1. Network Architecture

In our experiments, we use both PointNet and PointNet++ as our baseline architectures. For the S3DIS dataset, we use PointNet as our baseline for fair comparison with the 3D object detection system described in the PointNet paper [2]. The network architecture is the same as the semantic segmentation network as stated in PointNet except for the last two layers. Our $F$ is the last $1 \times 1$ conv layer with BatchNorm and ReLU in PointNet with 256 output channels. $F_{SIM}, F_{CF}, F_{SEM}$ are $1 \times 1$ conv layers with output channels (128, 128, 128), respectively.

For the NYUV2 dataset, we use PointNet++ as our baseline. We use the same notations as PointNet++ to describe our architecture:

$SA(K,r,[l_1,...,l_d])$ is a set abstraction (SA) level with $K$ local regions of ball radius $r$ using a PointNet architecture of $d \times 1$ conv layers with output channels $l_i (i = 1,...,d)$. $FP(l_1,...,l_d)$ is a feature propagation (FP) layer with $d \times 1$ conv layers. Our network architecture is:

\[
SA(1024, 0.1, [32, 32, 64]), \\
SA(256, 0.2, [64, 64, 128]), \\
SA(128, 0.4, [128, 128, 256]), \\
SA(64, 0.8, [256, 256, 256]), \\
SA(16, 1.2, [256, 256, 512]), \\
FP(512, 256), \\
FP(256, 256), \\
FP(256, 256), \\
FP(128, 128, 128, 128).
\]

$F_{SIM}, F_{CF}, F_{SEM}$ are $1 \times 1$ conv layers with output channels (128, 128, 128) respectively.

For our experiments on the ShapeNet part dataset, PointNet++ is used as our baseline. We use the same network architecture as in the PointNet++ paper [3].

2. Experiment Settings

2.1. S3DIS Dataset

Block Merging We divide each scene into $1m \times 1m$ blocks with overlapping sliding windows in a snake pattern of stride $0.5m$ as is shown in Figure 1. The entire scene is also divided into a $400 \times 400 \times 400$ grid $V$. $V_k$ is used to indicate the instance label of cell $k$ where $k \in [0, 400 \times 400 \times 400]$. Given $V$ and point instance labels for each block $PL$ where $PL_{ij}$ represents the instance label of $j$th point in block $i$, a BlockMerging algorithm (refer to Algorithm 1) is derived to merge object instances from different blocks.

In Figure 2, we show more qualitative results of instance segmentation with SGPN.
blocks and each block is uniformly sampled into 4096 points for training. All points in the block are used at test time. Each point is represented by a 9D vector (XYZ, RGB, and normalized location with respect to the room scene). PointNet++ is used as the baseline. The network architecture is:

\begin{align*}
SA(1024, 0.1, [32, 32, 64]), \\
SA(256, 0.2, [64, 64, 128]), \\
SA(64, 0.4, [128, 128, 256]), \\
SA(16, 0.8, [256, 256, 512]), \\
FP(256, 256), \\
FP(256, 256), \\
FP(256, 128), \\
FP(128, 128, 128, 128).
\end{align*}

And $F_{SIM}, F_{CF}, F_{SEM}$ are $1 \times 1$ conv layers with output channels $(128, 128, 128)$ respectively. Table 1 illustrates the quantitative comparison results with Seg-Cluster. The metric is average precision (AP) with IoU threshold 0.5. Figure 3 shows instance segmentation results on ScanNet.

References


Figure 2: Instance segmentation results on S3DIS with SGPN. Different colors represent different instances. The colors of the same object in ground truth and prediction are not necessarily the same.
Figure 3: Instance segmentation results on ScanNet with SGPN. Different colors represent different instances. The colors of the same object in ground truth and prediction are not necessarily the same.