文本检测–PixelLink

<Excerpt in index | 首页摘要>

PixelLink: Detecting Scene Text via Instance Segmentation

KeyWords Plus: AAAI-2018

- **relevant blog**： 文本行检测之PixelLink 知乎 文本行检测之PixelLink
- **paper**： PixelLink
- **coding**： Github

<The rest of contents | 余下全文>

Introduction

1、论文创新点

The main idea of the paper

- A novel scene text detection algorithm based on **instance segmentation**（Outside regression）
- A CNN model is trained to perform two kinds of **pixel-wise** predictions: **text/non-text prediction** and **link prediction**.

2、文本检测

Most state-of-the-art scene text detection algorithms are deep learning based methods that depend on bounding box regression and perform **at least** two kinds of predictions:

1、text/nontext classification

2、location regression

Classification: 输入是图片，输出是判断概率结果，判断是一只猫，这个叫做**分类任务。**

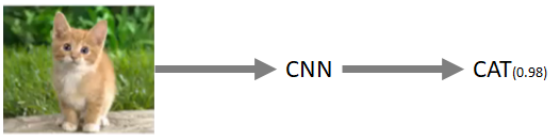
Regression: 输入是图片，输出是坐标，寻找到这张图里面猫在哪个位置，这个叫做**回归任务。**

- Classification:

Input: Image

Output: Class label

Evaluation metric: Accuracy

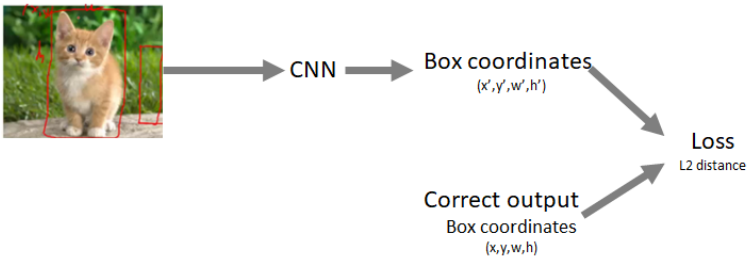


- Localization:

Input: Image

Output: Box in the image(x,y,w,h)

Evaluation metric: Intersection over Union



如上图所示，就是分类和回归问题的主要区别

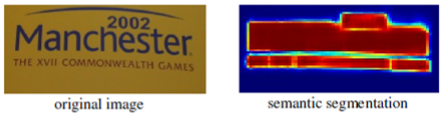
- CTPN (Tian et al. 2016)
- TextBoxes (Liao et al. 2017)
- SegLink (Shi, Bai,and Belongie 2017)
- EAST (Zhou et al. 2017).

Text/Nontext classification

Bounding box regression

Regression plays **a key role** in the acquisition of bounding boxes in these methods.

text/non-text prediction can also be considered as a kind of **semantic segmentation** that contains full location information



original image　　semantic segmentation

Figure 1: Text instances often lie close to each other, making them hard to separate via semantic segmentation.

Can we detect scene text just via **semantic segmentation**?

如上图所示,虽然文本框的回归在以往的文本检测中扮演着重要的角色，但是他并不是不可缺少的，因为基于像素点的文本二分类也包括着位置坐标信息。

但是这里就有个问题了，如上图所示，如果单纯的用文本二分类（语义分割）来定位检测文本，因为文本之间相距较近导致无法分割开来。

所以作者引入了实例分割来解决这个问题，下面就是语义分割与实例分割得区别。

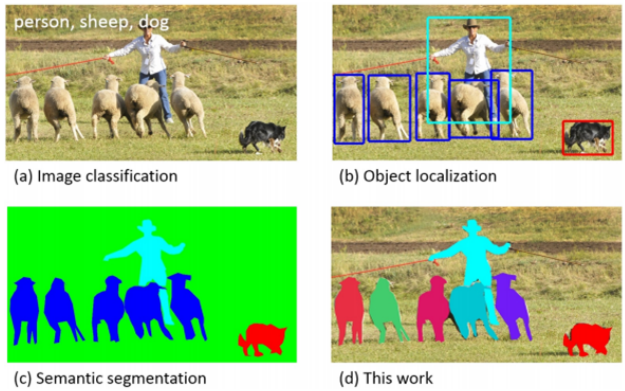
3、实例分割

- Semantic Segmentation

The implement of classification per pixel

- Instance Segmentation

Not only classify each pixel,but also on the basis we need to classify different Instances



(a) Image classification　(b) Object localization

(c) Semantic segmentation　(d) This work

如上图所示：

场景识别（Scene Recognition）：将不同的图片按照其所示环境进行分类，意在缩小计算机与人类对于场景理解的差异。

物体检测（Object Detection）：物体检测主要包含两个问题，一个是判断属于某个特定类的物体是否出现在图中；二是对该物体进行定位。主要实现，输入测试图片，输出检测到物体的类别和位置。

语义分割（Semantic Segmentation）：判断图片中的某一像素属于**某个物体类别。**同一个物体的不同实例不需要单独分割出来。如测试图片中有多头羊，**只需区分出羊这个大类**，不需要具体分出羊1，羊2，等。

实例分割（Instance Segment）：实例分割是**物体检测与语义分割的综合体。**相对于物体检测的方框，实例分割可精确到物体的边缘；相对于语义分割，实例分割可以标注出图上同一类别物体的**不同个体（牛1，牛2，牛3等）。**

网络框架

1、整体框架

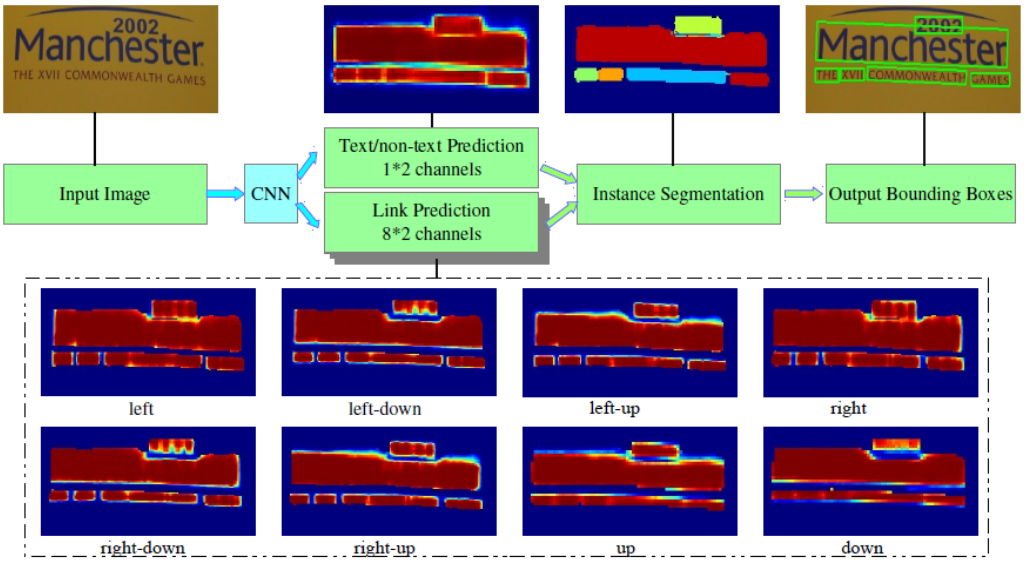


Figure 2: Architecture of PixelLink. A CNN model is trained to perform two kinds of pixel-wise predictions: text/non-text prediction and link prediction. After being thresholded, positive pixels are joined together by positive links, achieving instance segmentation. *minAreaRect* is then applied to extract bounding boxes directly from the segmentation result. Noise predictions can be efficiently removed using post-filtering. An input sample is shown for better illustration. The eight heat-maps in the dashed box stand for the link predictions in eight directions. Although some words are difficult to separate in text/non-text prediction, they are separable through link predictions.

PixelLink

1.**Text/Non-text prediction**

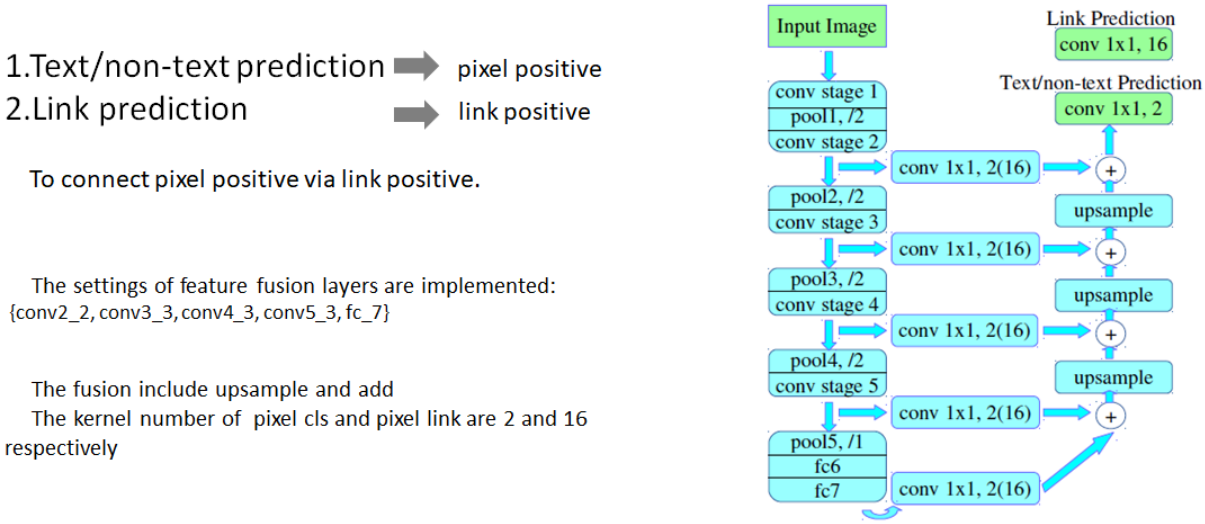
2.**Link Prediction**(`innovation`)

Although it's hard to distinguish among texts, we can solve this problem via link prediction.

PixelLink detects text via instance segmentation, where predicted positive pixels are joined together into text instances **by predicted positive links**. Bounding boxes are then directly **extracted from this segmentation result.**

2、网络框架

Structure of PixelLink+VGG16 2s. fc6 and fc7 are converted into convolutional layers. The **upsampling** operation is done through bilinear **interpolation** directly. Feature maps from different stages are fused through a cascade of **upsampling and add operations**. All pooling layers except pool5 take a stride of 2, and pool5 takes 1. Therefore, the size of fc7 is the same as conv5 3, and no upsampling is needed when adding scores from these two layers. 'conv 1 1,2(16)' stands for a 11 convolutional layer with **2 or 16 kernels**, for text/non-text prediction or link prediction individually.



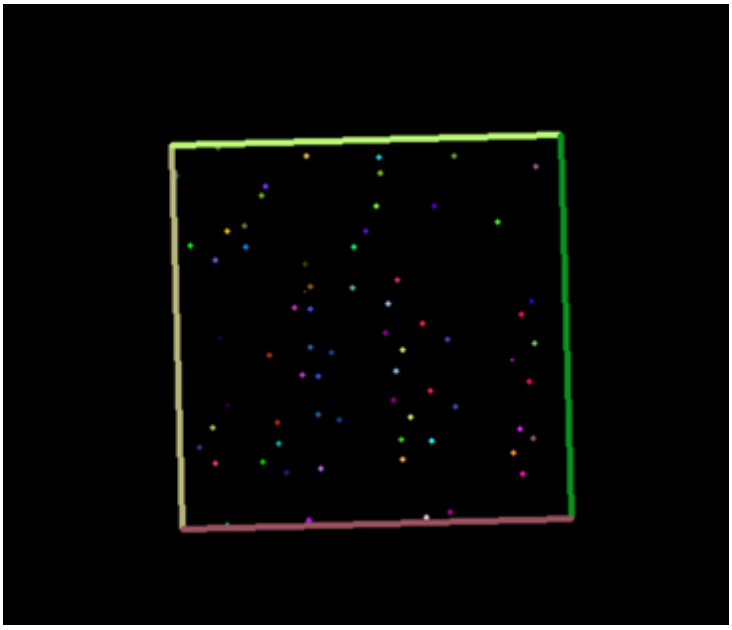
整个实现过程包括两部分：先通过深度学习网络预测`pixel positive`和`link positive`，并根据`link positive`连接`pixel positive`得到文本实例分割图，然后从分割图中直接提取文本行的bbox.具体步骤如下：

- 主干网络是沿用了SSD网络结构．具体来说：首先用**VGG16作为base net**，并将VGG16的最后两个**全连接层改成卷积层**．
- 提取不同层的feature map，对于PixelLink+VGG16 2s网络结构：提取了`conv2_2, conv3_3, conv4_3, conv5_3, fc_7`．
- 对已提取的特征层，采用自顶向下的方法进行融合，融合操作包括先向上采样，然后再进行add操作．注意：这里包含了两种操作：`pixel cls`和`pixel link`，对应的卷积核个数分别为 2 和16
- 对于网络输出层，包括**文本/非文本预测和Link预测**．

3、Linking Pixels Together

通过设定两个不同的阈值，可以得到`pixel positive`集合和`link positive`集合，然后根据link positive将pixel positive进行连接，得到`CCs(conected compoents)`集合，**集合中的每个元素代表的就是文本实例**．这里需要特别注意：给定两个相邻的pixel positive，它们之间的link预测是

由当前两个pixel共同决定的．而当前两个pixel需要连接(即两个像素属于同一个文本实例)的前提条件：**two link中至少有一个link positive．连接的规则采用的是Disjoint set data structure(并查集)的方法．**



4、Extraction of Bounding Boxes

基于上述语义分割的结果(即上述的CCs)，直接通过`opencv`的`minAreaRext`提取文本的带方向信息的外接矩形框(即带角度信息)，具体的格式为`((x,y),(w,h),θ)`，分别表示中心点坐标，当前bbox的宽和高，旋转角度．

这里和`SegLink`的关键区别：`PixelLink`是直接从分割结果中提取bbox，而`SegLink`采用的是边框回归.

5、Post Filtering after Segmentation

之所以要进行后处理，是因为在`pixel`进行连接的时候不可避免地会引入噪声．文中主要是对已检测的bbox通过一些简单的`几何形状判断`(包括bbox的宽，高，面积及宽高比等，这些主要是在对应的训练集上进行统计的出来的)进行filter．经过后处理后，可以提升`文本行检出的准确率`．

6、训练

像素间的`link`生成规则：给定一个像素，若其与邻域的 8 像素都属于同一个文本实例，则将其link标注为`positive`，否则标注为`negative`。

这里值得注意的是：`ground truth`的计算是在缩放后(具体缩放的比例与其预测的`feature map`大小一致)的原图上进行的。

Loss Function

The training loss is a weighted **sum of loss on pixels and loss on links**:

$$L = \lambda L_{pixel} + L_{link}.$$

1、Loss on Pixels

`Instance-Balanced Cross-Entropy Loss:`

$$L_{pixel} = \frac{1}{(1+r)S} W L_{pixel\_CE},$$

`Positive pixels weight:`

$$w_i = \frac{B_i}{S_i}.$$

$$B_i = \frac{S}{N}, S = \sum_{i}^{N} S_i, \forall i \in \{1, \ldots, N\}$$

`Negative pixels weight:`

Online Hard Example Mining (OHEM)

More specifically, `r*S negative pixels` with the highest losses are selected, by setting their weights to ones. `r` is the `negative-positive ratio` and is set to 3 as a common practice.

As a result, pixels in small instances have a higher weight, and pixels in large instances have a smaller weight.

2、Loss on Links

`Class-balanced cross-entropy loss:`

$$L_{link} = \frac{L_{link\_pos}}{rsum(W_{pos\_link})} + \frac{L_{link\_neg}}{rsum(W_{neg\_link})},$$

`Losses for positive and negative links are calculated separately and on positive pixels only:`

$$L_{link\_pos} = W_{pos\_link} L_{link\_CE},$$
$$L_{link\_neg} = W_{neg\_link} L_{link\_CE},$$

$$W_{pos\_link}(i,j,k) = W(i,j) * (Y_{link}(i,j,k) == 1),$$
$$W_{neg\_link}(i,j,k) = W(i,j) * (Y_{link}(i,j,k) == 0),$$

Coding

数据处理

使用TFRecords读取数据

`tfrecord`数据文件是一种将图像数据和标签统一存储的`二进制文件`，能**更好的利用内存**，在`tensorflow`中快速的复制，移动，读取，存储等.

优点:

- **读取速度快**

缺点:

- **需要额外生成TFRecords，占用较大内存（当然生成好后就不用管了）**

Feature map 融合、上采样

```
def _fuse_by_cascade_conv1x1_upsample_sum(self, num_classes, scope):
    import config
    num_layers = len(config.feat_layers)

    with tf.variable_scope(scope):
        smaller_score_map = None
        for idx in range(0, len(config.feat_layers))[::-1]: #[4, 3, 2, 1,
 0]
            current_layer_name = config.feat_layers[idx]
            current_layer = self.end_points[current_layer_name]
            current_score_map = self._score_layer(current_layer,
                                  num_classes, current_layer_name)
            if smaller_score_map is None:
                smaller_score_map = current_score_map
            else:
                upscore_map = self._upscore_layer(smaller_score_map, curr
ent_score_map)
                smaller_score_map = current_score_map + upscore_map

    return smaller_score_map
```

Result

1、数据集得分

论文：

Method	Recall	Precision	F
PixelLink+VGG16 2s	83.6	86.4	84.5
PixelLink+VGG16 4s	82.3	84.4	83.3
PixelLink+VGG16 2s MS	**87.5**	88.6	**88.1**
PixelLink+VGG16 4s MS	86.5	88.6	87.5
TextBoxes+VGG16	74	88	81
TextBoxes+VGG16 MS	83	89	86
EAST+PVANET2x MS?	82.7	92.6	87.4
SegLink+VGG16	83.0	87.7	85.3
CTPN + VGG16	83.0	**93.0**	87.7

ICDAR2015

实测：

`Recall`: **0.81608088**, `Precision`: **0.8543346**, `F-score`: **0.8347697**

2、分析

When it comes to `PixelLink`, neurons on the prediction layer only have to observe the status of `itself and its neighboring pixels` on the feature map. In another word, PixelLink has the least requirement on receptive fields and the `easiest task` to learn for neurons among listed methods.

The link design is important because it converts **semantic segmentation** into **instance segmentation**, indispensable for the separation of nearby texts in PixelLink.

PixelLink can be trained from scratch with `less data(1k)` in `fewer iterations`, while achieving on par or better performance on several benchmarks tha state-of-the-art methods based on location regression.

The main task of text detection is pixel classification, and Pixel link take this to extremes, the biggest innovation is link classification.

Content of PixelLink are all classification tasks without regression, the advantage is leaning easier , training faster, less data.

反馈与建议

- 微博：@柏林designer

- 邮箱：wwj123@zju.edu.cn