorrectly. *E.g.*, skateboard axles are often mistakenly labeled as 'deck' or 'wheel' (Figure 1a).
- Some object parts, *e.g.* 'fin' of some rockets, have incomplete range or coverage (Figure 1b).
- Some object part categories are labeled inconsistently between shapes. *E.g.*, airplane landing gears are seen labeled as 'body', 'engine', or 'wings' (Figure 1c).
- Some categories have parts that are labeled as 'other', which can be confusing for the classifier as these parts do not have clear semantic meanings or structures. *E.g.*, in the case of earphones, anything that is not 'headband' or 'earphone' are given the same label ('other') (Figure 1d).

The first two issues make evaluations and comparisons on the benchmark less reliable, while the other two make learning ill-posed or unnecessarily hard for the networks.

References

[1] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics*

Forum, 29(2):753–762, 2010. 1

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *Proc. ICLR*, 2015. 1

[3] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge: A retrospective. *IJCV*, 111(1):98–136, Jan. 2015. 1

[4] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014. 1