冠军的试炼

悟已往之不谏，知来者之可追

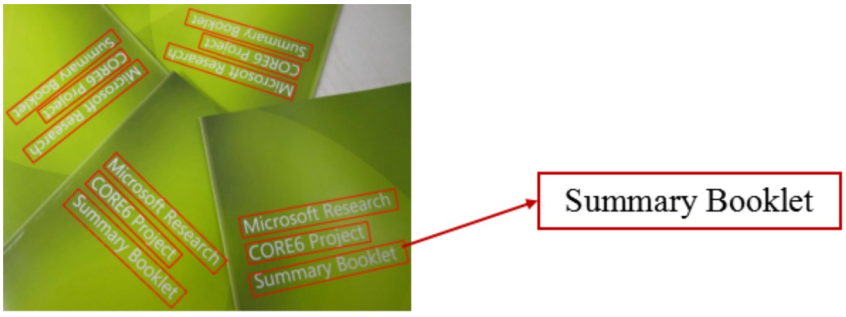
【OCR技术系列之五】自然场景文本检测技术综述（CTPN, SegLink, EAST）

文字识别分为两个具体步骤：文字的检测和文字的识别，两者缺一不可，尤其是文字检测，是识别的前提条件，若文字都找不到，那何谈文字识别。今天我们首先来谈一下当今流行的文字检测技术有哪些。

文本检测不是一件简单的任务，尤其是复杂场景下的文本检测，非常具有挑战性。自然场景下的文本检测有如下几个难点：

- 文本存在多种分布，文本排布形式多样；
- 文本存在多个方向；
- 多种语言混合。

我们先从直观上理解文本检测任务。给定一张图片，我们需要找出这张图里文字出现的所有位置位置，那这个任务其实跟目标检测任务差别不大，即找出每个物体在图片中的位置，并标出该包围框里的物体的类别。而文本检测就是，找出每个文本在图片中出现的位置，因为我们的类别只有2个（有文字和没文字），看起来就像一个简单的单类别目标检测的任务，自然而然我们就会想到用经典的目标检测网络来进行文本检测，比如经典的Faster R-CNN。



Faster RCNN

Faster RCNN来做文本检测从任务上分析是可行的，毕竟文本说到底还是一个Object。我们回顾一下Faster RCNN做目标检测的关键步骤有哪些：

公告

昵称：Madcola
园龄：2年7个月
粉丝：1334
关注：30
+加关注

‹ 2019年8月 ›

日	一	二	三	四	五	六
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类(69)

C++(1)
CUDA(1)
Linux编程(12)
OCR系列(8)
opencv探索(28)
STL(2)
波折岁月(4)
工具技巧(1)
机器学习之旅(5)
深度学习(4)
数字图像处理(3)

随笔档案(69)

2019年2月 (1)
2019年1月 (1)

1. 基础网络做特征提取；
2. 特征送入RPN做候选框提取；
3. 分类层对候选框内物体进行分类，回归层对候选框的(x,y,w,h)进行精细调整。

Faster RCNN做文本检测感觉问题不大，但是从效果来看，仅套用Faster RCNN来做文本检测效果并不好，原因在于，文本有自己独有的特点，这种通用的文本检测框架并不能很好地解决文本的这些特点。那文本有什么特点呢？我总结如下：

1. 文本大多数以长矩形形式存在，即长宽比一般较大或较小，这与普通的目标检测中的物体不一样（这些长宽比较接近1）
2. 普通物体（比如猫）存在明显的闭合边缘轮廓，而文本没有；
3. 文本中包含多个文字，而文字之间是有间隔的，如果检测做得不好，我们就会把每个字都当成文本行给框出来而非整行作为文本框，这与我们的期望不一样。

基于以上文本检测的特点，我们必须对Faster RCNN这类通用网络进行改进，设计出适合文本检测的全新网络架构。

CTPN（2016）

2016年出了一篇很有名的文本检测的论文：《Detecting Text in Natural Image with Connectionist Text Proposal Network》，这个深度神经网络叫做CTPN，直到今天这个网络框架一直是OCR系统中做文本检测的一个常用网络，极大地影响了后面文本检测算法的方向。

这个算法很有创新，我打算一步一步介绍其闪光点。我们回顾一下Faster RCNN做目标检测的一个缺点就是，没有考虑带文本自身的特点。文本行一般以水平长矩形的形式存在，而且文本行中每个字都有间隔。针对这个特点，CTPN剔除一个新奇的想法，我们可以把文本检测的任务拆分，第一步我们检测文本框中的一部分，判断它是不是一个文本的一部分，当对一幅图里所有小文本框都检测之后，我们就将属于同一个文本框的小文本框合并，合并之后就可以得到一个完整的、大的文本框了，也就完成了文本的检测任务。这个想法真的很有创造性，有点像“分治法”，先检测大物体的一小部分，等所有小部分都检测出来，大物体也就可以检测出来了。

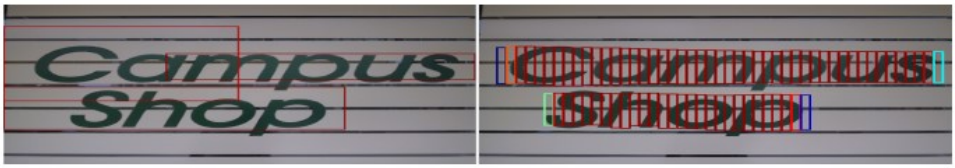


Fig. 2: **Left**: RPN proposals. **Right**: Fine-scale text proposals.

如图所示，左边的图是直接使用Faster RCNN中的RPN来进行候选框提取，可以看出，这种候选框太粗糙了，效果并不好。而右图是利用许多小候选框来合并成一个大文本预测框，可以看出这个算法的效果非常不错，需要说明的是，红色框表示这个小候选框的置信度比较高，而其他颜色的候选框的置信度比较低，我们可以看到，一个大文本的边界都是比较难预测的，那怎么解决这个边界预测不准的问题呢？后面会提到。

刚提到CTPN的其中一个闪光点，即检测小框代替直接检测大文本框。除了这个新意，CTPN还提出了在文本检测中应加入RNN来进一步提升效果。为什么要用RNN来提升检测效果？文本具有很强的连续字符，其中连续的上下文信息对于做出可靠决策来说很重要。我们知道RNN常用于序列模型，比如事件序列，语言序列等等，那我们CTPN算法中，把一个完整的文本框拆分成多个小文本框集合，其实这也是一个序列模型，可以利用过去或未来的信息来学习和预测，所以同样可以使用RNN模型。而且，在CTPN中，用的还是BiLSTM（双向LSTM），因为一个小文本框，对于它的预测，我们不仅与其左边的小文本框有关系，而且还与其右边的小文本框有关系！这个解释就很有说服力了，如果我们仅仅根据一个文本框的信息区预测该框内含不含有文字其实是很草率的，我们应该多参考这个框的左边和右边的小框的信息后（尤其是与其紧挨着的框）再做预测准确率会大大提升。

2018年12月 (2)
2018年10月 (1)
2018年9月 (3)
2018年5月 (1)
2018年4月 (2)
2018年2月 (6)
2018年1月 (3)
2017年12月 (4)
2017年11月 (3)
2017年10月 (1)
2017年9月 (4)
2017年8月 (3)
2017年7月 (5)
2017年6月 (4)
2017年5月 (17)
2017年4月 (1)
2017年2月 (2)
2017年1月 (5)

如上图所示，如果我们单纯依靠1号框内的信息来直接预测1号框中否存在文字（或者说是不是文本的一部分），其实难度相当大，因为1号框只包含文字的很小一部分。但是如果我们把2号框和3号框的信息都用上，来预测1号框是否存在文字，那么我们就会有比较大的把握来预测1号框确实有文字。我们还可以看看为什么边缘的文本框的置信度会较中间的低呢？个人认为很大一部分原因就在于因为这些框都位于总文本的边缘，没有办法充分利用左右相邻序列的信息做预测（比如位于最左的文本框丢失了其右边的信息）。这就是双向LSTM的作用，把左右两个方向的序列信息都加入到学习的过程中去。

Fig. 3: **Top**: CTPN without RNN. **Bottom**: CTPN with RNN connection.

我们已经对CTPN这个算法的总体思路有了一点理解，那关键问题来了，我们怎么把这些小文本框准确地检测出来呢？

CTPN借助了Faster RCNN中anchor回归机制，使得RPN能有效地用单一尺寸的滑动窗口来检测多尺寸的物体。当然CTPN根据文本检测的特点做了比较多的创新。比如RPN中anchor机制是直接回归预测物体的四个参数（x,y,w,h），但是CTPN采取之回归两个参数(y,h)，即anchor的纵向偏移以及该anchor的文本框的高度，因为每个候选框的宽度w已经规定为16个像素，不需要再学习，而x坐标直接使用anchor的x坐标，也不用学习，所以CTPN的思路就是只学习y和h这两个参数来完成小候选框的检测！跟RPN相类似，CTPN中对于每个候选框都使用了K个不同的anchors（k在这里默认是10），但是与RPN不同的是，这里的anchors的width是固定的16个像素，而height的高度范围为11~273（每次对输入图像的height除以0.7，一共K个高度）。当然CTPN中还是保留了RPN大多数的思路，比如还是需要预测候选框的分数score（该候选框有文本和无文本的得分）。

这么多小尺度候选框怎么才能串联成一个完整的文本行呢？

文本行构建很简单，通过将那些text/no-text score > 0.7的连续的text proposals相连接即可。文本行的构建如下。首先，为一个proposal Bi定义一个邻居（Bj）：Bj->Bi，其中：

1. Bj在水平距离上离Bi最近
2. 该距离小于50 pixels
3. 它们的垂直重叠(vertical overlap) > 0.7

另外，如果同时满足Bj->Bi和Bi->Bj，会将两个proposals被聚集成一个pair。接着，一个文本行会通过连续将具有相同proposal的pairs来进行连接来构建。

接下来我们就较为细节地学习一下这个CTPN经典网络。

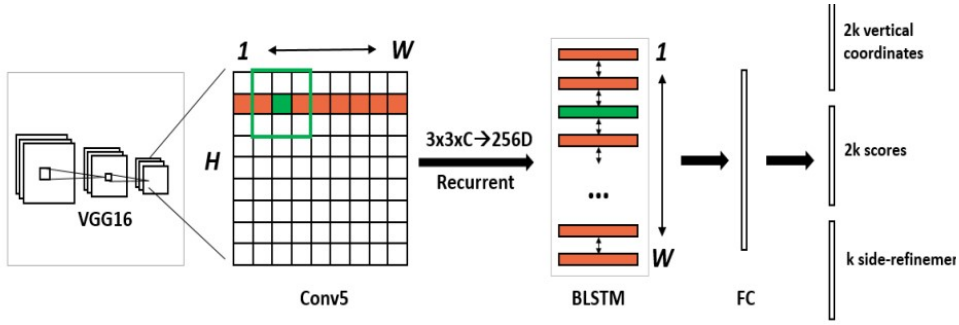
R-CNN(206008)
3. OpenCV探索之路（二十四）图像拼接和图像融合技术(88663)
4. CNN网络架构演进：从LeNet到DenseNet(58616)
5. OpenCV探索之路（二十三）：特征检测和特征匹配方法汇总(54127)
6. 【OCR技术系列之四】基于深度学习的文字识别（3755个汉字）(51233)
7. C++ STL快速入门(45935)
8. OpenCV探索之路（六）：边缘检测（canny、sobel、laplacian）(45154)
9. Linux编程之UDP SOCKET全攻略(41919)
10. OpenCV探索之路（十四）：绘制点、直线、几何图形(37784)

评论排行榜

1. 【OCR技术系列之四】基于深度学习的文字识别（3755个汉字）(82)
2. 【Keras】基于SegNet和U-Net的遥感图像语义分割(73)
3. OpenCV探索之路（二十四）图像拼接和图像融合技术(67)
4. 【OCR技术系列之八】端到端不定长文本识别CRNN代码实现(59)
5. 【Keras】从两个实际任务掌握图像分类(33)

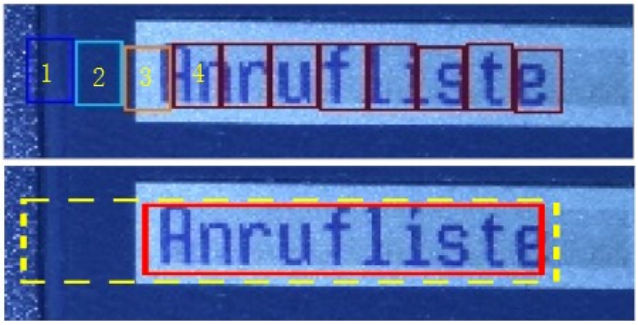
推荐排行榜

1. 基于深度学习的目标检测技术演进：R-CNN、Fast R-CNN、Faster R-CNN(92)
2. 卷积神经网络CNN总结(61)
3. 我的2018：OCR、实习和秋招(22)
4. 【OCR技术系列之四】基于深度学习的文字识别（3755个汉字）(21)
5. 【Keras】基于SegNet和U-Net的遥感图像语义分割(20)
6. CNN网络架构演进：从LeNet到DenseNet(20)
7. OpenCV探索之路（二十四）图像拼接和图像融合技术(18)
8. 我在北京实习的四个月(15)
9. 【OCR技术系列之一】字符识别技术总览(13)
10. 读研以来的一些感想：名校好在哪里？(13)



首先CTPN的基础网络使用了VGG16用于特征提取，在VGG的最后一个卷积层CONV5，CTPN用了3×3的卷积核来对该feature map做卷积，这个CVON5 特征图的尺寸由输入图像来决定，而卷积时的步长却限定为16，感受野被固定为228个像素。卷积后的特征将送入BLSTM继续学习，最后接上一层全连接层FC输出我们要预测的参数：2K个纵向坐标y，2k个分数，k个x的水平偏移量。看到这里大家可能有个疑问，这个x的偏移到底是什么，为什么需要回归这个参数？如果需要X的参数，为什么不在候选框参数回归时直接预测成（x,y,h）三个参数呢，而要多此一举把该参数单独预测？这个X的作用作者提到这也是他们论文的一大亮点，称之为Side-refinement，我理解为文本框边缘优化。我们回顾一下上面提到的一个问题，文本框检测中边缘部分的预测并不准确。那么改咋办，CTPN就是用这个X的偏移量来精修边缘问题。这个X是指文本框在水平方向的左边界和右边界，我们通过回归这个左边界和右边界参数进而可以使得我们对文本框的检测更为精准。在这里想举个例子说明一下回归这个x参数的重要性。

我们观察下图，第一幅图张我们看到我们有很多小候选框，位于左边的候选框我标记为1、2、3、4号框,1号框和2号框为蓝色，表明得分不高我们不把这两个框合并到大文本框内，对于3号框和4号框那就比较尴尬了，如果取3号框作为文本框的边缘框，那么显然左边边缘留白太多，精准度不够，但如果去掉3号框而使用4号框作为左边缘框，则有些字体区域没有检测出来，同样检测精度不足。这种情况其实非常容易出现，所以CTPN采取了Side-refinement 思路进一步优化边缘位置的预测即引入回归X参数，X参数直接标定了完整文本框的左右边界，做到精确的边界预测。第二幅图中的红色框就是经过Side-refinement后的检测结果，可以看出检测准确率有了很大的提升。 side-refinement确实可以进一步提升位置准确率，在SWT的Multi-Lingual datasets上产生2%的效果提升。



再看多几幅图，体验一下Side-refinement后的效果。

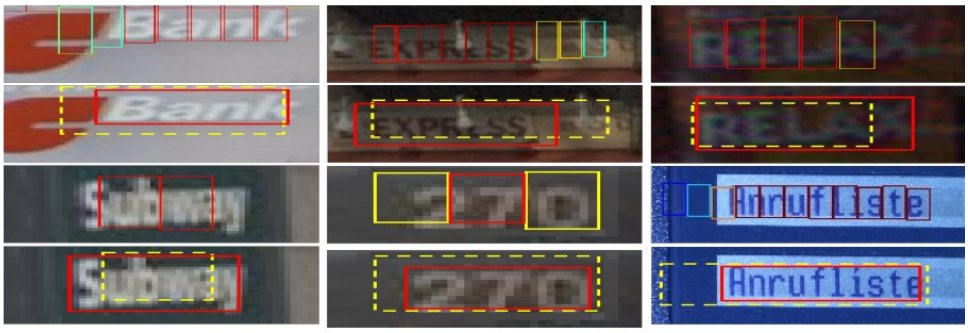


Fig. 4: CTPN detection with (red box) and without (yellow dashed box) th
side-refinement. Color of fine-scale proposal box indicate a text/non-text score

最后总结一下CTPN这个流行的文本检测框架的三个闪光点：

- 将文本检测任务转化为一连串小尺度文本框的检测；

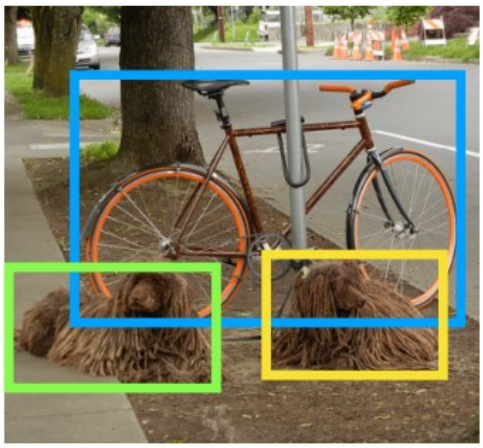
- 引入RNN提升文本检测效果；
- Side-refinement（边界优化）提升文本框边界预测精准度。

当然，CTPN也有一个很明显的缺点：对于非水平的文本的检测效果并不好。CTPN论文中给出的文本检测效果图都是文本位于水平方向的，显然CTPN并没有针对多方向的文本检测有深入的探讨。那对于任意角度的文本检测应该采取什么的算法思路呢？下面的SegLink算法给出了一个新奇的解决方案。

SegLink（2017）

CVPR2017的一篇spotlight论文《Detecting Oriented Text in Natural Images by Linking Segments》介绍以一种可以检测任意角度文本的检测算法，我们一般称这个算法为SegLink，这篇论文既融入CTPN小尺度候选框的思路又加入了SSD算法的思路，达到了当时自然场景下文本检测state-of-art的效果。

现在我想先介绍为什么要针对多方向的文本检测做特定的研究。对于普通目标检测，我们并不需要对其做所谓的多方向目标检测，比如下面这个检测任务，我们直接把单车和狗的位置找出来即可。

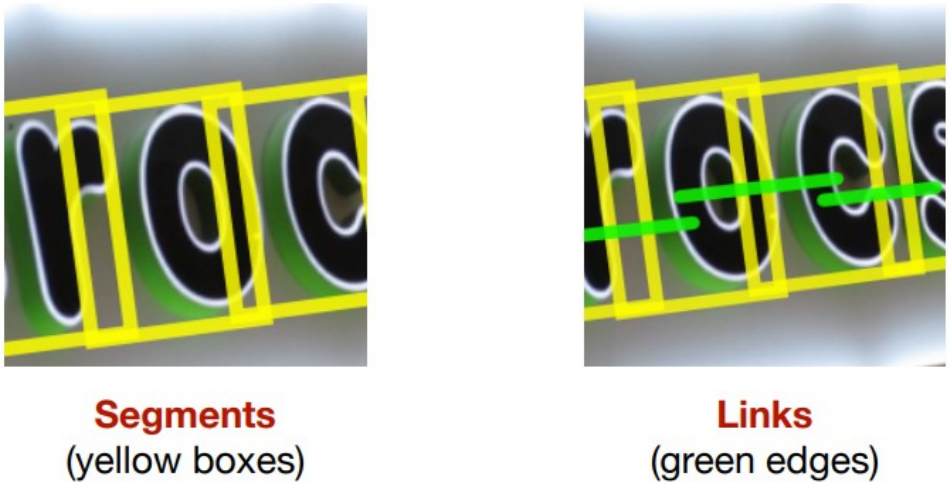


但是对于文本检测任务可不一样，文本的特点就是高宽比特别大或小，而且文本通常存在一定的旋转角度，如果我们对于带角度的文本仍然使用目标检测那个思路回归四个参数（x,y,w,h）来指定一个目标的位置的话（如下图红色框），那显然误差太大了，这个检测效果并不是我们所能接受的。作为对比，下图的绿色框的检测效果才是我们的终极目标。那么怎么基于原来经典的目标检测算法做相应的优化以适应这种检测效果的要求呢？

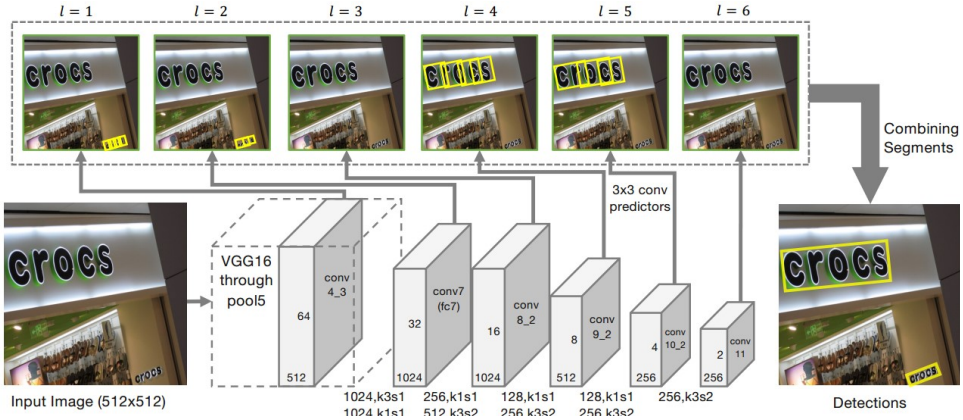
一个最直接的思路就是让模型再学习一个参数θ！这个θ表示文本框的旋转角度，也就是我们最终要回归的参数从原来的(x, y, w, h) 变成 (x, y, w, h, θ)。SegLink确实也采取了这个思路，除此之外，他还提出了Segment和Linking两个重要概念，这个才是这篇CVPR论文的核心创新点。

什么是segment？segment从中文上理解为文本行的一部分，这一部分可以是一个字符或文本行的任意一部分。如下图示，黄色框表示一个segment，一个完整的文本行中包含多个segment，每个sgment之间通过link（图中的绿色线）连接组合起来。那么Segment做文本检测的思路其实跟CTPN的思路很像，先检测文本行的一部分，再把他们连接起来构成一个完整文本行。

我们把图片的关键部位放大看看细节：首先每个segment是有一定的重合区域的，然后每两个segment连接的部位是两个segment的中心点。每一个segment和link仅依靠局部图像的纹理信息即可完成检测，而无需整张图像的信息。



接下来我们通过分析SegLink的网络架构进一步理解SegLink如何做到高效的多角度文本检测。下图是SegLink的网络架构，显然这个架构采取了SSD的思路，首先使用VGG16作为backbone进行特征提取，其中VGG16的全连接层（fc6,fc7）替换成卷积层（conv6,conv7），再接卷积层conv8到conv11。值得说明的是，conv4~conv11之间的尺寸依次减少（每一层是前一层的1/2）。这个做法是为了做多尺度下的目标检测，即大的feature map擅长做小物体的检测，而小的feature map则擅长检测大物体。借助多个不同尺度的feature map，从6个feature layer上检测segment和link，我们就可以检测出不同尺寸的文本行了。



观察后面的卷积层可以发现，对不同层的feature map使用3×3的卷积层产生最终的输出(包括segment和link)，不同特征层输出的维度是不一样的，因为除了conv4_3层外，其它层存在跨层的link。这里segment是text的带方向bbox信息(它可能是个单词，也可能是几个字符，总之是文本行的部分)，link是不同segment的连接信息(文章将其也增加到网络中自动学习)。

当所有segments都被检测出来后，我们就可以通过融合规则（combining segments），将各个feature map的segment的box信息和link信息进行融合，得到最终的文本行。

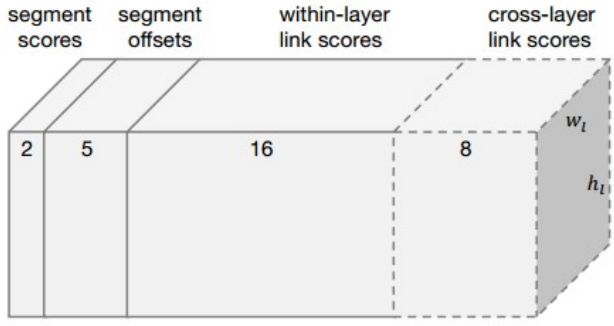
SegLink所使用的目标函数由三个部分构成，是否是text的二类分类的softmax损失，box的smooth L1 regression损失，是否link的二类的softmax损失。λ1和λ2控制权重，最后都设为1。

$$L(y_s, c_s, y_l, c_l, \hat{s}, s) = \frac{1}{N_s} L_{conf}(y_s, c_s) + \lambda_1 \frac{1}{N_s} L_{loc}(\hat{s}, s) + \lambda_2 \frac{1}{N_l} L_{conf}(y_l, c_l)$$

现在计算一下每个feature map输出的参数有哪些呢?

- segment的位置信息：(x, y, w, h, θ)，一共5个参数
- 每个segment内的分类分数，即判断框内有字符还是无字符的分数（2分类），共2个参数
- 同层（within-layer）的每个segment的link的分数，表示该方向有link还是没link（2分类问题），而一个segment有八邻域所以有八个方向，参数一共有2×8=16
- 相邻层(cross-layer)之间也存在link，同样是该方向有link还是没link（2分类问题），而link的个数是4个，所以参数总数为2×4=8

下图很清楚地表示出每个feature map输出的参数有多少个，输出参数总数为（2+5+16+8=31）。假设当前的feature map的尺度为(w,h)，那么该层卷积后输出为w×h×31。

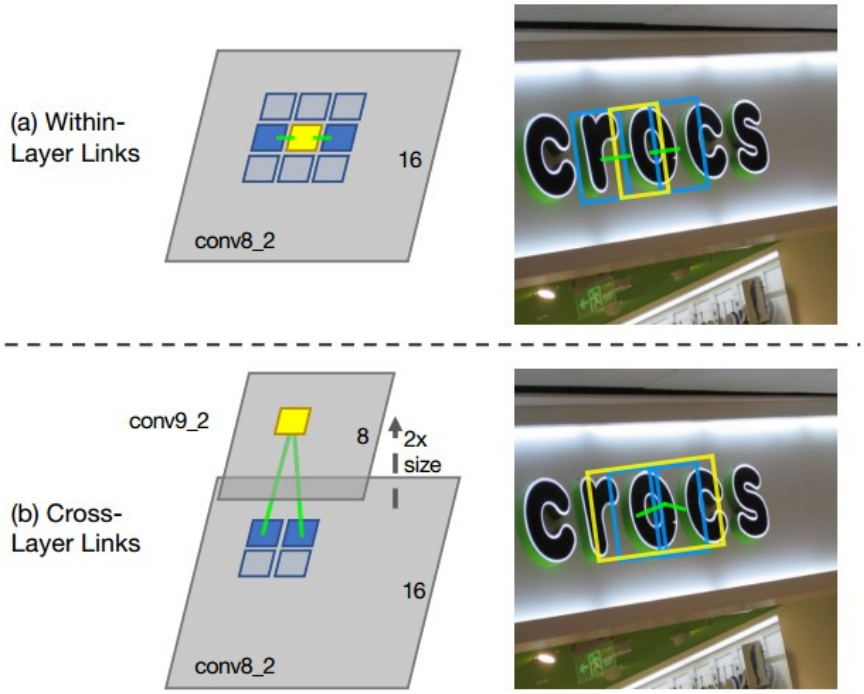


这里想再谈谈Within-Layer Link和Cross-Layer Link的作用。

within-layer link表示在同一层feature layer里，每个Segment与8邻域内的segment的连接状况，如下图(a)所示。且每个link有2维，一维是正分，表示两个segment属于同一文本，一维是负分，表示两个segment不属于同一文本。所以，每个predictor输出16（8×2）维向量。

cross-layer link：在不同的feature layer上有可能会检测到同一文本的segments，造成冗余，cross-layer link的

提出就是为了解决这个问题。cross-layer link连接了两个相邻feature layer上的segments，如图(b)所示。需要注意的是，由于下采样使后一层为前一层scale的1/2，定义一个segment的cross-layer邻居为前一层4邻域更小的segment，即前一层是后一层的邻居，但后一层不是前一层的邻居，所以conv4_3的feature layer没有cross-layer邻居。图中所示的黄框为当前层的segment，蓝框为上一层更小更细的segment，绿色的线代表cross-layer link有连接，属于同一文本，在后续的combine算法中会将他们融合，即去除了冗余。



读到这里我们已经知道如何获取segment和相应的link了，那接下来要做的就是怎么把这些link和segment合并成一个完整的文本行。先贴一下论文中使用到的合并算法：

Algorithm 1 Combining Segments

Input: $\mathcal{B} = \{s^{(i)}\}_{i=1}^{|\mathcal{B}|}$ is a set of segments connected by links, where $s^{(i)} = (x_s^{(i)}, y_s^{(i)}, w_s^{(i)}, h_s^{(i)}, \theta_s^{(i)})$.
Find the average angle $\theta_b := \frac{1}{|\mathcal{B}|}\sum_{\mathcal{B}} \theta_s^{(i)}$.
For a straight line $(\tan\theta_b)x + b$, find the b that minimizes the sum of distances to all segment centers $(x_s^{(i)}, y_s^{(i)})$.
Find the perpendicular projections of all segment centers onto the straight line.
From the projected points, find the two with the longest distance. Denote them by (x_p, y_p) and (x_q, y_q).
$x_b := \frac{1}{2}(x_p + x_q)$
$y_b := \frac{1}{2}(y_p + y_q)$
$w_b := \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} + \frac{1}{2}(w_p + w_q)$
$h_b := \frac{1}{|\mathcal{B}|}\sum_{\mathcal{B}} h_s^{(i)}$
$b := (x_b, y_b, w_b, h_b, \theta_b)$
Output: b is the combined bounding box.

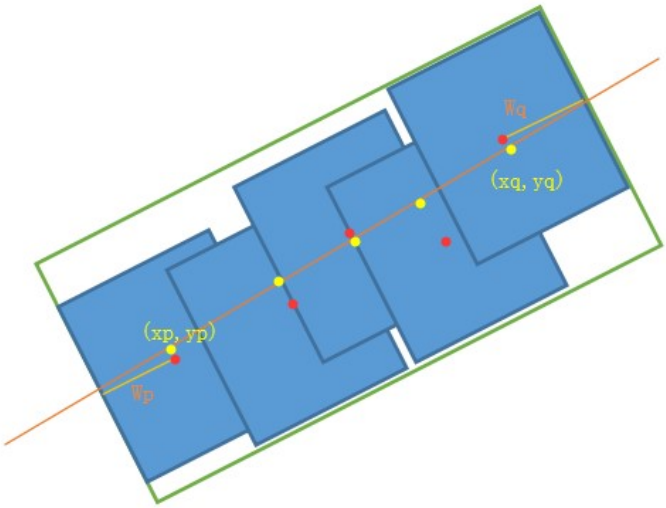
看起来真的是头都大，其实思想很简单，我尝试用中文解释一下：

1. 假设我们有一个集合B，里面有很多相关联的segment待合并；
2. 每一个segment都有自己的角度θ，那我们求集合B中所有segment角度的平均值θ_b；
3. 求一条直线L使得所有segment的中心到这条直线的距离最小，也就是最小二乘法线性回归啦；
4. 每个segment的中心往直线L做垂直投影；
5. 从所有投影点中选出相距最远的两个点，记做（xp,yp）和（xq,yq）；
6. 最终合并好的文本框的位置参数记为$(x_b, y_b, w_b, h_b, \theta_b)$那么$x_b := 1/2(x_p + x_q,\ y_b := 1/2(y_p + y_q)$
7. 文本行的宽度wb就是两个最远点的距离（即(xp,yp)和(xq,yq)）再加上最远两个点所处的segment的宽度

的一半(Wp和Wq)。

8. 文本行高度hb就是所有segment高度求平均值

我画了下图辅助理解合并算法，橙色直线是拟合出的最佳直线，红色点表示segment的中心，黄点表示红点在直线上的投影，绿框就是合并后的完整本文框。



这样子我们就求解完一个完整文本框的所有参数，也就完成了segment合并成文本行的任务。

SegLink算法对于各种角度的文本检测具有很强的鲁棒性。

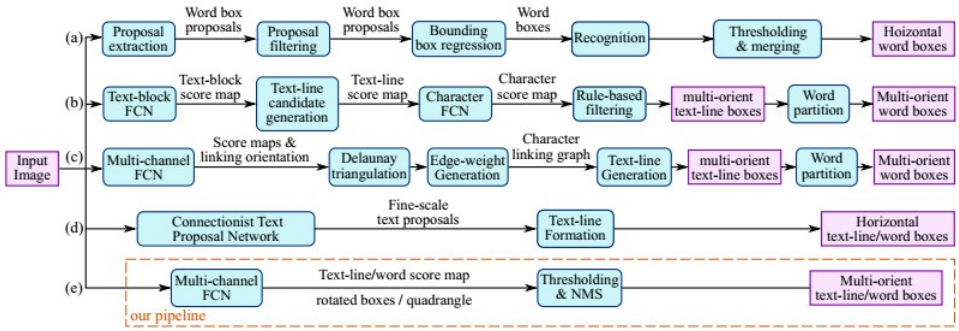
SegLink论文中并没有提到该算法能不能检测弯曲的文本，从理论上解读，SegLink是可以做到的。比如下图，只是合并算法要做一些改变而已。

Seglink can detect text of curved shape

EAST（2017）

对于以上把完整文本行先分割检测再合并的思路，有人提出质疑，觉得这种做法比较麻烦，把文本检测切割成多个阶段来进行，这无疑增大了文本检测精度的损失和时间的消耗，对于文本检测任务上中间处理越多可能效果越差。所以有篇CVPR2017的文章提出，我们有一种方法能优雅且简洁地完成多角度文本检测，这个算法叫做EAST，论文为《EAST: An Efficient and Accurate Scene Text Detector》。

通过下图我们知道，一个文本检测有多个阶段，就以region proposals系的检测算法为例，他们通常包含候选框提取、候选框过滤、bouding box回归、候选框合并等阶段，EAST的作者认为，一个文本检测算法被拆分成多个阶段其实并没有太多好处，实现真正端到端的文本检测网络才是正确之举。所以EAST的pipeline相当优雅，只分为FCN生成文本行参数阶段和局部感知NMS阶段，网络的简洁是的检测的准确性和速度都有了进一步的提高。



我们从网络架构来理解EAST做文本检测的优势。首先EAST采取了FCN的思路，一开始我以为EAST是一个通过语义分割来解决文本检测的难题，深入阅读后才发现并不是，而只是借助了FCN的架构做特征提取和学习，最终还是一个回归问题，在EAST最后预测出相应的文本行参数。

EAST网络分为特征提取层+特征融合层+输出层三大部分。

特征提取层： backbone采取PVANet来做特征提取，接下来送入卷积层，而且后面的卷积层的尺寸依次递减（size变为上一层的一半），而且卷积核的数量依次递增（是前一层的2倍）。抽取不同level的feature map，这样可以得到不同尺度的特征图，目的是解决文本行尺度变换剧烈的问题，size大的层可用于预测小的文本行，size小的层可用于预测大的文本行。

特征合并层，将抽取的特征进行merge．这里合并的规则采用了U-net的方法，合并规则：从特征提取网络的顶部特征按照相应的规则向下进行合并，这里描述可能不太好理解，具体参见下述的网络结构图。

网络输出层：网络的最终输出有5大部分，他们分别是：

- score map：一个参数，表示这个预测框的置信度；
- text boxes: 4个参数，（x,y,w,h），跟普通目标检测任务的bounding box参数一样，表示一个物体的位置；

- text rotation angle: 1个参数，表示text boxe的旋转角度；
- text quadrangle coordinates：8个参数，表示任意四边形的四个顶点坐标，即(x1,y1),(x2,y2),(x3,y3),(x4,y4)。

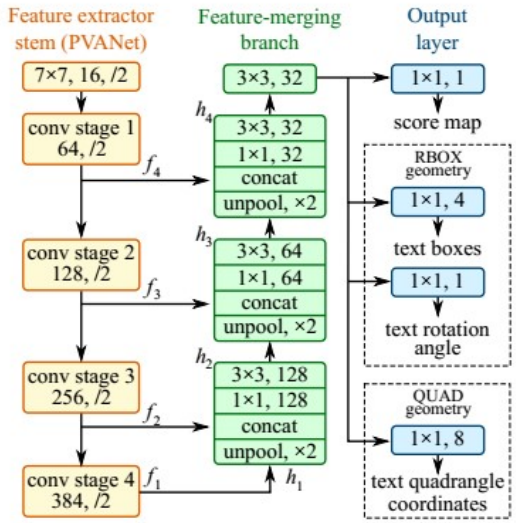
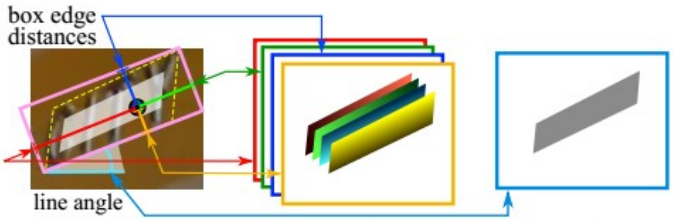


Figure 3. Structure of our text detection FCN.

所以从整体看来，EAST就是借助FCN架构直接回归出文本行的(x, y, w, h, θ)+置信度+四边形的四个坐标！非常简洁！但是看到这里或许会有个问题，为什么要生成四边形的四个坐标呢？(x, y, w, h, θ)这个参数不足以解决文本行定位问题？还真不能，看看下面这个图片。



对于这种带放射变换的文本行(可能还有的是透视变换)，呈现出来的形状是平行四边形（黄色虚线为ground true），如果我们以(x, y, w, h, θ)来表示这个文本的位置，就是粉色框所示，显然不合适。所以对于这种场合，直接预测四边形的四个顶点坐标才是正确之举。

EAST目标函数分两部分，如下，第一部分是分类误差，第二部分是几何误差，文中权衡重要性，λg=1。

$$L = L_s + \lambda_g L_g$$

Ls称为分类误差函数

，采用 class-balanced cross-entropy，这样做可以很实用的处理正负样本不均衡的问题。

$$\begin{aligned} L_s &= \text{balanced-xent}(\hat{\mathbf{Y}}, \mathbf{Y}^*) \\ &= -\beta \mathbf{Y}^* \log \hat{\mathbf{Y}} - (1-\beta)(1-\mathbf{Y}^*) \log(1-\hat{\mathbf{Y}}) \end{aligned}$$

其中β=反例样本数量/总样本数量 （balance factor）

$$\beta = 1 - \frac{\sum_{y^* \in \mathbf{Y}^*} y^*}{|\mathbf{Y}^*|}.$$

Lg为几何误差函数

$$L_g = L_{\text{AABB}} + \lambda_\theta L_\theta.$$

对于RBOX，采用IoU loss

$$L_{\text{AABB}} = -\log \text{IoU}(\hat{\mathbf{R}}, \mathbf{R}^*) = -\log \frac{|\hat{\mathbf{R}} \cap \mathbf{R}^*|}{|\hat{\mathbf{R}} \cup \mathbf{R}^*|}$$

角度误差则为：

$$L_\theta(\hat{\theta}, \theta^*) = 1 - \cos(\hat{\theta} - \theta^*).$$

对于QUAD采用smoothed L1 loss

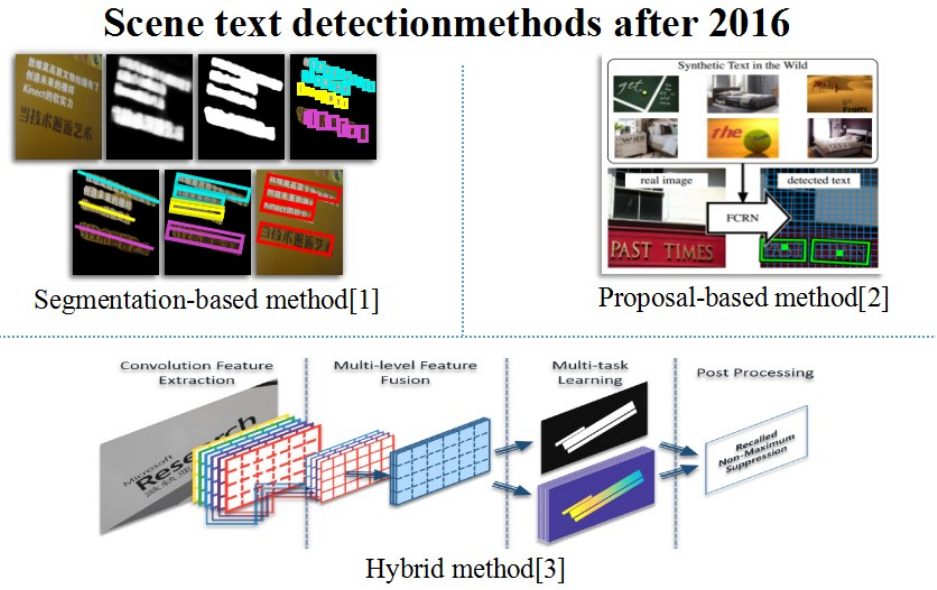
CQ={x1,y1,x2,y2,x3,y3,x4,y4}，NQ*指的是四边形最短边的长度

$$\begin{aligned} L_g &= L_{\text{QUAD}}(\hat{\mathbf{Q}}, \mathbf{Q}^*) \\ &= \min_{\tilde{\mathbf{Q}} \in P_{\mathbf{Q}^*}} \sum_{\substack{c_i \in C_{\mathbf{Q}}, \\ \tilde{c}_i \in C_{\tilde{\mathbf{Q}}}}} \frac{\text{smoothed}_{L1}(c_i - \tilde{c}_i)}{8 \times N_{\mathbf{Q}^*}} \end{aligned}$$

下面看看EAST文本检测的效果，注意观察一些带放射变换or透视变换的文本行的检测效果。

总结

文本介绍了我一直关注且实验效果都相当不错的三个经典的文本检测算法，他们都是基于深度学习，可以这么说，现在做文本检测都是深度学习的天下了。当然深度学习流派做文本检测远不止以上几个思路，比如基于语义分割的思路做文本检测的、基于角点检测做文本检测、各种方法混合的文本检测的论文也非常多，同时也取得非常不错的效果。可以说，现在基于深度学习的文本检测论文可谓百花齐放。



分类: OCR系列

好文要顶 关注我 收藏该文

Madcola
关注 - 30
粉丝 - 1334
+加关注

6 0

« 上一篇：如何免费使用谷歌搜索
» 下一篇：【OCR技术系列之六】文本检测CTPN的代码实现

posted @ 2018-10-12 10:18 Madcola 阅读(17752) 评论(5) 编辑 收藏

评论列表

#1楼 2018-11-09 17:34 **ParanoidY**

博主好，文章写的非常透彻，读下来有两个小疑问：
1. CTPN中先检测部分文本再合并的思路，小候选框的ground truth的设置原则是什么，如果随意设定其宽度的话，效果也优于整体检测吗，怎么理解这种分割的优势？
2. EAST用FCN的框架去回归效果好是因为它特征融合这一步比较好吗？好再哪里呢？不太理解这个地方。。
论文读的少，对深度学习理解有限，希望博主不吝赐教~

支持(0) 反对(0)

#2楼[楼主] 2018-11-10 10:02 **Madcola**

@ 这不是我这就是我
我的小小见解：
1. CTPN论文提到的小候选框的固定宽度为16，其实这个16是根据VGG16网络最后输出的一层的feature map来定的，因为前面已经进行了4次pooling，所以该层的feature map的一个像素对应原图16个像素。我觉得这是论文固定宽度设为16的原因。至于是否任意设定宽度对效果有影响，我觉得是有的，比如过小宽度，不能反映文字的信息，可能连人眼都没法确定该候选框是否有字符，何况模型。过大宽度，使得合并出来的候选框可能有过大的精度损失。

2.我觉得就是特征融合这部分的贡献大，对各个尺度的文本信息都学习到了。

支持(0) 反对(0)

#3楼 2019-02-28 05:26 **Aaron4Fun**

感谢作者的分享，请问一下我们用CTPN或者EAST进行文体提取之后，需要进行仿射变换后再输入进CRNN进行文本识别吗？如果是一些弧形的字体的话需要什么其他方法吗？谢谢

支持(0) 反对(0)

#4楼 2019-04-16 23:50 **征服天堂jj**

博主您好，请问下EAST中源码给的数据集是ICDAR2015，图片大小是固定的，我自己的数据集大小不是固定的，可以直接使用此网络吗，还有ICDAR数据集的标签是四个坐标8个参数，外加一个框中的文本，一个九个参数，最后一个参数我可以随便使用类似“text”去代替吗，或者是否可以不需要这个参数，希望不腻赐教，谢谢

支持(0) 反对(0)

#5楼 2019-05-10 15:08 **junboli**

写的很透彻

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

相关博文：
· 【OCR技术系列之六】文本检测CTPN的代码实现
· EAST自然场景文本检测
· FasterR-CNN:TowardsReal-TimeObjectDetectionwithRegionProposalNetworks论文理解
· 基于深度学习的目标检测技术演进：R-CNN、Fast R-CNN、Faster R-CNN
· 目标提取深度神经网络分析权衡 trade offs