问题

我的项目中涉及到这两个网络，而网络的最重要的部分除了巧妙的结构设计外，当属损失函数的设计了，所以很多时候面试官都会问我，损失函数使用的是啥，我想他们期待我给的答案是自己设计的损失函数吧，但是做项目的时候我那么菜，根本没有能力去设计好吧（说得好像现在有似的.....），所以都是使用人家原有的损失函数，并没做修改。

时间久了，这些东西当再次被面试官问起的时候，希望我至少能把人家论文中的损失函数形式讲清楚吧。

简单介绍下这两个网络

目标检测的框架中包含4个关键模块，包括region proposal（生成ROI）、feature extraction（特征提取网络）、classification（ROI分类）、regression（ROI回归）。而faster-rcnn利用一个神经网络将这4个模块结合起来，训练了一个端到端的网络。

1. **特征提取网络**：它用来从大量的图片中提取出一些不同目标的重要特征，通常由conv+relu+pool层构成，常用一些预训练好的网络（VGG、Inception、Resnet等），获得的结果叫做特征图；
2. **生成ROI**：在获得的特征图的每一个点上做多个候选ROI（这里是9），然后利用分类器将这些ROI区分为背景和前景，同时利用回归器对这些ROI的位置进行初步的调整；
3. **ROI分类**：在RPN阶段，用来区分前景（于真实目标重叠并且其重叠区域大于0.5）和背景（不与任何目标重叠或者其重叠区域小于0.1）；在Fast-rcnn阶段，用于区分不同种类的目标（猫、狗、人等）；
4. **ROI回归**：在RPN阶段，进行初步调整；在Fast-rcnn阶段进行精确调整；

总之，其整体流程如下所示：

```
首先对输入的图片进行裁剪操作，并将裁剪后的图片送入预训练好的分类网络中获取该图像对应的特征图；
然后在特征图上的每一个锚点上取9个候选的ROI（3个不同尺度，3个不同长宽比），并根据相应的比例将其映射到原始图像中（因为特征提取网络一般有conv和pool组成，但是只有pool会改变特征图的大小，因此最终的特征图大小和pool的个数相关）；
接着将这些候选的ROI输入到RPN网络中，RPN网络对这些ROI进行分类（即确定这些ROI是前景还是背景）同时对其进行初步回归（即计算这些前景ROI与真实目标之间的BB的偏差值，包括Δx、Δy、Δw、Δh）,然后做NMS（非极大值抑制，即根据分类的得分对这些ROI进行排序，然后选择其中的前N个ROI）；
接着对这些不同大小的ROI进行ROI Pooling操作（即将其映射为特定大小的feature_map,文中是7x7），输出固定大小的feature_map；
最后将其输入简单的检测网络中，然后利用1x1的卷积进行分类（区分不同的类别，N+1类，多余的一类是背景，用于删除不准确的ROI），同时进行BB回归（精确的调整预测的ROI和GT的ROI之间的偏差值），从而输出一个BB集合。
```

整个**Mask R-CNN**算法的思路很简单，就是在原始Faster-rcnn算法的基础上面增加了FCN来产生对应的MASK分支。即Faster-rcnn + FCN，更细致的是 RPN + ROIAlign + Fast-rcnn + FCN。

FCN算法是一个经典的语义分割算法，可以对图片中的目标进行准确的分割。它是一个端到端的网络，主要的模快包括卷积和反卷积，即先对图像进行卷积和池化，使其feature map的大小不断减小；然后进行反卷积操作，即进行插值操作，不断的增大其feature map，最后对每一个像素值进行分类。从而实现对输入图像的准确分割。

损失函数

Faster R-CNN 是目标检测网络，所以其损失函数由两部分组成：$L = L_{cls} + L_{box}$

Mask R-CNN 是实例分割网络，所以其损失函数由三部分组成：$L = L_{cls} + L_{box} + L_{mask}$

其中 Mask R-CNN 损失函数中的前两项与 Faster R-CNN 中的是一样的。

mask损失函数L_{mask}

对于每一个ROI，mask分支有 $K * m * m$ 维度的输出，其对K个大小为m*m的mask进行编码，每一个ROI有K个类别。我们使用了per-pixel sigmoid，并且将Lmask定义为the average binary cross-entropy loss 。对应一个属于GT中的第k类的ROI，Lmask仅仅在第k个mask上面有定义（其它的k-1个mask输出对整个Loss没有贡献）。我们定义的Lmask允许网络为每一类生成一个mask，而不用和其它类进行竞争；我们依赖于分类分支所预测的类别标签来选择输出的mask。这样将分类和mask生成分解开来。这与利用FCN进行语义分割的有所不同，它通常使用一个per-pixel sigmoid和一个multinomial cross-entropy loss ，在这种情况下mask之间存在竞争关系；而由于我们使用了一个per-pixel sigmoid和一个binary loss ，不同的mask之间不存在竞争关系。经验表明，这可以提高实例分割的效果。

具体公式如下：

$$L_{mask} = \frac{1}{m^2}\sum_{i}^{K}(1^k)\sum_{1}^{m^2}[-y * log(sigmoid(x)) - (1-y) * log(1 - sigmoid(x))] \tag{1}$$

其中：

- 1^k表示当第 k 个通道对应目标的真实类别时为1，否则为0 ；
- y 表示当前位置的mask的label值，为0或1 ；
- x 当前位置的输出值，sigmoid(x) 表示输出x经过sigmoid函数变换后的结果；

Faster R-CNN 的损失函数

Faster RCNN的的损失主要分为RPN的损失和Fast RCNN的损失，计算公式如下，并且两部分损失都包括**分类损失**（cls loss）和**回归损失**（bbox regression loss）。

$$L(\{p_i\}\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{2}$$

RPN 的分类损失

RPN网络的产生的anchor只分为前景和背景，前景的标签为1，背景的标签为0。在训练RPN的过程中，会选择256个anchor，256就是公式中的N_{cls}。这里的损失是经典的二分类交叉熵损失。

p_i为 anchor 预测为目标的概率;

$$GT\text{标签：}\ p_i^* = \begin{cases} 0 & negative\ label \\ 1 & positive\ label \end{cases} ;$$

$L_{cls}(p_i,\ p_i^*)$是两个类别（目标 vs.非目标）的对数损失：

$$L_{cls}(p_i, p_i^*) = -\log\left[p_i^* p_i + (1 - p_i^*)(1 - p_i)\right]$$

假设我们RPN网络的特征图大小为38×50，那么就会产生38×50×9=17100个anchor，然后在RPN的训练阶段会从17100个anchor中挑选Ncls个anchor用来训练RPN的参数，其中挑选为前景的标签为1，背景的标签为0。

Fast RCNN分类损失

RPN的分类损失时二分类的交叉熵损失，而Fast RCNN是**多分类的交叉熵损失**（当你训练的类别数>2时，这里假定类别数为5）。在Fast RCNN的训练过程中会选出128个rois，即**Ncls = 128**，标签的值就是0到4。

回归损失

回归损失这块就RPN和Fast RCNN一起讲，公式为：$\lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$

其中 $t_i = \{t_x, t_y, t_w, t_h\}$ 是一个向量，表示anchor，RPN训练阶段（rois，FastRCNN阶段）**预测的偏移量**，x, y, w, h分别表示锚盒 anchor 的中心坐标、宽和高。

t_i^* 是与 t_i 维度相同的向量，表示anchor，RPN训练阶段（rois，FastRCNN阶段）相对于gt**实际的偏移量**

$$L_{reg}(t, t_i^*) = R(t_i - t_i*)$$

R是 smoothL1 函数，不同之处是这里**σ = 3，RPN训练（σ = 1，Fast RCNN训练）**

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 \times 1/\sigma^2 & \text{if } |x| < 1/\sigma^2 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

其中 $x = t_i - t_i^*$

对于每一个 anchor 计算完 $L_{reg}(t_i, t_i^*)$ 部分后还要乘以P*，如前所述，**P*有物体时（positive）为1，没有物体（negative）时为0，意味着只有前景才计算损失，背景不计算损失。**

参数 λ 理解为为平衡分类损失和回归损失而引入的权重参数。

参考资料

Mask R-CNN 论文笔记
Mask R-CNN详解
Faster-rcnn详解
【Faster RCNN】损失函数理解
Mask-RCNN 算法及其实现详解(讲得非常好)