Deep Classification and Semantic Segmentation of 3D Point cloud

Maryam Jameela

Workshop: Deep Learning

Date: June 21, 2021 13:00 pm to 15:00 pm Mountain Daylight Time

Agenda

- System Configuration
- Classification
  - PointNet
  - ModelNet40
- Segmentation
  - 3D U-Net
  - Utility Data

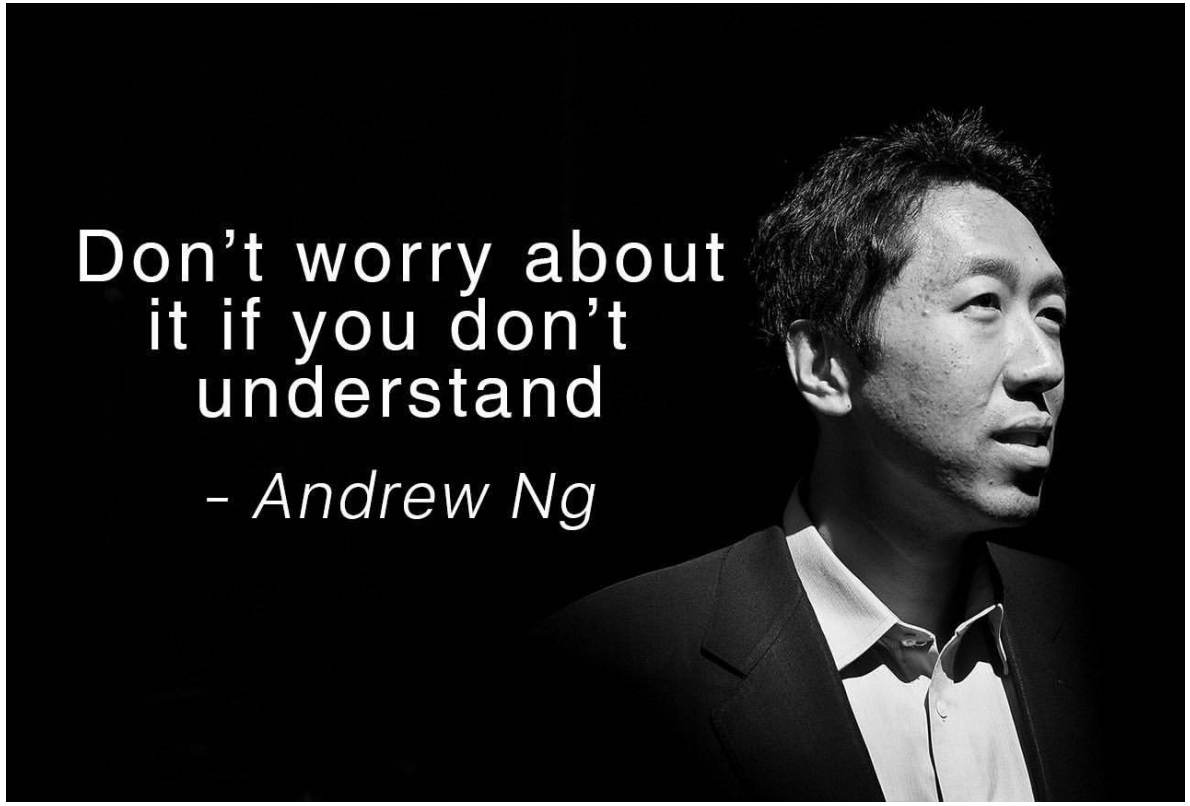
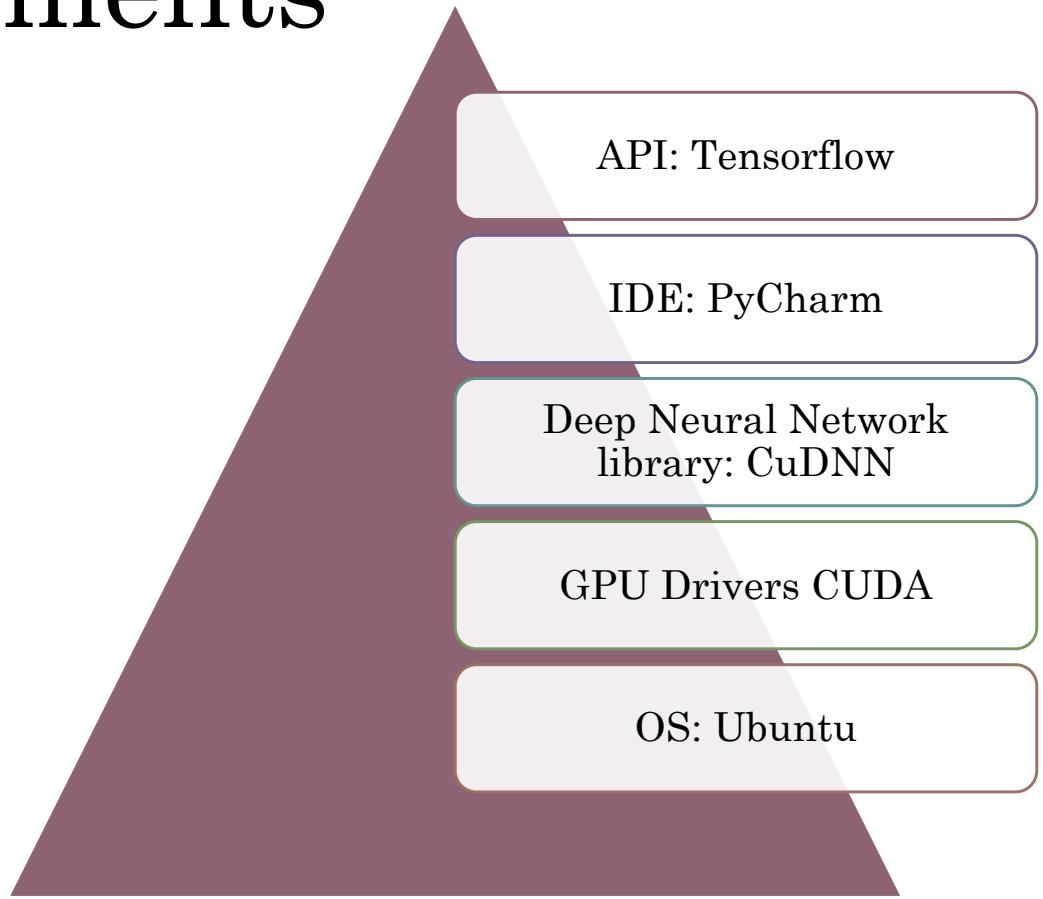


Image Retrieved from: https://medium.datadriveninvestor.com/thoughts-on-andrew-ngs-machine-learning-course-7724df76320f

System Configuration

System Requirements

- Operating System
  - Ubuntu 18.04
- Environment
  - CUDA 10.0
  - CuDNN 7.4
  - Pycharm 2021
- Install
  - Tensorflow 1.14
  - Numpy
  - H5py



Compatibility Table

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow-2.5.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.4.0	3.6-3.8	GCC 7.3.1	Bazel 3.1.0	8.0	11.0
tensorflow-2.3.0	3.5-3.8	GCC 7.3.1	Bazel 3.1.0	7.6	10.1
tensorflow-2.2.0	3.5-3.8	GCC 7.3.1	Bazel 2.0.0	7.6	10.1
tensorflow-2.1.0	2.7, 3.5-3.7	GCC 7.3.1	Bazel 0.27.1	7.6	10.1
tensorflow-2.0.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10.0
tensorflow_gpu-1.15.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10.0
tensorflow_gpu-1.14.0	2.7, 3.3-3.7	GCC 4.8	Bazel 0.24.1	7.4	10.0
tensorflow_gpu-1.13.1	2.7, 3.3-3.7	GCC 4.8	Bazel 0.19.2	7.4	10.0

https://www.tensorflow.org/install/source#linux

CUDA

CUDA Toolkit 10.0 Archive

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System	Windows, **Linux**, Mac OSX
Architecture	**x86_64**, ppc64le
Distribution	Fedora, OpenSUSE, RHEL, CentOS, SLES, **Ubuntu**
Version	**18.04**, 16.04, 14.04
Installer Type	runfile (local), **deb (local)**, deb (network), cluster (local)

Download Installers for Linux Ubuntu 18.04 x86_64

The base installer is available for download below.
There is 1 patch available. This patch requires the base installer to be installed first.

Base Installer — Download (1.6 GB)

Installation Instructions:

1. `sudo dpkg -i cuda-repo-ubuntu1804-10-0-local-10.0.130-410.48_1.0-1_amd64.deb`
2. `sudo apt-key add /var/cuda-repo-<version>/7fa2af80.pub`
3. `sudo apt-get update`
4. `sudo apt-get install cuda`

Other installation options are available in the form of meta-packages. For example, to install all the library packages, replace "cuda" with the "cuda-libraries-10-0" meta package.
For more information on all the available meta packages click here.

https://developer.nvidia.com/cuda-10.0-download-archive?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=deblocal

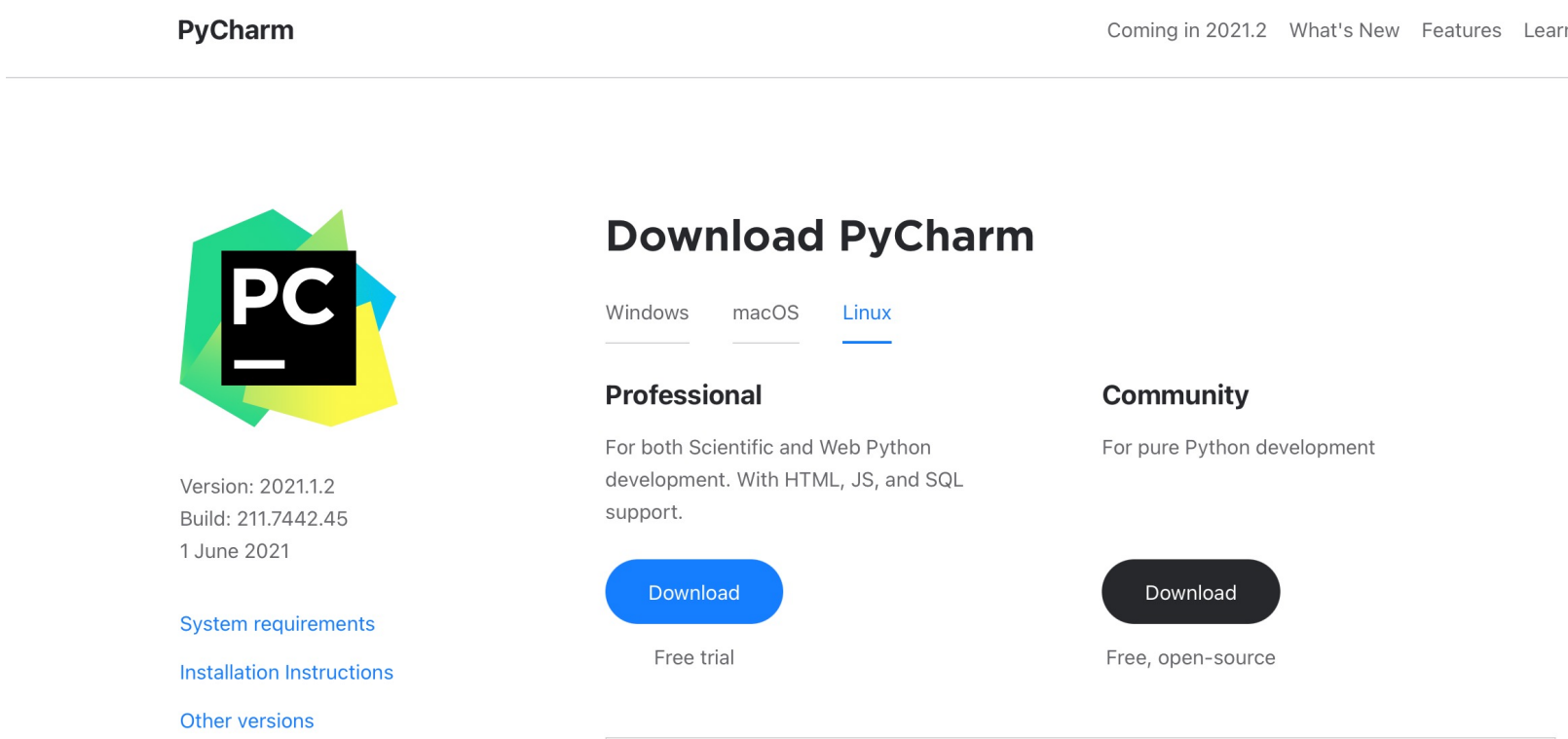
CuDNN

1. Register a nvidia developer account and download cudnn
2. Check where your cuda installation is.
3. Copy the files:

```
$ cd folder/extracted/contents
$ sudo cp include/cudnn.h /usr/local/cuda/include
$ sudo cp lib64/libcudnn* /usr/local/cuda/lib64
$ sudo chmod a+r /usr/local/cuda/lib64/libcudnn*
```

https://developer.nvidia.com/cudnn

PyCharm

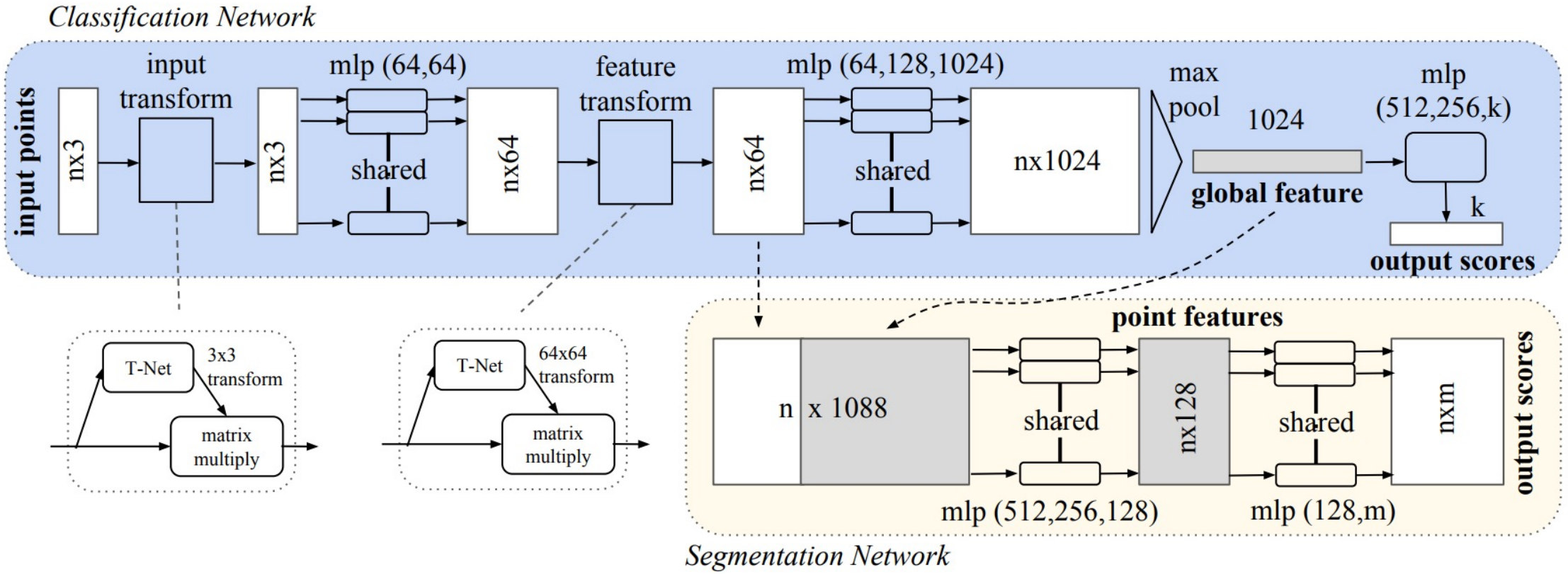


https://www.jetbrains.com/pycharm/download/#section=linux

Tensorflow

- pip3 install tensorflow==1.14.0
- pip3 install tensorflow-gpu==1.14.0
- pip3 install numpy
- pip3 install h5py

Demo

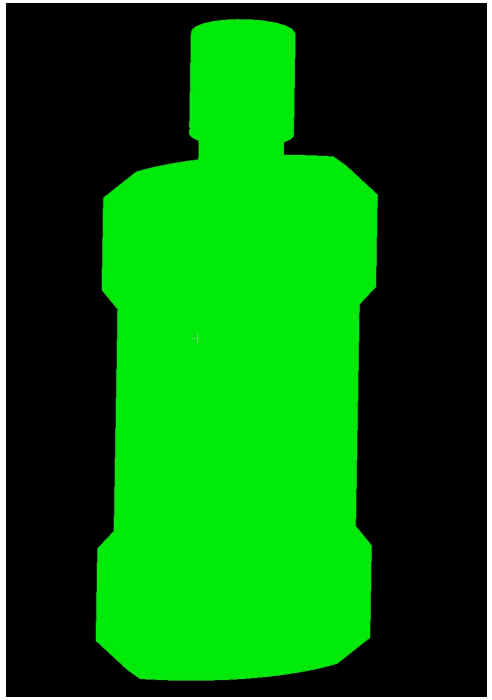


PointNet: Classification Demo

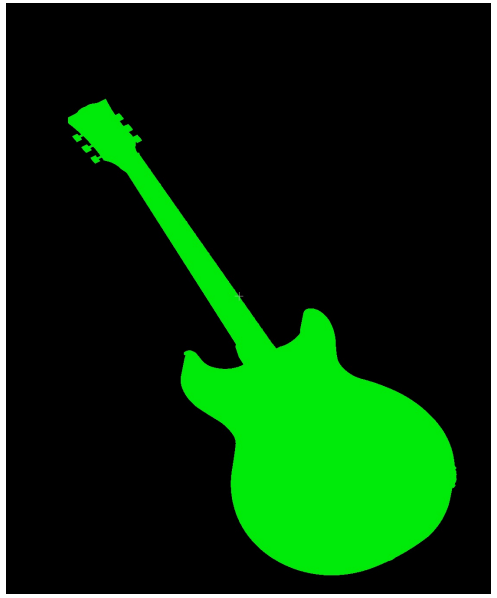
Pointnet: Deep Learning on Point Sets for 3d Classification and Segmentation [Qi et al., 2017]

Dataset

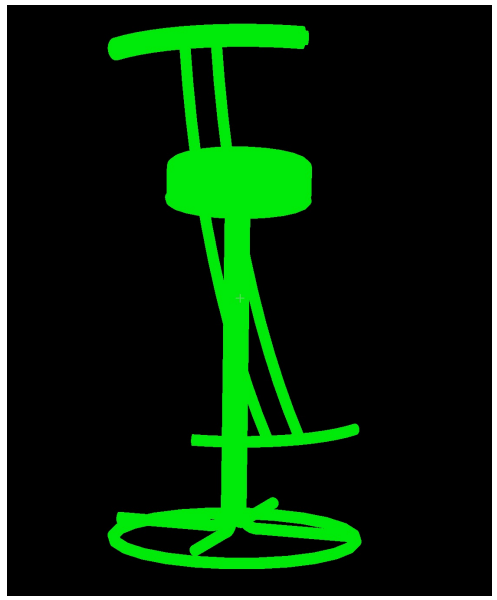
- ModelNet40: 40 Classes
- Examples:



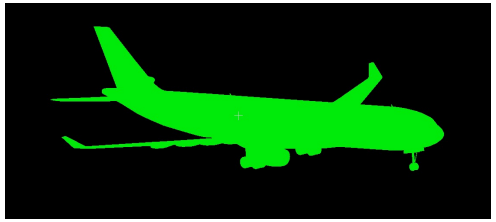
Bottle



Guitar

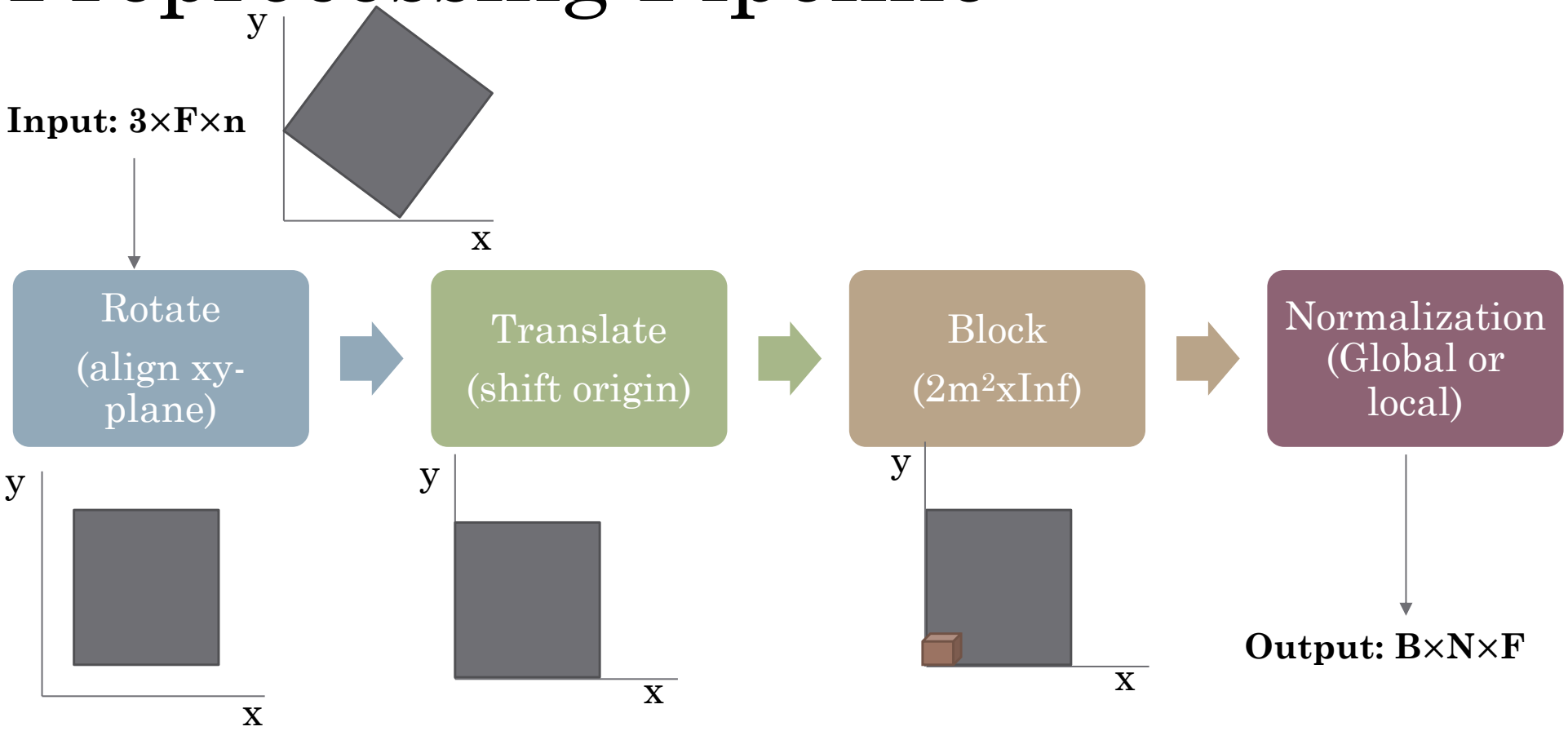


Stool



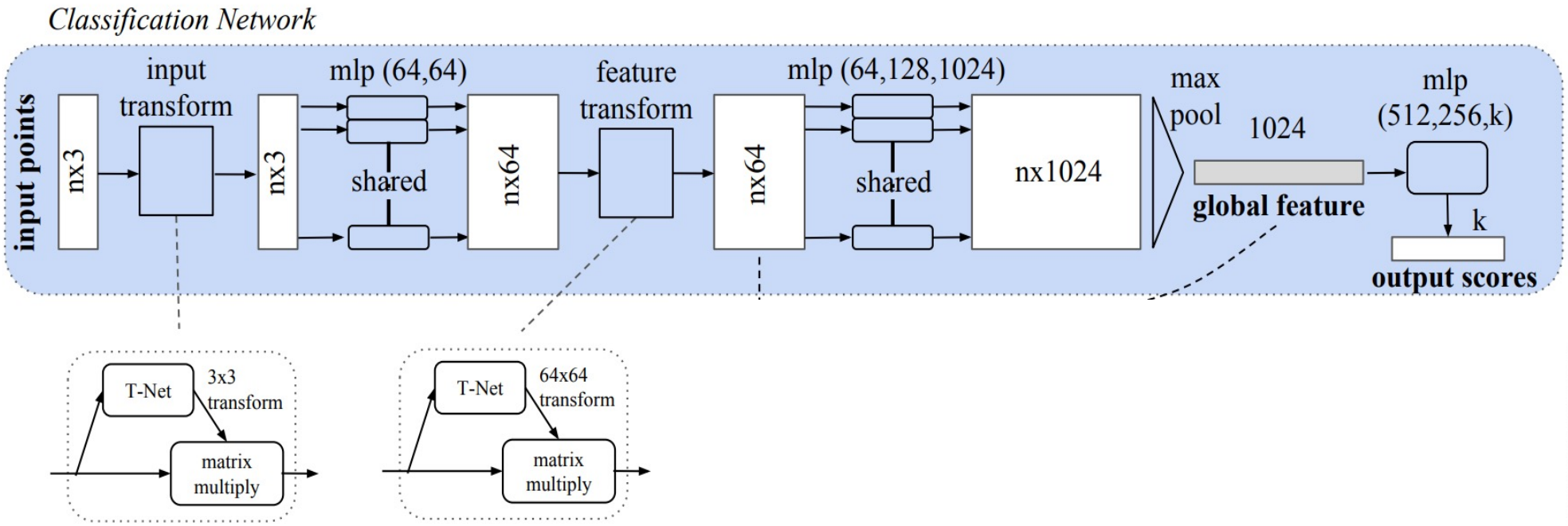
Airplane

Preprocessing Pipeline

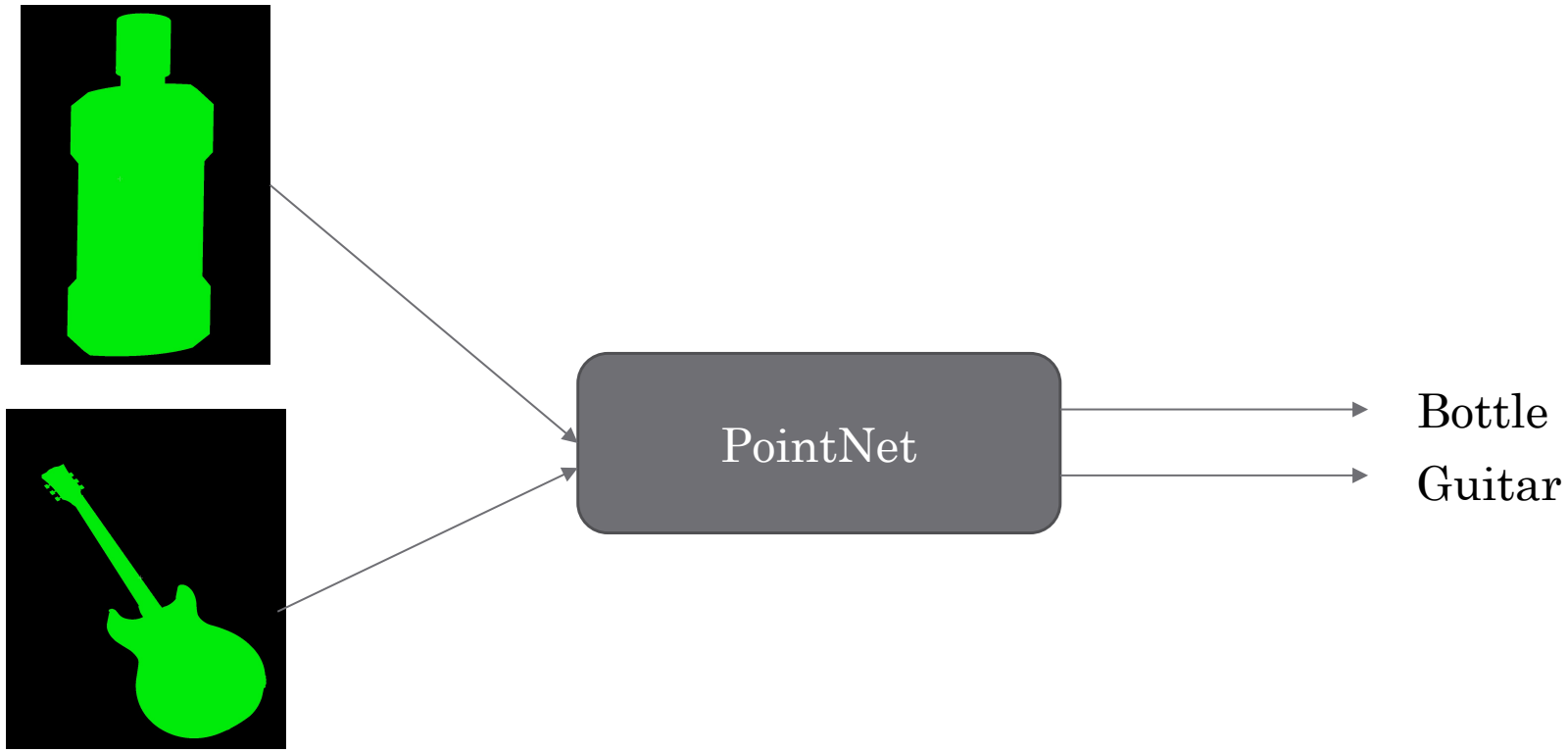


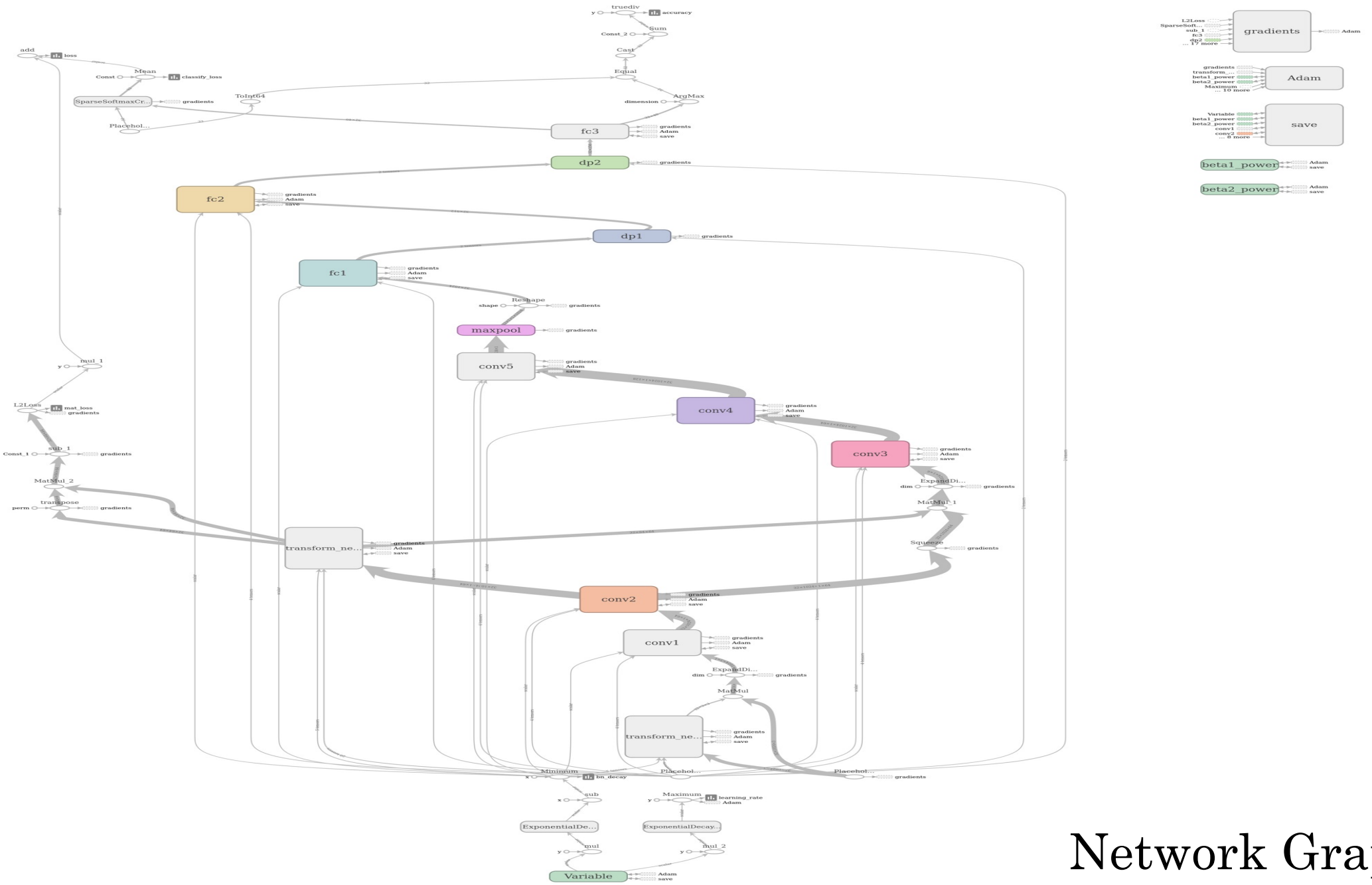
F: Features
B: Blocks
N: Maximum number of points in a block
n: Total number of points in point cloud

Network


Classification Network
input points
nx3
input transform
nx3
mlp (64,64)
shared
nx64
feature transform
nx64
mlp (64,128,1024)
shared
nx1024
max pool
1024
global feature
mlp (512,256,k)
k
output scores
T-Net
3x3 transform
matrix multiply
T-Net
64x64 transform
matrix multiply

Classification





Network Graph

Input T-Net

First Spatial Transformer is to adjust the point cloud in the space. Intuitively, rotating an angle, such as turning an object to the front, is conducive to classification or segmentation.

README.md × train.py × pointnet_cls.py × home/.../train.py ×

```
17
18      def get_model(point_cloud, is_training, bn_decay=None):
19          """ Classification PointNet, input is BxNx3, output Bx40 """
20          batch_size = point_cloud.get_shape()[0].value
21          num_point = point_cloud.get_shape()[1].value
22          end_points = {}
23
24          with tf.variable_scope('transform_net1') as sc:
25              transform = input_transform_net(point_cloud, is_training, bn_decay, K=3)
26          point_cloud_transformed = tf.matmul(point_cloud, transform)
27          input_image = tf.expand_dims(point_cloud_transformed, -1)
```

Feature T-Net

The feature transformer is to convert the features into high semantic information of the scene

```
18    def get_model(point_cloud, is_training, bn_decay=None):
19        """ Classification PointNet, input is BxNx3, output Bx40 """
20        batch_size = point_cloud.get_shape()[0].value
21        num_point = point_cloud.get_shape()[1].value
22        end_points = {}
23
24        with tf.variable_scope('transform_net1') as sc:
25            transform = input_transform_net(point_cloud, is_training, bn_decay, K=3)
26        point_cloud_transformed = tf.matmul(point_cloud, transform)
27        input_image = tf.expand_dims(point_cloud_transformed, -1)
28
29        net = tf_util.conv2d(input_image, 64, [1,3],
30                             padding='VALID', stride=[1,1],
31                             bn=True, is_training=is_training,
32                             scope='conv1', bn_decay=bn_decay)
33        net = tf_util.conv2d(net, 64, [1,1],
34                             padding='VALID', stride=[1,1],
35                             bn=True, is_training=is_training,
36                             scope='conv2', bn_decay=bn_decay)
37
38        with tf.variable_scope('transform_net2') as sc:
39            transform = feature_transform_net(net, is_training, bn_decay, K=64)
40        end_points['transform'] = transform
41        net_transformed = tf.matmul(tf.squeeze(net, axis=[2]), transform)
42        net_transformed = tf.expand_dims(net_transformed, [2])
43
```

Global Feature, Dropout & Classification

README.md | train.py | pointnet_cls.py | home/.../train.py

```
44      net = tf_util.conv2d(net_transformed, 64, [1,1],
45                           padding='VALID', stride=[1,1],
46                           bn=True, is_training=is_training,
47                           scope='conv3', bn_decay=bn_decay)
48      net = tf_util.conv2d(net, 128, [1,1],
49                           padding='VALID', stride=[1,1],
50                           bn=True, is_training=is_training,
51                           scope='conv4', bn_decay=bn_decay)
52      net = tf_util.conv2d(net, 1024, [1,1],
53                           padding='VALID', stride=[1,1],
54                           bn=True, is_training=is_training,
55                           scope='conv5', bn_decay=bn_decay)
56
57      # Symmetric function: max pooling
58      net = tf_util.max_pool2d(net, [num_point,1],
59                               padding='VALID', scope='maxpool')
60
61      net = tf.reshape(net, [batch_size, -1])
62      net = tf_util.fully_connected(net, 512, bn=True, is_training=is_training,
63                                    scope='fc1', bn_decay=bn_decay)
64      net = tf_util.dropout(net, keep_prob=0.7, is_training=is_training,
65                            scope='dp1')
66      net = tf_util.fully_connected(net, 256, bn=True, is_training=is_training,
67                                    scope='fc2', bn_decay=bn_decay)
68      net = tf_util.dropout(net, keep_prob=0.7, is_training=is_training,
69                            scope='dp2')
70      net = tf_util.fully_connected(net, 40, activation_fn=None, scope='fc3')
71
72      return net, end_points
73
74
```

Experimental Configuration

- Configuration
  - 70-30 Split Training and Test
  - Block size= $2m^2$xInf,
  - Batch size=32x2048x6
  - 1 GPU GTX 2080
  - Training Time: 12 hours (250 Epochs)
  - Inference: 30 seconds
- Evaluation Configuration
  - Accuracy

Training

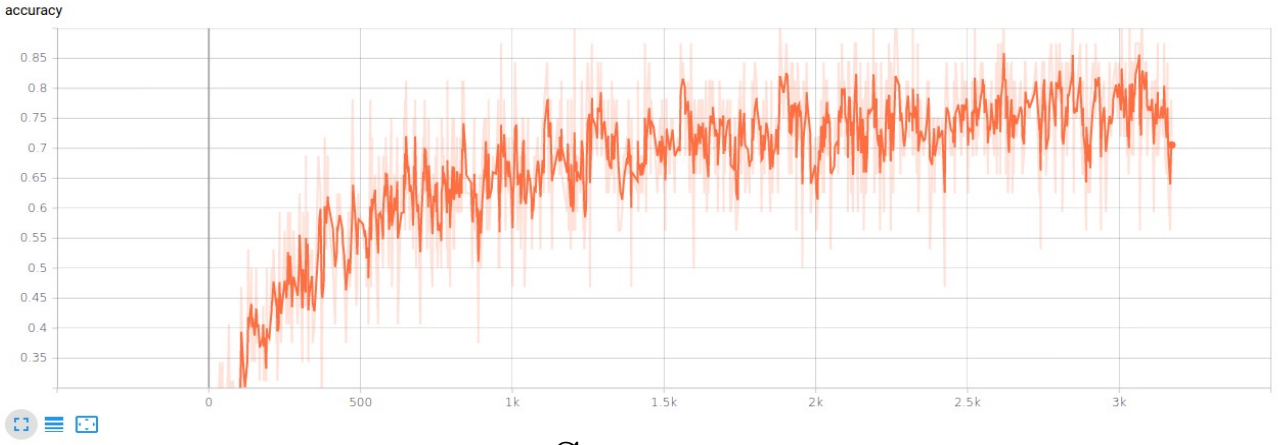
- python3 train.py
- Hyperparameter:

```
parser = argparse.ArgumentParser()
parser.add_argument('--gpu', type=int, default=0, help='GPU to use [default: GPU 0]')
parser.add_argument('--model', default='pointnet_cls', help='Model name: pointnet_cls or pointnet_cls_basic [default: pointnet_cls]')
parser.add_argument('--log_dir', default='log', help='Log dir [default: log]')
parser.add_argument('--num_point', type=int, default=1024, help='Point Number [256/512/1024/2048] [default: 1024]')
parser.add_argument('--max_epoch', type=int, default=250, help='Epoch to run [default: 250]')
parser.add_argument('--batch_size', type=int, default=32, help='Batch Size during training [default: 32]')
parser.add_argument('--learning_rate', type=float, default=0.001, help='Initial learning rate [default: 0.001]')
parser.add_argument('--momentum', type=float, default=0.9, help='Initial learning rate [default: 0.9]')
parser.add_argument('--optimizer', default='adam', help='adam or momentum [default: adam]')
parser.add_argument('--decay_step', type=int, default=200000, help='Decay step for lr decay [default: 200000]')
parser.add_argument('--decay_rate', type=float, default=0.7, help='Decay rate for lr decay [default: 0.8]')
FLAGS = parser.parse_args()
```

- tensorboard --logdir log

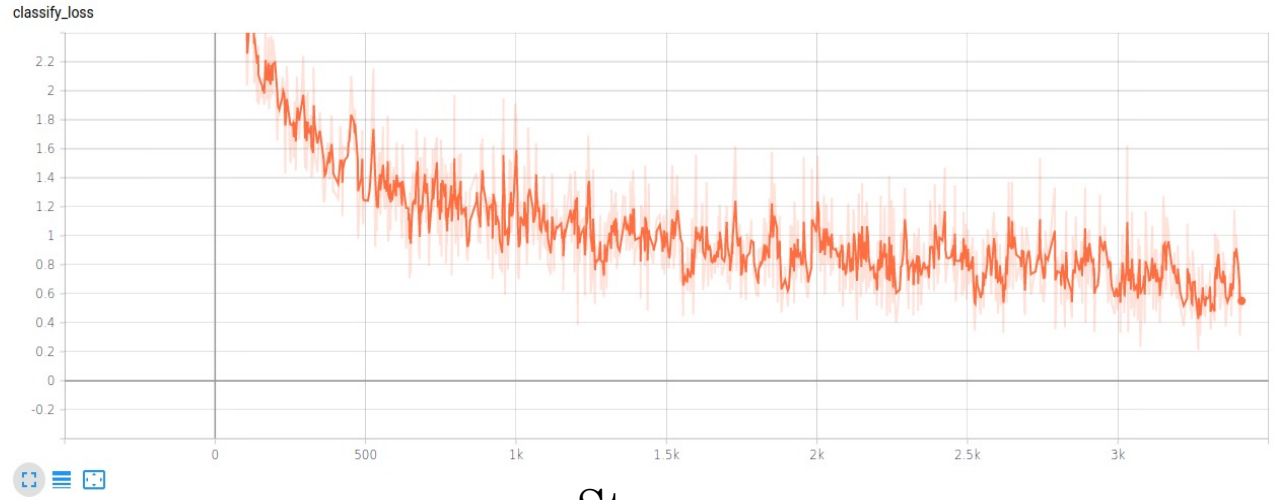
Convergence

Accuracy



Steps

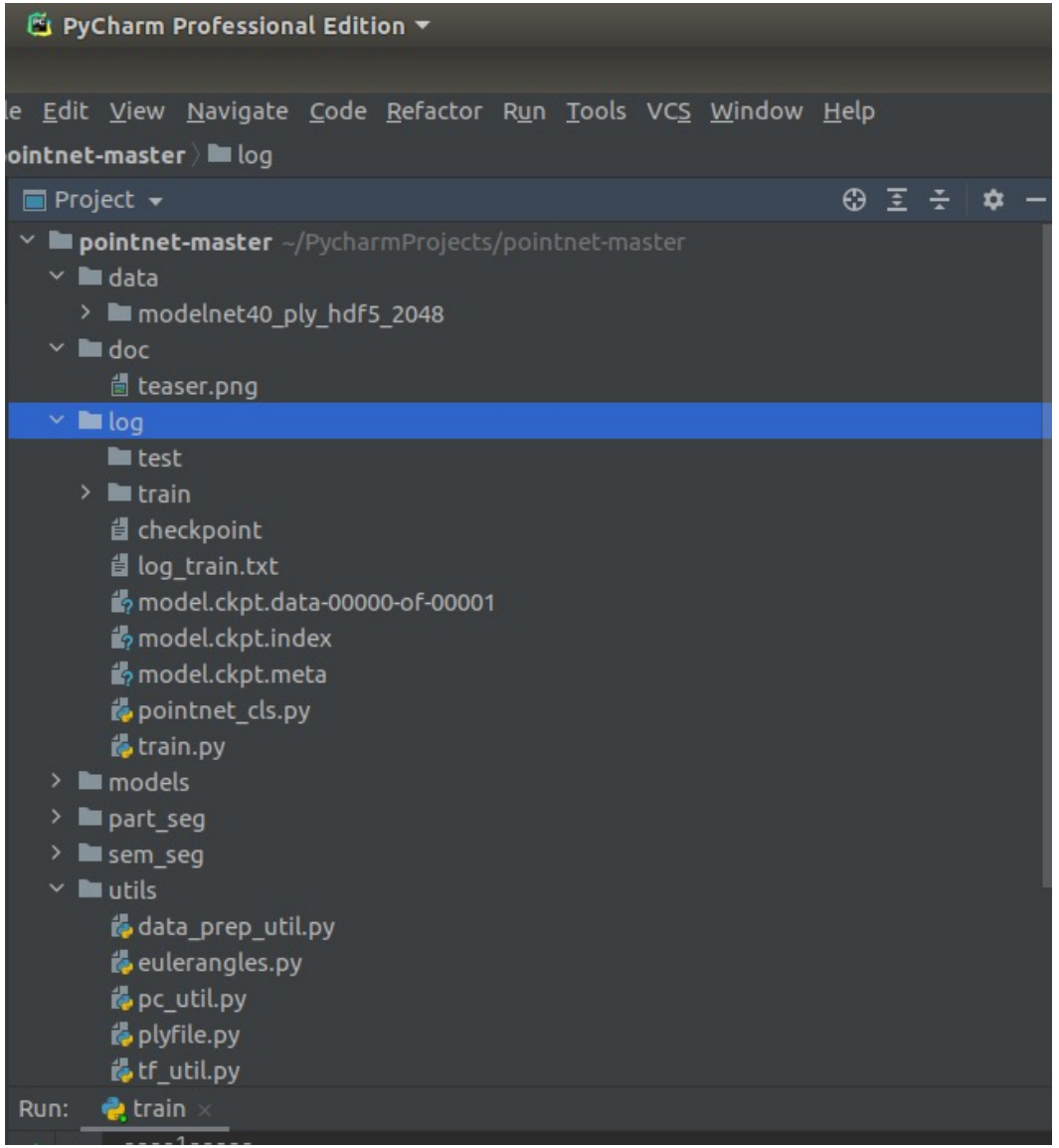
Loss



Steps

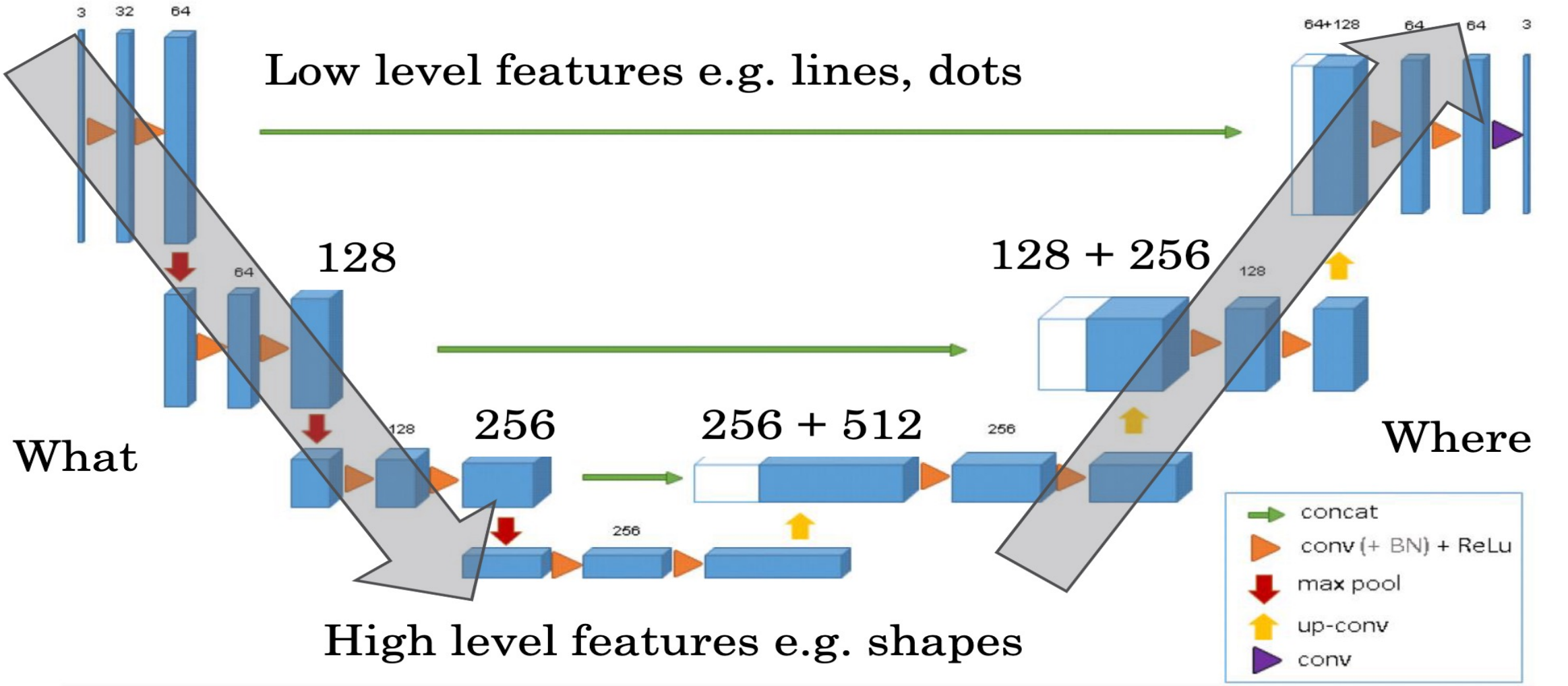
Saved Models

- Weights:
  model.ckpt.data-00000-of-00001
- Weight Index:
  - model.ckpt.index
- Metadata of tensors
  - Model.ckpt.meta



Evaluation Script

- python evaluate.py --visu

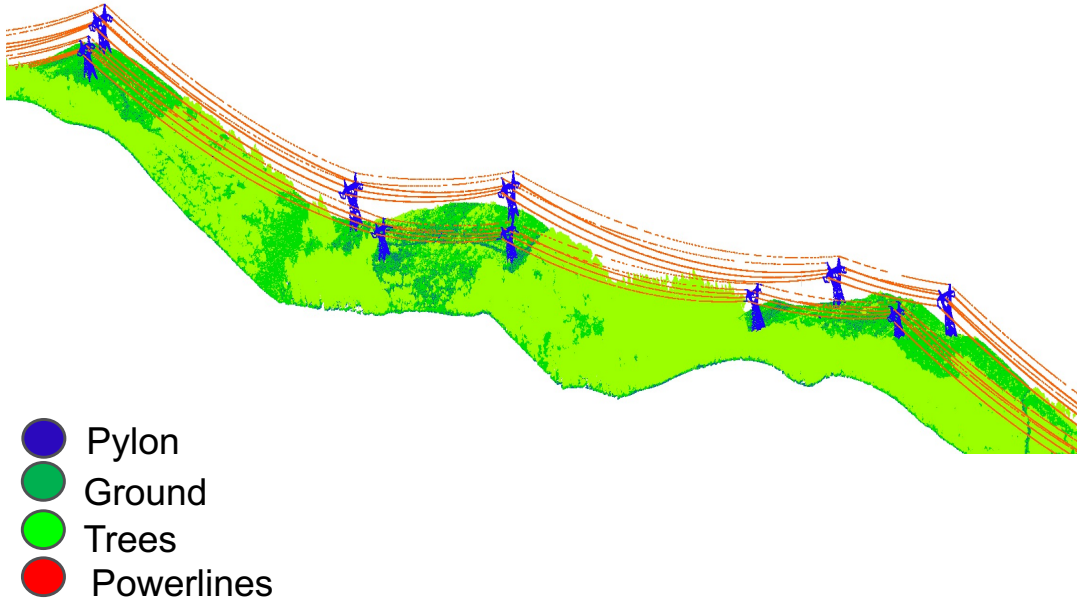


3D U-Net: Semantic Segmentation Demo

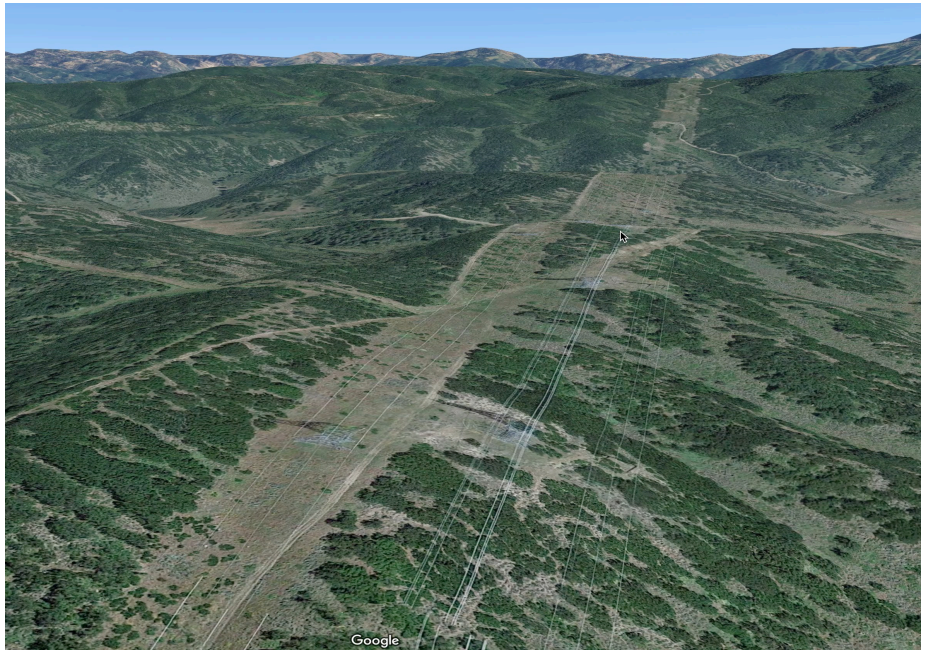
3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation [Çiçek., et al. 2016]

Dataset

- Coverage area: 66.7 $km^{2,}$
- Each scene area ≅ 2.9 km^2, total no. points ≅ 2m
- Sensor: Riegl Q560 laser scanner

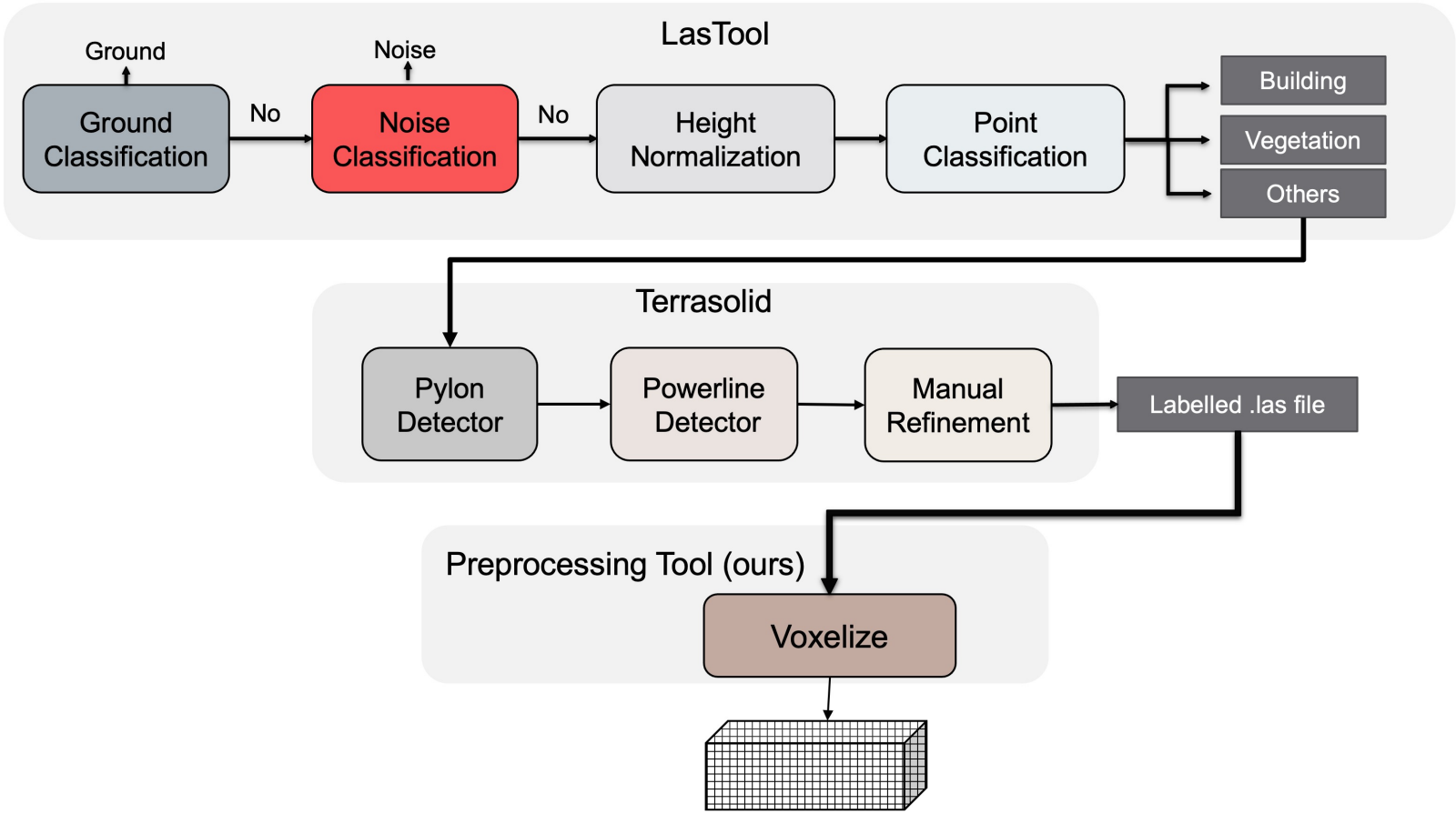


Labelled Point cloud



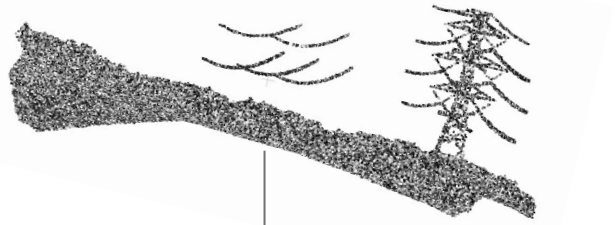
Steamboat Springs, Colorado, USA

Preprocessing Pipeline

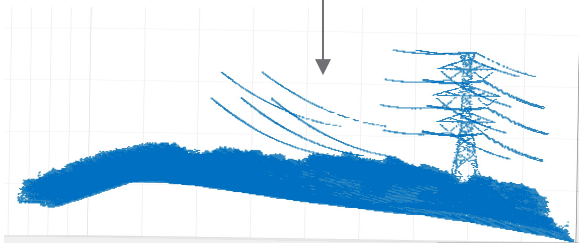


Preprocessing Tool (ours)

Input: Raw 3D point cloud

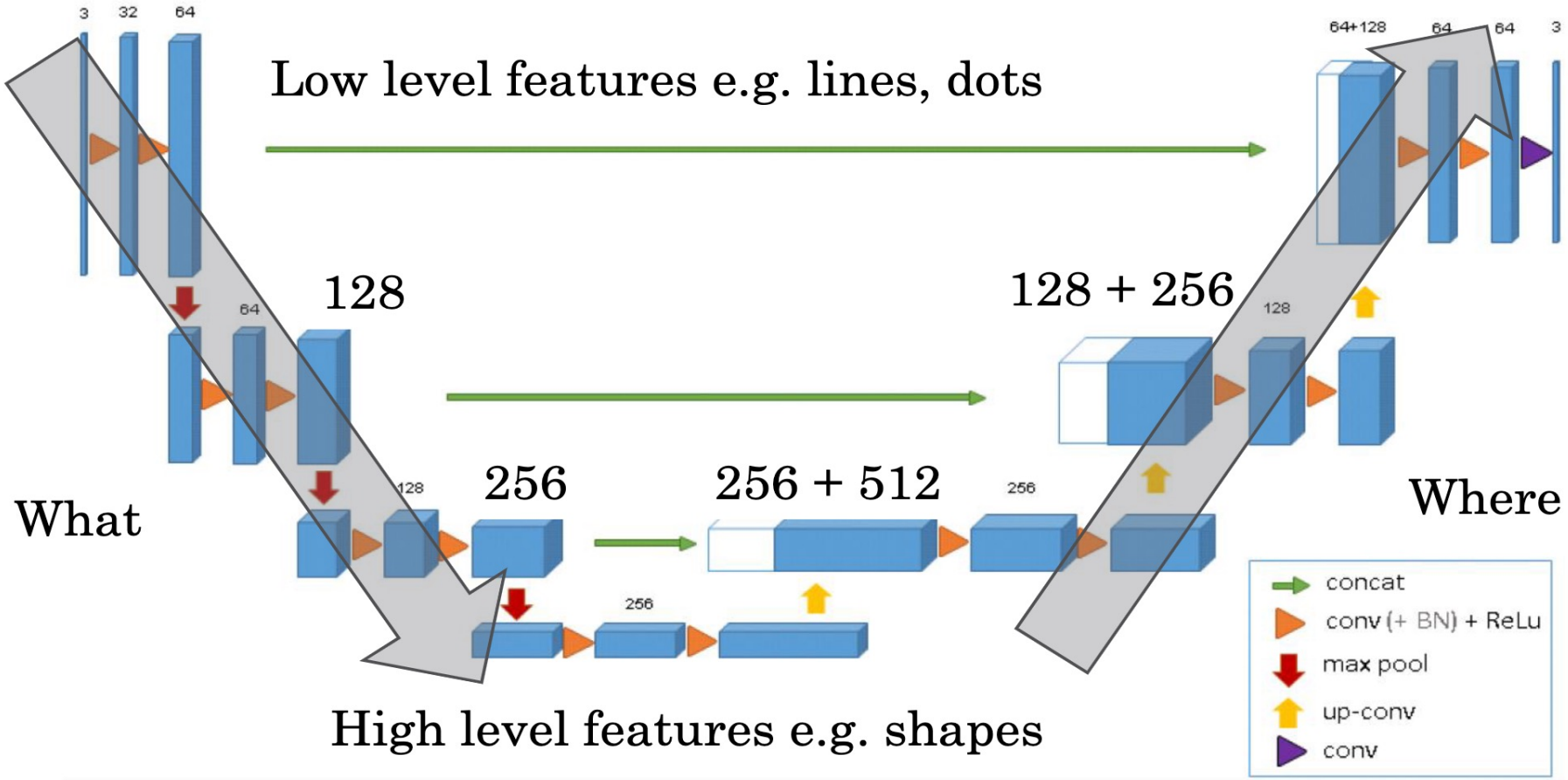


Rotate (align xy-plane) → Translate (shift origin) → Split Scenes → Voxelization ($1m^3$) → Normalization (Global or local)

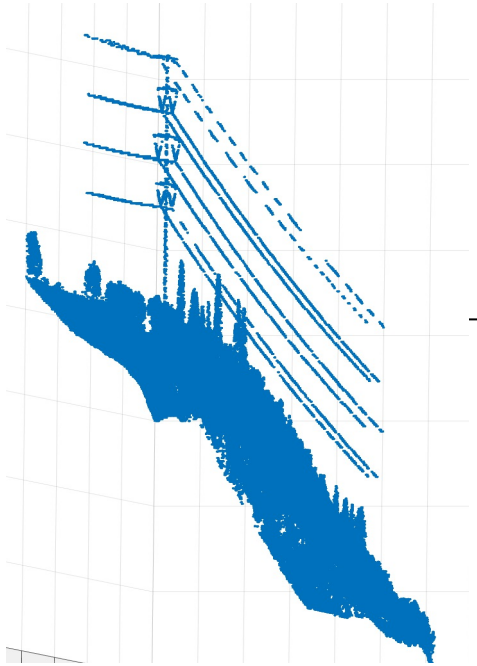


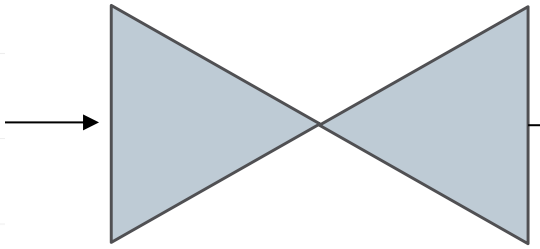
Output: Normalized voxel grid

Network

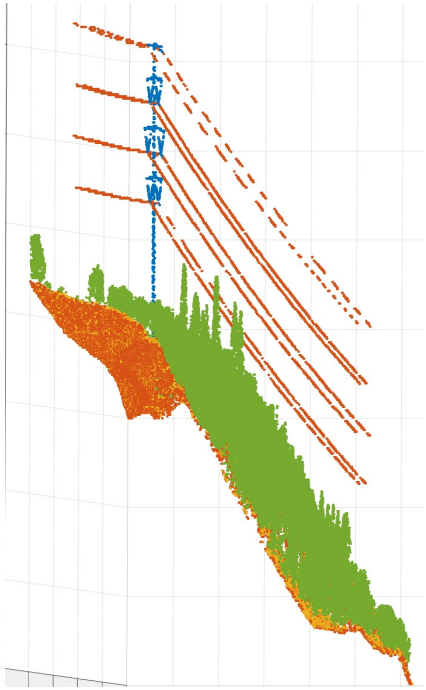

3
32
64
64+128
64
64
3
Low level features e.g. lines, dots
128
64
128 + 256
128
What
128
256
256 + 512
256
Where
256
High level features e.g. shapes
concat
conv (+ BN) + ReLu
max pool
up-conv
conv

Semantic Segmentation





3D U-Net



Input:
Occupancy Grid [0-1]

Output:
Multi Segmentation

Experimental Configuration

- Incremental K-Cross Validation Training
- Voxel size= $1m^3$,
- Input grid size = 32x32x32 and Batch size=4 grids
- 2 GPU RTX 6000
- Training Time: 48-60 hours
- Inference: 2-3 mins

Training

- python3 train.py
- Hyperparameter:

```
parser = argparse.ArgumentParser()
parser.add_argument('--gpu', type=int, default=0, help='GPU to use [default: GPU 0]')
parser.add_argument('--model', default='model', help='Model name [default: model]')
parser.add_argument('--category', default=None, help='Which single class to train on [default: None]')
parser.add_argument('--log_dir', default='log', help='Log dir [default: log]')
parser.add_argument('--XYZ', type=int, default=32, help='Point Number [default: 2048]')
parser.add_argument('--max_epoch', type=int, default=30, help='Epoch to run [default: 201]')
parser.add_argument('--batch_size', type=int, default=4, help='Batch Size during training [default: 32]')
parser.add_argument('--learning_rate', type=float, default=0.001, help='Initial learning rate [default: 0.001]')
parser.add_argument('--momentum', type=float, default=0.9, help='Initial learning rate [default: 0.9]')
parser.add_argument('--optimizer', default='adam', help='adam or momentum [default: adam]')
parser.add_argument('--decay_step', type=int, default=100000, help='Decay step for lr decay [default: 200000]')
parser.add_argument('--decay_rate', type=float, default= 0.8, help='Decay rate for lr decay [default: 0.7]')
parser.add_argument('--no_rotation', action='store_true', help='Disable random rotation during training.')

FLAGS = parser.parse_args()
```

Encoder

main.py | Unet3D_network.py

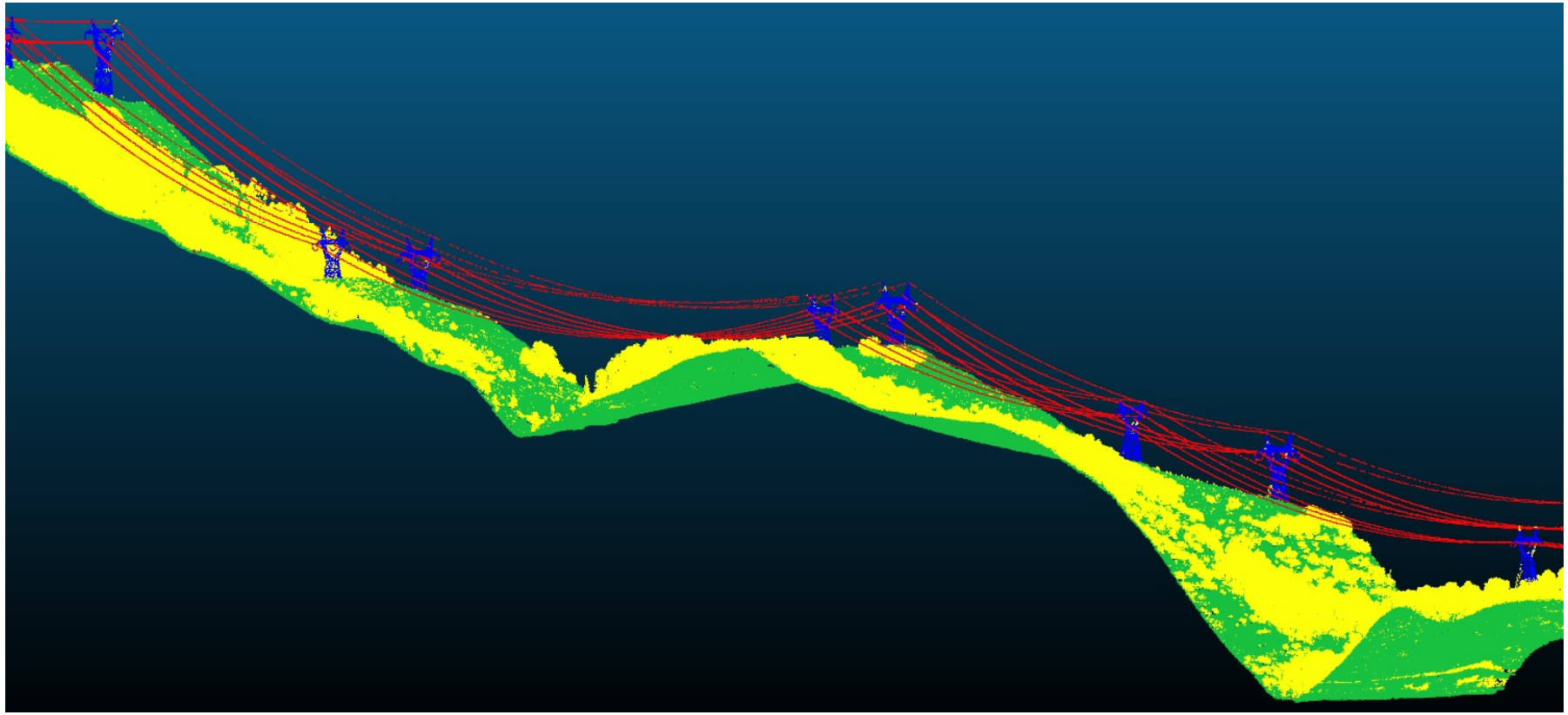
```
32      input_image = point_cloud
33
34      # Encoder
35      net = tf_util.conv3d(input_image, base_fil, [3, 3, 3],
36                           padding=pad, stride=[1, 1, 1],
37                           bn=True, is_training=is_training,
38                           scope='conv1', bn_decay=bn_decay)
39      net_f_1 = tf_util.conv3d(net, 2*base_fil, [3, 3, 3],
40                               padding=pad, stride=[1, 1, 1],
41                               bn=True, is_training=is_training,
42                               scope='conv2', bn_decay=bn_decay)
43      net = tf_util.max_pool3d(net_f_1, [2, 2, 2], stride=[2, 2, 2], scope="maxpool1", padding=pad)
44      net = tf_util.conv3d(net, 2*base_fil, [3, 3, 3],
45                           padding=pad, stride=[1, 1, 1],
46                           bn=True, is_training=is_training,
47                           scope='conv3', bn_decay=bn_decay)
48      net_f_2 = tf_util.conv3d(net, 4*base_fil, [3, 3, 3],
49                               padding=pad, stride=[1, 1, 1],
50                               bn=True, is_training=is_training,
51                               scope='conv4', bn_decay=bn_decay)
52      net_f_2 = tf.layers.dropout(net_f_2, rate=drop_r, training=is_training)
53      net = tf_util.max_pool3d(net_f_2, [2, 2, 2], stride=[2, 2, 2], scope="maxpool2", padding=pad)
54      net = tf_util.conv3d(net, 4*base_fil, [3, 3, 3], padding=pad, stride=[1, 1, 1],
55                           bn=True, is_training=is_training,
56                           scope='conv5', bn_decay=bn_decay)
57      net_f_3 = tf_util.conv3d(net, 8*base_fil, [3, 3, 3],
58                               padding=pad, stride=[1, 1, 1],
59                               bn=True, is_training=is_training,
60                               scope='conv6', bn_decay=bn_decay)
61      net_f_3 = tf.layers.dropout(net_f_3, rate=drop_r, training=is_training)
62      net = tf_util.max_pool3d(net_f_3, [2, 2, 2], stride=[2, 2, 2], scope="maxpool3", padding=pad)
63      net = tf_util.conv3d(net, 8*base_fil, [3, 3, 3],
64                           padding=pad, stride=[1, 1, 1],
65                           bn=True, is_training=is_training,
66                           scope='conv7', bn_decay=bn_decay)
```

Decoder

main.py | Unet3D_network.py

```
67      # Decoder
68      net = tf_util.conv3d_transpose(net, 16*base_fil, [2, 2, 2], padding=pad, stride=[2, 2, 2], bn=True,
69                                     is_training=is_training, scope='deconv1', bn_decay=bn_decay)
70      net_concat_3 = tf.concat([net_f_3, net], axis=-1)
71      net = tf_util.conv3d(net_concat_3, 8*base_fil, [3, 3, 3],
72                           padding=pad, stride=[1, 1, 1],
73                           bn=True, is_training=is_training,
74                           scope='conv8', bn_decay=bn_decay)
75      net = tf_util.conv3d(net, 8*base_fil, [3, 3, 3],
76                           padding=pad, stride=[1, 1, 1],
77                           bn=True, is_training=is_training,
78                           scope='conv9', bn_decay=bn_decay)
79      net = tf.layers.dropout(net, rate=drop_r, training=is_training)
80      net = tf_util.conv3d_transpose(net, 8*base_fil, [2, 2, 2],
81                           padding=pad, stride=[2, 2, 2],
82                           bn=True, is_training=is_training, scope='deconv2', bn_decay=bn_decay)
83      net_concat_2 = tf.concat([net_f_2, net], axis=-1)
84      net = tf_util.conv3d(net_concat_2, 4*base_fil, [3, 3, 3],
85                           padding=pad, stride=[1, 1, 1],
86                           bn=True, is_training=is_training,
87                           scope='conv10', bn_decay=bn_decay)
88      net = tf_util.conv3d(net, 4*base_fil, [3, 3, 3],
89                           padding=pad, stride=[1, 1, 1],
90                           bn=True, is_training=is_training,
91                           scope='conv11', bn_decay=bn_decay)
92      net = tf.layers.dropout(net, rate=drop_r, training=is_training)
93      net = tf_util.conv3d_transpose(net, 4*base_fil, [2, 2, 2],
94                           padding=pad, stride=[2, 2, 2],
95                           bn=True, is_training=is_training, scope='deconv3', bn_decay=bn_decay)
96      net_concat_1 = tf.concat([net_f_1, net], axis=-1)
97      net = tf_util.conv3d(net_concat_1, 2*base_fil, [3, 3, 3], padding=pad, stride=[1, 1, 1],
98                           bn=True, is_training=is_training,
99                           scope='conv12', bn_decay=bn_decay)
100     net = tf_util.conv3d(net, 2*base_fil, [3, 3, 3],
101                          padding=pad, stride=[1, 1, 1],
102                          bn=True, is_training=is_training,
103                          scope='conv13', bn_decay=bn_decay)
104     net = tf.layers.dropout(net, rate=drop_r, training=is_training)
105     net = tf_util.conv3d(net, 5, [1, 1, 1], padding=pad, stride=[1, 1, 1], bn=False, is_training=is_training,
106                          scope='conv14', bn_decay=bn_decay, activation_fn=None)
```

Results



Conclusion

- There are pros and cons of point and voxel-based methods. Selection of architecture is based on problem domain and dataset.
- Tensorflow is rich api for deep learning programming.
- Hyperpramter tuning can help achieve optimal results
- Data preprocessing pipeline and experimental configurations are key steps.

Acknowledgements

3D Mobile Mapping AI Project
(Teledyne Optech and NSERC)

Questions?