周报8

上周在读PSPNet论文的中途去对ResNet进行了细研究，理解了背后的数学本质原理：残差网络之所以更容易训练，得益于学习残差能够缓解了梯度弥散。本周我们继续研究PSPNet，从数学原理上理解该网络为何更加优秀。

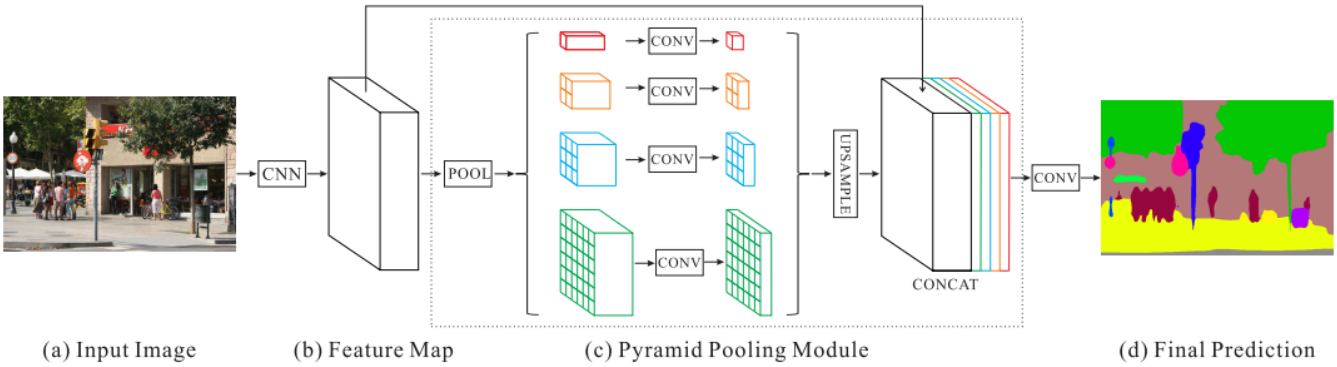
Pyramid Scene Parsing Network

PSPNet之所以在语义分割任务中更加出色，是因为作者考虑了**语境关系和不同感受野的全局文本信息。**

Pyramid Pooling Module 金字塔池化模块

感受野的大小决定了全局信息量，因此作者从**感受野**（**receptive field**）出发，发现网络中更深的层的感受野是比理论上要小很多，这就导致了许多网络无法充分地包含重要的global scenery。因此作者为了提高全局信息能力，搭建了该结构。

接下来我配合实现的代码来理解该网络结构，并与FCN结构进行对比，从而搞清楚为什么该网络结构在语义分割任务中更优秀。



对于这样一个网络主结构，从全局的视角来看，它分为两部分：一部分是由ResNet作为主干提取网络，得到上图中的（b）Feature Map，而另一部分是根据不同的池化操作，将输入进来的Feature Map进行划分为4类：

- 上图中红色部分为将输入进来的特征层整个进行平均池化操作
- 上图中黄色部分是将特征层划为 2 x 2 个子区域，然后在每一个子区域中进行平均池化
- 蓝色是划分为 3 x 3，然后平均池化
- 绿色是划分为 6 x 6，然后平均池化

```
def __init__(self, in_channel, each_out_channel, num_bins):
    super(PPM, self).__init__()
    self.features = []
    for n in num_bins:
```

```
        self.features.append(nn.Sequential(
            nn.AdaptiveAvgPool2d(output_size=n),
            nn.Conv2d(in_channel, each_out_channel, kernel_size=1, bias=False),
            nn.BatchNorm2d(each_out_channel),
            nn.ReLU(inplace=True)
        ))
    self.features = nn.ModuleList(self.features)
```

上述代码中可以看到在PPM结构中，我们根据传入进来的num_bins（作者这里对应的是[1, 2, 3, 6]），也就是每一个金字塔层所划分的子区域大小，对其进行池化，卷积，标准化，激活等操作。而对于每一个子区域进行平均池化，能够实现聚合不同区域的上下文信息，从而提高获取全局信息的能力。

根据上图，我们可以看见整个金字塔池化模型**将主特征提取到的Feature Map和我们根据不同bins大小的池化卷积后的特征层进行堆叠（连接concat）**。因此作者使用**双线性插值**的方法，将池化后的特征层上采样为feature map的大小进行拼接。

```
def forward(self, x):
    out = [x]
    for feature in self.features:
        feat_x = feature(x)
        feat_x_ = F.interpolate(feat_x, [x.shape[2], x.shape[3]], mode = 'bilinear', align_
corners = True)
        out.append(feat_x_)
    return torch.cat(out, 1)
```

上述代码中，传入的 x 是feature map，针对每一个金字塔池化层，我们都对其进行 feature(x) 操作，然后进行 F.interpolate 双线性插值进行上采样，最后利用 cat 方法在维度1上进行连接。

```
def forward(self, x, y=None, preact=False):
        B, C, H, W = x.shape
        H = int((H - 1) / 8 * self.zoom_factor + 1)
        W = int((W - 1) / 8 * self.zoom_factor + 1)

        # 对于ResNet的4个stage
        x0 = self.layer0(x)
        x1 = self.layer1(x0)
        x2 = self.layer2(x1)
        x3 = self.layer3(x2)
        x = self.layer4(x3)

        if self.use_ppm or self.use_aspp:
            x = self.enrich_module(x)
        x = self.cls(x)
        if self.zoom_factor != 1:
            x = F.interpolate(x, size=(H, W), mode='bilinear', align_corners=True)
```

```
18
19          if self.training and y is not None:
20              ### [requires implementation] calculate losses for training
21              aux = self.aux(x3)
22              if self.zoom_factor != 1:
23                  aux = F.interpolate(aux, size=(H, W), mode='bilinear', align_corners=True)
24              main_loss = self.criterion(x, y)
25              # 这里添加了辅助学习
26              aux_loss = self.criterion(aux, y)
27              return x, main_loss, aux_loss
28          else:
29              return x
```

主体的PSPNet我也搭建好了，上述代码中只展示forward函数，根据论文中的描述，我们要对图像进行缩放，因此我们根据传入的缩放系数 zoom_factor计算出相应的图像的高和宽。

然后传入已经预训练好的ResNet50模型的4个stage对其进行特征提取，其中代码的第14行是对应论文中最后的conv。

如果我们的缩放系数不等于1的话，我们要复原出图像，因此这里还需要根据缩放后的H和W进行上采样。

```
1 self.aux = nn.Sequential(
2     nn.Conv2d(1024, 256, kernel_size=3, padding=1, bias=False),
3     nn.BatchNorm2d(256),
4     nn.ReLU(inplace=True),
5     nn.Dropout2d(p=dropout),
6     nn.Conv2d(256, classes, kernel_size=1)
7 )
```

最后这里，我人为添加了辅助学习loss，理论上讲效果会更加优秀。到这里，总算把PSPNet的结构和代码吃透了，同时也学会了很多额外知识，例如做上采样可以用插值的方法。这周也把ResNet代码吃透了这里就不涉及了。这周很大的一个感受就是，一定要先明白理论，再上手代码。

个人想法

这种金字塔池化结构从感觉上确实能提高获取全局信息的能力，但是很多点不知道从数学上该如何去解释，例如为何最后将上采样后的feature map堆叠起来效果就会更好？如果按这样的思路将feature map划分为不同大小后进行平均池化，那么是不是我划分的越细，金字塔层数越多，最后的效果会更卓越？这类金字塔式结构这些都是可以思考的地方和后期替换掉GG-CNN网络后可以进行修改调参的点。