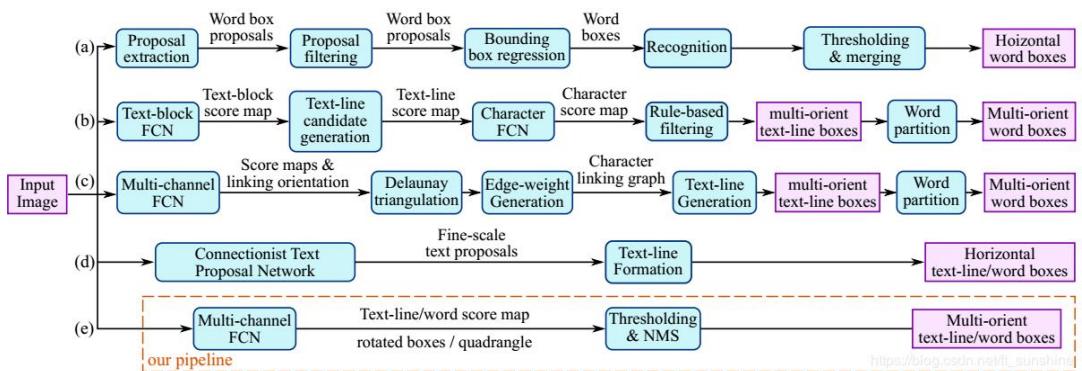


1. 现有工作

1.1 现有总结



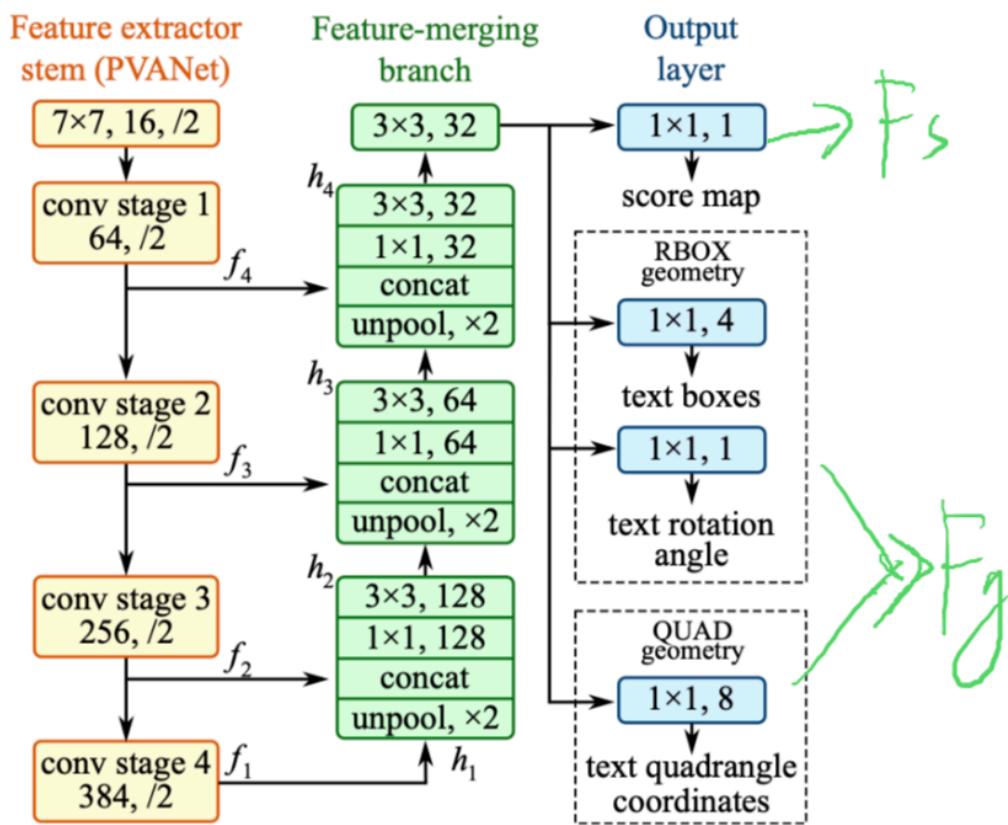
1.2 CTPN存在问题

在水平文本的检测方面效果比较好，但是对于竖直文本或者倾斜的文本，该方法的检测就很差



2.EAST模型介绍

基于FCN的模型EAST，该模型只需要两阶段(全卷积网络和NMS)，就可以完成场景文本检测的任务，全程端到端结构；



图一

EAST的网络结构总共包含三个部分：

- feature extractor stem (特征提取分支)
- feature-merging branch (特征合并分支)
- output layer (输出层)

2.1 feature extractor stem (特征提取分支)

在特征提取分支部分，主要由四层卷积层组成，可以是一些预训练好的卷积层，作者采用的是VGG16中pooling-2到pooling-5每一层得到的feature map。

记每一层卷积层卷积后得到feature map为，如图1所示，从上到下唉，每一层feature map对应的尺度刚好为输入图像的

$$\frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4} \quad .$$

2.2 feature-merging branch (特征合并分支)

在特征合并分支部分，其实作者借鉴了U-net的思想，只是U-net采用的是反卷积的操作，而这里采用的是反池化的操作，具体的计算大致如下，对于一个，首先经过一层反池化操作，得到与上一层卷积feature map同样大小的特征，然后将其与进行拼接，拼接后再依次进入一层和的卷积层，以减少拼接后通道数的增加，得到对应的，在特征合并分支的最后一层，是一层的卷积层，卷积后得到的feature map最终直接进入输出层。具体的计算公式如下：

$$g_i = \begin{cases} \text{unpool}(h_i) & \text{if } i \leq 3 \\ \text{conv}_{3 \times 3}(h_i) & \text{if } i = 4 \end{cases}$$

$$h_i = \begin{cases} f_i & \text{if } i = 1 \\ \text{conv}_{3 \times 3}(\text{conv}_{1 \times 1}([g_{i-1}; f_i])) & \text{otherwise} \end{cases}$$

2.3 output layer (输出层)

在输出层部分，主要有两部分，一部分是用单个通道的1x1卷积得到score map（分数图），记为Fs，另一部分是多个通道的1x1卷积得到geometry map（几何形状图），记为Fg，在这一部分，几何形状可以是RBOX（旋转盒子）或者QUAD（四边形）。对于RBOX，主要有5个通道，其中四个通道表示每一个像素点与文本线上、右、下、左边界距离（axis-aligned bounding box，AABB），记为R，另一个通道表示该四边形的旋转角度θ。对于QUAD，则采用四边形四个顶点的坐标表示，每个点的坐标为，因此，总共有8个通道。关于RBOX和QUAD的表示可以见表1：

Geometry	channels	description
AABB	4	$\mathbf{G} = \mathbf{R} = \{d_i i \in \{1, 2, 3, 4\}\}$
RBOX	5	$\mathbf{G} = \{\mathbf{R}, \theta\}$
QUAD	8	$\mathbf{G} = \mathbf{Q} = \{(\Delta x_i, \Delta y_i) i \in \{1, 2, 3, 4\}\}$

Table 1. Output geometry design

表一

再来说一下输出层的几个特征图(score map + geometry map)：

- 对于检测形状为RBOX，则输出包含文本得分(score map)和文本形状(AABB boundingbox和rotate angle)，也就是一起有6个输出，这里AABB分别表示相对于top, right, bottom, left边的偏移。
- 对于检测形状为QUAD，则输出包含文本得分(score map)和文本形状

(8个相对于corner vertices的偏移), 也就是一起有9个输出, 其中QUAD有8个, 分别为 $(\Delta x_i, \Delta y_i), i \in [1, 2, 3, 4]$ 。

注: RBOX和QUAD二者选一, 一般RBOX的效果会略好。

3. 真实标签生成

3.1 score map

对于score map, 不失一般性的, 这里考虑QUAD的情况, 在RBOX也似类似的标签生成方式。EAST对真实标签的四边形区域会进行放缩, 放缩的方式如下:

首先, 记四边形

$$Q = \{p_i | i \in \{1, 2, 3, 4\}\}$$

其中, p_i 表示四边形顺时针方向的四个顶点, 然后计算每个顶点 p_i 的参考长度 r_i , 说简单一点, 其实就是计算每个顶点相邻两条边的最短边的长度, 其计算公式如下:

$$r_i = \min(D(p_i, p_{(i \bmod 4)+1}), D(p_i, p_{((i+2) \bmod 4)+1}))$$

其中, $D(p_i, p_j)$ 表示 p_i 和 p_j 的欧式距离

接着, 对于四边形每一对对边, 将两条边的长度与他们的均值进行对比, 以确定出哪对对边是长边, 然后对两条长边优先进行放缩, 放缩的方式是对每个顶点沿着边向内部分别移动 $0.3r_i$, 如图2(a):

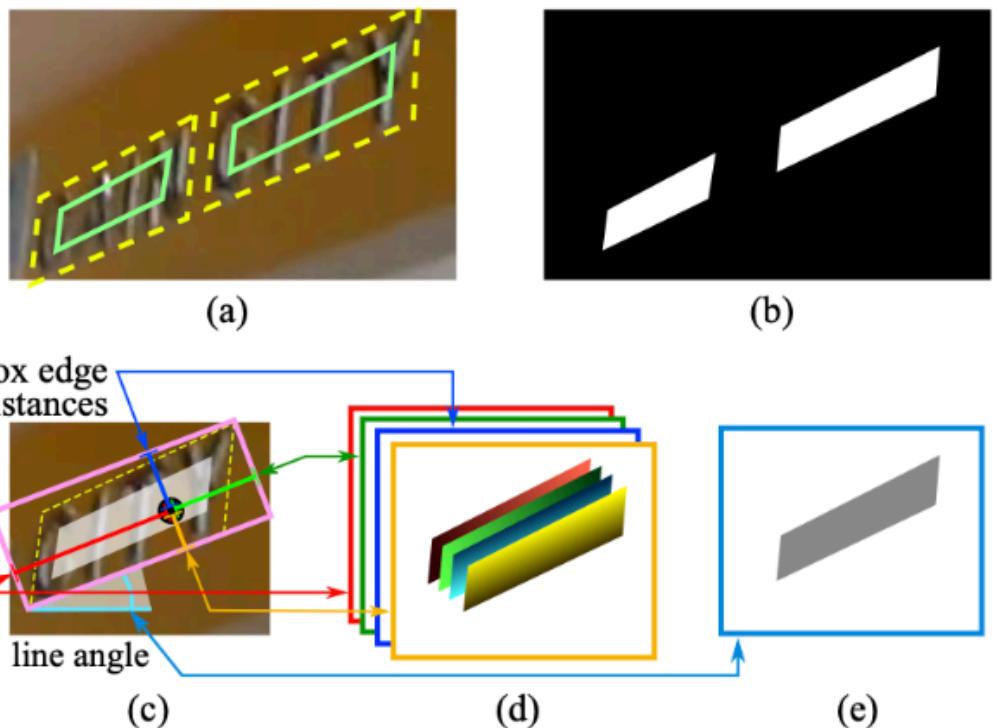


Figure 4. Label generation process: (a) Text quadrangle (yellow dashed) and the shrunk quadrangle (green solid); (b) Text score map; (c) RBOX geometry map generation; (d) 4 channels of distances of each pixel to rectangle boundaries; (e) Rotation angle.

图2

3.2 geometry map

对于geometry map，由前面我们知道有两种类型，分别是QUAD和RBOX，对于score map为正例的像素点，其QUAD对应的标签直接是他们与四个顶点的偏移坐标，即顶点的差值，而对于RBOX，则首先会选择一个最小的矩形框住真实的四边形，然后计算每个正例像素点与该矩形四条边界的距离。具体的如图2(c)-(e)所示。

4. 损失函数

由于输出层有两个分支，因此对应着两个损失函数，可以表达如下：

$$L = L_s + \lambda_g L_g$$

其中：

Ls和**Lg**: 分别表示score map和geometry map的损失函数

λ_g

表示权重，在论文中作者设置为1。

4.1 Ls

对于Ls,为了解决类别不平衡的问题，作者引入了平衡交叉熵损失函数，其表达形式如下：

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

其中， $\hat{Y} = F_s$ 是预测出来的分数， Y^* 是真实的标签， β 是每一张图像中负例的占比，其计算公式如下：

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

4.2 Lg

4.2.1 RBOX

当geometry map采用的是RBOX时，对于RBOX中的AABB，作者采用的是IoU损失函数，其表达形式如下：

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

其中， $\hat{\mathbf{R}}$ 表示预测到的矩形， \mathbf{R}^* 表示真实的矩形， $|\hat{\mathbf{R}} \cap \mathbf{R}^*|$ 表示两个矩形的重叠面积，其对应的宽和高计算方式如下：

$$\begin{aligned} w_1 &= \min(\hat{d}_2, d_2^*) + \min(\hat{d}_4, d_4^*) \\ h_1 &= \min(\hat{d}_1, d_1^*) + \min(\hat{d}_3, d_3^*) \end{aligned}$$

其中， d_1, d_2, d_3 and d_4 分别代表一个像素点到矩形上、右、下、左边界的距离， $|\hat{\mathbf{R}} \cup \mathbf{R}^*|$ 表示两个矩形的总区域，其计算公式如下：

$$|\hat{\mathbf{R}} \cup \mathbf{R}^*| = |\hat{\mathbf{R}}| + |\mathbf{R}^*| - |\hat{\mathbf{R}} \cap \mathbf{R}^*|$$

由于RBOX还有一个通道是表示旋转角度，因此，对于角度的损失函数计算如下：

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

其中， $\hat{\theta}$ 表示预测到的角度， θ^* 是真实的角度，最后，RBOX的损失函数如下：

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

其中， $\lambda\theta$ 表示权重，作者在实验时取的是10。

4.2.2 QUAD

此时损失函数的计算方式与RBOX不一样，作者采用的是smoothed-L1损失函数。记一个四边形Q对应的坐标集合为：

$$\mathcal{C}_Q = \{x_1, y_1, x_2, y_2, \dots, x_4, y_4\}$$

则QUAD对应的损失函数如下：

$$L_g = L_{\text{QUAD}}(\hat{\mathbf{Q}}, \mathbf{Q}^*) = \min_{\hat{\mathbf{Q}} \in P_{\mathbf{Q}^*}} \sum_{c_i \in \mathcal{C}_Q} \frac{\text{smoothed}_{L1}(c_i - \tilde{c}_i)}{8 \times N_{\mathbf{Q}^*}}$$

其中， $N_{\mathbf{Q}^*} = \min_{i=1}^4 D(p_i, p_{(i \bmod 4) + 1})$ ，表示每个四边形的最小边长，而 P_Q 是与 Q^* 等价的四边形集合，唯一的不同就是 P_Q 是经过排序，因为原始数据中， Q^* 的标注是无序的。

5.局部感知NMS

当预测结束后，需要对文本线进行构造，为了提高构造的速度，作者提出了一种局部感知NMS算法，其基本思想是假设相邻的像素点之间是高度相关的，然后按行逐渐合并几何形状，当相邻两个几何形状满足合并条件（这里的合并条件作者没有具体讲清楚）时，对他们的坐标按照分数进行加权，其计算公式如下：

$$a = \text{WEIGHTEDMERGE}(g, p)$$

$$a_i = V(g)g_i + V(p)p_i$$

$$V(a) = V(g) + V(p)$$

其中， g 、 p 分别表示两个满足合并的几何形状， $V(g)$ 、 $V(p)$ 分别表示他们的分数，分别对应第*i*个坐标， a_i 、 $V(a)$ 分别对应合并后的坐标和分数，当合并完成后，会将合并后的几何形状作为一个整体继续合并下去，直到不满足合并条件，将此时合并后的几何形状作为一个文本线保存到S当中，重复该过程，直到所有的几何形状都遍历一遍为止。具体的算法过程如下图：

Algorithm 1 Locality-Aware NMS

```
1: function NMSLOCALITY(geometries)
2:    $S \leftarrow \emptyset$ ,  $p \leftarrow \emptyset$ 
3:   for  $g \in \text{geometries}$  in row first order do
4:     if  $p \neq \emptyset \wedge \text{SHOULDERGE}(g, p)$  then
5:        $p \leftarrow \text{WEIGHTEDMERGE}(g, p)$ 
6:     else
7:       if  $p \neq \emptyset$  then
8:          $S \leftarrow S \cup \{p\}$ 
9:       end if
10:       $p \leftarrow g$ 
11:    end if
12:  end for
13:  if  $p \neq \emptyset$  then
14:     $S \leftarrow S \cup \{p\}$ 
15:  end if
16:  return STANDARDNMS( $S$ )
17: end function
```

6.优缺点总结

优点：

- 结构简单， pipeline短， 模型训练和预测的速度快
- 可以适用于单词或文本行级别的文本检测，并且文本框的形状可以是任意四边形，对竖直、倾斜文本的检测效果要比CTPN好
- 模型采用全卷积神经网络，参数量更加轻量级，并且速度也更快

缺点：

- 对长文本的检测不够准确
- 模型的感受野有限，当遇到一些比较大的文字时，可能就没法识别出来

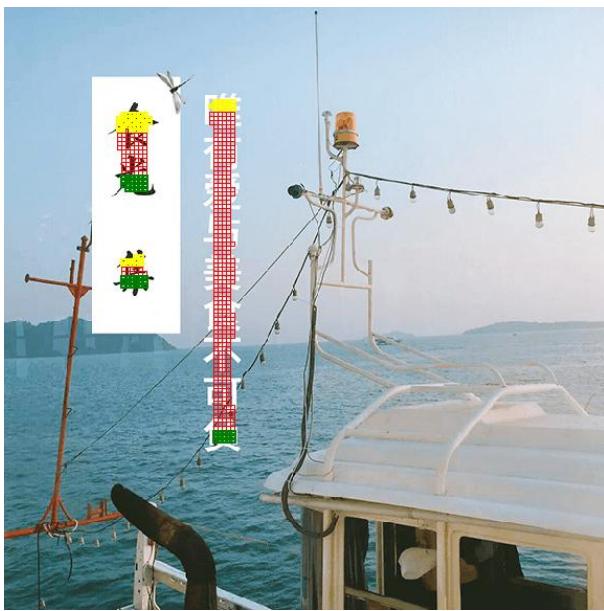
7.改进

7.1 方法一

由于原始的EAST模型是对文本线内的每一个像素点的坐标进行预测，然后对其按照score进行加权平均，作为最终文本线的预测坐标，但是这种方法会有一个缺点，也就是容易导致文本线的坐标容易受到内部像素点的影响，导致预测的文本线偏小，没法准确地框住完整的文本，如下图所示：



因此，huoyijie对其进行了改进，他的改进思想是既然对文本线内的所有像素点的坐标进行预测比较难，那就只对文本线的边界点进行预测，然后只对边界点的坐标进行加权平均，这样就可以加快模型的收敛速度和精度，如下图所示，是他改进方法之后的效果，其中黄色和绿色点就是边界点。



这种想法确实是挺妙的，不过这种方法虽然提高了对文本检测的精度，但是在构建文本线时确有问题，当对边界点的预测不准确时，比如漏了左侧边界或者右侧边界时，就会导致整个文本线都构造失败，

因此，反而使得模型的预测效果更差。

因此，笔者对代码中的文本线构造方法进行修改，笔者的思想是既然采用边界预测可以使得对文本的打点更加准确，那可以直接根据这些像素点的分布，用一个矩形框把它们直接框起来，那这样计算漏掉了某一侧或全部的边界点，文本线也可以构建成功，但是这种方法也有一个不好的地方就是对于**倾斜的文本**，只能用矩形框住，而没法用随意的四边形框住。笔者在作者原来代码的基础上进行小部分修改后：

```
def nms(predict, activation_pixels,
threshold=cfg.side_vertex_pixel_threshold):
    region_list = []
    for i, j in zip(activation_pixels[0], activation_pixels[1]):
        merge = False
        for k in range(len(region_list)):
            if should_merge(region_list[k], i, j):
                region_list[k].add((i, j))
                merge = True
        # Fixme 重叠文本区域处理，存在和多个区域邻接的pixels，先都
        # merge试试
        if not merge:
            region_list.append({(i, j)})
    D = region_group(region_list)
    quad_list = np.zeros((len(D), 4, 2))
    score_list = np.zeros((len(D), 4))
    # TODO(linchuhai):这里确定每个文本框的坐标还需要进一步修改
    for group, g_th in zip(D, range(len(D))):
        cord_list = []
        for row in group:
            for ij in region_list[row]:
                cord_list.append((ij[0], ij[1]))
        cord_list = np.array(cord_list)
        min_i, min_j = np.amin(cord_list, axis=0)
```

```
max_i, max_j = np.amax(cord_list, axis=0)
quad_list[g_th, 0] = np.array([(min_j - 1) * cfg.pixel_size, (min_i - 1) *
cfg.pixel_size])
quad_list[g_th, 1] = np.array([(min_j - 1) * cfg.pixel_size, (max_i + 1) *
cfg.pixel_size])
quad_list[g_th, 2] = np.array([(max_j + 1) * cfg.pixel_size, (max_i + 1) *
cfg.pixel_size])
quad_list[g_th, 3] = np.array([(max_j + 1) * cfg.pixel_size, (min_i - 1) *
cfg.pixel_size])
```

因为参考代码已经写的比较完善，所以这次没有对代码进行大幅度修改，就不贴出模型其他部分的代码了。最后，模型的效果如下：





7.2 方法二



计算机工程与应用
Computer Engineering and Applications
ISSN 1002-8331,CN 11-2127/TP

《计算机工程与应用》网络首发论文

题目： 基于改进 EAST 的自然场景文本定位算法
作者： 杨飚，杜晓宇
网络首发日期： 2019-04-29
引用格式： 杨飚，杜晓宇. 基于改进 EAST 的自然场景文本定位算法[J/OL]. 计算机工程与应用. <http://kns.cnki.net/kcms/detail/11.2127.TP.20190426.1608.008.html>

1、网络结构优化

- 空洞卷积ASPP

2、loss 的改进

- 将原论文中的 class balanced cross-entropy loss 换为 dice loss

- 对 Geo-map 加入 Instance-Balanced