# Supplemental Materials: Finding Your (3D) Center

We will here give further details of our architecture. For simplicity we ignore batch dimensions. All branches for both `loss` and `scene` networks have a shared PointNet++ encoder shown in Tbl. 1 with input 4,096 and 32,768 respectively. We use the original "Single Scale Grouping" architecture from Qi et al. [1]. Unlike the original implementation no batch normalization, weight regularization or dropout layers are used. We define the implemented architectures in Tbl. 1 and Tbl. 2.

**Table 1.** PointNet++ encoder consisting of four "Set Abstraction" (SA) layers. Input is either 32,768 or 4,096 points for training `scene` or `loss` networks respectively. *Points* indicates the number of points selected from the fartest point sampling algorithm. *Samples* is the maximum points per neighborhood defined using a ball point query with radius $r$.

| Layer | MLP | Points | Samples | Radius | Activation | Output Shape |
|---|---|---|---|---|---|---|
| Input | | | | | | $(32768/4096) \times 3$ |
| SA | [64, 64, 128] | 1024 | 32 | 0.1 | ReLU | $1024 \times 128$ |
| SA | [64, 64, 128] | 256 | 32 | 0.2 | ReLU | $256 \times 128$ |
| SA | [128, 128, 256] | 64 | 64 | 0.4 | ReLU | $64 \times 256$ |
| SA | [256, 512, 128] | None | None | None | ReLU | $1 \times 128$ |

Our *local* encoder is a 3 layer SSG network, which typically consists of half the number of units for each respective MLP layer.

**Table 2.** PointNet++ local encoder consisting of 3 SA layers. See Tbl. 1 caption for further details regarding columns.

| Layer | MLP | Points | Samples | Radius | Activation | Output Shape |
|---|---|---|---|---|---|---|
| Input | | | | | | $4096 \times 3$ |
| SA | [32, 32, 64] | 1024 | 32 | 0.1 | ReLU | $256 \times 128$ |
| SA | [64, 64, 128] | 256 | 32 | 0.2 | ReLU | $64 \times 256$ |
| SA | [128, 128, 128] | None | None | None | ReLU | $1 \times 128$ |

Our loss network architecture has a shared PointNet++ encoder with three branches; gradient, property and objectness. Each branch consists of two fully-connected layers with a ReLU activation for the hidden layer. We use `softmax` for both single and multi-class examples. For single class where $k = 2$ the first

class represents `empty` and the second class represents `present`. In a multi-class setting $k$ is the number of classes. We do not have a `empty` class for multi-class. Both gradient and property branches do not have a final layer activation function.

**Table 3.** Loss network architecture. PNet++ is defined in Tbl. 1.

| Branch | Layer | Kernel | Activation | Output Shape |
|---|---|---|---|---|
| Trunk | PNet++ | | ReLU | $1 \times 128$ |
| Gradient | FC | $1 \times 1$ | ReLU | $1 \times 512$ |
| | FC | $1 \times 1$ | None | $1 \times 3$ |
| Property | FC | $1 \times 1$ | ReLU | $1 \times 512$ |
| | FC | $1 \times 1$ | None | $1 \times 5$ |
| Objectness | FC | $1 \times 1$ | ReLU | $1 \times 512$ |
| | FC | $1 \times 1$ | Softmax | $1 \times k$ |

The *scene* network architecture is similar to the loss network but is designed for multiple proposals $n$ as apposed to the loss network where $n = 1$. The property and objectness branches take patch codes generated by the local encoder. The patches are local crops from the center branch proposals. Note that the output of the PointNet++ encoder and local encoder are both $1 \times 128$. These latent codes are concatenated for input into the scene network property and objectness branches.

**Table 4.** Scene network architecture. PNet++ and PNet++ local are defined in Tbl. 1 and 2 respectively.

| Branch | Layer | Kernel | Activation | Output Shape |
|---|---|---|---|---|
| Trunk | PNet++ encoder | | ReLU | $1 \times 128$ |
| Center | FC | $1 \times 1$ | ReLU | $n \times 512$ |
| | FC | $1 \times 1$ | None | $n \times 3$ |
| Property | PNet++ local | - | ReLU | $n \times 128$ |
| | FC | $1 \times 1$ | ReLU | $n \times 512$ |
| | FC | $1 \times 1$ | None | $n \times 5$ |
| Objectness | PNet++ local | - | ReLU | $n \times 128$ |
| | FC | $1 \times 1$ | ReLU | $n \times 512$ |
| | FC | $1 \times 1$ | Softmax | $n \times k$ |

All network weights were initialized using the Glorot Normal initialization. We do not employ regularization or batch normalization in any of the networks.

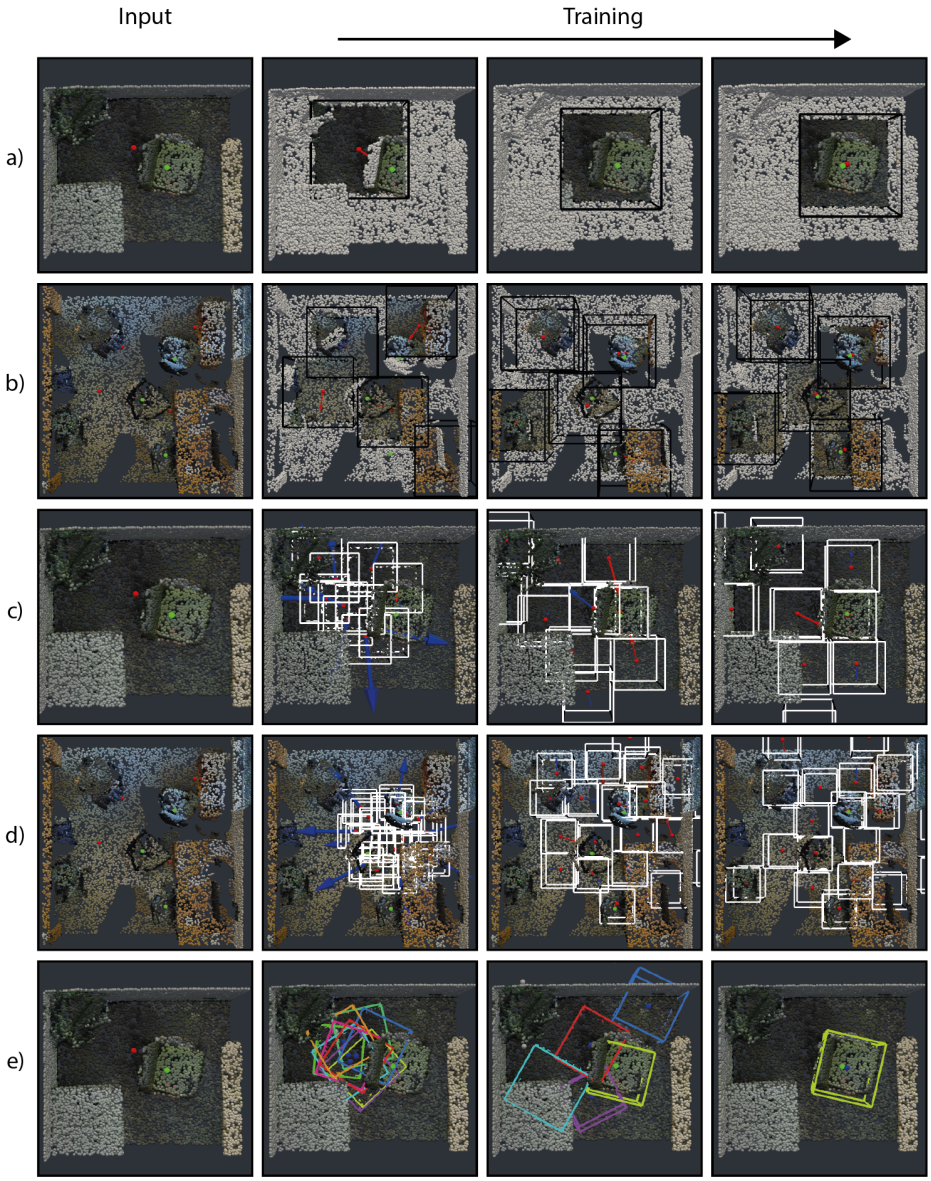


**Fig. 1.** Visualizations for the *scene network* during training. **a)** A scene consisting of a single proposal and single object. Colored points indicate those passed seen by the *loss network* for gradient estimation (red arrow). **b)** A scene consisting of 5 proposals and 5 objects. **c)** Here we show 10 proposals with 1 object. Our overlap penalty (blue arrow) is applied to all but the closest proposal to an object. **d)** 20 proposals with 5 objects. **e)** 10 proposals with 1 object. Here we visualize the final scene network output which consists of location, bounding box and orientation. Proposals with an objectness score below a threshold are shown as grey dots. For full video sequences visit the project page at: `dgriffiths3.github.io`.

# Bibliography

[1] Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv:1706.02413 (2017)