# 论文记录

1 Shape Robust Text Detection with Progressive Scale Expansion Network

这篇文章介绍了Progressive Scale Expansion Network (PSENet)，目的是为了识别弯曲文字，提到了现有的一些方法，主要有两种，regression-based approaches 和segmentation-based approaches，他们在处理不同角度的文字效果还可以，但是对弯曲文字的处理效果一般。因此为了识别弯曲文字，这里提出了PSENet。

PSENet方法：1) starting from the kernels with minimal scales (instances can be distinguished in this step); 2) expanding their areas by involving more pixels in larger kernels gradually; 3) finishing until the complete text instances (the largest kernels) are explored. 大致就是从最小的kernel开始，然后慢慢扩大周围的pixel，直到识别出了完整的文字。

（但是我有比较疑问的，就是怎么解决文字的重叠问题以及同一行文字的隔断问题？重叠就不过多解释了，就是上下行的文字可能因为弯曲的原因导致了一定的重合；隔断问题就是，比如像i这个字分为两部分，怎么才能确认他们是同一行呢？或者如果文字是竖着排列呢？是有个想法啦，能不能用新的文字框边缘的平滑度来判断呢？）

这里提到了PSENet的两个优点，1是可以识别任意形状，2是不会上下行混在一起，具体的应该要看算法才能得知了。

2 Baek\_Character\_Region\_Awareness\_for\_Text\_Detection\_CVPR\_2019

这篇文章也是用来解决识别弯曲句子的。其中提到的方法为Character Region Awareness For Text detection(CRAFT)。

这篇文章的主要方法就是先在图像中找到每个字母，然后将字母连在一起形成句子。 producing the character region score and affinity score. The region score is used to localize individual characters in the image, and the affinity score is used to group each character into a single instance.

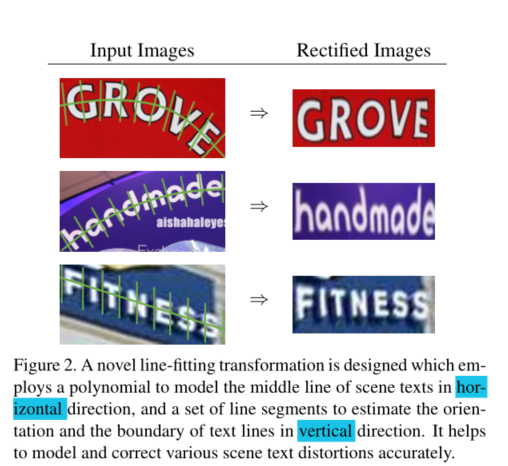
但是这里有个没懂的地方，他说现在的数据库没有提供字符层面的文字，所以用了一种方法解决，但是我没看懂那是什么方法。 we propose a weakly supervised learning framework that estimates character-level ground truths in existing real word-level data sets.

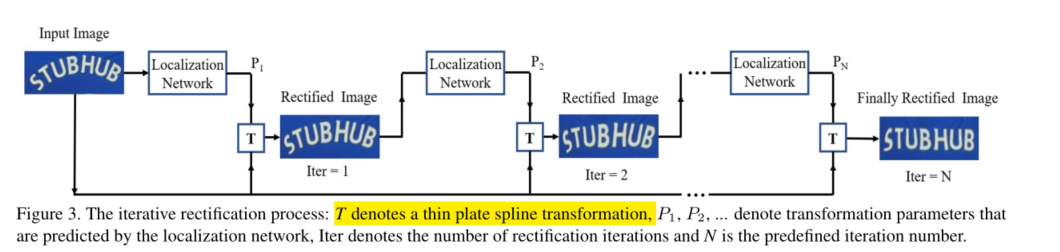
然后这篇文章在分析其他的方法时比较详细的提到了4种，相比上面多了两种。分别是1 Regression-based text detectors：using box regression；2 Segmentation-based text detectors： seek text regions at the pixel level；3 End-to-end text detectors：trains the detection and recognition modules simultaneously；4 Character-level text detectors。给出了各个方法的简单分析及优缺点。（没讲到第二种的缺点）

（这个方法没有过多的想法吧，我觉得这种方法其实更适合汉字，先找汉字再找汉字之间的联系，如果放到英文中使用这个方法的话觉得很奇怪啊，找到字母后要通过affinity score连成单词再连成句子，觉得效率堪忧？）

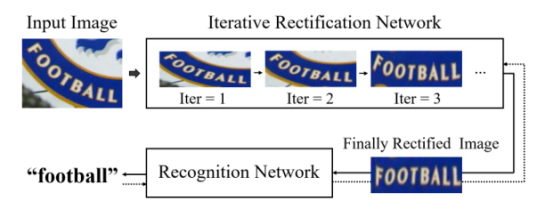
3 ESIR end to end scene text recognition via iterative image rectification

这篇文章的方法就比较有意思，是通过对图片进行扭曲处理，把原本弯曲的文字扭成在一条横线上， end-to-end trainable scene text recognition system (ESIR)。先用多项式去拟合文字的水平中线，然后用用竖直的线分割文字。employs a polynomial to model the middle line of scene texts in horizontal direction, and a set of line segments to estimate the orientation and the boundary of text lines in vertical direction.





然后这篇一直提到这是 iteratable的，可以利用迭代的方法去增加鲁棒性和准确性。可以看到下图有个反向的虚线，但是具体是个怎么运行的就不好说了，要用到RNN。

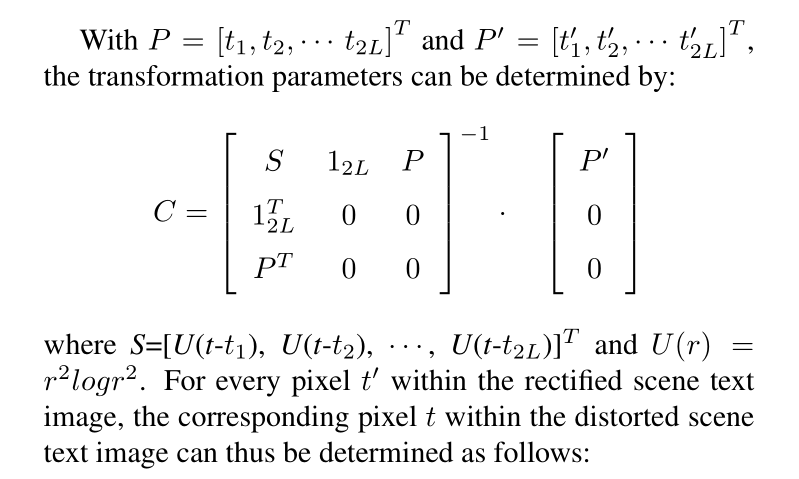


说到这个文章有3重贡献，First, it proposes a novel line-fitting transformation that is flexible and robust for scene text distortion modeling and correction. Second, it designs an iterative rectification framework that clearly improves the scene text rectification and recognition performance with no extra annotations. Third, it develops an end-to-end trainable system that is robust to parameter initialization and achieves superior scene text recognition performance across a number of public datasets. 感觉是废话吧。

在介绍其他的方法时，本文章把方法都分成了两类：bottom up approach和 top-down approach。

（感觉，这个方法太复杂了，工程量太大了而且很难做，感觉这个或许可以出来弯曲的句子，但是对于因为视角而变的扭曲的字母，这个rectification代价太大了，不仅需要rec，还需要拉伸以及压缩才能让文字变正常。再者，由于已经分了水平和垂直两部分，那是不是意味着没办法处理竖直的文字，那么延伸出来，是不是如果句子倾斜到了一定的角度，就会水平和竖直的处理就会出现误差？And既然是利用多项式来做的话，那如果出现环形的文字是不是大概就是无解了？）

Ok了，这个文章看完了。就是用多项式拟合然后分割再把图片进行修改（第一部分，也是关键部分），这个过程会有3L + K + 1个参数，其中K+1个是多项式的，3L是每个小分割的，分别是斜率、截距以及长度。



然后用这个公式算出C，算出C后就可以利用这个C来确定修改后的图片每个像素点的位置。这边提到了‘boundary effect’ problem，是说可能会丢弃某些在文字内的像素，然后解决办法是将得到的参数每次都直接用在原图片上，而不是用在上一次迭代的图上。

还有个问题是比较难训练，如果起始太随机的话可能会导致训练的时候收敛出问题，给出的办法是定义了一个额外的p0，p0和p’很相近，然后用P = P0 +△P去得到P，但是...P’是个啥你倒是说说啊...

突然想到的问题，如果这个把像素点重新排列了，那没像素点的地方咋办？这个是boundary effect’的一部分嘛？要咋整？从原图上扣像素点填充过来吗？

But，这玩意不开源...mad

正在重看这篇文章，看的时候多注意这几个问题吧，这里时怎么做到多次回归的？正常来说，如果已经得到了拟合的多项式，应该一次rectification就够了吧，像aster也是找到边框点之后直接rectify了一次，那这里为什么需要多次？一次和多次的区别在哪？一次难道就没办法完成需要的任务吗？为什么一次不能完成所需要的任务？如果的确一次不能完成需要的任务的话，那为什么多次可以？既然已经决定用多次的话，如何判断某一次是否已经结束还是要继续迭代？需要用什么来引导迭代？以及关于转换的问题，在转换过程中，原图和中间图分别扮演什么角色？说是通过中间图获得parameter，然后利用原图来rectify，但是本文并没有给出具体是怎么rectify的，这可能还需要查资料。拟合出来的多项式到底有什么用呢？在后面似乎就没出现过多项式了..而且最后也没有去考虑多项式该怎么取...再者，要怎么监督这个卷积呢，怎么优化呢？

为什么需要多次回归。在这篇文章里没讲，但是它放出来了用aster来rectify出来的结果的对照。那可以就这么想吧，因为客观原因，可能是薄板插值的固有属性，导致每次修改出来的图片没办法做到和原图相差太远吧，没有数学上的严格推理我好特么难受啊。那这样的话，只能默认多次就是比一次好咯whatever



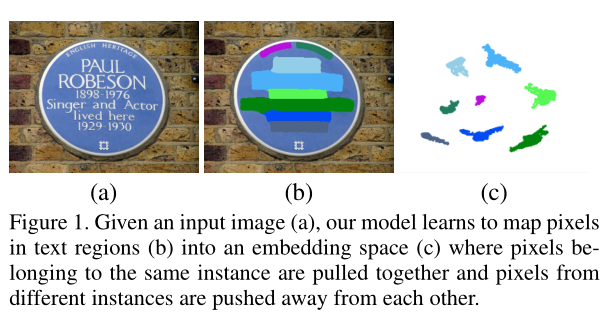
如何判断某一次迭代是否继续？Emmm这里根本就不用判断...这边是固定的迭代次数..所以也没有什么是需要引导迭代的，就相同的网络，不停地用就好了。那既然能用迭代的办法，主要是因为这是一个通过同一个模型的处理多次就可以得到改进，那再attention上有没有这种模型呢？为什么这个rectify问题有呢，主要是因为在最开始找点的时候已经确定了最终完美的状态，也就是变成直的。但是attention有没有固定可知的最终状态呢？感觉..没有的样子诶...毕竟每个不同图片的字数是不一样的。

然后我突然又想到，能不能利用GT的字符信息，卷出来字符的个数呢？但是卷出来字符个数的话，不就跟attention的作用一样了吗...但是如果能捐出来字符个数N的话，那相当于最终的结果也是知道的，就只要把输入图片N等分，然后每个区域的中心基本上就是attention的中心。那有什么办法能够比较准确的卷出来字符个数吗？字符级别的？PSENET？然后又出现了一个问题，就是空格的问题，以及有时候并不一定平均分割是好的。那如果用max pooling来找字符，能不能用min pooling来找空格？但是空格是没有GT标注的相当于就没办法训练..而且N等分可能不太行，因为边上很可能会留白。这个均分的只能先无视了吧，因为这个字符大小也会有影响，比如某一个字符占了两个位置，后两个字符占了一个位置，这样还是没办法解决问题。

所以，怎么样得到最终的完美情况呢？能不能确立最终结果是个20个字符空格，然后去填充呢？那到底要填充几个还是不确定啊。那能不能就让attention在整个图像上滑动，每划一个距离算一次在这个attention点时卷出的各个字符的概率，然后设定阈值，比如0.6，超过这个阈值后继续往右划，直到再次超过阈值，把attention的中心点再放在这里？或者能不能做一个位置和概率的坐标图？然后取每个的波峰处作为某个字符的位置？或者说就是attention的中心点？似乎不好优化？和LSTM的区别在哪呢..emmm是和CTC比较像...

4 learning shape aware embedding for scene text detection

这篇其实没太懂。 our method maps pixels onto an embedding space where pixels belonging to the same text are encouraged to appear closer to each other and vise versa. 大概就是把每个句子都放到一个中间空间中，然后在同一个句子间的像素点距离更近。但是我很奇怪，这个论文的目的不就是detection吗？如果已经能判断出来某些像素点在同一个句子的话那不就意为着句子已经被detect了？



然后这里提到了一个 Shape-Aware Loss，can adaptively adjust pulling and pushing force on the embedding feature based on scales and adjacency of text instances.文章倒是对这个SA Loss 很满意，或许之后看看具体叭。

Utilizes information from both embedding space and segmentation space generated by two parallel branches from our proposed network.这句话也没太理解，所以是两个space都用？后面那个space咋产生的？真的不太懂，之后问问吧。

5 Robust\_Curve\_Text\_Detection\_With\_Conditional\_Spatial\_Expansion

这个和第一篇论文很类似，Conditional Spatial Expansion (CSE) mechanism，也是先找一个小点，然后扩大。 starts with a seed arbitrarily initialized within a text region and progressively merges neighborhood regions based on the extracted local features by a CNN and contextual information of merged regions。好像有点不太一样的是第一篇只是找初始点旁边的像素点，然后分析合并，这个是根据图像的特点来扩展的，因此在文中写到了，对于具有类似形状的句子，能够更快更好的识别，extremely discriminative especially when texts are close to each other。

这边提到了文章的几个贡献，一个是变得更简单，一个是说可以无缝和其他existing object detection workflows结合，还有就是说自己在识别的时候初始点不会太大的影响结果。

（看了这些后觉得..不必要看relate work..都大同小异...都是在嘴炮自己的多好多好，别人的多差多差..这大概就是所谓学究吧...）

6 look more than once an accurate detector for text of arbitrary shapes

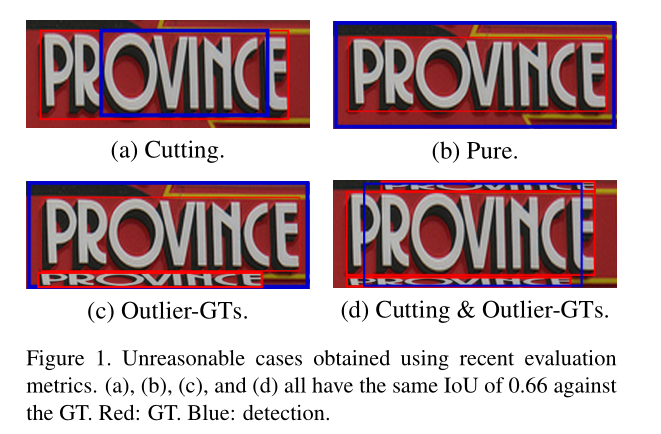
这篇文章介绍了text detector namely LOMO, which localizes the text progressively for multiple times (or in other word, LOok More than Once)。貌似出发点是说别的detector都是试图看一次就知道，不符合人类的视觉系统，因为对于长文章人类一次也只能看到一部分。（The mainstream methods force the detectors to localize text of different length with only once perception, which is contrary to human visual system）而LOMO就可以像人类视觉一样多看几次来确定...

LOMO分为了3个部分，direct regressor (DR)， iterative refinement module (IRM)，shape expression module (SEM)，DR是像其他大多数的方法一样产生一个四边形文字框，DR generates text proposals firstly, then IRM refines the quadrangle proposals. Relying on position attention mechanism, IRM can be aware of these locations and refine the input proposal closer to the entire annotation。这里position attention mechanism好像在其他文章的related work里面看到过，之后可以找找看。而对于不规则形状的文字，SEM regresses the geometry attributes of text instances, i.e., text region, text center line and corresponding border offsets. can reconstruct a more precise polygon expression也就是利用这些特征，可以得到文字的具体形状。

（大概看了个一知半解吧，不是很明白中间的IRM到底有什么必要..）

7 Tightness-aware Evaluation Protocol for Scene Text Detection

这篇不同于上面的文章，这不是一个将方法的文章，而是设定标准的文章。文章中提到了现有的判断text detection好坏的模型存在着一些问题，然后提出来自己的标准。它提出的现有文章的问题，1) They are not goal-oriented; 2)they cannot recognize the tightness of detection methods;3)existing one-to-many and many-to-one solutions involve inherent loopholes and deficiencies.所以，this paper proposes a novel evaluation protocol called Tightness-aware Intersect-over-Union (TIoU) metric that could quantify completeness of ground truth, compactness of detection,and tightness of matching degree.



这4个图片的ToU是一样的，但明显效果是不同的。a不完全，b范围略大但是还算清楚，c有噪音，d为a和c的结合。然后文章给出了导致这种结果的原因， 1)The detections in Figs. 1 (a) and (d) cut the GT region; 2)the detections in Figs. 1 (c) and (d) both contain outlier-GT; and 3) recent metrics use binary results (0 or 100%) to represent the final recall or precision score.

为了解决， use the occupation ratio of detection to GT and occupation ratio of outlier-GT to detection as penalty factors。然后就提到说这篇文章的方法有3个好处，Completeness， Compactness，Tightness-aware。具体的标准以及之前的标准要具体看文章了。

（我这边的问题是，之前的方法真的有这么明显的问题吗？如果有的话为什么还会一直沿用呢？它这里给的4张图片，a b分数一样似乎还是可以能够理解的，但是c为什么？是不是因为下面那行小字和大字是一样的？如果不一样的话，ToU会不会就会下降的很厉害呢？毕竟两行一样的字这样的极端情况也不会出现太多叭）

8 Arbitrary\_Shape\_Scene\_Text\_Detection\_With\_Adaptive\_Text\_Region

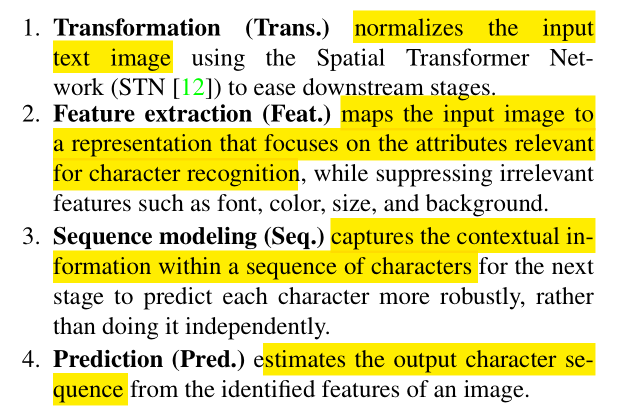
（这篇看完最直观的就是，废话好多，当然可能是因为我只看了前半部分吧，然后其实也没有特别理解这个方法..）

a text region proposal network (Text-RPN) is first used for obtaining text proposals. Then, text proposals are verified and refined with a refinement network, three branches including text/nontext classification, bounding box refinement and recurrent neural network (RNN) based adaptive text region representation.In the RNN, a pair of boundary points are predicted each time step until the stop label is predicted.

大致就是先找文本框，然后利用RNN去预测边界点，直到预测到了结束标签。

9 What is wrong with scene text recognition model

这篇文章是为数不多的把所有的都看完的文章，这篇文章是把比较经典的模型全都整合在了一起，然后把他们分成了4个阶段：



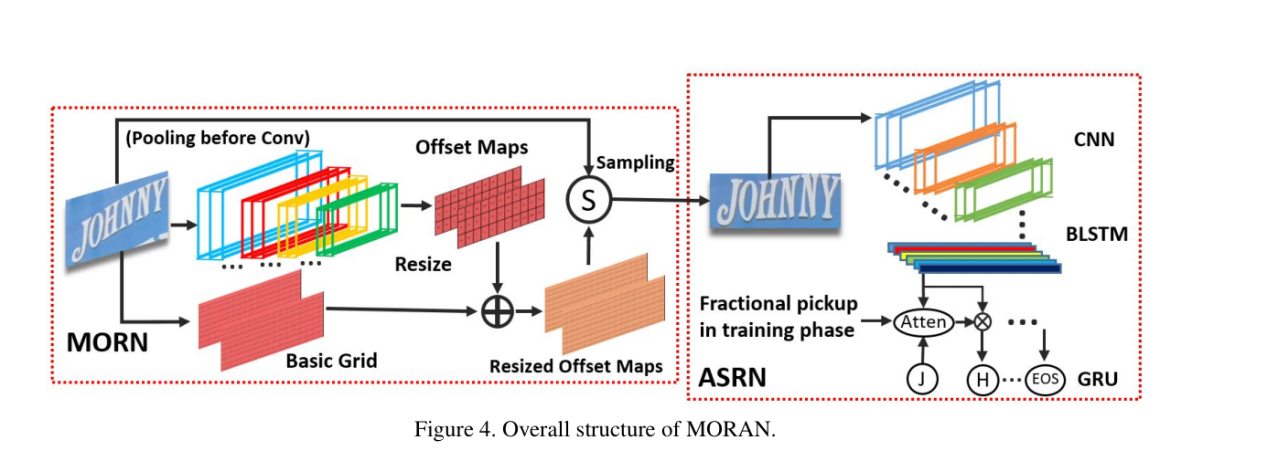
第一阶段是处理图像用的，这里面就一个model即Spatial Transformer Network（STN）。

11 MORAN A Multi-Object Rectified Attention Network

这篇文章和第3个差不多的样子，也是先把图片调整一下，然后再做文字识别。但是看图片的修改效果的话感觉是不如第3篇文章的。

它的multi-object rectified attention network (MORAN)的方法实际上分为两部分，a multi-object rectification network and an attention-based sequence recognition network或者叫MORN和ASRN。

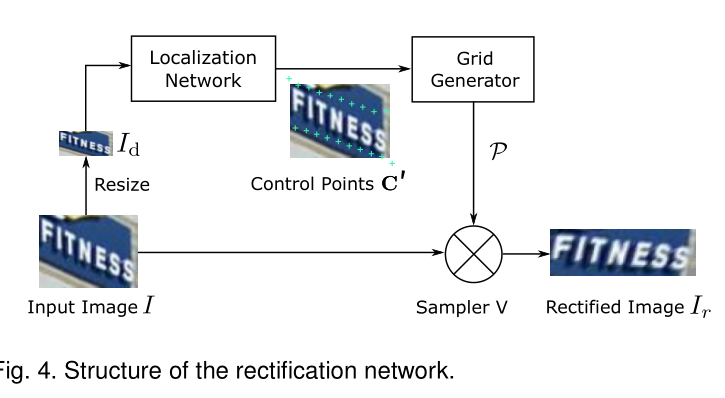
对于MORN，它先提到了现有的affine transformation network和deformable convolutional network的不足，（这两个明天也要查一下），然后给出了自己的卷积网络，感觉也没什么特殊的，但是最后变成了2×3×11，两个channel，然后就相当于把图片划分成了33个区域，然后两个channel分别是x 和y的偏移量，得到偏移量后根据basic的网格就可以得到变化后的图片。根据它的图片显示，这个操作会将一些用不到的背景给剔除掉，也是蛮有意思是的。



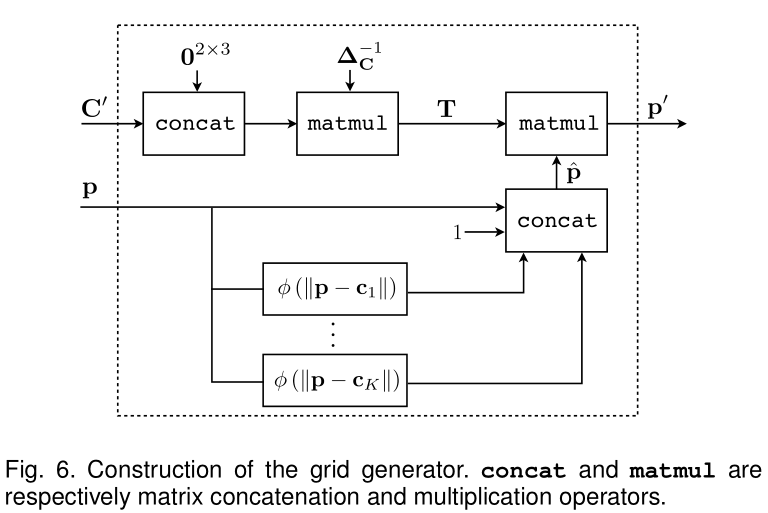
后面ASRN，和CRNN其实还是有点像，但是加入了注意力机制，然后说了一堆没懂的东西，明天好好看看注意力机制再回来。

12 ASTER\_Attentional\_Scene\_Text\_Recognizer

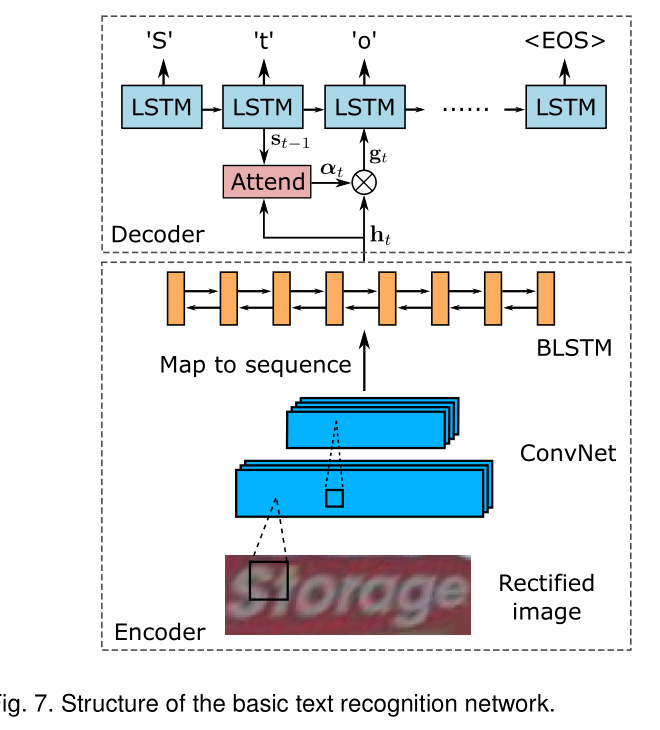
这是我看的第三篇先修改图像然后再进行recognition的了。这篇文章转换图片的方法是先找到图片上下的控制点，然后再根据控制点来修改图片。第3篇是找拟合多项式，第11篇是把整个图片卷积后找每块的偏移量。还是有点区别的。



找控制点的Localization Network：要得到K个控制点，在I上下采样得到Id，然后用Id回归出控制点C，（这里我很奇怪为啥回归出来的是C不是C’...讲道理应该是C’，C是自己设置的）经过网络回归后要reshape成为R2XK。然后通过Grid Generator一通变换，就可以得到每个像素点的对应了。然后sampler也用了比较特殊的方法以防止像素出问题，不过在这里没仔细讲，大概得看看引用的论文。所以看这的图的话应该是C’要比T先得到，但是C’怎么得到的呢？



然后在文字识别部分，先是通过crnn得到序列，再用注意力机制和LSTM来得到每个字符，最后用的LSTM是单向的，但是它这边选择了从左向右和从右向左各得到了一个序列，然后选择哪个识别分数比较高的。这里的attention体现在gt的生成过程，a是通过e用softmax函数得到的，怎么解释这个？算是隐藏层中加入了attention weight吗？大概就是这个样子吧。



13 Focusing Attention Towards Accurate Text Recognition in Natural Images

这篇文章不再是关于图像转换的了，而着重讲的是注意力机制和注意力机制的调整。这边的 FAN (the abbreviation of Focusing Attention Network）是分成了两个部分，an attention network (AN)和 a focusing network (FN)，简单说即使AN就是普通的attention机制的网络，然后FN是对他的调整。

FN对于AN的每个输入，都算出其在原图上的 receptive field ，然后算出这个field 的中点，就是注意力中心，然后把那块割下来后算一个 energy distribution，然后算一个 probability distribution，再用 focusing loss function去评估即可。虽然其实不是很懂这是啥原理...

读完17后又回来读了一遍这个文章，感觉attention大都是跟softmax有关，这篇文章虽然是没有提到softmax，但是它算概率分布的那个公式明明就是softmax，所以attention的本质大概就是用到softmax函数吧，然后这篇和17对处理attention的准确度的方法都差不多啊，都是看目前的attention中点和GT的中点差的远不远，理论上attention应该就是得用这个方法，但是每个方法获得当前attention中点的方法不一样。或许这上面可以做文章？然后对于中心和GT的差距，有没有其他的loss算法呢？

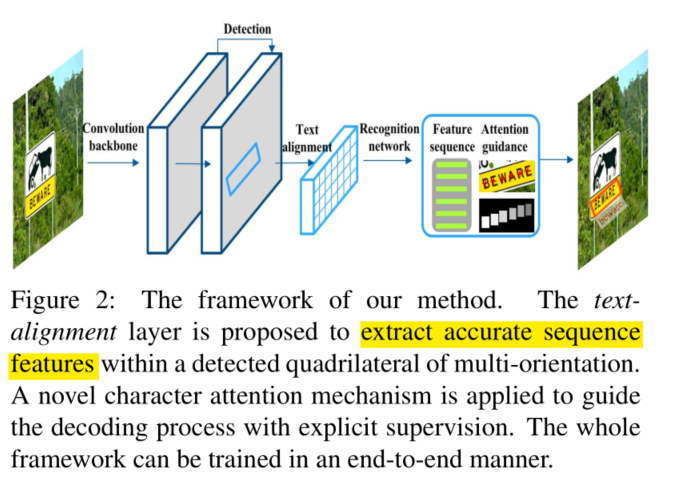
现在的源码情况：

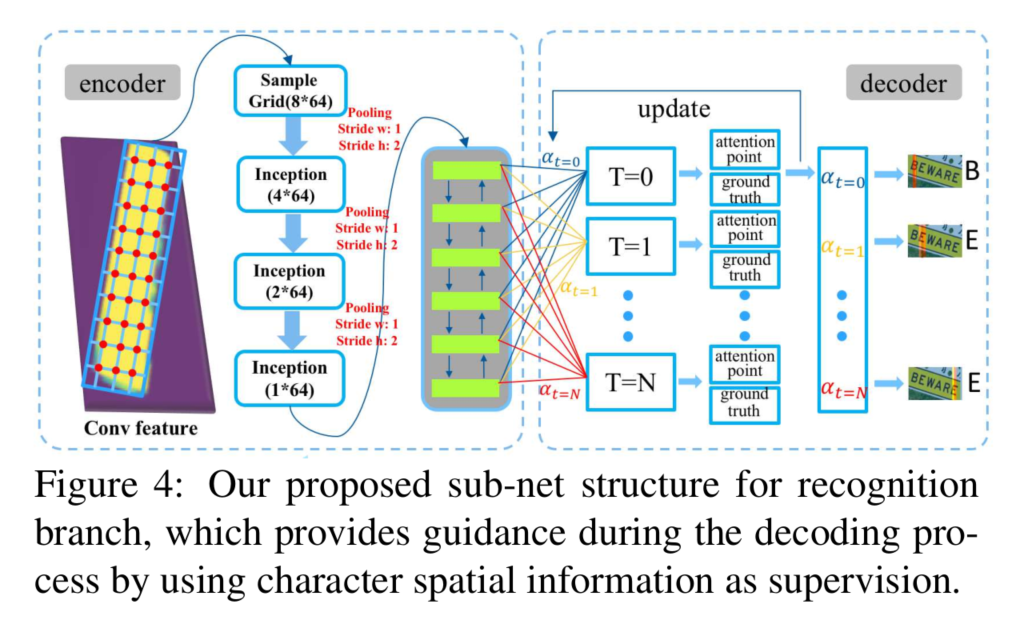
1. 论文9，把大部分的常用策略都写出来的一个整合的代码
2. 论文11，MORAN，有源码
3. 论文12，ASTER，但是是TensorFlow的代码
4. 论文13，FAN，其中的FN有可借鉴性，但是无源码
5. 论文3，ESIR，比较想仿照的，但是无源码

17 He\_An\_End-to-End\_TextSpotter\_CVPR\_2018\_paper

其实我在看这篇文章的时候，觉得text detection和recognition应该是不能分开单独做的，在做detect的时候肯定会用到一些信息，那些信息对后来的识别也一定会有帮助，如果分开做，就相当于前面的检测部分的信息都直接扔了，然后再开启新的问题，这肯定是不好的。

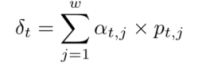
应该我觉得能够最大程度上发挥文字这部分的办法应该是，先检测把字的部分检测出来，然后能不能用什么办法（比如分类？），然后把文字和背景分离出来分别着色，然后再进行识别，然后如果可以的话识别的时候加上语言模型，这应该就是最大化正确率的方法。可问题是，中间那个分离背景似乎是有点难的，虽然的确是可以用到检测中得到的信息，但是似乎还是比较难实现？如果都能把背景分辨出来了，那岂不是文字自然就出来了？





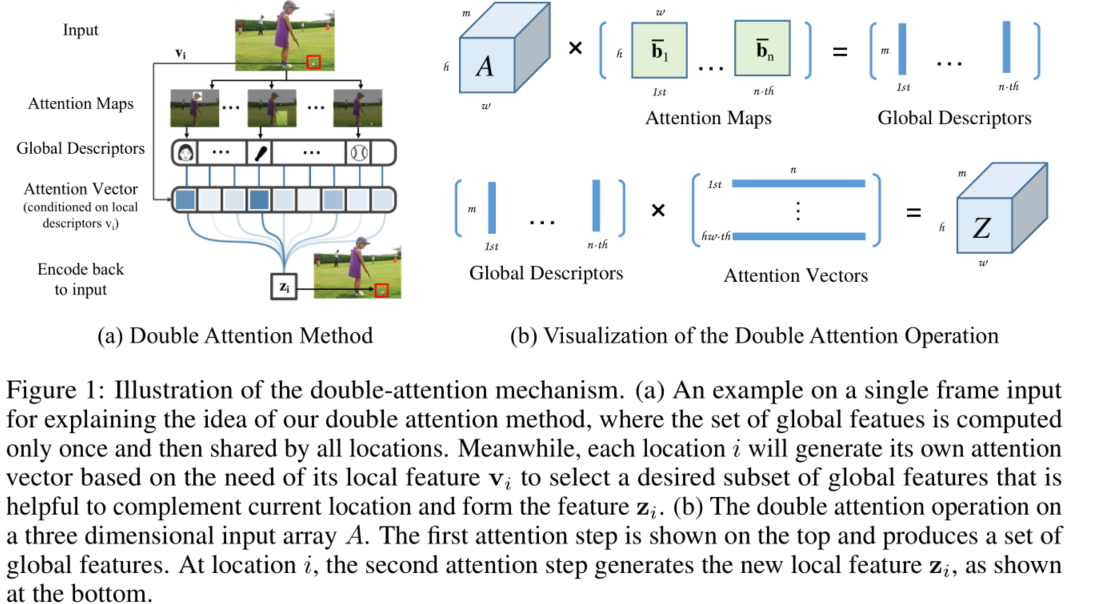
在进入下一篇文章前还是要把这篇文章给归纳总结一下。付博是让我注意一下这里的attention，但是感觉..没啥太大的区别的样子啊...这篇论文可能还是得先把PVANet以及MASKRCNN看了才能看懂吧。

这边在详细介绍的时候，说的Character mask是啥玩意啊真的不太懂，然后说到的lmask又是个啥...

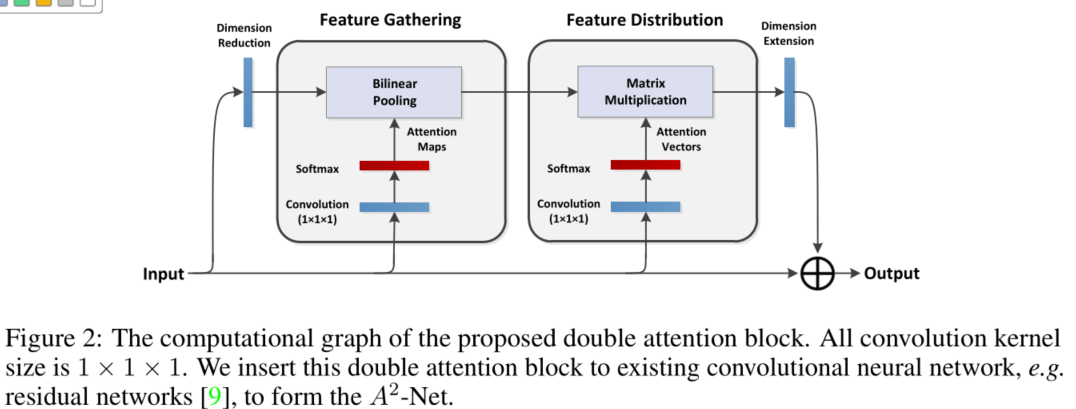
这篇文章对attention的处理是加了一个额外的align loss，用这个算出来目前的attention的中心点在哪里，然后用align loss去量化这个中心点和GT的中心点的距离。似乎是和前面那个增加了一个focusing net的做法类似。但是这篇文章不仅如此吧，不只有识别部分，也有检测的部分，而且最后算的总loss是和前面的检测的loss一起的。这篇文章得仔细看看。

15 A2 -Nets Double Attention Networks

这边提到两层attention，第一层从完整的空间中，利用second-order attention pooling来获取特征并放入一个压缩集。第二层attention是选择性的挑取每个location的特征。



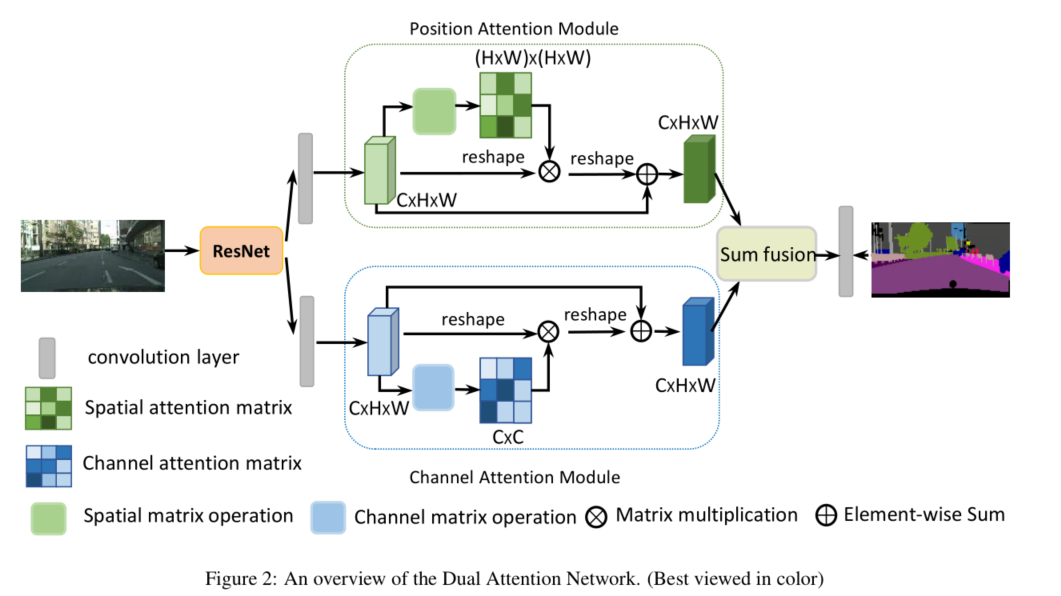
这里提到了一个bilinear pooling，说是能够得到second-order statistics，而传统的max pooling和average pooling就只能得到 first-order statistics，什么意思？

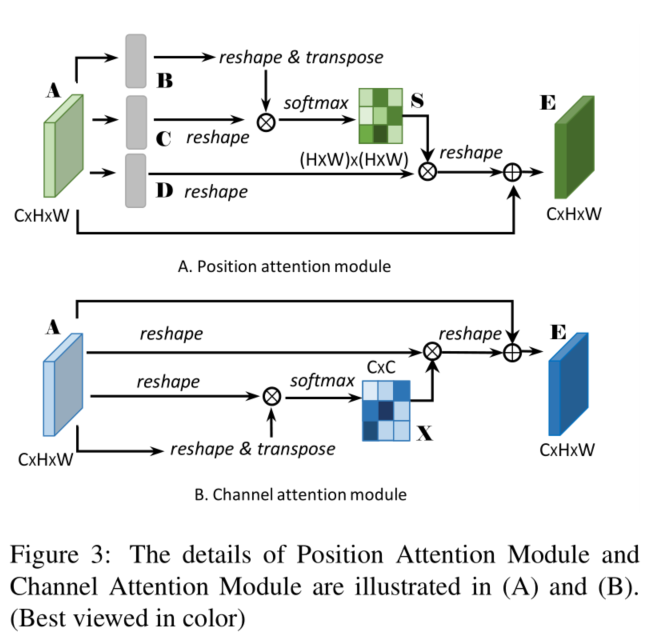


看了一堆公式，但是似乎还是不太懂这种两层的attention是几个意思。第一层attention，是用两个卷积分别卷出两个不同的feature map，然后两个矩阵相乘，然后做了一个变换吧bi变成了bi-，然后用A乘上经过softmax的B，（B经过softmax后每一列的和都是0），得到gi。以图上的注释来说的话，经过softmax的B-应该就是attention map，而且既然是attention map，那通过的卷积应该总会是比较特别的吧？然后第二层attention，就是用刚刚得到的global descriptions和attention vectors相乘，然后就得到最终结果了，这个attention vectors也是先通过一层卷积，然后再通过softmax函数，但是还是和刚刚同样的问题，作为attention vector的东西，难道不需要特殊的卷积吗..大概就是不需要吧。

这个就相当于是串联的attention了，但是这么用attention的意义在哪里啊？

14 Dual Attention Network for Scene Segmentation





在这里的图中，C的转置和B相乘，然后通过softmax函数，得到的结果中的sji为什么表示 the i th position’s impact on j th position？或者是所有的类似的矩阵乘法以及经过softmax后都是这样的含义吗？

然后这边有个问题就是，没有loss的情况下这两个attention要怎么学习啊..？

所以从这两个图里已经可以比较清楚地明白这里的dual attention是干嘛的了，这边应该就是付博说的并联的attention，第一个attention得到的是一个N\*N的attention map，是关于position的，第二个是C\*C的，关于channel的，然后通过这两个并行的attention进行sum fusion就可以得到最终的结果。但是因为没做过分割，其实不知道最终输出的结果是个啥。

那这种dual attention要怎么用在文字上呢？想想文字一般都有几个维度需要知道呢？其实就以这篇来说，我也不是很懂为什么要分为channel和position两个attention，总共是4维的输入，按理说可以划分成4个维度？那么，对于文字识别来说的话，有什么是值得分成两个attention的吗？我是之前一直在想，可不可以单独拿出来一个卷积层用来得到文字的中心线呢？不过不知道和最开始那篇的差别大不大，待会回去再看看。Attention的本质是什么呢？

我还是再回到上一篇看看串行的attention吧