MSN-Code

https://github.com/Colin97/MSN-Point-Cloud-Completion

val.py

```
with torch.no_grad(): #验证关闭梯度计算
    partial # 10 5000 3 10个网络补全数据
    gt      # 10 5000 3 10个原始完整数据

    output1 ,output2, expansion_penalty  =
network(partial.transpose(2,1).contiguous())
    # jump to model.py class MSN [10,8192,3] [10,8192,3] 1
    dist, _ = EMD(output1, gt, 0.002, 10000)
    emd1 = torch.sqrt(dist).mean()
    dist, _ = EMD(output2, gt, 0.002, 10000)
    emd2 = torch.sqrt(dist).mean()
    return emd1, emd2, expansion_penalty
```

model.py

```
class MSN(nn.Module):
    def __init__(self, num_points = 8192, bottleneck_size = 1024, n_primitives =
16):
        super(MSN, self).__init__()
        self.num_points = num_points
        self.bottleneck_size = bottleneck_size
        self.n_primitives = n_primitives
        self.encoder = nn.Sequential(                #编码器
        PointNetfeat(num_points, global_feat=True),
        nn.Linear(1024, self.bottleneck_size),
        nn.BatchNorm1d(self.bottleneck_size),
        nn.ReLU()
        )
        self.decoder = nn.ModuleList([PointGenCon(bottleneck_size = 2
+self.bottleneck_size) for i in range(0,self.n_primitives)])
        self.res = PointNetRes()
        self.expansion = expansion.expansionPenaltyModule()

    def forward(self, x):
        partial = x         # [10 5000 3]
        x = self.encoder(x) # [10 1024]
        outs = []
        for i in range(0,self.n_primitives):
            rand_grid =
Variable(torch.cuda.FloatTensor(x.size(0),2,self.num_points//self.n_primitives))
#[10 2 512]
            rand_grid.data.uniform_(0,1) # 生成0 1 之间的实数
```

```
            y = x.unsqueeze(2).expand(x.size(0),x.size(1),
rand_grid.size(2)).contiguous() #对x升维度[10 1024 512]
            y = torch.cat( (rand_grid, y), 1).contiguous() # 拼接y [10 1026 512]
            outs.append(self.decoder[i](y)) # 00-15  [10,3,512]

        outs = torch.cat(outs,2).contiguous()    # [10,3,8192]
        out1 = outs.transpose(1, 2).contiguous() # [10,8192,3]

        dist, _, mean_mst_dis = self.expansion(out1,
self.num_points//self.n_primitives, 1.5)# dist         [10,8192] 路径
mean_mst_dis [10]      平均最小生成树路径
        loss_mst = torch.mean(dist)# 平均loss

        id0 = torch.zeros(outs.shape[0], 1, outs.shape[2]).cuda().contiguous()#
[10,1,8192]
        outs = torch.cat( (outs, id0), 1) #[10,1,8192]+[10,3,8192]=[10,4,8192]
        id1 = torch.ones(partial.shape[0], 1,
partial.shape[2]).cuda().contiguous()#[10,1,5000]
        partial = torch.cat( (partial, id1), 1)#[10,1,5000]+[10,3,5000]=
[10,4,5000]
        xx = torch.cat( (outs, partial), 2)#[10,4,8192]+[10,4,5000]=
[10,4,8192+5000]

        # [10,3,8192] 8192 10
        resampled_idx = MDS_module.minimum_density_sample(xx[:, 0:3,
:].transpose(1, 2).contiguous(), out1.shape[1], mean_mst_dis)  # [10,8192]
        xx = MDS_module.gather_operation(xx, resampled_idx) # [10,4,8192]
        delta = self.res(xx) # xx[10,4,8192] delta=[10,3,8192]
        xx = xx[:, 0:3, :]   # [10,3,8192]
        out2 = (xx + delta).transpose(2,1).contiguous()  # [10,8192,3]
        return out1, out2, loss_mst
```

架构

encoder编码

```
self.encoder = nn.Sequential(                #编码器
        PointNetfeat(num_points, global_feat=True),
        nn.Linear(1024, self.bottleneck_size),
        nn.BatchNorm1d(self.bottleneck_size),
        nn.ReLU()
        )

input  [3,5000] 3个通道数 5000个特征
output [1024]   1024维瓶颈向量
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
        pointnetfeat                  [-1, 1024]
            Linear-7                  [-1, 1024]       1,049,600
       BatchNorm1d-8                  [-1, 1024]           2,048
              ReLU-9                  [-1, 1024]               0
```

class PointNetFeat()

```
input  [3,5000] 3个通道数 5000个特征
output [1024]   1024维瓶颈向量
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv1d-1              [-1, 64, 5000]             256
       BatchNorm1d-2              [-1, 64, 5000]             128
              relu                [-1, 64, 5000]
            Conv1d-3             [-1, 128, 5000]           8,320
       BatchNorm1d-4             [-1, 128, 5000]             256
              relu               [-1, 128, 5000]
            Conv1d-5            [-1, 1024, 5000]         132,096
       BatchNorm1d-6            [-1, 1024, 5000]           2,048
               max                [-1, 1024, 1]
              view                   [-1, 1024]
```

---

decoder解码-16个表面元素形成复杂的形状

```
    self.decoder = nn.ModuleList([PointGenCon(bottleneck_size = 2
+self.bottleneck_size) for i in range(0,self.n_primitives)])

    x #[10 1024]
    for i in range(0,self.n_primitives): # 16次
            rand_grid =
Variable(torch.cuda.FloatTensor(x.size(0),2,self.num_points//self.n_primitives))
# 随机网格[10 2 512]
            rand_grid.data.uniform_(0,1) # 网格内生成0 1之间的实数
            y = x.unsqueeze(2).expand(x.size(0),x.size(1),
rand_grid.size(2)).contiguous()          # 对x升维度[10 1024 512]
            y = torch.cat( (rand_grid, y), 1).contiguous() # 拼接y和随机网格[10
1026 512]
            outs.append(self.decoder[i](y)) # 00-15  [10,3,512]
```

class PointGenCon()

```
input  [10,1026,512] 1026个通道数 512个点
output [10,3,512]    3个通道数 512个点
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
           Conv1d-10            [-1, 1026, 512]       1,053,702
      BatchNorm1d-11            [-1, 1026, 512]           2,052
              relu
           Conv1d-12             [-1, 513, 512]         526,851
      BatchNorm1d-13             [-1, 513, 512]           1,026
              relu
           Conv1d-14             [-1, 256, 512]         131,584
      BatchNorm1d-15             [-1, 256, 512]             512
```

```
                relu
            Conv1d-16               [-1, 3, 512]             771
              Tanh-17               [-1, 3, 512]               0
```

扩展惩罚

```
self.expansion = expansion.expansionPenaltyModule()

dist, _, mean_mst_dis = self.expansion(out1, self.num_points//self.n_primitives,
1.5)
# dist         [10,8192] 路径
# mean_mst_dis [10]      平均mst路径
```

class expansionPenaltyFunction()

```
class expansionPenaltyFunction(Function):
    @staticmethod
    def forward(ctx, xyz, primitive_size, alpha): # xyz[10,8192,3] 512 1.5
        assert(primitive_size <= 512)
        batchsize, n, _ = xyz.size()
        assert(n % primitive_size == 0)
        xyz = xyz.contiguous().float().cuda()
        dist = torch.zeros(batchsize, n, device='cuda').contiguous() #[10,8192]
        assignment = torch.zeros(batchsize, n, device='cuda',
dtype=torch.int32).contiguous() - 1 # [10,8192] 值为-1
        neighbor = torch.zeros(batchsize, n * 512,  device='cuda',
dtype=torch.int32).contiguous() # [10,8192*512]
        cost = torch.zeros(batchsize, n * 512, device='cuda').contiguous()#
[10,8192*512]
        mean_mst_length = torch.zeros(batchsize, device='cuda').contiguous() #
[10]
        expansion_penalty.forward(xyz, primitive_size, assignment, dist, alpha,
neighbor, cost, mean_mst_length)
        ctx.save_for_backward(xyz, assignment)
        return dist, assignment, mean_mst_length / (n / primitive_size)

    @staticmethod
    def backward(ctx, grad_dist, grad_idx, grad_mml):
        xyz, assignment = ctx.saved_tensors
        grad_dist = grad_dist.contiguous()
        grad_xyz = torch.zeros(xyz.size(), device='cuda').contiguous()
        expansion_penalty.backward(xyz, grad_xyz, grad_dist, assignment)
        return grad_xyz, None, None
```

最小密度采样

```
resampled_idx = MDS_module.minimum_density_sample(xx[:, 0:3, :].transpose(1,
2).contiguous(), out1.shape[1], mean_mst_dis) # [10,8192]
```

class MinimumDensitySampling()

```
class MinimumDensitySampling(Function):
    @staticmethod
    def forward(ctx, xyz, npoint, mean_mst_length):
        # xyz [10,13192,3] npoint 8192 mean_mst_length 10
        # 使用迭代半径点采样来选择一组具有最大最小距离的npoint要素
        # npoint : 采样集中的要素数量
        # mean_mst_length : 扩展惩罚模块的平均边缘长度

        idx = torch.zeros(xyz.shape[0], npoint, requires_grad= False,
device='cuda', dtype=torch.int32).contiguous()
        MDS.minimum_density_sampling(xyz, npoint, mean_mst_length, idx)
        return idx

    @staticmethod
    def backward(grad_idx, a=None):
        return None, None, None
```

---

res残差

```
self.res = PointNetRes()
delta = self.res(xx) #xx[10,4,8192] delta=[10,3,8192]
```

class PointNetRes()

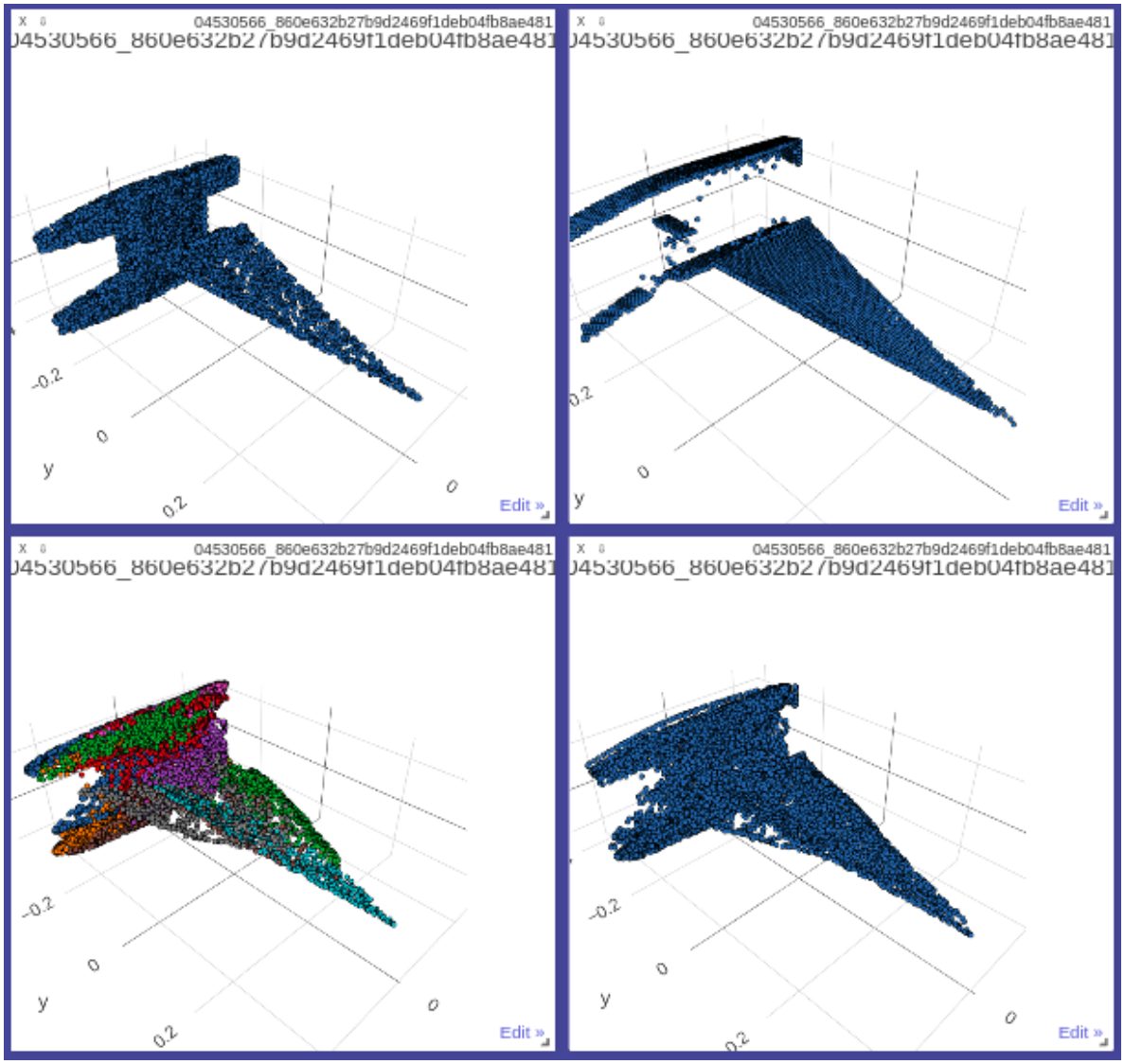
```
input  [10,4,8192]  4个通道数 8192个点
output [10,3,8192]  3个通道数 8192个点
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
          Conv1d-139             [-1, 64, 8192]             320
     BatchNorm1d-140             [-1, 64, 8192]             128
                relu                                          # >>pointfeat [-1,
64, 8192]
          Conv1d-141            [-1, 128, 8192]           8,320
     BatchNorm1d-142            [-1, 128, 8192]             256
                relu
          Conv1d-143           [-1, 1024, 8192]         132,096
     BatchNorm1d-144           [-1, 1024, 8192]           2,048
                max                [-1, 1024]
    repeat(1,1,8192)           [-1, 1024, 8192]                     # >>x [-1,
1024, 8192]
    cat(x,pointfeat)           [-1, 1088, 8192]
          Conv1d-145            [-1, 512, 8192]         557,568
     BatchNorm1d-146            [-1, 512, 8192]           1,024
                relu
```

```
          Conv1d-147            [-1, 256, 8192]         131,328
     BatchNorm1d-148            [-1, 256, 8192]             512
               relu
          Conv1d-149            [-1, 128, 8192]          32,896
     BatchNorm1d-150            [-1, 128, 8192]             256
               relu
          Conv1d-151              [-1, 3, 8192]             387
            Tanh-152              [-1, 3, 8192]               0
```

summary

```
input  [10,3,5000] 3个通道数 5000个特征
output []
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
        PointGenFeat
            Linear-7                 [-1, 1024]       1,049,600
       BatchNorm1d-8                 [-1, 1024]           2,048
              ReLU-9                 [-1, 1024]               0
      16*PointGenCon
expansionPenaltyModule-138           [-1, 8192]               0
         PointNetRes
================================================================
Total params: 29,525,859
Trainable params: 29,525,859
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.057220
Forward/backward pass size (MB): 581.984375
Params size (MB): 112.632214
Estimated Total Size (MB): 694.673809
----------------------------------------------------------------
```

Demo


04530566_860e632b27b9d2469f1deb04fb8ae481
Edit »