深蓝学院
shenlanxueyuan.com

三维点云处理
第五章作业讲评

主讲人 蔡毅

题目

深蓝学院
shenlanxueyuan.com

- 自己实现PointNet或者跑通PointNet源码
- 提交一个报告，内容包含training/testing loss/accuracy Curve

评分准则

- 优秀：自己实现或基于源码进行较大修改，Accuracy在84%以上。
- 良好：1.自己实现或修改源码，Accuracy在80%-84%之间。

  2.跑开源代码（没有修改），Accuracy在84%以上的。
- 及格：精度在80%以上的其他情况。
- 不及格： Accuracy < 80%。

```python
class ModelNetDataset():
    def __init__(self, root, batch_size=32, npoints=1024, split='train', normalize=True,
                 normal_channel=False, modelnet10=False, cache_size=15000, shuffle=None):
        self.root = root
        self.batch_size = batch_size
        self.npoints = npoints
        self.normalize = normalize

        self.catfile = os.path.join(self.root, 'modelnet40_shape_names.txt')
        self.cat = [line.rstrip() for line in open(self.catfile)]
        self.classes = dict(zip(self.cat, range(len(self.cat))))
        self.normal_channel = normal_channel

        shape_ids = {}
        shape_ids['train'] = [line.rstrip() for line in open(os.path.join(self.root, 'modelnet40_train.txt'))]
        shape_ids['test']= [line.rstrip() for line in open(os.path.join(self.root, 'modelnet40_test.txt'))]

        assert(split=='train' or split=='test')
        shape_names = ['_'.join(x.split('_')[0:-1]) for x in shape_ids[split]]
        # list of (shape_name, shape_txt_file_path) tuple
        self.datapath = [(shape_names[i], os.path.join(self.root, shape_names[i], shape_ids[split][i])+'.txt')
                         for i in range(len(shape_ids[split]))]
```

读 label名

将数据划分训练、测试集

```python
def _get_item(self, index):
    if index in self.cache:
        point_set, cls = self.cache[index]
    else:
        fn = self.datapath[index]
        cls = self.classes[self.datapath[index][0]]
        cls = np.array([cls]).astype(np.int32)
        point_set = np.loadtxt(fn[1], delimiter=',').astype(np.float32)
        # Take the first npoints
        point_set = point_set[0:self.npoints,:]
        if self.normalize:
            point_set[:,0:3] = pc_normalize(point_set[:,0:3])
        if not self.normal_channel:
            point_set = point_set[:,0:3]
        if len(self.cache) < self.cache_size:
            self.cache[index] = (point_set, cls)
    return point_set, cls
```

得到数据集和对应label

训练测试集准备

几种数据增广操作

```python
def _augment_batch_data(self, batch_data):
    if self.normal_channel:
        rotated_data = provider.rotate_point_cloud_with_normal(batch_data)
        rotated_data = provider.rotate_perturbation_point_cloud_with_normal(rotated_data)
    else:
        rotated_data = provider.rotate_point_cloud(batch_data)
        rotated_data = provider.rotate_perturbation_point_cloud(rotated_data)

    jittered_data = provider.random_scale_point_cloud(rotated_data[:, :, 0:3])
    jittered_data = provider.shift_point_cloud(jittered_data)
    jittered_data = provider.jitter_point_cloud(jittered_data)
    rotated_data[:, :, 0:3] = jittered_data
    return provider.shuffle_points(rotated_data)
```

以下是几种数据增广的实现

```python
def rotate_point_cloud(batch_data):
    """ Randomly rotate the point clouds to augument the dataset
        rotation is per shape based along up direction
        Input:
          BxNx3 array, original batch of point clouds
        Return:
          BxNx3 array, rotated batch of point clouds
    """
    rotated_data = np.zeros(batch_data.shape, dtype=np.float32)
    for k in range(batch_data.shape[0]):
        rotation_angle = np.random.uniform() * 2 * np.pi
        cosval = np.cos(rotation_angle)
        sinval = np.sin(rotation_angle)
        rotation_matrix = np.array([[cosval, 0, sinval],
                                    [0, 1, 0],
                                    [-sinval, 0, cosval]])
        shape_pc = batch_data[k, ...]
        rotated_data[k, ...] = np.dot(shape_pc.reshape((-1, 3)), rotation_matrix)
    return rotated_data
```

```python
def rotate_perturbation_point_cloud(batch_data, angle_sigma=0.06, angle_clip=0.18):
    """ Randomly perturb the point clouds by small rotations
        Input:
          BxNx3 array, original batch of point clouds
        Return:
          BxNx3 array, rotated batch of point clouds
    """
    rotated_data = np.zeros(batch_data.shape, dtype=np.float32)
    for k in range(batch_data.shape[0]):
        angles = np.clip(angle_sigma*np.random.randn(3), -angle_clip, angle_clip)
        Rx = np.array([[1,0,0],
                       [0,np.cos(angles[0]),-np.sin(angles[0])],
                       [0,np.sin(angles[0]),np.cos(angles[0])]])
        Ry = np.array([[np.cos(angles[1]),0,np.sin(angles[1])],
                       [0,1,0],
                       [-np.sin(angles[1]),0,np.cos(angles[1])]])
        Rz = np.array([[np.cos(angles[2]),-np.sin(angles[2]),0],
                       [np.sin(angles[2]),np.cos(angles[2]),0],
                       [0,0,1]])
        R = np.dot(Rz, np.dot(Ry,Rx))
        shape_pc = batch_data[k, ...]
        rotated_data[k, ...] = np.dot(shape_pc.reshape((-1, 3)), R)
    return rotated_data
```

```python
def random_scale_point_cloud(batch_data, scale_low=0.8, scale_high=1.25):
    """ Randomly scale the point cloud. Scale is per point cloud.
        Input:
            BxNx3 array, original batch of point clouds
        Return:
            BxNx3 array, scaled batch of point clouds
    """
    B, N, C = batch_data.shape
    scales = np.random.uniform(scale_low, scale_high, B)
    for batch_index in range(B):
        batch_data[batch_index,:,:] *= scales[batch_index]
    return batch_data
```

```python
def shift_point_cloud(batch_data, shift_range=0.1):
    """ Randomly shift point cloud. Shift is per point cloud.
        Input:
          BxNx3 array, original batch of point clouds
        Return:
          BxNx3 array, shifted batch of point clouds
    """
    B, N, C = batch_data.shape
    shifts = np.random.uniform(-shift_range, shift_range, (B,3))
    for batch_index in range(B):
        batch_data[batch_index,:,:] += shifts[batch_index,:]
    return batch_data
```

```python
def jitter_point_cloud(batch_data, sigma=0.01, clip=0.05):
    """ Randomly jitter points. jittering is per point.
        Input:
          BxNx3 array, original batch of point clouds
        Return:
          BxNx3 array, jittered batch of point clouds
    """
    B, N, C = batch_data.shape
    assert(clip > 0)
    jittered_data = np.clip(sigma * np.random.randn(B, N, C), -1*clip, clip)
    jittered_data += batch_data
    return jittered_data
```

以上是截取训练集和测试集划分的核心步骤，以及部分数据增广的操作

更多完整代码请参考PointNet++的数据预处理

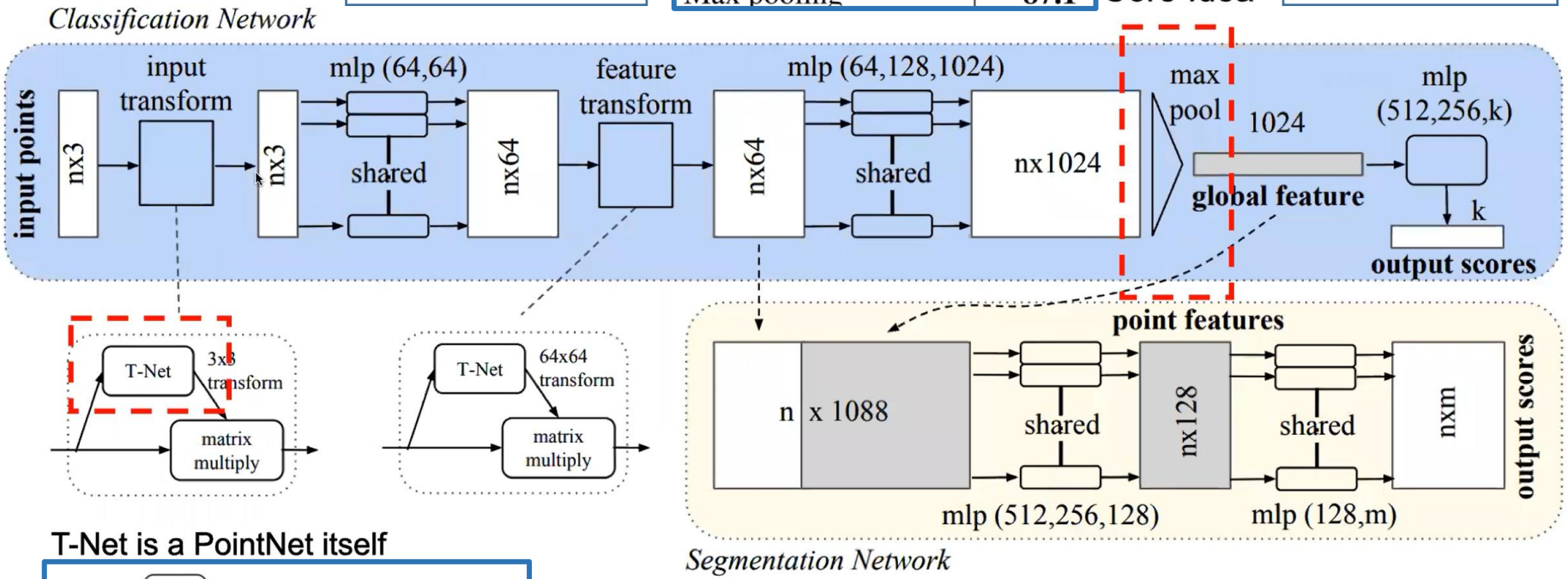
https://github.com/charlesq34/pointnet2

Attention sum	83.0
Average pooling	83.8
Max pooling	**87.1**

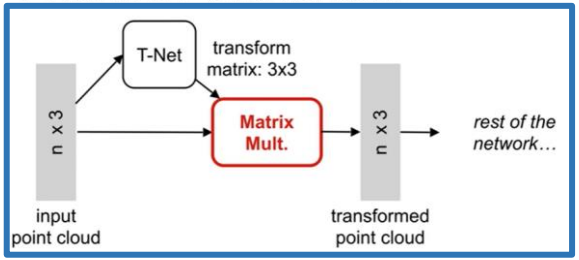
N*3 → N*1024

Core Idea

1*1024 →1*K



T-Net is a PointNet itself



PointNet

```
# Point functions (MLP implemented as conv2d)
net = tf_util.conv2d(input_image, 64, [1,3],
                     padding='VALID', stride=[1,1],
                     bn=True, is_training=is_training,
                     scope='conv1', bn_decay=bn_decay)
net = tf_util.conv2d(net, 64, [1,1],
                     padding='VALID', stride=[1,1],
                     bn=True, is_training=is_training,
                     scope='conv2', bn_decay=bn_decay)
net = tf_util.conv2d(net, 64, [1,1],
                     padding='VALID', stride=[1,1],
                     bn=True, is_training=is_training,
                     scope='conv3', bn_decay=bn_decay)
net = tf_util.conv2d(net, 128, [1,1],
                     padding='VALID', stride=[1,1],
                     bn=True, is_training=is_training,
                     scope='conv4', bn_decay=bn_decay)
net = tf_util.conv2d(net, 1024, [1,1],
                     padding='VALID', stride=[1,1],
                     bn=True, is_training=is_training,
                     scope='conv5', bn_decay=bn_decay)
```

MLP

Max pooling

```
# Symmetric function: max pooling
net = tf_util.max_pool2d(net, [num_point,1],
                         padding='VALID', scope='maxpool')
```

MLP

```
# MLP on global point cloud vector
net = tf.reshape(net, [batch_size, -1])
net = tf_util.fully_connected(net, 512, bn=True, is_training=is_training,
                              scope='fc1', bn_decay=bn_decay)
net = tf_util.fully_connected(net, 256, bn=True, is_training=is_training,
                              scope='fc2', bn_decay=bn_decay)
net = tf_util.dropout(net, keep_prob=0.7, is_training=is_training,
                      scope='dp1')
net = tf_util.fully_connected(net, 40, activation_fn=None, scope='fc3')
```

Loss

```
def get_loss(pred, label, end_points):
  """ pred: B*NUM_CLASSES,
    label: B, """
  loss = tf.nn.sparse_softmax_cross_entropy_with_logits(logits=pred, labels=label)
  classify_loss = tf.reduce_mean(loss)
  tf.summary.scalar('classify loss', classify_loss)
  return classify_loss
```

pointnet_cls_basic

```
with tf.variable_scope('transform_net1') as sc:
    transform = input_transform_net(point_cloud, is_training, bn_decay, K=3)
point_cloud_transformed = tf.matmul(point_cloud, transform)
input_image = tf.expand_dims(point_cloud_transformed, -1)
```

3*3 T-Net

```
with tf.variable_scope('transform_net2') as sc:
    transform = feature_transform_net(net, is_training, bn_decay, K=64)
end_points['transform'] = transform
net_transformed = tf.matmul(tf.squeeze(net, axis=[2]), transf
net_transformed = tf.expand_dims(net_transformed, [2])
```

64*64 T-Net

Loss

```
def get_loss(pred, label, end_points, reg_weight=0.001):
    """ pred: B*NUM_CLASSES,
        label: B, """
    loss = tf.nn.sparse_softmax_cross_entropy_with_logits(logits=pred, labels=label)
    classify_loss = tf.reduce_mean(loss)
    tf.summary.scalar('classify loss', classify_loss)

    # Enforce the transformation as orthogonal matrix
    transform = end_points['transform'] # BxKxK
    K = transform.get_shape()[1].value
    mat_diff = tf.matmul(transform, tf.transpose(transform, perm=[0,2,1]))
    mat_diff -= tf.constant(np.eye(K), dtype=tf.float32)
    mat_diff_loss = tf.nn.l2_loss(mat_diff)
    tf.summary.scalar('mat loss', mat_diff_loss)

    return classify_loss + mat_diff_loss * reg_weight
```

T-Net

```
def input_transform_net(point_cloud, is_training, bn_decay=None, K=3):
    input_image = tf.expand_dims(point_cloud, -1)
    net = tf_util.conv2d(input_image, 64, [1,3],
                         padding='VALID', stride=[1,1],
                         bn=True, is_training=is_training,
                         scope='tconv1', bn_decay=bn_decay)
    net = tf_util.conv2d(net, 128, [1,1],
                         padding='VALID', stride=[1,1],
                         bn=True, is_training=is_training,
                         scope='tconv2', bn_decay=bn_decay)
    net = tf_util.conv2d(net, 1024, [1,1],
                         padding='VALID', stride=[1,1],
                         bn=True, is_training=is_training,
                         scope='tconv3', bn_decay=bn_decay)
    net = tf_util.max_pool2d(net, [num_point,1],
                             padding='VALID', scope='tmaxpool')

    net = tf.reshape(net, [batch_size, -1])
    net = tf_util.fully_connected(net, 512, bn=True, is_training=is_training,
                                  scope='tfc1', bn_decay=bn_decay)
    net = tf_util.fully_connected(net, 256, bn=True, is_training=is_training,
                                  scope='tfc2', bn_decay=bn_decay)
```

pointnet_cls

优秀作业

融合 pointnet和 bps[1] 模型，想法是:
pointnet 的全局特征是由每个 point 的特征通过 max-pooling 得到的，point 和 point之间没有encoder到其它信息，而bps的全局特征则刚好相反，它encoder都是所有point的特征，但单独每个 point 的特征提取不够充分; 我主要是想看下它们的全局特征是否起到了互补作用。

model	test accuracy
PointNet (Official)	89.2%
Ours PointNet	89.2%
Ours PointNetFuseBps	88.6%

Reference:Efficient Learning on Point Clouds with Basis Point Sets

部分私信

- 有同学私信我说，时间都花在搭GPU环境

  折腾GPU环境可以说是新手入门的必经之路，多花点时间搜索教程，简单的问题基本上都有现成教程。

- 希望多提供一些深度学习的内容

  这个课程毕竟是点云处理为体系的课程，深度学习只是其中一部分，所以有关深度学习的更系统详细的内容可能需要大家额外努力了。

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening