**Fashion属性标签识别**

第29名 全球某工商 解决方案

**实验环境：**

OS：Windows7

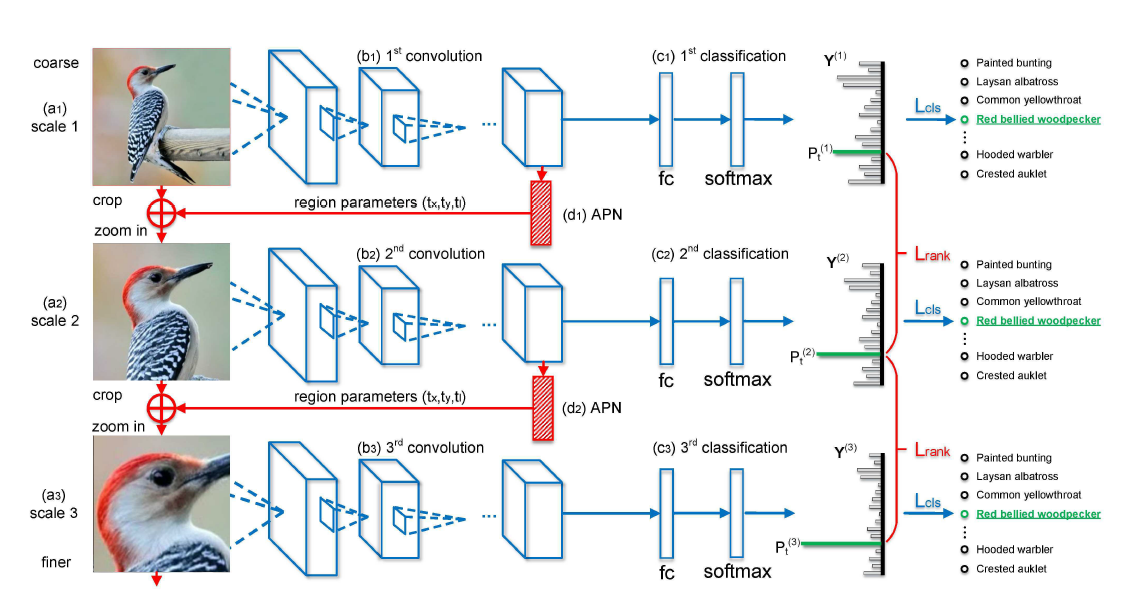
Python：3.5

Tensorflow：1.4.0

GPU：GeForce GTX 980Ti 6G 一块

**整体思路：**

我们团队受CVPR 2017论文RA-CNN的Look Closer To See Better思想的启发（论文链接：<http://openaccess.thecvf.com/content_cvpr_2017/papers/Fu_Look_Closer_to_CVPR_2017_paper.pdf>），将该论文的整体流程框架应用到我们团队的解决方案。



RA-CNN的整体框架如上图所示。RA-CNN有三个过程。第一个过程是将原始整体图片（用whole表示）建立一个模型然后通过APN网络定位到目标整体区域，再提取感兴趣区域（用ROI表示）（这里提取的是目标的整体部分）。第二个过程是将第一个提取到的ROI建立一个模型然后通过APN网络定位到目标的局部区域，再提取感兴趣区域（ROI）（这里提取的是目标的局部部分）。第三个过程是基于第二个过程提取到的ROI去建立一个模型。

最后将三个过程进行融合。

我们团队借鉴并实现了RA-CNN的第一个过程和第二个过程。我们团队用Grad-CAM ++（<https://arxiv.org/abs/1710.11063>）算法替换了APN网络用于定位目标的整体区域，并提取ROI。最终将第一个过程和第二个过程融合。详情请见下下一章：**详细解决方案**。

**数据预处理**

注：whl（whole）代表原始图片的意思，roi（Region Of Interest）代表感兴趣区域图片的意思

1. 预热Skirt数据集处理

预热Skirt数据集和正式数据集有部分重复，但不完全重复。去重后有2023张skirt图片是不重复的。再将2023张图片移动到base/Images/skirt\_length\_labels文件下进行合并。（代码见fashion-release\utils\process\_skirt.py和move\_skirt.py）

1. 数据集划分训练集和验证集

使用sklearn库的StratifiedKFold的按类别数量进行分层10折抽样，选择其中的9份作为训练集其中的一份作为验证集（验证集用于选择最优模型）。（代码见fashion-release\utils\whl\_csv\gen\_xxx\_csv.py）。

1. 制作TfRecords数据集，高效读取数据

将划分好的训练集，验证集，还有测试集制作成tfrecords文件。（代码见fashion-release\lib\tf\_data\whl\_data\create\_xxx\_tfrecords.py）

1. 读取数据并做随机数据增强
2. 数据增强：随机裁剪（先resize到(299+32)\*(299+32)再随机裁剪到299\*299大小），随机水平翻转，随机distort color，随机resize方法（有四种resize方法0：双线性差值。1：最近邻居法。2：双三次插值法。3：面积插值法。）resize图片。
3. 标签one-hot。
4. 将输入规范到[-1，1]之间（模型选择的是Inception\_resnet\_v2）作为模型的输入。

注：测试集不做数据增强操作。测试集选择的resize方法是双线性插值。

（代码见fashion-release\lib\tf\_data\read\_tfrecords.py）

**详细解决方案**

我们团队选择**Inception\_resnet\_v2**模型进行**迁移学习**。去除了最后一层的全连接层，接上八个属性类别的全连接层。

**Loss函数** 交叉熵函数

**优化器** SGDM

**网络结构**代码见fashion-release\lib\networks\inception\_resnet\_v2.py。网络结构片段如下：

**def setup**(self):  
 **with** slim.arg\_scope(inception\_resnet\_v2\_arg\_scope()):  
 net, end\_points = inception\_resnet\_v2.inception\_resnet\_v2(self.data, num\_classes=**None**, create\_aux\_logits=**False**, is\_training=self.phase\_train, reuse=self.reuse)  
 self.layers['inception\_resnet\_v2'] = net  
 self.layers['PreGAP'] = end\_points['PreGAP']

(self.feed('inception\_resnet\_v2')  
 .dropout(self.keep\_prob, name='dropout1')  
 .auto\_fc(8, name='cls\_score\_coat', relu=**False**)  
 .softmax(name='cls\_prob\_coat'))  
  
 (self.feed('inception\_resnet\_v2')  
 .dropout(self.keep\_prob, name='dropout2')  
 .auto\_fc(5, name='cls\_score\_collar', relu=**False**)  
 .softmax(name='cls\_prob\_collar'))  
  
 (self.feed('inception\_resnet\_v2')  
 .dropout(self.keep\_prob, name='dropout3')  
 .auto\_fc(5, name='cls\_score\_lapel', relu=**False**)  
 .softmax(name='cls\_prob\_lapel'))  
  
 (self.feed('inception\_resnet\_v2')  
 .dropout(self.keep\_prob, name='dropout4')  
 .auto\_fc(5, name='cls\_score\_neck', relu=**False**)  
 .softmax(name='cls\_prob\_neck'))  
  
 (self.feed('inception\_resnet\_v2')  
 .dropout(self.keep\_prob, name='dropout5')  
 .auto\_fc(10, name='cls\_score\_neckline', relu=**False**)  
 .softmax(name='cls\_prob\_neckline'))  
  
 (self.feed('inception\_resnet\_v2')  
 .dropout(self.keep\_prob, name='dropout6')  
 .auto\_fc(6, name='cls\_score\_pant', relu=**False**)  
 .softmax(name='cls\_prob\_pant'))  
  
 (self.feed('inception\_resnet\_v2')  
 .dropout(self.keep\_prob, name='dropout7')  
 .auto\_fc(6, name='cls\_score\_skirt', relu=**False**)  
 .softmax(name='cls\_prob\_skirt'))  
  
 (self.feed('inception\_resnet\_v2')  
 .dropout(self.keep\_prob, name='dropout8')  
 .auto\_fc(9, name='cls\_score\_sleeve', relu=**False**)  
 .softmax(name='cls\_prob\_sleeve'))

预训练好的inception\_resnet\_v2模型权值文件下载链接<https://github.com/tensorflow/models/tree/master/research/slim>选择inception\_resnet\_v2下载。

inception\_resnet\_v2 code: <https://github.com/tensorflow/models/blob/master/research/slim/nets/inception_resnet_v2.py>

**训练过程如下：**

1. 随机交替多任务训练出一个检测模型（**相当于RA-CNN第一个过程**）

（1）这里的随机交替多任务训练是指这个step训练coat, 下个step训练skirt,一直这样随机交替下去训练。（这里训练是指在inception\_resnet\_v2基础上微调）

初始学习率：0.001每5个epoch以0.75比例下降总共15个epoch。

batch\_size为16。

训练代码见fashion-release\experiment\AlternateMultiTask\train.py（运行入口，制作好tfrecords文件就能运行）

交替多任务训练器代码见fashion-release\lib\trainers\amt\_trainer.py

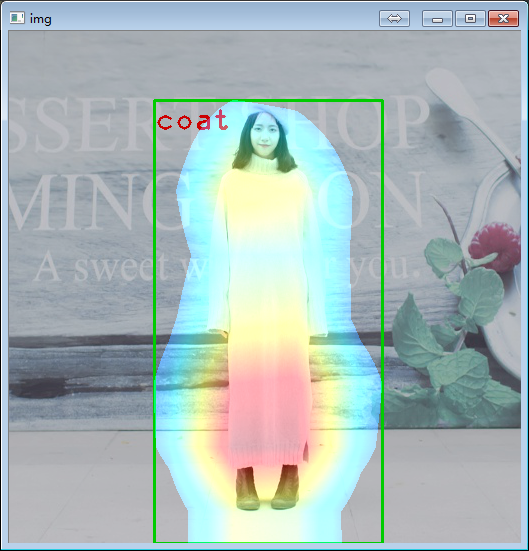
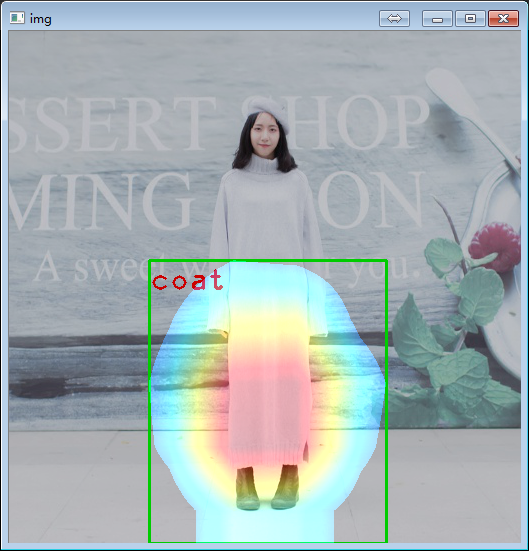
验证每个epoch保存的模型并选择最高acc的模型。代码见fashion-release\experiment\AlternateMultiTask\validate.py

（2）将上述训练好的分类模型通过CAM（Class Activitation Map）算法提取ROI

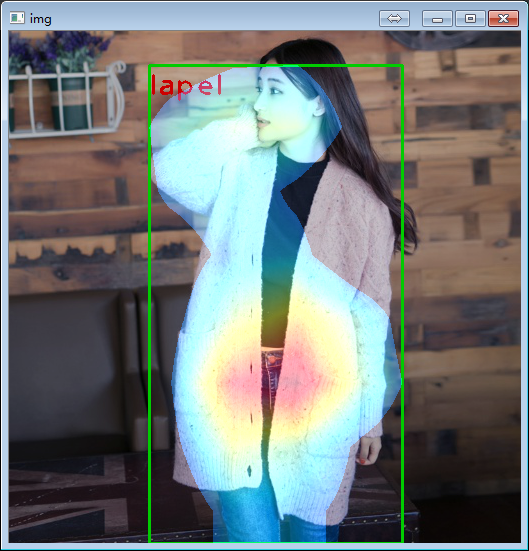
这里我们团队选择grad\_cam（<https://arxiv.org/abs/1610.02391>）和grad\_cam++试验了一下。代码见fashion-release\experiment\CAM\gen\_grad\_cam.py和gen\_grad\_cam\_plus.py。

grad-cam++突出显示整个类别。而grad-cam突出显示最显著的部分。

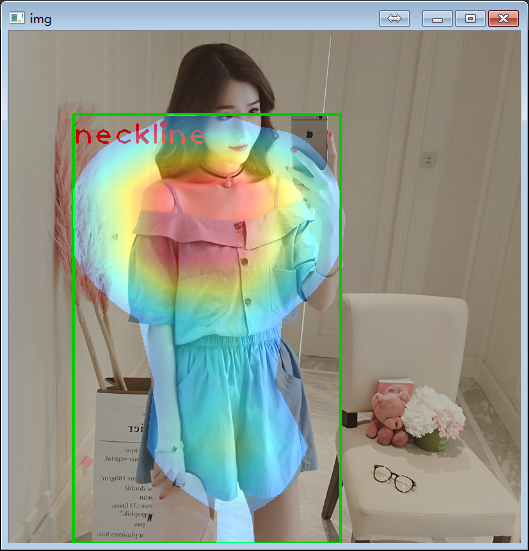
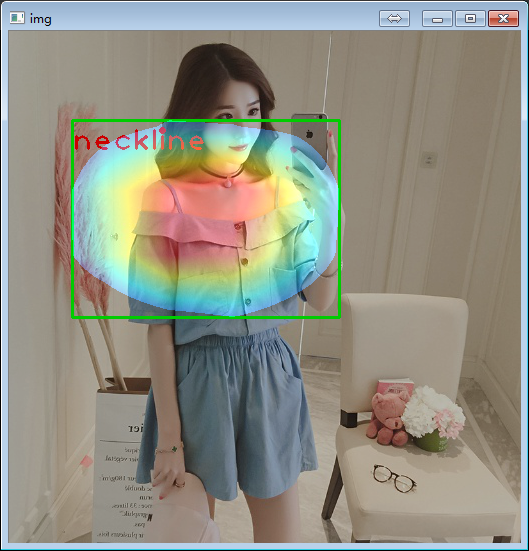
**grad-cam和grad-cam++效果对比图：**（左grad-cam右grad-cam++相同的阈值为64）

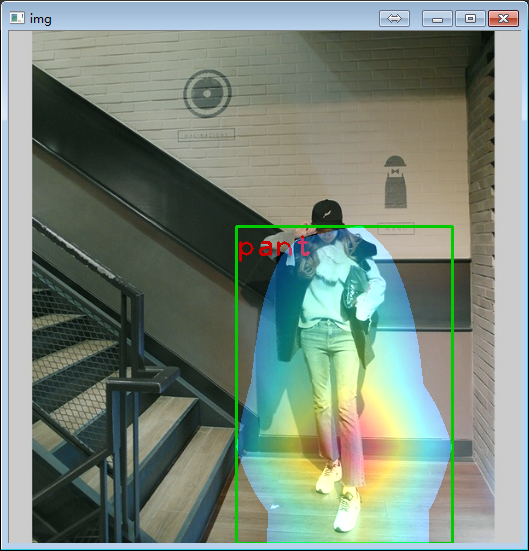
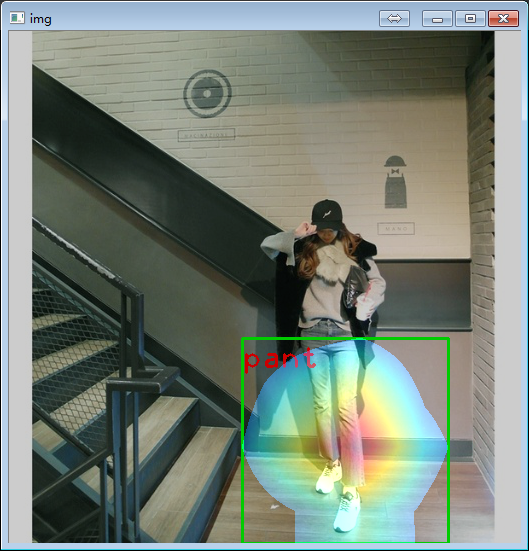


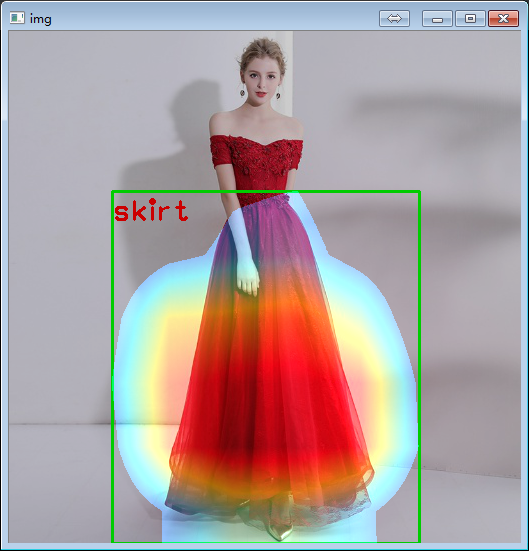


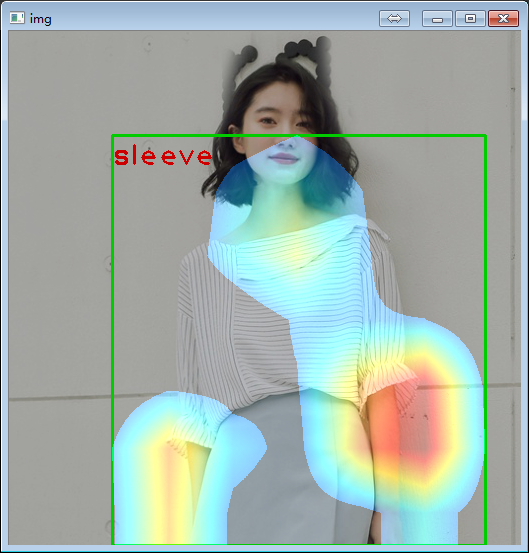
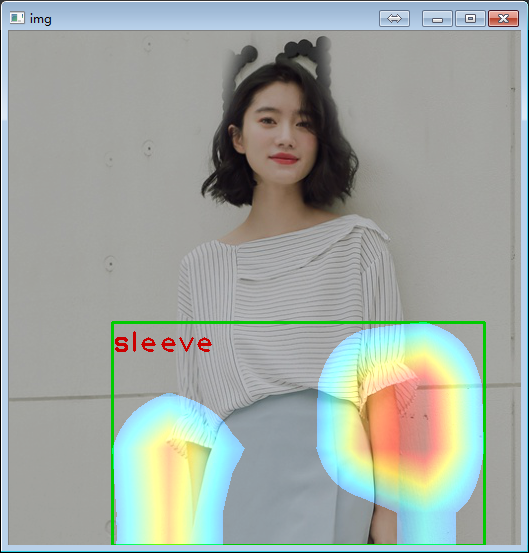












上图的bouding box是将cam 热力图resize到原图大小然后二值化为灰度图，设置阈值为64通过Opencv的boundingRect方法即可计算出Bounding Box。

1. 裁剪ROI图片并制作tfrecords文件

我们团队选择grad-cam++将上述计算的bounding box裁剪下来生成ROI图片。代码见fashion-release\experiment\CAM\gen\_roi.py

并制作roi.tfrecords文件代码见fashion-release\lib\tf\_data\roi\_data\create\_xxx\_tfrecords.py

还要制作mix.tfrecords文件（是将whl+roi放一起）代码见fashion-release\lib\tf\_data\mix\_data\create\_xxx\_tfrecords.py

1. 将原始图片whl和ROI图片放一起训练一个分类模型base\_model（**相当于RA-CNN第二个过程**）

将whl+roi（即mix.tfrecords文件）数据进行随机多任务交替训练（在inception\_resnet\_v2基础上微调），训练出一个base\_model。（训练参数和检测模型训练参数一致）

1. 在base\_model基础上进行单任务微调。

将roi数据在base\_model基础上8个任务分别微调。

单任务微调代码见fashion-release\ experiment\SingleTask\script.py（运行入口）

单任务训练器代码见fashion-release\lib\trainers\st\_trainer.py

初始学习率为0.0005, 每3个epoch 0.75的比例下降, 总共12个epoch

batch\_size为16

**预测过程**

第一个过程预测whl数据得到结果1，第二个过程预测roi数据得到结果2，将结果1答案和结果2而答案**简单线性加权（0.5的比例）融合**。

第一个过程预测代码见fashion-release\experiment\AlternateMultiTask\predict.py

第二个过程预测代码见fashion-release\experiment\SingleTask\predict.py

**融合**代码见fashion-release\utils\fusion\process\_sub.py

第一个过程预测whl数据分数96.xxAP88.xxBP

第二个过程预测roi数据分数97.xxAP90.xxBP

将两个过程融合98.xxAP91.xxBP。AP和BP都会提升非常明显。

我的理解是第一个过程更加注重非真实标签预测。第二个过程更加注重真实标签预测。

模型从远到近的不同的视角看一个目标，融合分数提升明显。