BMVC_TextCohesion

<Excerpt in index | 首页摘要>

TextCohesion: A Accurate Detector for Detecting Text of Arbitrary Shapes

KeyWords Plus: **CVPR** **ICDAR** **AAAI** **ECCV**

- **relevant URL**： 2018-2019 International Conferences Top Computer Science Conferences
- **Paper**：https://arxiv.org/abs/1904.12640

<The rest of contents | 余下全文>

Our idea

本论文的主要贡献：
（1）、**提出了两个概念1、TS(Text Skeleton) 和 （DPR）Directional pixel regions。**首先利用TS完美的区分分割开两个文本实例，同时每个文本实例像素对TS有个类似聚心力的作用，可以完美分割开文本分界线边缘的像素点。
（2）、**引入了文本候选区TCR（Text Candidate Region）的概念**，即引入了周围像素构成上下文的联系关系对tcl进一步进行判断，排除一些误测。
（3）、**引入TS的confidence机制**，对低概率的TS文本进行滤波操作，除去了大部分FP现象。

(a) (b) (c)

python数字图像处理（19）：骨架提取与分水岭算法
Loss加权推荐分水岭算法，真的很好用

- 1、预测**TCL中心线**
- 2、**TR预测**
- 3、像素权重，**大文本小权重，小文本大权重**
- 4、**预测聚心力和正常， 四个方向**

```
BMVC，英国机器视觉会议
举办时间：9月10日-12日
举办地点：卡迪夫大学
截稿时间：4月29日
H5指数：42
官网：https://bmvc2019.org/
推特@BMVC 2019：
https://mobile.twitter.com/bmvc2019
```

Schedule:

3.14 - 4.14: 完成四个数据集上的对比实验，完成论文初稿。
4.14 - 4.29：进一步完善论文，看看有什么需要补充的。

1、TR权重加上

1、 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019)
Location： Long Beach Convention & Entertainment Center, Los Angeles CA, United States
Date：Jun 15 - Jun 21, 2019
Paper Submission Deadline：Nov 16, 2018 (92)
http://cvpr2019.thecvf.com/

2、Association for the Advancement of Artificial Intelligence（AAAI 2019)
Location：Hilton Hawaiian Village, Waikiki Beach, Honolulu, Hawaii, United States
Date：Jan 27 - Feb 1, 2019
Paper Submission Deadline:Sep 5, 2018 (20)
https://aaai.org/Conferences/AAAI-19/

3.International Joint Conference on Artificial Intelligence（IJCAI 2019)
Location：Macao, China
Date：Aug 10 - Aug 16, 2019
Paper Submission Deadline：Feb 25, 2019
http://www.ijcai19.org/

4. IEEE International Conference on Computer Vision (ICCV 2019)
Location： Seoul, South Korea
Date: Oct 27 - Nov 3, 2019
Paper Registration Deadline:March 15, 2019 (11:59PM PST)
Paper Submission Deadline: March 22, 2019 (11:59PM PST)
http://iccv2019.thecvf.com

5. ICDAR2019
Submission Deadline: Feb. 15, 2019
http://icdar2019.org/dates/

Data processing

2D中如何判断一点在另一个点的那个方位

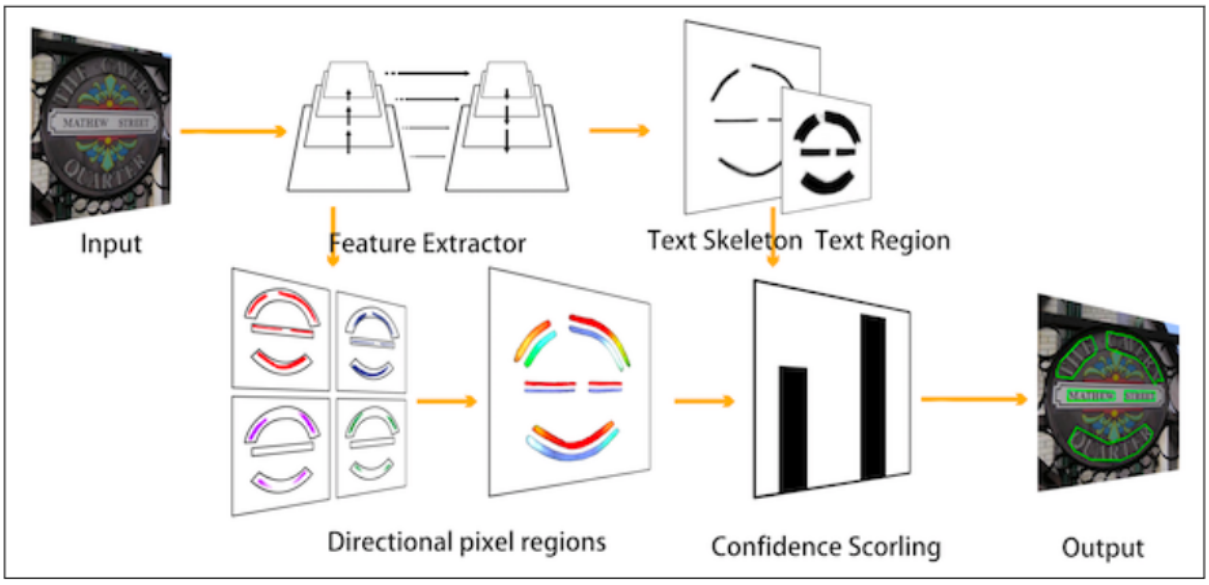
1、计算坐标得到**TS，TPR**

2、得到 上下左右向心力的**link label**

3、tcl加权计算，**大文本有小权重，小文本有大权重**

4、**引入文本周围的全局信息对文本检测的辅助判定**

Pipeline



主要是利用改进版的FCN对图像进行特征提取，对提取得到的DPR,TS,TR等元素进行后处理。最后进过一个confidence scorling机制可得到最终的文本区域。

Loss

Pytorch - Cross Entropy Loss

OHEM(Online Hard Example Mining)

难分样本指的是模型对某个样本学习困难，难以学得其特征。而数据不平衡会导致某一类别在模型中学习迭代次数较少，逐渐成为一种难分样本。

一般解决办法：

1、focal loss：通过模型预测的概率pt，使用(1-Pt)来代表样本难分程度。可以理解为模型对某个样本预测属于其真实label的概率越高，则说明该样本对此模型比较容易学习，反之则难分。

2、《ScreenerNet: Learning Self-Paced Curriculum for Deep Neural Networks》论文提出一个附加网络来帮助主网络区分样本难易程度。

3、《Fine-tuning Convolutional Neural Networks for Biomedical Image Analysis》论文通过对一张图像进行数据增强生成多张图像，然后使用模型预测每张图像的概率。根据多张相同label的增强图像的概率分布区分其样本难易程度。

4、《OHEM: Training Region-based Object Detectors with Online Hard Example

Mining》论文提出先使用模型输出概率，据此选出部分难分样本，然后根据这些样本，更新网络参数。

torch.nn.functional.cross_entropy

在pytorch中若模型使用CrossEntropyLoss这个loss函数，则不应该在最后一层再使用softmax进行激活。https://blog.csdn.net/zziahgf/article/details/80196376

> class torch.nn.CrossEntropyLoss(weight=None, size_average=True, ignore_index=-100, reduce=True)[source]

```
weight(Tensor, optional) - 每个类别class 的权重. 默认为值为 1 的 Tensor.
size_average(bool, optional) - 默认为 True.
  size_average=True, 则 losses 在 minibatch 结合 weight 求平均average.
  size_average=False, 则losses 在 minibatch 求相加和sum.
  当 reduce=False 时,忽略该参数.
ignore_index(int, optional) - 指定忽略的 target 值, 不影响 input 梯度计算.
  当 size_average=True, 对所有非忽略的 targets 求平均.
reduce(bool, optional) - 默认为 True.
  reduce=True, 则 losses 在 minibatch 求平均或相加和.
  reduce=False, 则 losses 返回 per batch 值, 并忽略 size_average.
输入 - input x, (N,C)(N,C), C=num_classesC=num_classes 类别总数.
输入 - target y, (N)(N), 每个值都是 0≤targets[i]≤C-10≤targets[i]≤C-1
输出 - 如果 reduce=True, 输出标量值. 如果 reduce=False, 输出与输入target一致,
(N)(N)
输入 - input x, (N,C,d1,d2,...,dK)(N,C,d1,d2,...,dK), K≥2K≥2 适用于 KK-dim
 场景
输入 - target y, (N,d1,d2,...,dK)(N,d1,d2,...,dK), K≥2K≥2 适用于 KK-dim 场景
```

Smooth_L1_Loss

1、Log loss

$$L = -y \cdot log(y^{'}) - (1-y) \cdot log(1-y)$$

其中： $y = y_{truth}$, $y^{'} = y_{pred}$

对公式进行拆分：

$$L = \begin{cases} -log(y^{'}) & y = 1 \\ -log(1-y^{'}) & y = 0 \end{cases}$$

最终的loss是y=0和y=1两种类别的loss相加，这种方法有一个明显缺点，**当正样本数量远远小于负样本的数量时，即y=0的数量远大于y=1的数量，loss函数中y=0的成分就会占据主导，使得模型严重偏向背景。**

所以对于背景远远大于目标的分割任务，Log loss效果非常不好。

本论文的Loss如下，我这边就不在详细总结了：

The proposed model is trained end-to-end,with the following loss functions as the objectives:

$$L = \lambda L_{TS} + L_{DPR} + L_{TF} + L_{TA} \tag{3}$$

where L_{TS} is a cross entropy loss basically. But put the same weight on all positive pixel is unfair, in which case the large instance contributes greater loss while the smaller one a little. The total loss should treat all samples equally regardless of size.

$$L_{TS} = \frac{B}{S_i} \sum_{n=1}^{N} CrossEntropy(TS_i, \widehat{TS_i}) \tag{4}$$

where B is the sum of the text areas, S_i represent the size of ith instance. $TS_i, \widehat{TS_i}$ are the predicted ith pixel belonging to TS and the corresponding ground truth respectively.

$$L_{DPR} = \sum_{n=1}^{4} \sum_{i \in DPR_n} SmothL1(DPR_i, \widehat{DPR_i}) \tag{5}$$

where n represents the ith DPR. $DPR_i, \widehat{DPR_i}$ represent the th pixel falling into and its ground truth. We use sm1 as a loss in case of outlier effects.

For L_{TF}, L_{TA},we also choose crossentropy loss. L_{TF} counts for whether a text are treat as a true plosive candidate or not and L_{TA} is used to prevent points out of boundaries.

Post processing after Segmentation

后处理步骤：

1、得到预测结果后，先进行一些滤波操作：

1、tcl面积小于多少的舍去 对应`filters_TCL`

2、…

2、按照预测结果检测像素块轮廓，拟合成坐标：

cv2.findContours

检测物体轮廓。返回轮廓坐标： blog

3、approxPolyDP 简化边界：

cv2.approxPolyDP

用指定的精度逼近多边形曲线： blog

针对论文我这边先后尝试了两种不同思路的后处理：

1、让同一个TA区域内的不在TS中的像素去寻找他们各自的TS

主要分成以下几步：

（1）、**先对网络输出的TS、TA、DP进行阈值化**，其中TS 的阈值最为重要，这个可以在之后进行阈值搜索寻找到最合适的阈值，这个阈值选的不好会差十几个百分点的指标都是可以的。

（2）、**对TS进行滤波**，面积小于多少的过滤，不在TA内的过滤，不在TCR内的过滤。

（3）、**利用TS将不同的文本实例区分开来**，即用 `find_contours` 函数按照TS将一张图分为多少个文本实例（这就是本论文的关键）

（4）、**得到与每一个TS与之对应的TA区域，有且仅有一个**，同理用 `find_contours` 函数按照TA分成很多不同的TA实例与TS比较得到与每一个TS相交面积最大的TA，该TA即为相对于的TA

（5）、**在对应的TA中进行两重负循环对每个TA中不在TS中的像素到TS的连接**，首先判断该像素是否为改文本实例像素，否的话进行判断，是否为四个方向的内部像素点（判断依据：周围八个像素至少有六个与自己同方向）

（6）、**对每个方向的内部像素点进行对应方向上寻找对应的TS**，在一定范围内，并且途中没有其他方向的像素，找到得TS即为该文本的像素点，该文本的像素点路径上所有的像素点都属于该文本实例。

2、连接与TS相交最大面积的各个方向 上的像素块

主要分成以下几步：

（1）、**先对网络输出的TS、TA、DP进行阈值化**，其中TS 的阈值最为重要，这个可以在之后进行阈值搜索寻找到最合适的阈值，这个阈值选的不好会差十几个百分点的指标都是可以的。

（2）、**对TS进行滤波**，面积小于多少的过滤，不在TA内的过滤，不在TCR内的过滤。

（3）、**利用TS将不同的文本实例区分开来**，即用 `find_contours` 函数按照TS将一张图分为多少个文本实例（这就是本论文的关键）

（4）、**将各个不同文本的TS进行膨胀操作，并且与周围方向像素块求交集得到最高的方向像素块**，同理用 `find_contours` 函数对四个方向分别进行操作，并与TS求并集，有相交的才可能是同一个文本内的像素点。

（5）、**分成上下和左右两种情况对像素进行连接形成一个文本实例**，在上下方向上按照TS的最大最小X坐标进行负循环，找到对应X上最远的像素块的Y，将这个范围的像素归为该文本实例，同理左右像素进行连接。

结论：

最终我选择的是第二种方案，也是改进后的方案，第二种方案的速度是第一种的6倍左右，并且同一个模型精度更高。

Visualization

热力图资料搜索
python如何实现可视化热力图
Python可视化：Seaborn库热力图使用进阶
python可视化——热力图

Datasets

SynthText

Contains about **800K** synthetic images.

TotalText

Newly-released benchmark for text detection. Besides horizontal and multi-Oriented text instances.The dataset is split into **training and testing sets with 1255 and 300 images**, respectively.

CTW1500

another dataset **mainly consisting of curved text**. It consists of **1000 training images and 500 test images**. Text instances are annotated with polygons with **14 vertexes.**

ICDAR 2015

MSRA-TD500

A dataset with **multi-lingual, arbitrary-oriented and long text lines.** It includes **300 training images and 200 test images** with text line level annotations

implementation details

2019.2.14

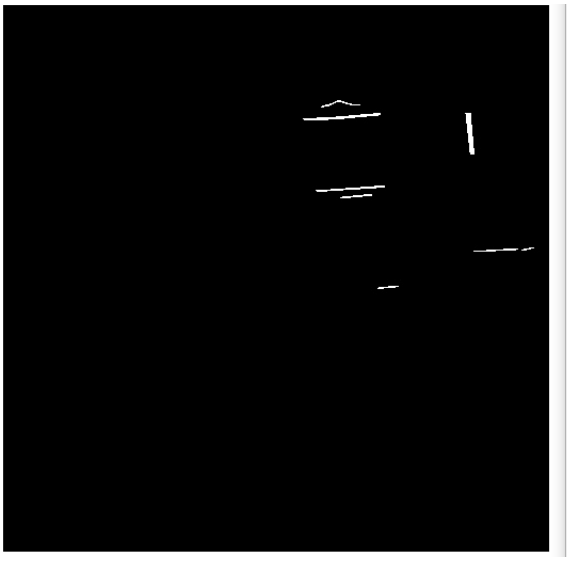
阈值至关重要:

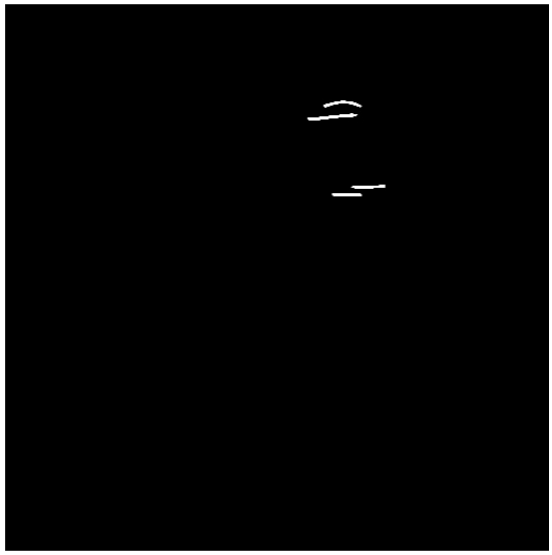
tr_thresh：0.2
tcl_thresh：0.4
up_thresh：0.1
down_thresh:0.1
left_thresh:0.1
right_thresh:0.1

```
Precision:   0.7041604010025037
Recall:  0.693284936479127
F-score: 0.6986803501880151
```

improvement
1、小文本检测
local_precision: 0.0
local_recall: 0.0


WhiteSpot
Breakfast, Lunch & Dinner
WELCOME TO
Lot 2 - 1608 West Georgia St.





2019.2.24

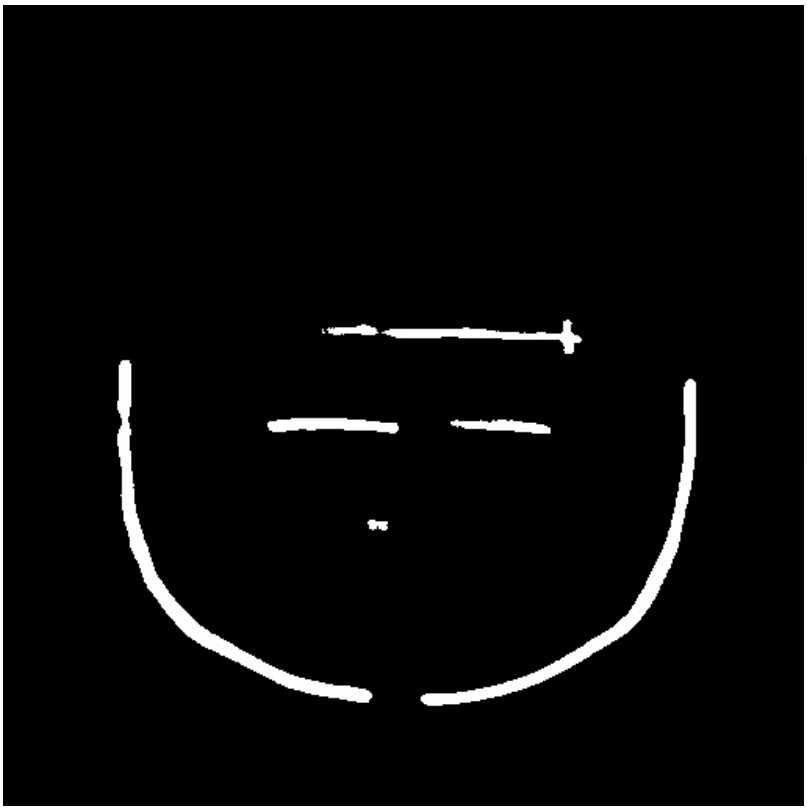
textcohesion_VGG16_26.pth

```
Precision:   0.7478172588832468
Recall:  0.7286751361161509
F-score: 0.7381221125855904
```

2、误测。

Local_precision: 0.2857142857142857

Local_recall: 1.0



pred

2019.3.3

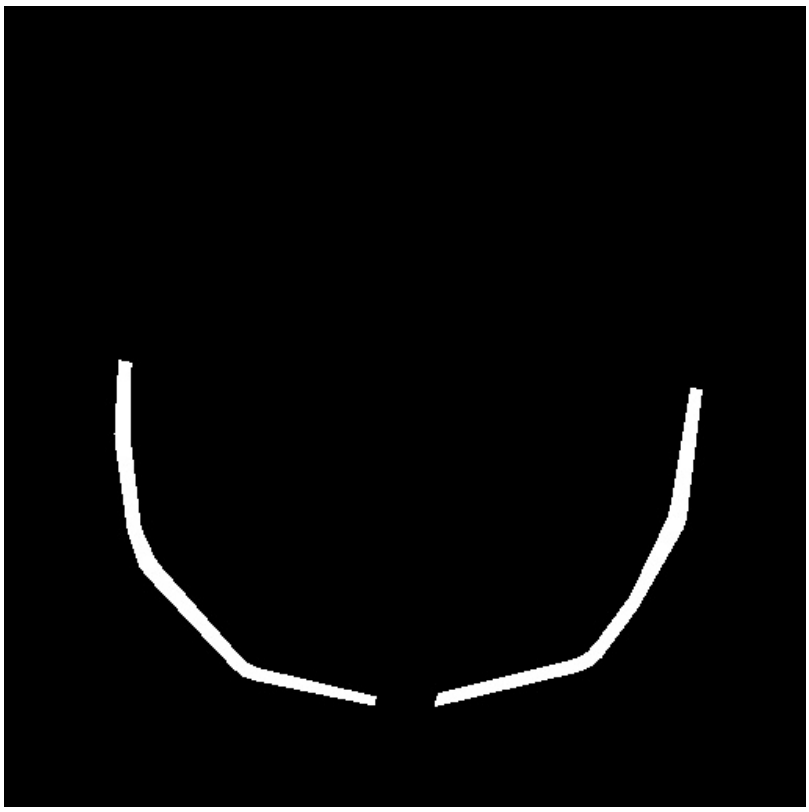
textcohesion_VGG16_28.pth

```
 precision:  0.7908
3644
 recall:  0.7228675
1
 f_score:  0.755325
99
```

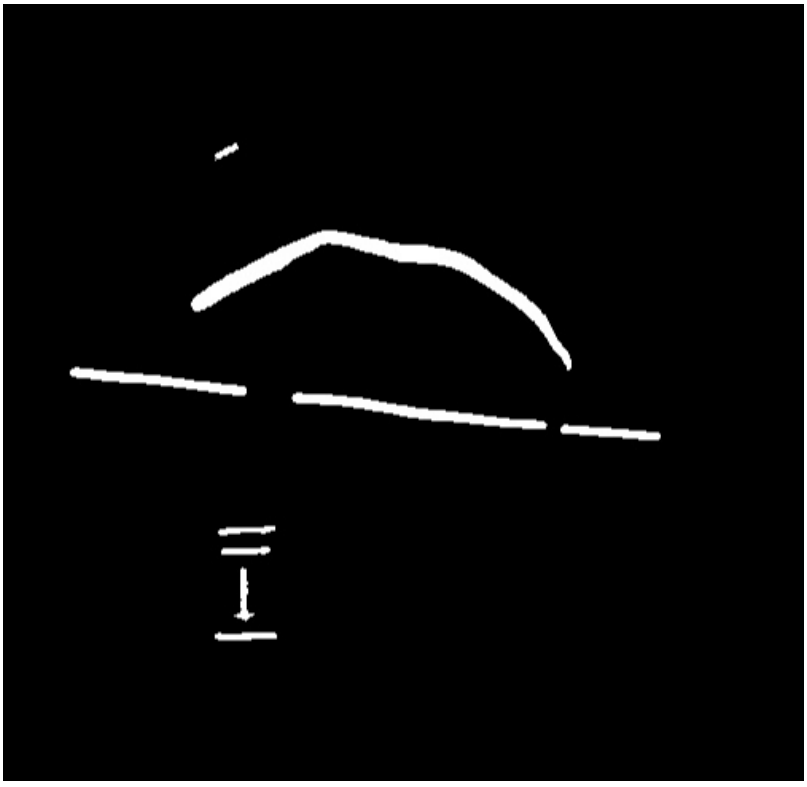
3、检测不到
Local_precision: 0
Local_recall: 0

(./1551591108881.png) 4、检测不到 + 误测
Local_precision: 0.6444444444444444
Local_recall: 0.5499999999999999

Experience



label



pred

label

		真实类别	
		0	1
预测类别	0（Negative）	TN（True Negative）	FN（False Negative）
	1（Positive）	FP（False Positive）	TP（True Positive）

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

1、Total Text

arxiv	2018.11.21	Supervised Pyramid Context Network	0.829	AAAI 2019
arxiv	2018.12.4	TextField	0.806	
arxiv	2018.7.4	TextSnake	0.783	ECCV 2018
arxiv	2019.1.9	MSR	0.786	

2019.4.4 ours TextCohesion
precision: 0.881, recall: 0.814, f_score: 0.846

2、CTW1500

arxiv	2018.11.30	TextMountain	0.832	
arxiv	2018.12.4	TextField	0.814	
arxiv	2018.7.4	TextSnake	0.756	ECCV 2018
arxiv	2019.1.9	MSR	0.807	

textcohesion_best_model.pth
2019.3.12 ours TextCohesion 0.863

3、ICDAR2015

arxiv	2018.11.30	TextMountain	0.872	
arxiv	2018.12.4	TextField	0.824	
arxiv	2018.7.4	TextSnake	0.826	ECCV 2018

textcohesion_best_model.pth
2019.3.12 ours TextCohesion 0.70

4、ICDAR2019_Arbitrary-Shaped

arxiv	2018.11.30	TextMountain	0.872	
arxiv	2018.12.4	TextField	0.824	
arxiv	2018.7.4	TextSnake	0.826	ECCV 2018

textcohesion_best_model.pth
2019.3.12 ours TextCohesion

调参技巧:

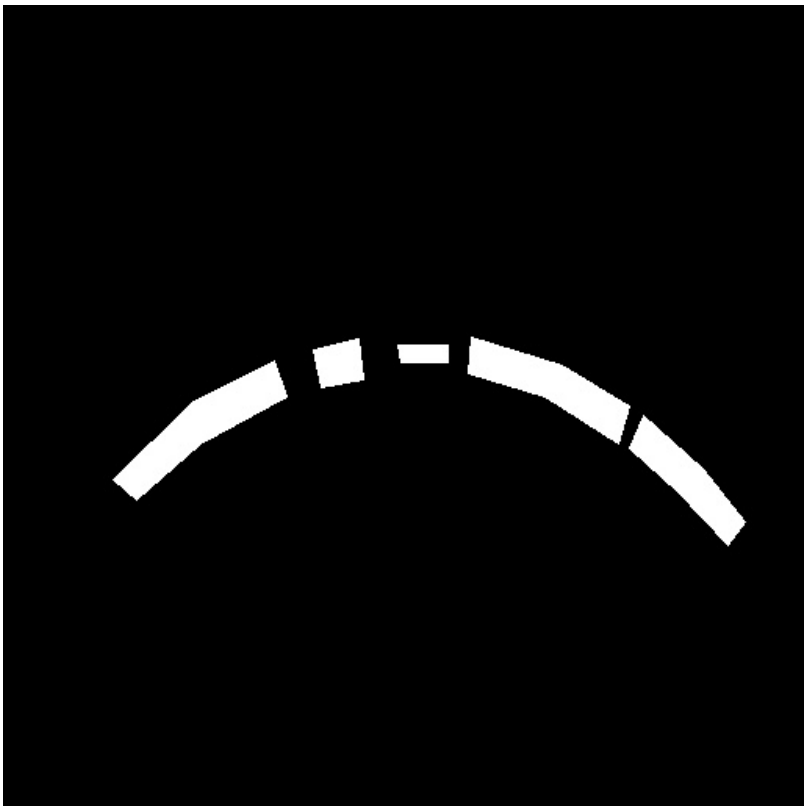
1、改动 TCL的阈值
2、修改TCL的filter阈值，小于多少面积的舍去
3、获得TCL的confidence值（即对TCL内的像素求和取平均），将一定概率下的TCL滤去

Demo

2019.2.24

预训练：textcohesion_VGG16_0.pth
`textcohesion_VGG16_24.pth`

```
Precision:   0.7478172588832468
Recall:  0.7286751361161509
F-score: 0.7381221125855904
```



TR

`textcohesion_best_model.pth`

2019.3.13

TCL_TEL_7

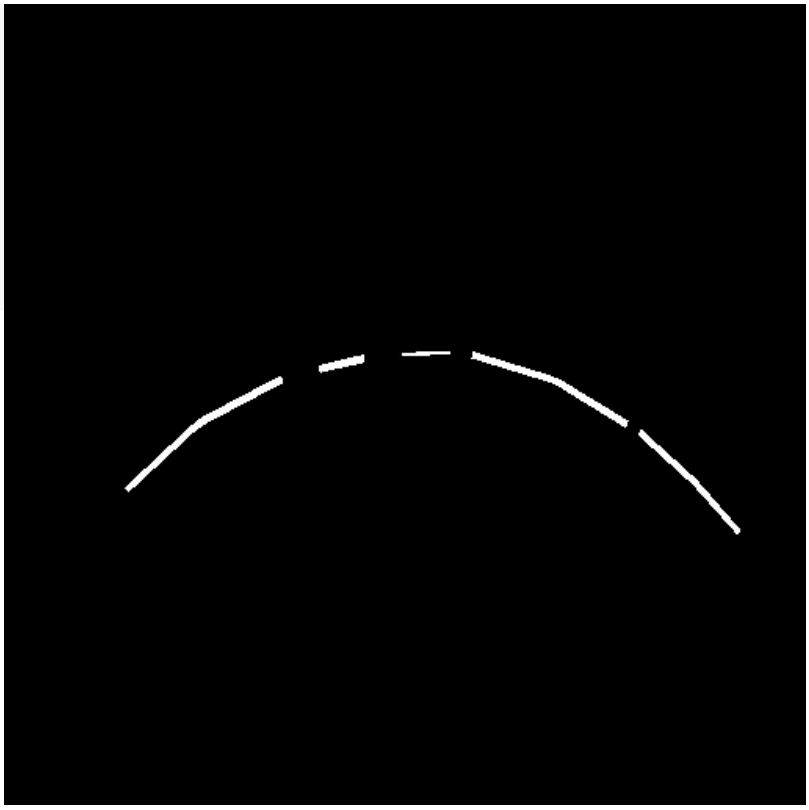
预训练:

textcohesion_VGG16_1.pth

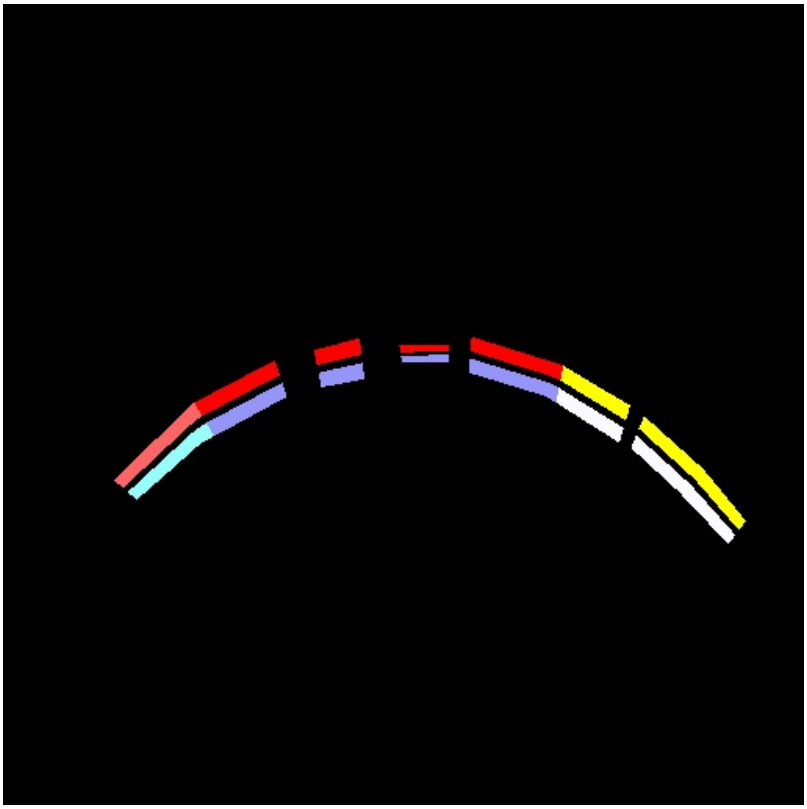
```
 precision:0.787418
218419726
 recall:0.750907441
0163324
 f_score:0.76872955
44689221
```

反馈与建议

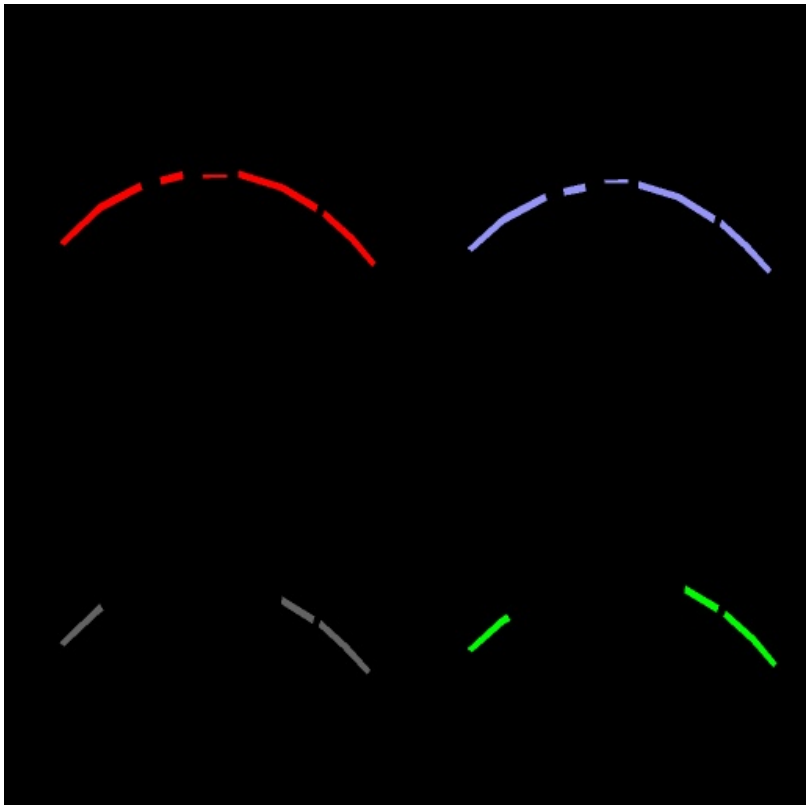
- 邮箱：weijia_wu@yeah.net



TCL

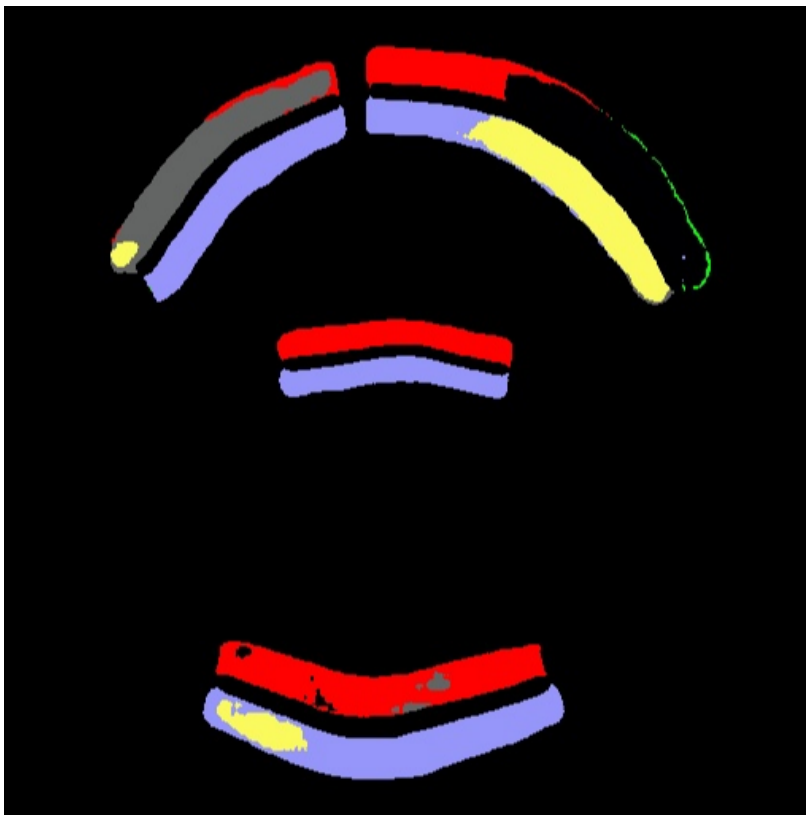


four directions

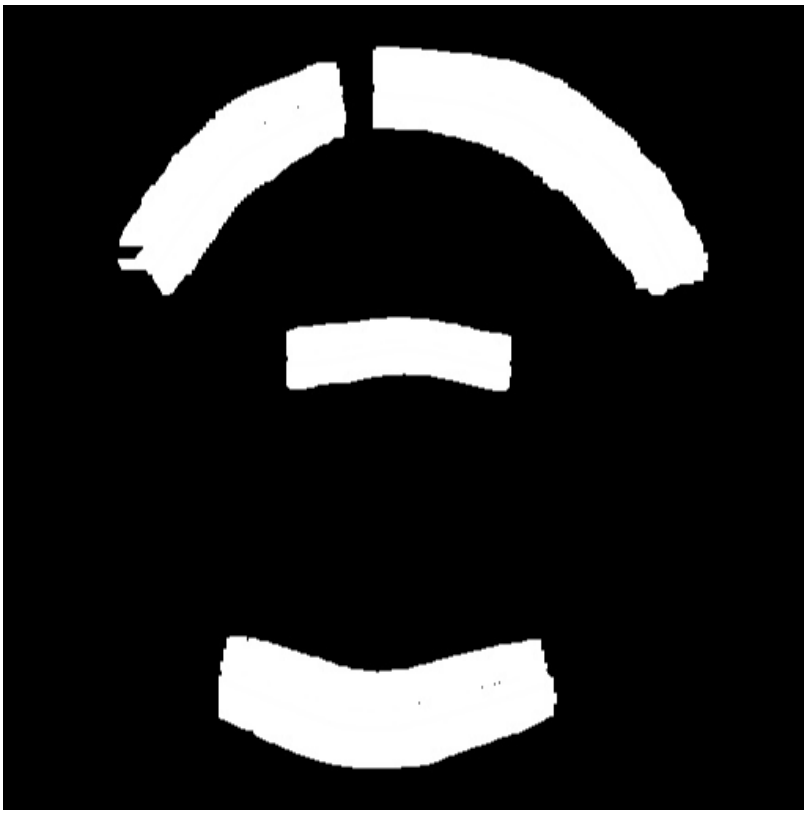


four directions

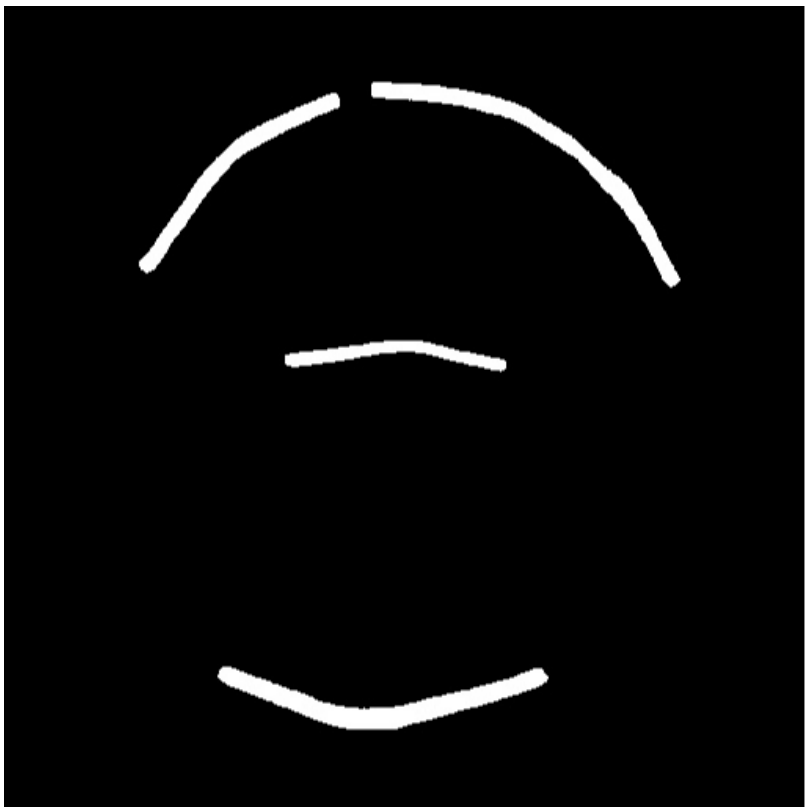
TR



TCL



TR



TCL