

复旦大学 IMC 出品

ICDAR2019 竞赛报告

By 字符组

IMC 字符组

2019/5/20

目录

检测任务.....	2
竞赛简介.....	2
ICDAR2019-LSVT.....	2
ICDAR2019-ArT.....	3
采用模型.....	3
Textsnake[1]	3
Mask R-CNN[2].....	6
Mask Scoring R-CNN[3]	7
Cascade Mask R-CNN[4].....	8
CTD(Curve Text Detector)[5]	9
EAST[6]	10
Advanced EAST[7]	11
PSENet[8]	13
PANet[9]	14
SPCNet[10]	15
DenseASPP[11].....	17
M2Det[12].....	18
Deformable Convolution[13].....	20
不同论文的组合架构:	21
消融实验分析（Ablation Study）	23
所发现的问题.....	27
涉及的创新点.....	29
识别任务.....	31
竞赛.....	31
ICDAR2019-LSVT.....	31
ICDAR2019-ArT.....	32
采用模型.....	33
MORAN	36
GRCNN.....	38
消融实验分析（Ablation Study）	39
所发现的问题.....	55
涉及的创新点.....	59
参赛人员名单及分工.....	62
检测组.....	62
识别组.....	62
特别鸣谢.....	62

检测任务

竞赛简介

本次 ICDAR2019 竞赛我们主要参加了两个子竞赛：ArT 和 LSVT。

ICDAR2019-LSVT

ArT 数据集包含训练集 5603 张图片，测试集 4503 张图片，绝大部分图片是街景和标志广告牌等。任务特点是数据集中将近 1/3 的文本区域都是弯曲文本，并且检测的提交结果也不同于 ICDAR2015 和 2017 的要求仅需要给出包围文本区域的任意四边形，而是需要用足够多的点将文本区域准确恰好地分割出来。



图 1 LSVT 数据集图片样例



GT: 依佳人网咖



GT: 赫伦芭蒂国际艺术教育

图 2 LSVT 数据集弱标注数据样例

ICDAR2019-ArT

LSVT 数据集包含全标注训练集 30000 张图片，弱标注训练集 400000 张图片，测试集 20000 张图片，文本区域特点相对而言与 ICDAR2015 和 2017 的数据集比较接近，但也存在一定数量的弯曲文本，绝大部分图片也是来自于街景和标志广告牌等。任务特点是存在大量的弱标注数据，这些图片的标注信息中没有文本区域位置信息，并且仅对图片中的其中一个文本区域提供了文本内容的标注信息。提交结果同样要求将文本区域恰好准确地分割出来。



图 3 ArT 数据集图片样例

采用模型

Textsnake[1]

1) 算法介绍

算法 Textsnake，发表于 [ECCV2018 \(旷视科技\)](#)

论文链接：<https://arxiv.org/abs/1807.01544>

代码链接：<https://github.com/princewang1994/TextSnake.pytorch>

Textsnake 是基于 U-Net 结构的 One-stage 分割算法。Motivation 如下：自然场景中的文本形状有时不一定是规则的任意朝向四边形，还有很大一部分是弯曲或者是不规则形状的文本。如果对于这些文本依旧采用原来的基于四边形的算法，所检测得到的文本框往往带有很多多余的背景信息，导致识别算法效果不佳。如何检测并分离开不同的弯曲文本实例成为了 OCR 文本检测中目前的一个热点

难题。

Textsnake 算法通过预测文本区域的 mask (称为 tr_mask) 及各文本中心区域的 mask (称为 tcl_mask)，并通过后处理算法从中心区域采样出一条由较少关键点组成的中心线(论文中原始的后处理算法中需用到从每个像素点指向沿中心线下一点方向的下一个像素点的角度，用 $\sin \theta$ 和 $\cos \theta$ 来描述)。最后对采样得到的中心线上每一个点均去预测一个圆的半径，将所得圆的 mask 结合 (elementwise-or 操作) 为文本区域的 mask。这样的方法能够利用中心线分隔较远的特性将相距较近的文本分离开来，并通过用圆盘拼接的方法完成对任意形状文本区域的建模。

2) 模型结构

(1) 架构图

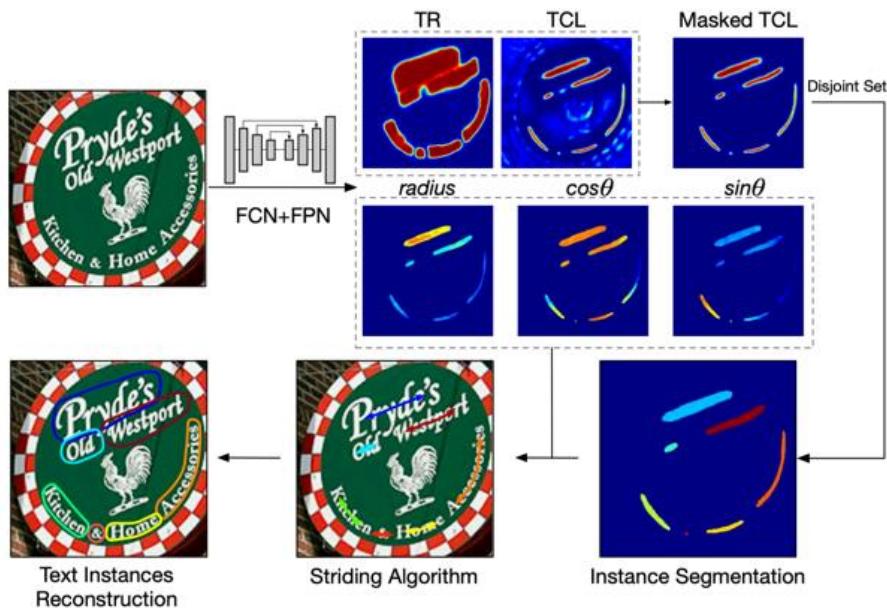


图 4 Textsnake 算法框架图

(2) 方法细节

该算法基于 U-Net 结构，backbone 网络选择了 VGG-16，最后预测得到大小和输入图像相同的结果，channel 数为 7，分别代表 tr_mask(2 维)、tcl_mask(2 维)、 $\sin \theta$ (1 维)、 $\cos \theta$ (1 维) 和半径 r (1 维)。

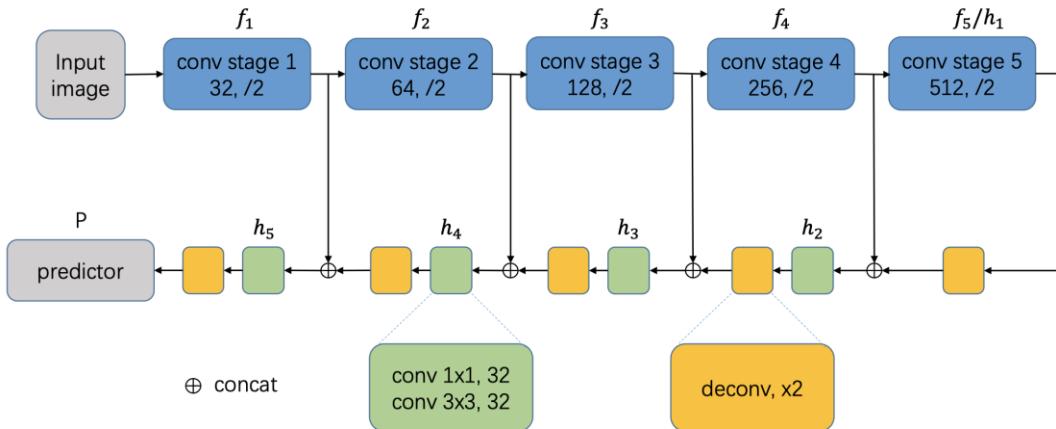


图 5 Textsnake 的网络结构

后处理算法中从 `tcl_mask` 中得到中心线的过程共分为 4 步, 如下图所示。

- (i) 首先在 `tcl_mask` 内部随机采样一个点, 并根据该位置对应的 $\sin \theta$ 和 $\cos \theta$ 求出相应指向下一个中心点的直线方向 (由于 $\sin \theta$ 和 $\cos \theta$ 的 gt 信息在生成时是每个圆盘内部的点相应值都是一样的, 所以在该随机采样的点的位置处所预测得到的 $\sin \theta$ 和 $\cos \theta$ 可以认为与在圆盘中心处的点所预测得到的 $\sin \theta$ 和 $\cos \theta$ 近似), 求该直线过采样点的垂线, 与 `tcl_mask` 边界相交于两点, 取得该两点的中点即得到了第一个中心线的采样点;
- (ii) 根据(i)中所得中心线采样点位置所预测得到的 $\sin \theta$ 、 $\cos \theta$ 和 r , 分别沿 $(0.5r\cos \theta, 0.5r\sin \theta)$ 和 $(-0.5r\cos \theta, -0.5r\sin \theta)$ 方向移动一步, 得到新的两个点;
- (iii) 对(ii)中所得到的两个点分别使用(i)中的方法, 取得新的中心线上的采样点;
- (iv) 不断重复(ii)(iii)步骤, 最终生成完整的中心线。

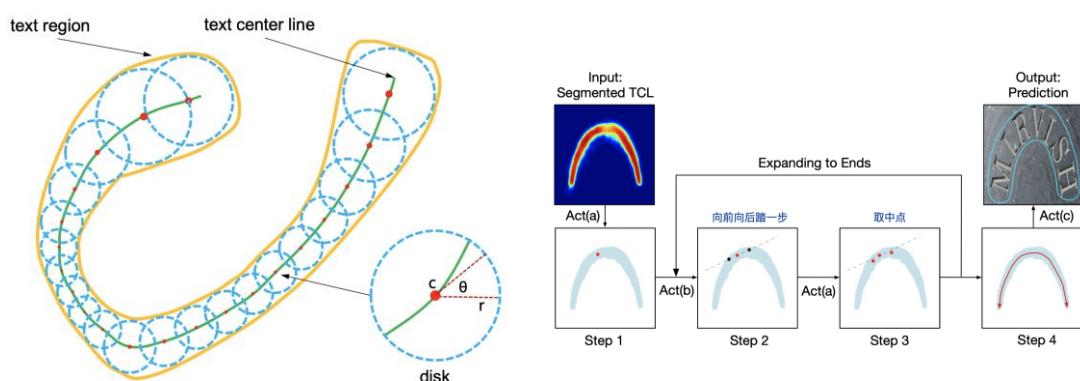


图 6 通过 `tcl_mask` 获取中心线的后处理算法图示

Mask R-CNN[2]

1) 算法介绍

Mask rcnn 在 Faster rcnn 的基础上引入了 ResNet+FPN 的 backbone、RoI Align、Mask 分支，FPN 结构融入了多尺度信息，RoI Align 改进了 Faster rcnn 中 RoI Pooling 的精度不足的问题。

算法 Mask rcnn，发表于 **ICCV 2017**（何凯明）。

论文地址：<https://arxiv.org/abs/1703.06870>。

Pytorch 源码地址：

<https://github.com/facebookresearch/maskrcnn-benchmark>。

2) 模型结构

(1) 架构图

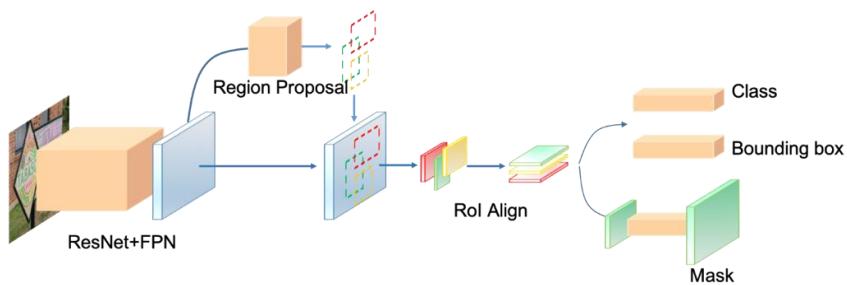
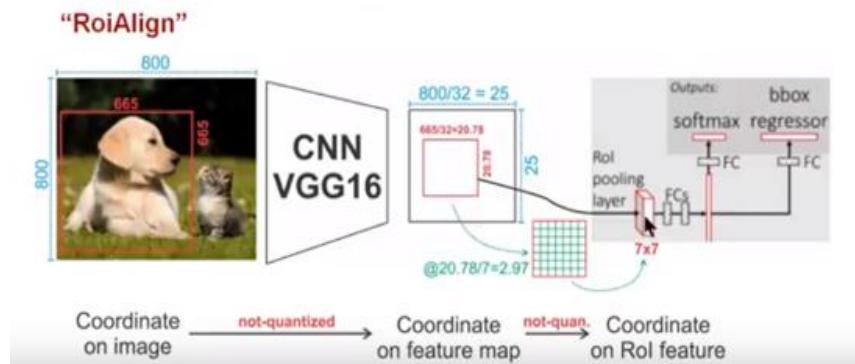


图 7 Mask rcnn 架构图

(2) 方法细节

为了得到固定大小 (7×7) 的 feature map，ROIAlign 并没有使用 RoI Pooling 的量化操作，比如 $665 / 32 = 20.78$ ，不用 20 来替代 20.78。这就是 ROIAlign 的初衷。



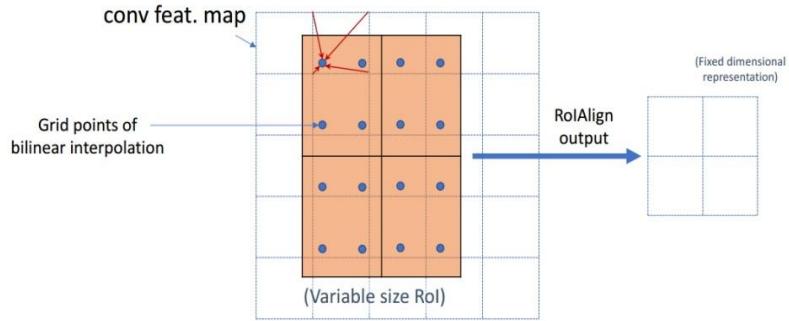


图 8 RoI Align 过程

如上图所示，蓝色的框表示卷积后获得的 feature map，黑色实线框表示 ROI feature，假设最后需要输出的大小是 2x2，那么利用双线性插值来估计这些蓝点(虚拟坐标点，又称双线性插值的网格点)处所对应的像素值，最后得到相应的输出。这些蓝点是 2x2Cell 中的随机采样的普通点，作者指出，这些采样点的个数和位置不会对性能产生很大的影响，你也可以用其它的方法获得。然后在每一个橘红色的区域里面进行 max pooling 或者 average pooling 操作，获得最终 2x2 的输出结果。

Mask Scoring R-CNN[3]

1) 算法介绍

算法 Mask scoring rcnn，发表于 CVPR 2019 (Oral 文章)。

论文地址：<https://arxiv.org/pdf/1903.00241.pdf>。

Pytorch 源码地址：https://github.com/zhuang22/maskscoring_rcnn。

Mask rcnn 存在的问题是：bounding box 的 classification confidence 不能代表 mask 的分割质量。classification confidence 高可以表示检测框的置信度高，但也会存在 mask 分割的质量差的情况，如下图。

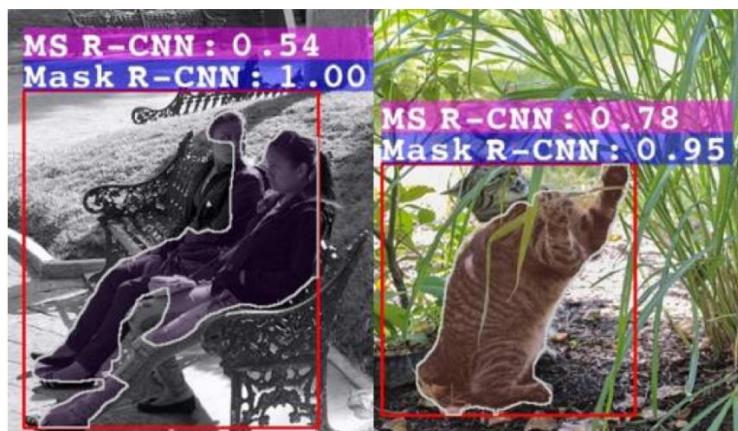


图 9 mask 的评分问题

2) 模型结构

Mask scoring rcnn 的方法，架构如下图，为了弥补 Maskrcnn 对于目标的评分容易受 Bbox 框中背景像素的影响，它在 Maskrcnn 中增加了一个支路来对分类进行重新打分，在 COCO 数据集上取得了不错的效果。

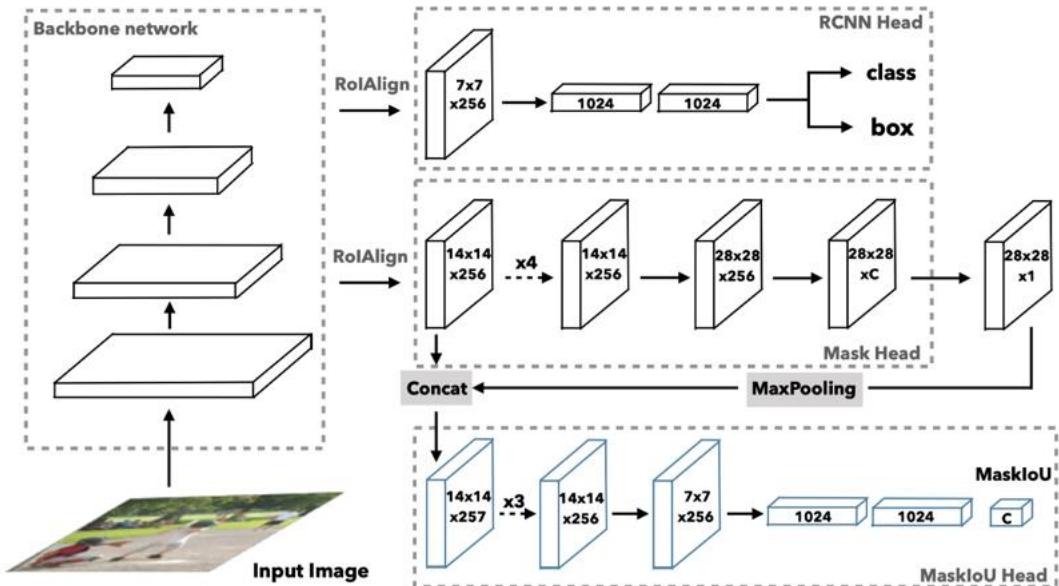


图 10 Mask scoring RCNN 的架构图

Cascade Mask R-CNN[4]

1) 算法介绍

算法 Cascade R-CNN: Delving into High Quality Object Detection, 发表于 **CVPR 2018**。

论文地址: <https://arxiv.org/abs/1712.00726>。

Pytorch 源码地址: <https://github.com/open-mmlab/mmdetection>。

RCNN 系列的模型中，**training** 阶段和 **inference** 阶段，**bbox** 回归器的输入分布是不一样的，**training** 阶段的输入 **proposals** 质量更高(被采样过, $\text{IoU} > \text{threshold}$)，**inference** 阶段的输入 **proposals** 质量相对较差（没有被采样过，可能包括很多 $\text{IoU} < \text{threshold}$ 的），这就是论文中提到 **mismatch** 问题。

为了提高检测精度，不能单纯地提高 **IoU** 阈值，那样会造成更严重的 **mismatch** 问题，因此，作者设计了 **cascade** 结构，采用多个 **stage**，用一个 **stage** 的输出去训练下一个 **stage**，举个例子，有三个串联起来的用 $0.5/0.6/0.7$ 的阈值训练出来的 **detector**，有一个 **IoU** 约为 0.55 的 **proposal**，经过 0.5 的 **detector**，**IoU** 变为 0.75 ；再经过 0.6 的 **detector**，**IoU** 变为 0.82 ；再经过 0.7 的 **detector**，最终

IoU 变为 0.87……比任何一个单独的 detector 的结果都要好。不仅仅只有 IoU 改善的好处，因为每经过 detector，proposal 的 IoU 都更高，样本质量更好了，那么即使下一个 detector 阈值设置得比较高，也不会有太多的样本被刷掉，这样就可以保证样本数量避免过拟合问题。

2) 模型结构

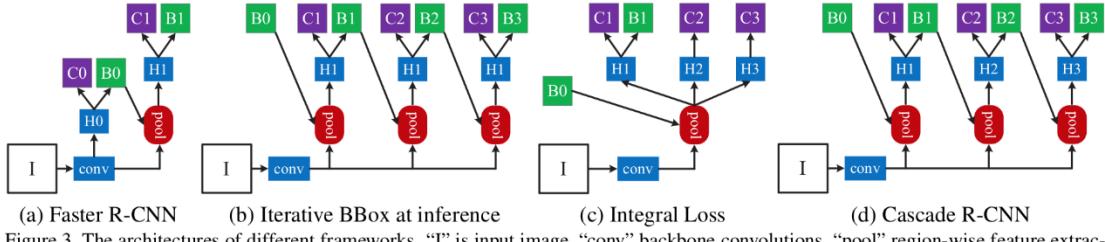


Figure 3. The architectures of different frameworks. “I” is input image, “conv” backbone convolutions, “pool” region-wise feature extraction, “H” network head, “B” bounding box, and “C” classification. “B0” is proposals in all architectures.

图 11 Cascade RCNN 的架构图

CTD(Curve Text Detector)[5]

1) 算法介绍

算法 Detecting Curve Text in the Wild: New Dataset and New Solution，发表于 arXiv:1712.02170（华南理工大学）。

论文地址：<https://arxiv.org/abs/1712.02170>

代码地址：<https://github.com/Yuliang-Liu/Curve-Text-Detector>

其 motivation 如下：传统的文本表征是参考通用目标检测，即使用 4 个点的 bounding box 来表示文本，这种方法在普通的数据集上取得了不错的效果。然而，面对弯曲文本复杂的形状，传统的四边形无法取得很好的效果，因为会引入大量的背景噪声，不利于识别任务。用多边形比四边形能更好的表征弯曲文本，所以作者提出利用 14 个等距点来表征弯取文本，完成弯曲文本检测的任务。

2) 模型结构

如下图所示，CTD 的框架结构总共分为三个分支：第一个是 text/non-text 分支，执行的是普通的二分类任务，用于判断是否存在目标 object；第二个分支，执行的是整个弯曲文本最小外接矩的 bounding box 的位置回归；第三个分支是 14 个点的坐标回归任务。包括采用类似 R-FCN 的方式进行画网格 pooling、以及用 RNN 来增加上下文信息做 smooth。

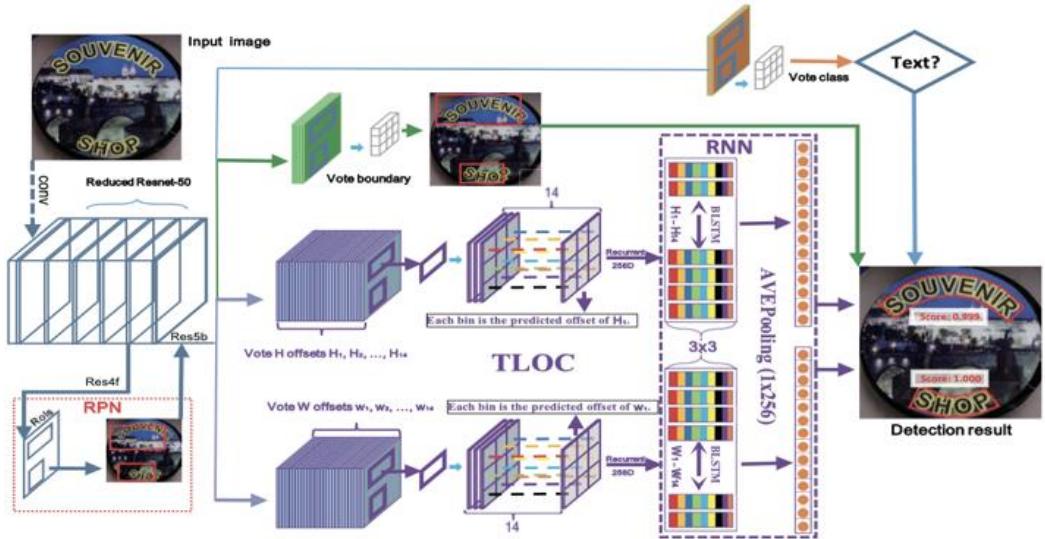


图 12 CTD 算法的网络框架图

如下图所示是 CTD 算法所采用的标注弯曲文本标注方法。一共采用 14 个点来标注弯曲文本，首先确定边界上的 1、2、3、4 序号点，随后利用等距线生成上下分别 5 个序号点。

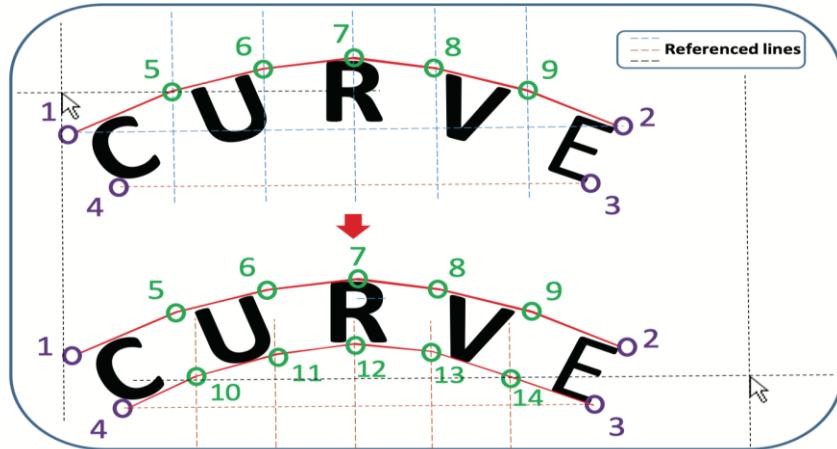


图 13 弯取文本的标注演示

EAST[6]

1) 算法介绍

算法 EAST: An Efficient and Accurate Scene Text Detector, 发表于 **CVPR2017**.

论文地址: <https://arxiv.org/pdf/1704.03155.pdf>

代码地址: <https://github.com/argman/EAST>

Motivation 如下：为了解决之前大多数方法只能检测水平朝向文本的缺陷，并且简化检测的框架，作者提出了一种高效且准确的文本检测模型，即 EAST。

2) 模型结构

(1) 架构图

EAST 是一种 one-stage 的基于检测框的字符检测算法，执行流程如下：

(i) 使用一个基础的特征提取网络，并采用 UNet 的结构作为 backbone，用于提取特征。

(ii) 在最后一个 feature map 上直接进行结果的预测。输出包含三部分，首先输出一个 score map，表示文本的热力图；其次每个点预测一个 RBOX（四个值表示该点距离四条边的距离，一个值表示旋转角度），表示一个有角度的矩形框，最后每个点预测一个任意四边形（8 个值表示该点距离四个顶点的 x, y 方向上的偏移）

(iii) 针对所有的检测框，先设置适当的阈值过滤掉不合适的框，然后使用 locality-aware NMS 步骤抑制冗余的候选框，这就得到最后的预测结果。

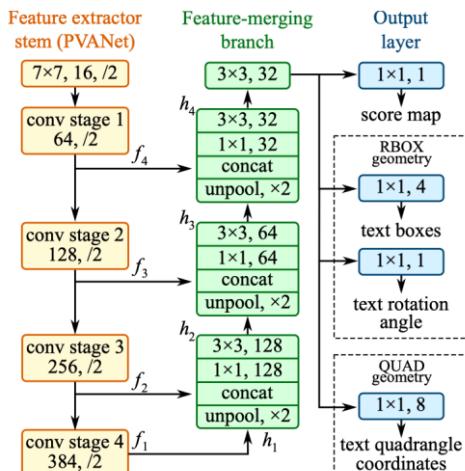


图 14 EAST 算法的网络结构图

Advanced EAST[7]

1) 算法介绍

AdvancedEAST 是 ICPR2018 比赛中公开的、针对 EAST 对长文本检测效果进行改进的算法。

代码地址：<https://github.com/huoyijie/AdvancedEAST>

2) 模型结构

(1) 架构图

如下所示是 Advanced EAST 算法的网络结构。网络前半部分的特征提取和特

特征融合层基本与 EAST 一致（关于骨干网络，由于时间较早仍为 VGG，现应使用 ResNet 等）。输出层有 7 个通道，分别是：是否是文本区域的二分类评分 **inside score**，是否属于文本框边界以及是头还是尾的二分类评分 **side vertex code**，边界/头尾像素对到该侧两个角点像素的距离 **side vertex geo**，即 $(\Delta x, \Delta y)$ 值。所有的边界像素回归的坐标的预测值的加权平均得到头或尾的短边两端的两个顶点坐标。

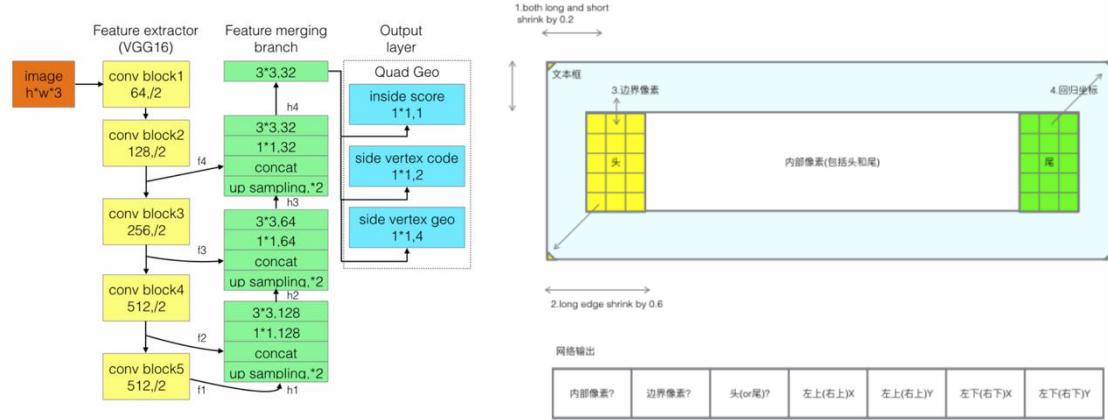


图 15 Advanced EAST 算法的网络框架图

(2) 方法细节

网络输出经过后处理得到最终预测结果，具体流程如下：

首先根据网络输出的 **inside score** 和给定阈值，得到文本区域的二值化图像；然后用如 OpenCV 中的 `connectedComponents` 的连通域函数来确定文本区域块 **region_group**；最后对每个区域块中所有头、尾像素回归的角点坐标值的加权平均，作为最终的角点坐标。四个角点坐标即可确定一个四边形文本框。

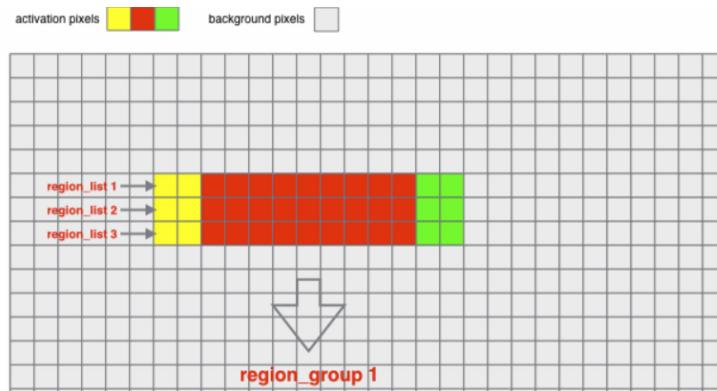


图 16 后处理步骤的图示

PSENet[8]

1) 算法介绍

算法 PSENet: Shape Robust Text Detection with Progressive Scale Expansion Network.发表于 CVPR2019.

论文地址: <https://arxiv.org/abs/1806.02559>

代码地址: <https://github.com/whai362/PSENet>

PSENet 是一种基于语义分割的字符检测方法，从像素级别上分割文字区域和背景区域。backbone 部分采用的是 Resnet+FPN 的结构，然后将得出的各个 feature map 融合用于最后的预测。其 motivation 如下：现有的基于检测框的文本检测方法很难很好地检测任意形状的文本，而且大多数基于像素的分割检测器又很难将彼此非常接近的文本实例分开。为了解决这些问题，作者提出了一种新的渐进尺度扩展网络 (PSENet)。

2) 模型结构

(1) 架构图

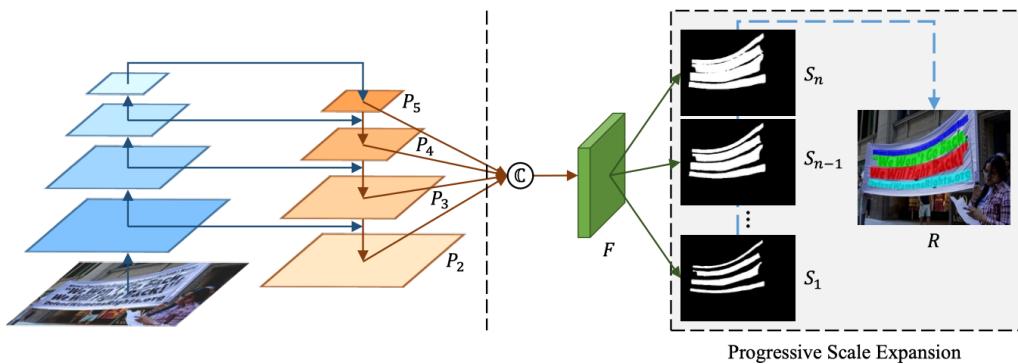


图 17 PSENet 算法的网络架构图

(2) 方法细节

PSENet 在训练时输出的结果包含多个文字区域的区域表示，从完整的文字区域，如图中 S_n ，然后逐阶段收缩文字区域 (S_{n-1} 到 S_1)，这种方法可以增大文字块之间的距离，易于区分粘连的文字块。预测时，以最小的文字块为基础，开始向外不断扩张，最后得到完整的文字区域。扩张算法如下图所示。

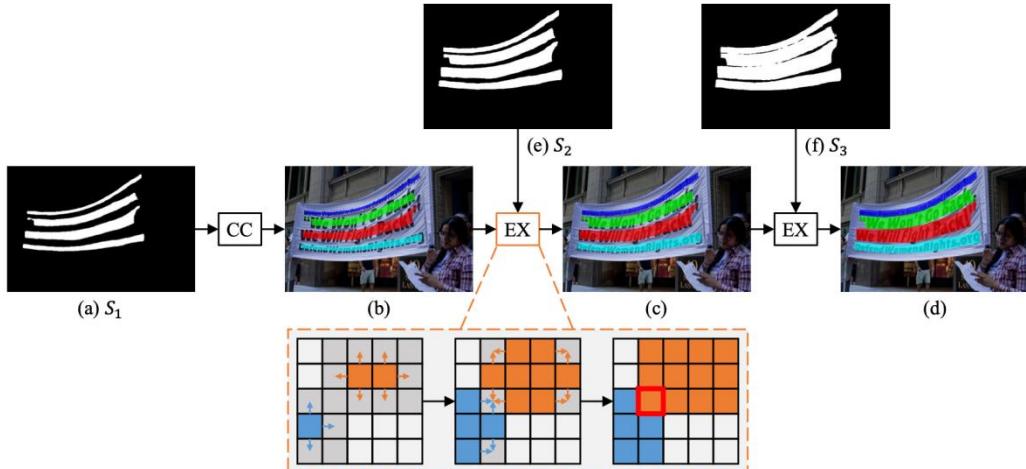


图 18 扩张算法

PANet[9]

1) 算法介绍

文章标题为 Path Aggregation Network for Instance Segmentation, 发表于 CVPR 2018 (香港大学),

2) 模型结构

(1) 架构图

框架如下:

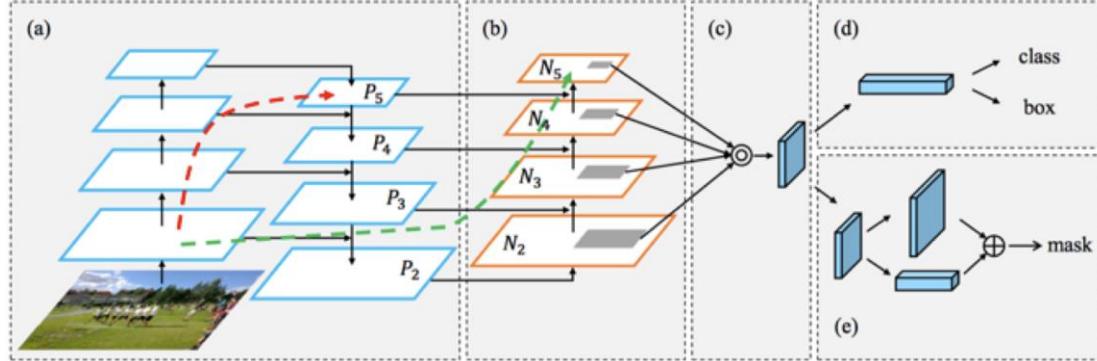


图 19 PANet 的网络架构图

(2) 方法细节

该文章是基于 Mask-RCNN 进行改进来做实例分割 (Instance Segmentation) 的网络, PANet 与 Mask-RCNN 的区别有四点:

(a) 添加了类似 FPN 结构的 Bottom-up 模块 (FPN 使用 Top-down 的结构将深层语义特征与浅层语义特征相结合, 使得浅层特征的语义信息更加丰富), Bottom-up 结构对特征图的语义进行再度提取, 能够丰富语义信息, 下图为

Bottom-up 的模块:

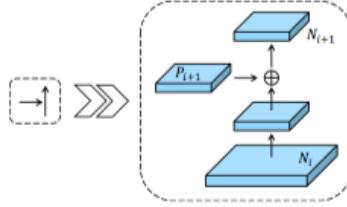


图 20 Bottom-up 模块图示

(b) 如 PANet 网络架构图中的红色虚线和绿色虚线, PANet 将 $1/4$ 大小的特征图中的特征进行 max-pooling 操作下采样之后直接 add 到深层特征图 P_5 和 N_5 上, 使得深层的语义特征使用浅层通用的特征, 浅层的特征通常对轮廓和物体的部分区域响应值较高, 同时拥有更精确的空间位置信息, 使得网络能够对空间位置信息更加敏感。

(c) 添加了 Adaptive Feature Pooling 模块。FPN 论文中将不同大小范围的 proposals 赋予不同大小的特征图, 仅仅相差十几个像素的 Proposal 会被赋给不同大小的特征图, 然而这两个 proposals 非常的相似, 作者通过实验发现不应该简单的根据大小来将 proposals 赋予不同的特征图, 一次使用 Adaptive Feature Pooling 模块来将 proposals 进行重新分配, 使得即使小的 proposals 也会使用深层的特征图的信息, 对于大的亦是如此。

(d) 在 Mask Branch 添加了一个全连接层分支。卷积方式能够基于不同空间位置的共享参数和局部感受野, 来对每个像素进行预测, 而全连接层在不同的空间位置进行预测, 对位置比较敏感, 因此可以更好的识别不同的空间位置。将卷积特征和全连接的特征有助于得到更好的 mask prediction, 因此有助于改善效果。模块结构如下, 不同的:

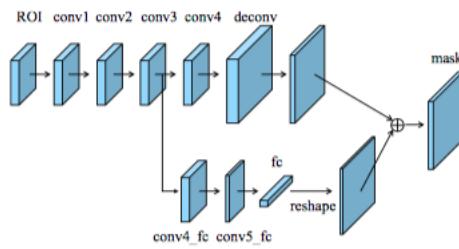


图 21 不同的部分

SPCNet[10]

1) 算法介绍

文章标题 Scene Text Detection with Supervised Pyramid Context Network, 文章发表于 CVPR 2018 (中国科学与技术大学和同济大学)。

2) 模型结构

(1) 架构图

整体的网络架构图下图所示：

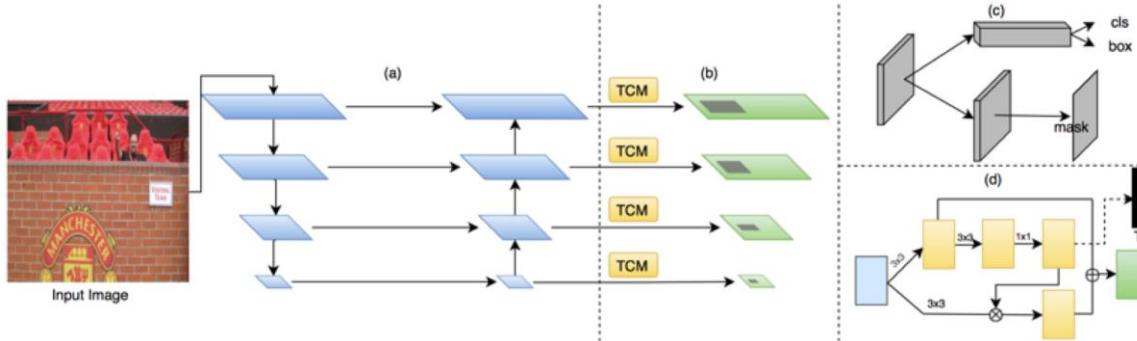


图 22 SPCNet 的网络架构图

(2) 方法细节

该文章基于 Mask-RCNN 进行改进，在 Mask-RCNN 的 FPN 模块的后面添加了 TCM 模块，来做特征图的 pixel 级别的 attention，对每个特征图上的每个像素赋予不同的权重。

改进主要有两点：

(a) 添加了 TCM (Text Context Module) 模块。FPN 每个 feature map 后面接一个 TCM 模块，对每个特征图，学习一个全图的语义分割 (Semantic Segmentation)，仅有 text 和 non-text 两种类别，包括了整张输入图片上的所有 text 区域，使得特征图对于 text 区域的激活值更高)，与 GT 求 Loss，将学得的语义分割 heatmap 广播到和 x 相同维度，然后与之相乘，将得到的特征图与上面经过 1 个 3×3 卷积支路的 x 相加，使得属于 Object 的像素点激活值更高，也就是给 Object 赋予更高的权重，完成 feature map 的 attention 机制。TCM 模块的结构如下图所示：

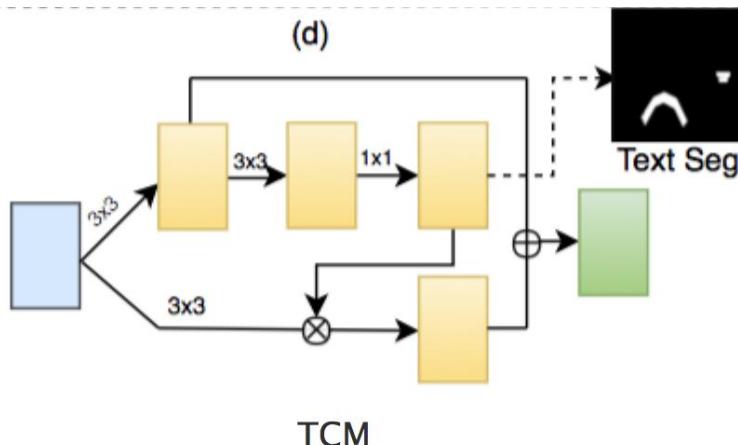


图 23 TCM 模块图示

(b)添加了 Re-Score Mechanism。作者觉得仅仅使用 Bbox 类别置信度的话，包含了许多背景信息在里面，因此作者使用 Instance Segmentation 得到的区域，映射回相应的特征图区域，求得像素激活值的平均大小，与 bbox 类别置信度使用类似加权的方式求得新的类别置信度分数，以此作为最终的分类置信度。

Re-Score 机制的示意图如下，详细请阅读论文：

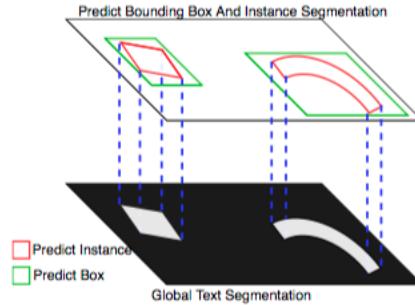


图 24 Re-Score 机制图示

DenseASPP[11]

1) 算法介绍

该文章的标题为 DenseASPP for Semantic Segmentation in Street Scenes，发表于 CVPR 2018 (DeepMotion)。

2) 模型结构

(1) 架构图

整体的网络架构如下图所示：

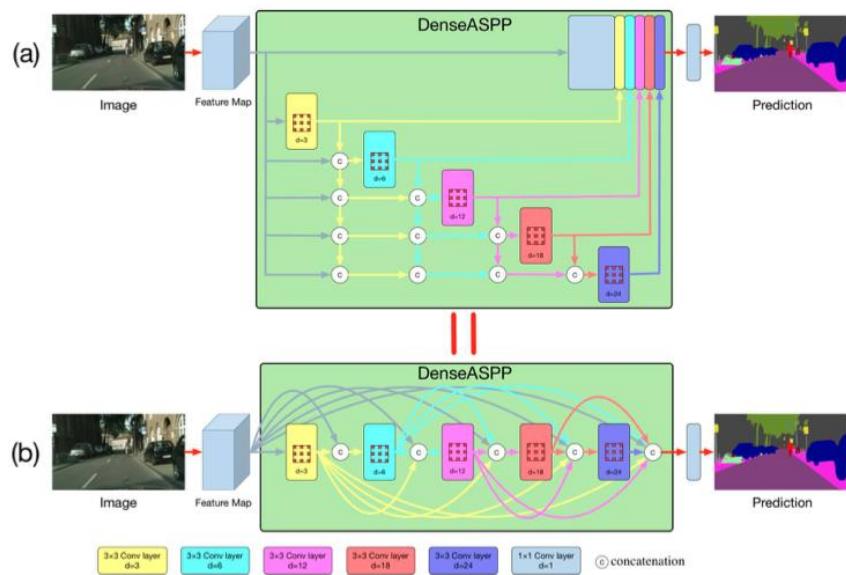


图 25 DenseASPP 网络架构图

(2) 方法细节

该论文是用来做语义分割的，进行像素级别的分类，因此只在 $1/4$ 大小的特征图后面，使用了不同大小的空洞率（dilate rate）的卷积核进行卷积，而后使用 DenseNet 的操作将其 concat 起来，使得最终得到的特征图拥有不同的感受野（更大的空洞率能够获得更大的感受野，详情请了解空洞卷积）。下图展示了结合不同大小空洞率的特征之后的感受野 k：

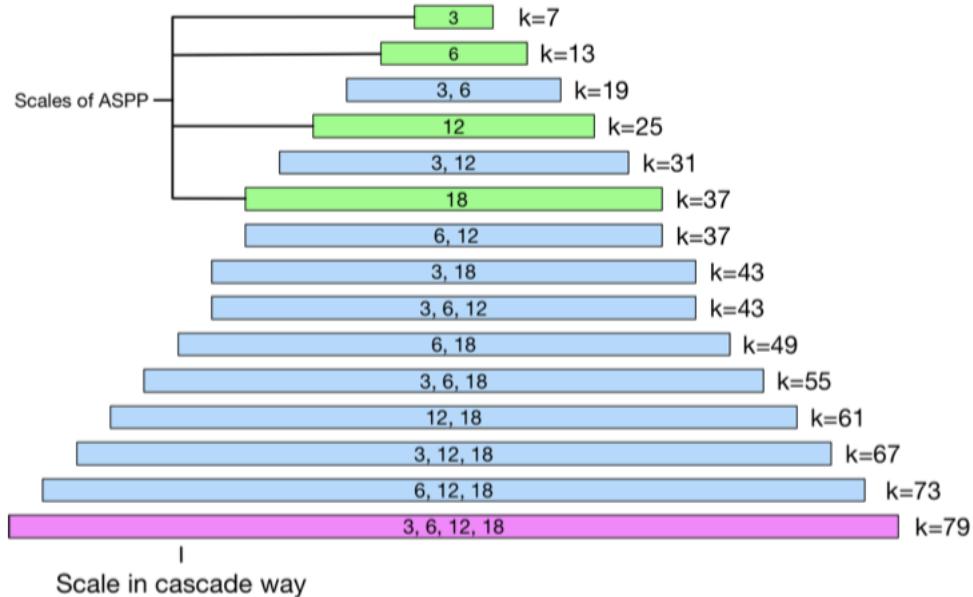


图 26 结合不同大小空洞率的特征之后的感受野图示

M2Det[12]

1) 算法介绍

该论文标题为：A Single-Shot Object Detector based on Multi-Level Feature Pyramid Network，发表于 [AAAI 2019](#)（中国科学与技术大学和北京大学）。

2) 模型结构

(1) 架构图

整体架构如下图所示：

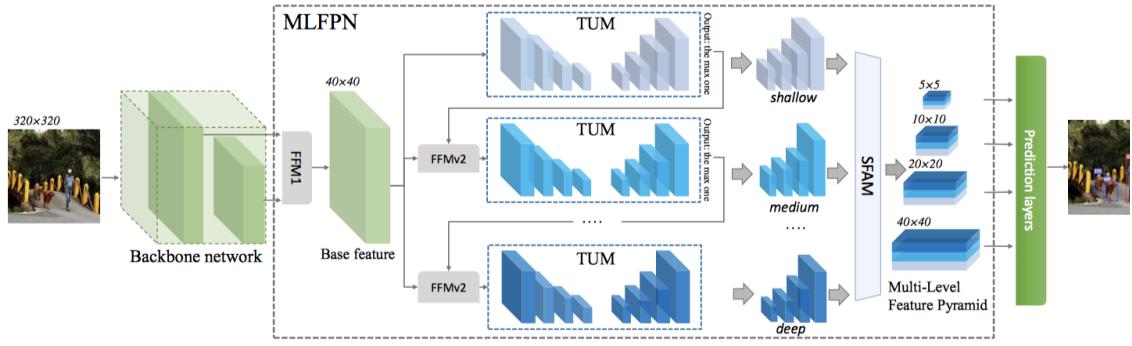


图 27 M2Det 的网络架构图

该论文最突出的贡献在于 TUM (Thinned U-shape Module) 模块，将特征图通过下采样不断减小，再通过反卷积不断变大，形成沙漏结构，然后将这个过程重复多次（每次都会重新 concat Base Feature 到输出上），将 TUM 多次提取得到的特征图按相应大小的特征图进行 concat（经过了 SE-Block 对通道进行 attention，即 SFAM 模块），再进行 Bbox 和 class 的预测。

SFAM 模块如下图所示：

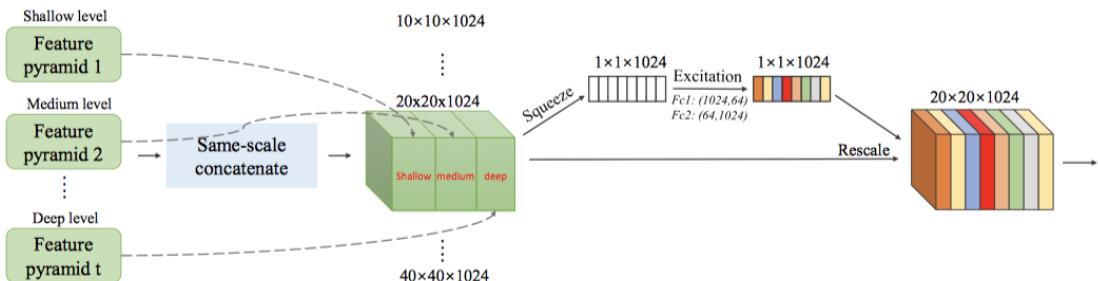


图 28 SFAM 模块图示

FFMv1、FFMv2 和 TUM 模块如下图所示：

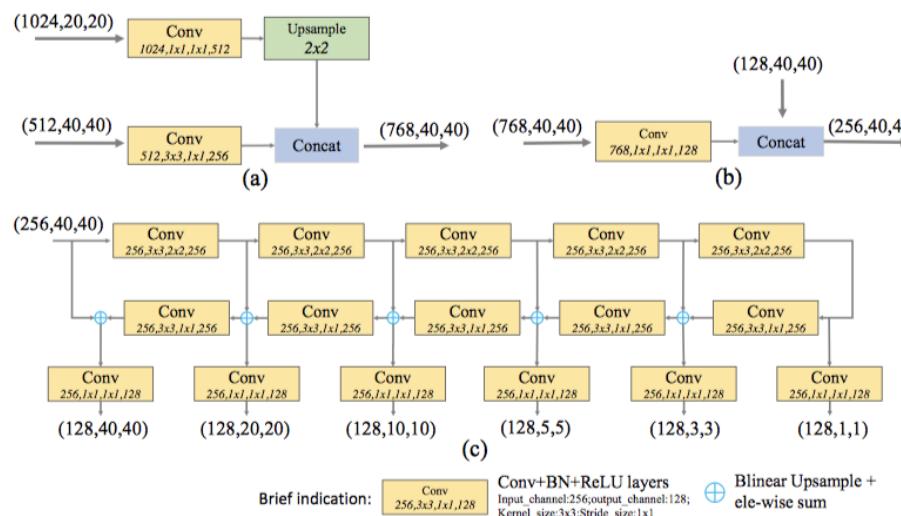


图 29 FFMv1、FFMv2 和 TUM 模块图示

Deformable Convolution[13]

1) 算法介绍

该文章的标题为: Deformable Convolutional Networks, 发表于 [ICCV 2017](#)。

2) 模型结构

(1) 架构图

该文章模块结构如下, 对于 conv 的采样点学习偏移, 使得卷积层学到了偏移, 不再是规则的矩形结构, 而是根据学习到的偏移, 在相应的偏移过的位置进行卷积操作。

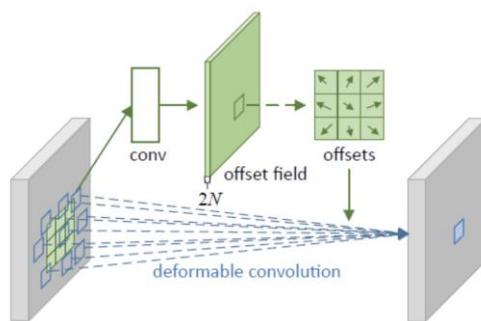


图 30 可变形卷积图示

(2) 方法细节

后续的改进 Deformable conv v2 添加了 Deformable RoI Pooling 等操作, 如下图所示:

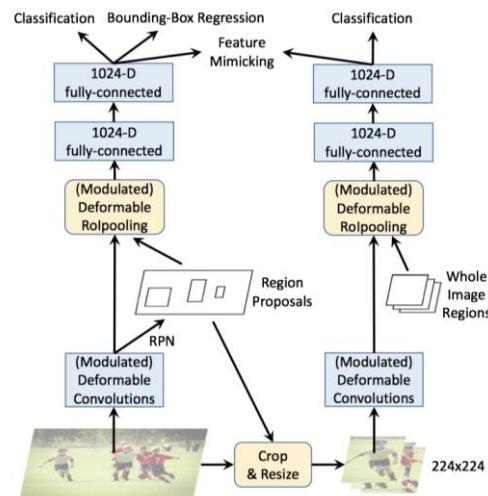


图 31 改进后的可变形卷积网络图示

不同论文的组合架构：

1) PANet + SPCNet + MaskRCNN:

网络架构如下图所示：

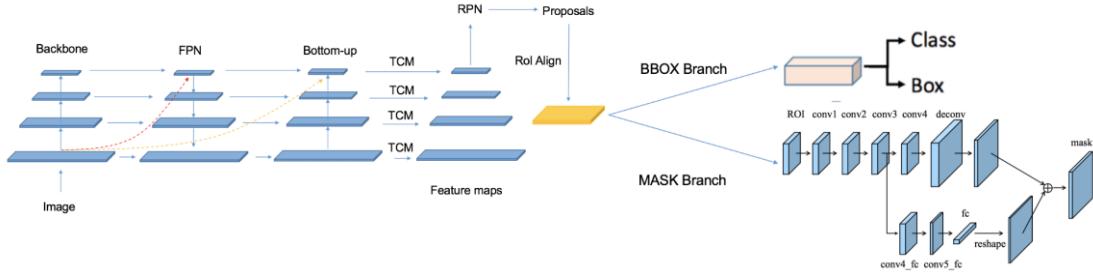


图 32 PANet + SPCNet + MaskRCNN 的网络架构图

该架构结合了 PANet、SPCNet 和 MaskRCNN，因为这些模块的结构互不冲突，因此可以很方便的相互嵌入，唯一的问题就在于以何种方式对模块进行堆叠和嵌入，而不是简单的嵌套，同时，不同的参数设置会有不同的影响，因为不同的结构有不同的最优参数设置。按照如上图所示的结构进行网络架构的改进，取得了不错的效果。

2) Cascade MaskRCNN + Deformable CONV:

网络架构如下图所示：

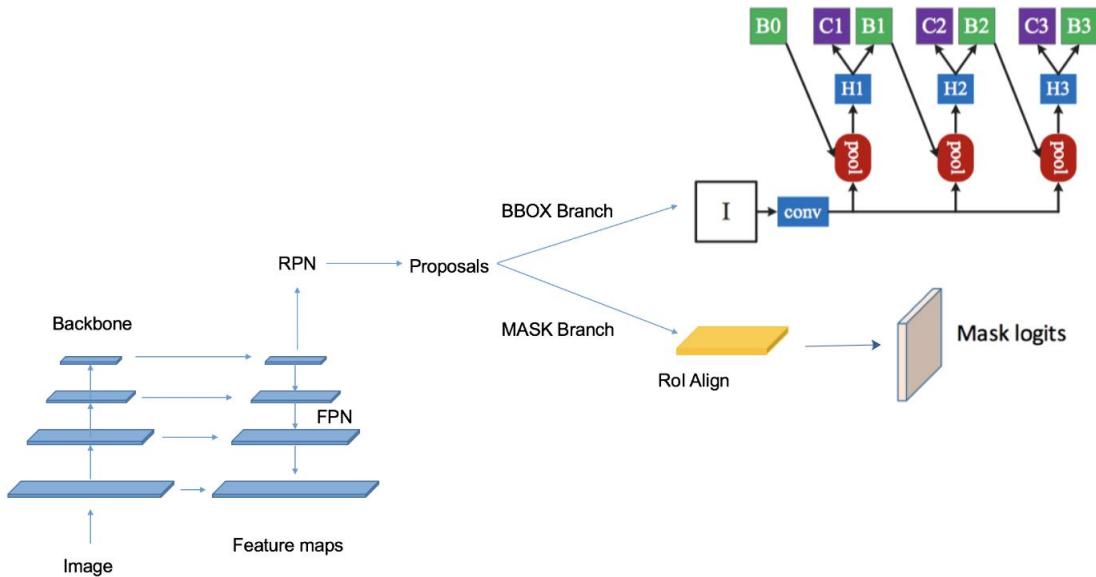


图 33 Cascade MaskRCNN + Deformable CONV 的网络架构图

该网络仅是普通的 Cascade MaskRCNN 架构，然后再 backbone 上添加了可变形卷积，提升效果相对于 Cascade MaskRCNN 较为明显，后续添加了 Deformable

RoI Pooling，但相对于未添加效果变化不大，当然这个和数据集也有关系

消融实验分析（Ablation Study）

1) Textsnake 算法相关实验

学习率初始值设置为 0.001，学习率下降策略统一为第 10000 步和第 30000 步各乘以 0.1，batchsize 设为 10，输入大小统一为 512*512，更换网络结构的部分实验结果如下：

表一：实验结果

	备注	Recall	Precision	F-measure
Textsnake	Baseline	0.7690	0.4694	0.5829
Textsnake	更换网络结 构为 Global Convolution Network	0.7357	0.5840	0.6512

2) MaskRCNN 及其变种相关实验

学习率初始值设置为 0.002，其他参数基本沿用 Maskrcnn 的默认值，Backbone 分别选用 ResNet50、ResNet101、ResNeXt101。部分实验结果如下：

表二：实验结果

	Backbone	Recall	Precision	F-measure
Mask rcnn	ResNet50	0.7317	0.8400	0.7821
Mscring rcnn	ResNet50	0.7234	0.8468	0.7803
Mscring rcnn	ResNet101	0.7165	0.8525	0.7786
Cascade rcnn	ResNeXt101	0.7460	0.8486	0.7940

学习率初始值设置为 0.002，其他参数基本沿用 Mask scoring rcnn 的默认值，开启和关闭 Mask IoU Head，测试网络召回率的变化，Backbone 采用 ResNet50，迭代次数 220000。部分实验结果如下：

表三：实验结果

	是否开启 Mask IoU Head	Recall	Precision	F-measure
Mscring rcnn	开启	0.7314	0.8374	0.7819
Mscring rcnn	关闭	0.7151	0.8431	0.7739

学习率初始值设置为 0.0025，对于不同的数据集，有不同的训练策略：

(1) LSVT：在 120k, 180k 迭代时衰减 1 / 10，总共跑 240k 轮迭代（约 9 个 epoch）

(2) ArT：在 60k, 120k 迭代时衰减 1 / 10，总共跑 180k 轮迭代（约 42 个 epoch）

其他参数基本沿用 MaskRCNN 的默认值，分别添加不同论文中的的模块进行不同的实验。部分实验结果如下：

表四：实验结果（LSVT，无 ArT）

	实验设置	Recall	Precision	F-measure
MaskRCNN (baseline)	Backbone 为 ResNet50 (后续都是 Resnet50)	0.7622	0.875	0.8147
MaskRCNN+ PANet	(1) 添加 fully-connected fusion 模块			+ 0.3
MaskRCNN+ PANet	(2) 添加 bottom-up 结构以及将浅层 语义特征融入深层语义特征图上			+ 0.2
MaskRCNN+ PANet	添加 (1) 和 (2)			+ 0.4
MaskRCNN+ SPCNet	添加论文中的 TCM 模块			+ 0.5
MaskRCNN+ M2Det	将 M2Det 的 TUM 模块放到 backbone 前两层 feature map 后面，输出的 feature maps 后面添加 FPN 模块减小通 道数且添加语义信息			略微下降
MaskRCNN+ M2Det	将 M2Det 的 TUM 模块放到 backbone 前两层 feature map 后面（该两层 feature map 将更深层次的语义 add 到 浅层补充语义），输出的 feature maps 后面添加 1×1 的卷积减小通道数。			+ 0.6 (显存消 耗过大，无法 使用，而且非 常耗时)
MaskRCNN+ DenseASPP	将 DenseASPP 模块加入到 FPN 后，每 个特征图都经过 DenseASPP 模块，共 享五个特征图的参数。			降 1.0

MaskRCNN+	放在 FPN 之前, resnet 之后, 仅对 stride=4, 8 的特征层使用 denseASPP 模块, 共享特征。	效果变化不大
MaskRCNN+ DenseASPP	在 FPN 之后, 只对 stride=4, 8 的两个特征图分别使用 denseASPP 模块, 不共享特征。	小物体和中等物体 recall 增加 1-2 个点, 大物体下降 2 个点
MaskRCNN+ DenseASPP	对每个特征图使用 DenseASPP, 且将原 paper 中每个 dilate rate 输出 256 通道变成 64 通道, 同时 不共享特征。	效果变化不大
MaskRCNN+ PANet+ SPCNet	PANet 的 bottom-up 后面接 SPCNet	+ 1.4
MaskRCNN+ PANet + DenseASPP	PANet 的 bottom-up 后面接 DenseASPP	+ 0.5
MaskRCNN+ PANet+ SPCNet + DenseASPP	PANet 的 bottom-up 后面接 SPCNet, SPC 后面接 DenseASPP	+ 0.8
MaskRCNN+ PANet+SPCNet (最好效果)	PANet 的 bottom-up 后面接 SPCNet	0.8029 0.8755 0.8376

表五：实验结果 (LSVT)

实验设置	Recall	Precision	F1-measure
Cascade MaskRCNN Backbone 为 (Baseline) ResNeXt101-64-4d	0.8029	0.8755	0.8376

Cascade MaskRCNN	在 backbone 上面添加 + Deformable Conv	deformable conv	降 0.3
Cascade MaskRCNN	在 backbone 上面添加 + Deformable Conv	deformable conv, 在 rcnn head 上使用 DeformableROIPooling	降 0.4
Cascade MaskRCNN	添加 PANet 和 +PANet+ DenseASPP	DenseASPP(未进行调 + Deformable Conv 试, 效果不佳)	变化不大
Cascade MaskRCNN	在 backbone 上面添加 + Deformable Conv	0.805 deformable conv (最终效果)	0.8923 0.8464

3) EAST 相关实验

Batchsize 设为 14, 训练总步数为 10000, 其他参数于沿用 EAST 的源码默认值。部分实验结果如下:

表四: 实验结果

模型	backbone	数据集	Precision	Recall	F-measure	实验
EAST-baseline	Res50+Unet	LSVT	0.7745	0.5962	0.6737	源代码
EAST-V1	Res50+Unet	LSVT	0.7765	0.6043	0.6769	数据增强
EAST-V2	Res50+Unet	LSVT	0.7616	0.6091	0.6776	调整 nms 过程
EAST-V3	Res50+Unet	LSVT	0.7652	0.6412	0.6978	改进 gt 的生成方法
EAST-V4	Res50+Unet	LSVT	0.7287	0.6174	0.6684	进行多尺度预测

4) PSENet 相关实验

Batchsize 设为 14, 训练总 epoch 为 20, 其他参数于沿用 PSENet 的源码默认值。部分实验结果如下:

表五: 实验结果

模型	backbone	数据集	Precision	Recall	F-measure	实验
PSENet-Baseline	Res50+FPN	LSVT	0.6886	0.5803	0.6298	源代码
PSENet-V1	Resxnet	LSVT	0.794	0.6534	0.7169	更改 backbone,

所发现的问题

- 当图片中存在紧密的多行斜向文字时，使用预测水平框的 Mask R-CNN 的话，会容易导致位置处于中间的水平框（与上下水平框 IoU 较大）会被抑制掉，从而导致召回率下降。



图 34 密集斜向文本容易有水平框框被抑制

- 对于使用旋转框，例如 (x,y,w,h,angle) ，来描述文本框的算法（如 RRPN、EAST 等），角度 angle 对预测框与实际框之间的 IoU 的影响要大于另外 4 个。相比于 $xywh$ ，角度的同等变化对最终预测框的精确度影响要更大，这也就导致我们对角度 angle 的预测要更为精准。目前的做法是训练时降低 angle_loss 的比重，但这样并没有从实际上解决这个问题。最好的方法应该是修改对旋转框的描述方式，但目前还没想到特别好的其他描述方式（实在没办法的话只能用 4 个点的 8 个坐标来描述？）。
- 使用 Mask R-CNN 算法来做检测任务的话，有时会存在一个框内框住多个物体（多行文字）的情况，之前尝试过使用二次分割的方法来解决，即将预测得到的框切割下来再输进去模型里再跑一次，但由于切割下来的图片在大小和数据分布上都与原模型的训练数据差别较大，大概需要单独训练一个专门处理切割后图片的模型用于二次分割。
- 分割方法得到的结果无法准确标出文字周边的 16 个点，稍稍影响后续的识别性能。即使目前能过获得紧凑的弯曲文本检测结果，但识别部分仍没有很好的利

用这个弯曲文本检测结果的方法。

5. Advanced EAST 的缺点，同时也是 one-stage 方法的共通缺点，性能依赖于第一步文本区域分类，如果连通域分析结果有断裂或黏连，都会直接影响后处理中文本框的生成。

6. 如下左图所示，这种英文单词的标注格式是分开的，而检测的时候容易用一个检测框把好几个英文单词框出来，导致错误的结果。如下右图这种，也是标注的时候将单词分开，检测的时候容易检测到一起。或者是较长的 GT 文本，容易用好几个框将文本分割开来。（红色的是多边形 Ground Truth 的最小外接矩形的 Ground Truth（即 Bbox 的 Ground Truth），其他颜色的是不同的 Bbox 结果和分割结果，上面的文本是 confidence。）



图 35 由于标注问题而产生的误检

7. 如下图这样的，文本过小，有些小文本 GT 却是分开的，而这种分开的小文本，极其容易使用一个框把它们全部框起来，如最上面的三个框展示的。这种一个框检测出多个 GT 是非常常见的。

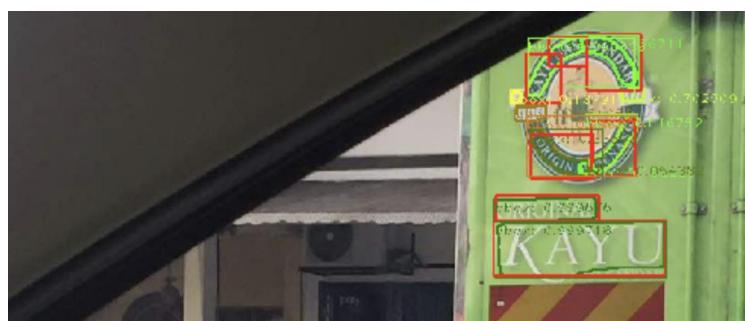


图 36 由于文本过小而带来的问题

8. 如下左图，有点乱，但只要看和 GT 匹配的就好了。图上两个电话号码 GT 是标在一起的，而这种距离非常大，容易分开检测，导致如上图的错误结果。同时蓝色大字“斯图 陶瓷”容易使用大框把它们框起来。底部的四个红色的字因为光照和分隔太开的原因，难以完全检测到。而对于下右图这个 GT，不同的纹理在标注的时候把它们标成一个 GT，导致检测的时候容易对它们进行分开检测，产生两个框，导致 FN 增加。

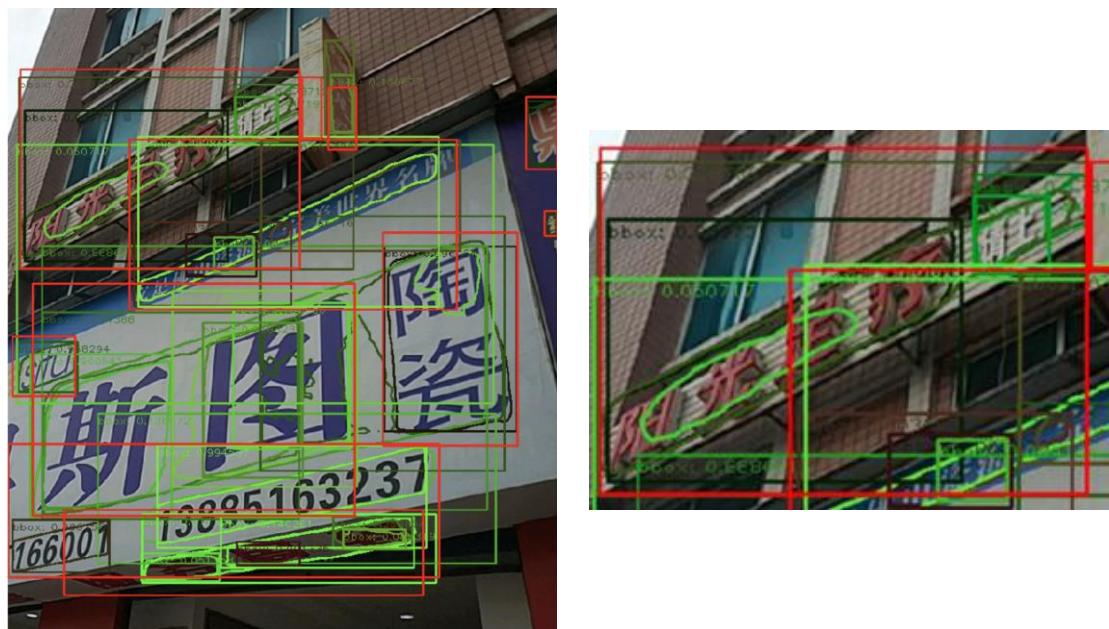


图 37 由于文字距离过大而带来的误检

涉及的创新点

1. 对于由 A 任务衍生出的复杂任务 B，使用 A 进行预训练，可以在 B 上有非常大的提升。例如 A 为普通的文本检测，B 为弯曲文本检测。
2. 在训练文本数据时，将 mask 分支的类别数设置为多个，比单单设置 2 个（背景、文本）效果好得多。
3. Re-inference
 - 1) 任务描述
 - 目前 Mask Score R-CNN 的检测结果 recall 值很低，需要进行后续处理提高 recall 分数；
 - 主要思想是先让网络产生所有的检测结果，将结果送入当下分数最高的检测网络中再进行一次 inference；

- 将 Re-inference 的结果坐标重新映射回原来的图片
- 2) 方法描述
- 将网络 NMS 部分的置信度阈值设为 0, 即保留所有的预测结果, 但是进行 iou 过滤;
 - 对每一个结果对应的原图区域 crop 出来, 保存;
 - 把 crop 生成出来的所有图片重新送入目前分数最高的网络进行跑一次 inference, 得到基于 crop 图片坐标系的 mask 和 bbx 坐标
 - 将 re-inference 的结果对应到原来图片的坐标系, 重新投影 mask 和 bbx, 得到二次推断的最终结果。
 - 将二次推断的结果和一次推断的结果进行 nms。
4. 使用 Top-down 和 bottom-up 结构, 设置不同的 Dilate rate, 然后 concat 相同大小的特征图, 使用不同大小的感受野捕获不同尺度的 Object。(效果未经验证, 实验还在进行)

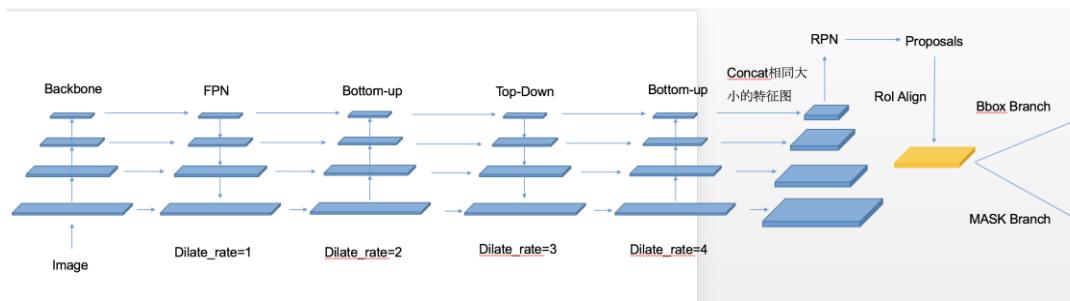


图 38 网络结构图示

识别任务

竞赛

本次识别组参加的比赛共有两个：ICDAR2019-LSVT 和 ICDAR2019-ArT。

ICDAR2019-LSVT

- (1) 任务：Task 2. End-to-end text spotting
- (2) 对于识别任务来说，全标注的一共有 50000 张；其中，划分训练集 27000 张，测试集 3000 张；正式测试 2 万张。

数据集中图片如图 1.jpg 所示，对应的 groundtruth 格式如图 2.jpg 所示。



1.jpg

```
{  
    "transcription": "TEL:",  
    "points": [  
        [ 25,  
          359  
        ],  
        [ 71,  
          364  
        ],  
        [ 70,  
          385  
        ],  
        [ 25,  
          382  
        ]  
    ],  
    "illegibility": false  
},  
2.jpg
```

(3) 数据集分析

分析：总体来说难度中等，主要以广告牌为主

难点：A. 广告字体形式变换较多

B. 存在一定的竖排文本

C. 存在少部分的多朝向字体（波浪、扇形等）

ICDAR2019-ArT

(1) 任务: Task 2. Scene Text Recognition

(2) 通过数据统计,有接近 1/4 的图片文本都是 Arbitrary-Shaped Text,一共 50029 张; 其中我们划分训练集 45000 张, 测试集 5029 张, 比例为 9:1; 全部已经裁剪。

数据集中原图如图 3.jpg 所示, 裁剪后的图片如图 4.jpg 所示。



3.jpg



4.jpg

(3) 数据集分析

分析：总体来说难度较大，主要以广告牌、商标为主，接近 1/4 都是任意朝向的文本，还存在部分手写体，以及分辨率较低的图片。

难点：A. 广告字体形式变换较多

B. 多朝向字体较多

采用模型

数据处理部分工作：

(1) 数据集处理：

1) LSVT：根据官方配对的 json 文件，对数据集中每个整图中需要进行识别的小图，根据坐标进行剪切旋转。由于坐标数量为 4-12 个点，并且不都是规整的平行四边形，采用坐标点的最小外接矩形策略进行剪切旋转，剪切效果如下图所示。

2) ArT：ArT 官方数据集不需要剪切，可直接使用。

3) 对两个数据集均选取高宽比阈值为 2 筛选出竖排图片。

4) 对两个数据集均根据 9: 1 的比例随机分出训练集和测试集。

5) 将 json 文件中 label 整理成 txt 文件以便后续训练模型使用。

6) LSVT 和 ArT 两个数据集划分训练集和测试集以及筛选出竖排图片之后数据量分布如下图，基本满足训练：测试=9: 1 的比例。



图 12. 从整图剪切需要识别的小图示例



图 13. 采用最小外接矩形剪切相比原来不规则四边形的效果。绿框为给出的坐标点，大图中红框为选取 4 个坐标点得到的框，在这种多于 4 点的情况下容易错误剪切，下图为取最小外接矩形得到的剪切效果。

图 14. 上图为原图，下图为剪切后的图。正确剪切和旋转可以减少干扰，更利于模型训练。

表 14. 两个数据集划分训练集和测试集以及筛选出竖排图片之后数据量分布

dataset	train		test	
	normal	vertical	normal	vertical
	41275	3428	4601	364
ArT	83%	7%	9%	1%
	192062	26980	21466	3129
LSVT	79%	11%	9%	1%

(2) 数据生成：生成数据包含背景/字体/大小/颜色/干扰/旋转角度等的随机选取，尽可能模拟真实场景图片。

- 1) 地址数据 5 万，语料源于真实地址构造方法，从市镇一直具体到门牌号；
- 2) 小广告 5 千，语料中 300 条源于张培尧师姐小广告项目语料，还有 700 条源于网上淘宝起名大全中收集的商店名，一共 1000 条语料；生成效果及真实比赛数据对比如下图所示。

Generate samples



True training samples



图 15. 生成的地址图片跟 ArT 数据集中图片对比

Generate samples

阿尔泰科技有限公司

携手三百四十九年同仁堂

鱼腥草提取物

高纯度小分子低聚肽

True training samples

上海栖格麦物业管理有限公司

上海惠赛实业有限公司

上海青浦香花房产经营有限公司

启德教育 出国留学语言培训

图 16. 生成的地址图片跟 ArT 数据集中图片对比

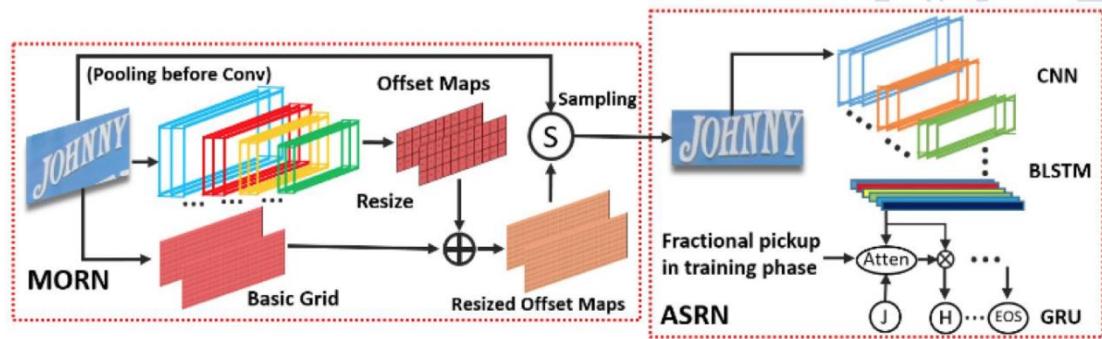
(3) 数据集收集及打包上传:

- 1) icdar2013;
- 2) icdar2015;
- 3) RCTW;
- 4) ICPR;
- 5) ICDAR2015-TRW;
- 6) 暴恐横幅（源于褚倩云师姐暴恐项目）；

MORAN

- (1) 论文地址: <https://arxiv.org/abs/1901.03003?context=cs.CV>
- (2) 源代码地址: https://github.com/Canjie-Luo/MORAN_v2

(3) 原理: MORAN 由两个部分组成: 一个是弯曲矫正网络 MORN, 一个是识别网络 ASRN。MORN(Multi-Object Rectification Network) 网络定义了一个从输入图像坐标 (x_1, y_1) 到输出图像坐标 (x_2, y_2) 的可微分映射。可微保证了可用过梯度方法训练。先将原图划分为几个部分 (3×11), 然后利用了一个 5 层 CNN 网络学习到了每个部分的 offset maps, 再将 offset maps 跟原图的每个部分加和, 利用双线性插值缩放到原图大小。通过矫正后输入图的 (x_1, y_1) 点的像素变为输出图的 (x_2, y_2) 点的像素。ASRN(Attention-based Sequence Recognition Network) 网络的主要结构就是常见的 CNN (论文 7 层 CNN, 代码 21 层 resnet, $1+5 \times 4=21$ 层) +2 层 BLSTM 结构。模型结构图如图 8 所示。



(4) 模型的亮点在于:

- 1) MORN 纠正模块采用弱监督的方式, 不需要额外的人工标注, 可以直接适用于本次比赛识别任务。
- 2) 针对现有的基于注意力的方法无法精确对齐特征区域与目标, 采用 fractional pickup 训练方法, 也即在训练阶段, 对特征图的不同部分采用不同尺度的拉伸, 在每次迭代中随机改变特征区域, 加强 ASRN 对变形文本的鲁棒性。
- 3) 采用 curriculum learning strategy, 也即分段训练增加模型训练的稳定性。

(5) 改进思路:

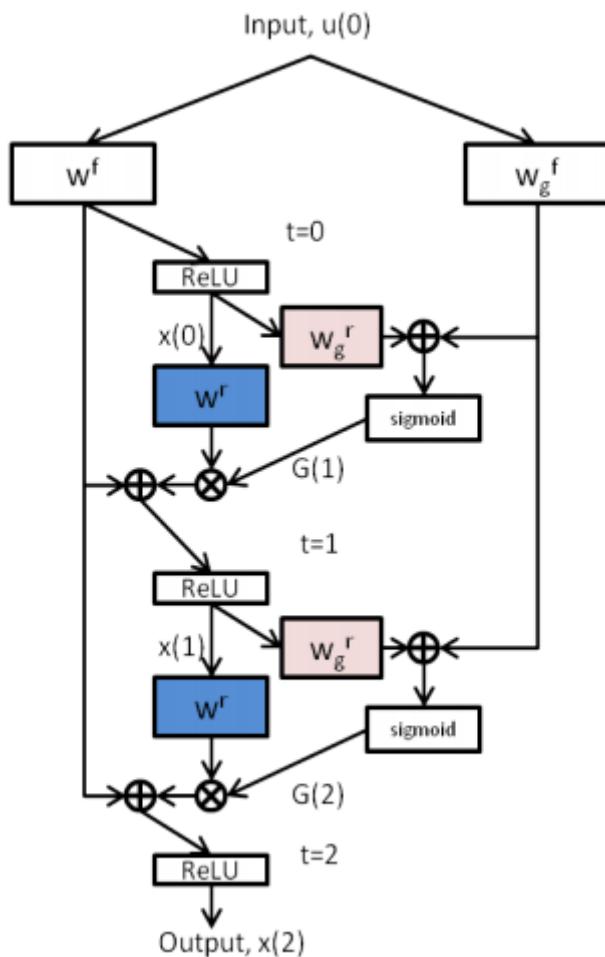
- 1) 现有模型仅针对英文字符和数字进行识别。
- 2) 现有的源码还没有实现 morn 部分将输入图片切成 33 个图片块并分别预测每个块的形变。
- 3) 现有的源码预测的形变仅有 y 方向, 这跟文章中说的输出 x 方向和 y 方向形变存在差异。
- 4) 针对 MORN 和 ASRN 部分的 CNN 进行改进, 换成其他更深的网络, 以获得更好的特征提取能力。需要提到的是采用过 Octave Convolution, 原文亮点是预设了通信中的高低频信号, 通过压缩低频信号来加速。
- 5) 尝试采用多个 CNN 提取特征并进行融合。

- 6) 对负责纠正弯曲文本的 morn 模块进行优化, 如换成其它的 tps 等纠正模块等, 改善整个网络的纠正效果。
 - 7) 对训练网络的方法/优化器/学习率等进行调优。
 - 8) 进行数据增强等数据方面的改进。
 - 9) 多个模型融合。
- 10) 加入 spatial attention, 对 feature map 进行空间编码, 融入空间的监督信息。

GRCNN

论文题目是《Gated Recurrent Convolutional Neural Network for OCR》，发表于 NeurIPS 2017，这篇论文提出并实现了核心组件 Gated Recurrent Convolutional Layer (GRCL) 来对输入的图像进行序列特征表示的提取，GRCL 包含具有循环连接的卷积层和用于控制上下文信息并平衡前馈信息和循环信息的门控。

整体算法是将特征提取、序列建模、序列转录集合到一个端到端、可训练的模型中，其中特征提取层由 GRCL 组成，提取输入图像的特征；在特征提取层的上面是 BiLSTM 来进行序列建模，最后一个部分是用 CTC 将每个时间序列的预测值转换为最后的序列输出。由此实现了一个从输入图片识别出图片中的字符序列的框架。



消融实验分析 (Ablation Study)

MORAN 实验部分：

实验部分如数据集无特殊说明均为 ArT 数据集。

(6.1) 针对 MORN 和 ASRN 部分的 CNN 的改进。

1) 将 MORN 的 CNN 部分进行替换。MORN-N 表示将 MORN 部分的 CNN 替换为 N。

表 1. 针对 MORN 的 CNN 部分进行改进的实验结果

模型	词准确率	编辑距离
----	------	------

MORAN(original)	46. 25%	0. 2940
MORN-SECNN	47. 72%	0. 2833
MORN-SE_ResNet18	47. 77%	0. 2813
MORN-SE_ResNet34	48. 22%	0. 2787
MORN-SE_ResNet50	48. 38%	0. 2775
MORN-DenseNet121	47. 88%	0. 2824
MORN-DenseNet169	48. 90%	0. 2776
MORN-DenseNet201	47. 22%	0. 2892

可以看到，将 MORN 部分原来的简单的 CNN 替换成更深的其他 CNN 确实能更好地捕捉字符形变的信息，有 1%-3% 的性能提升。

2) 将 ASRN 的 CNN 部分进行替换。ASRN-N 表示将 ASRN 部分的 CNN 替换为 N。

表 2. 针对 ASRN 的 CNN 部分进行改进的实验结果

模型	词准确率	编辑距离
MORAN(original)	46. 25%	0. 2940
ASRN-SECNN	47. 09%	0. 2802
ASRN-SE_ResNet18	47. 35%	0. 2845
ASRN-SE_ResNet34	47. 01%	0. 2862
ASRN-SE_ResNet50	48. 11%	0. 2772
ASRN-DenseNet121	45. 44%	0. 2978
ASRN-DenseNet169	45. 12%	0. 3033
ASRN-DenseNet201	47. 51%	0. 2831
ASRN-OctaveNet	46. 20%	-----

ASRN-OctaveNet w/spatial attention	47. 14%	----
--	---------	------

可以看到，将 ASRN 部分原来的简单的 CNN 替换成更深的其他 CNN 确实能更好地捕捉字符形变的信息，有 1%-2% 的性能提升。

3) 将提升效果最好的 MORN-DenseNet169 以及 MORN-SE_ResNet50 分别跟 ASRN-SE_ResNet50 进行合并。

表 3. 将改进后的 MORN 和 ASRN 进行组合的实验结果

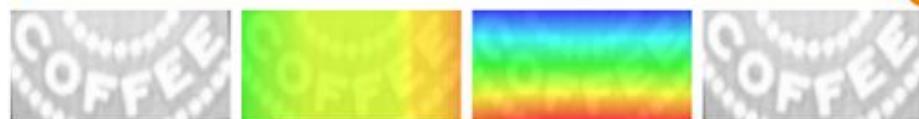
模型	词准确率	编辑距离
MORN+ASRN	46. 25%	0. 294
MORN-SE_ResNet50+ASRN-SE_ResNet50	49. 11%	0. 2759
MORN-DenseNet169+ASRN-SE_ResNet50	49. 22%	0. 272

(6.2) 可视化分析及 Ablation 实验

3) 对 MORN 纠正模块的输出以及纠正时依据的 CNN 学到的形变网格进行分析，其中部分样例如下图。

attention_map0/0

step 45 Wed Apr 10 2019 18:41:59 GMT+0800 (中国标准时间)



attention_map0/0

step 39 Wed Apr 10 2019 05:01:54 GMT+0800 (中国标准时间)



attention_map0/0

step 34 Wed Apr 10 2019 15:29:11 GMT+0800 (中国标准时间)



attention_map0/0

step 71 Wed Apr 10 2019 03:18:48 GMT+0800 (中国标准时间)



attention_map0/0

step 31 Wed Apr 10 2019 11:55:04 GMT+0800 (中国标准时间)



attention_map0/0

step 57 Tue Apr 09 2019 23:59:02 GMT+0800 (中国标准时间)



图 3. MORN 模块输出结果可视化的部分示例

可以看到，MORN 的纠正能力并不强，基本没有什么效果，甚至还会增大一些图片的形变，使之更难识别。为了进一步分析，进行了去除 MORN 模块的 Ablation 实验。

表 4. 去除 MORN 模块的 Ablation 实验

模型	词准确率	编辑距离
MORN+ASRN	46.25%	0.294
ASRN (CNN)	47. 59%	0. 2815
ASRN (GRCNN)	47. 75%	0. 2717
ASRN (SE_ResNet50)	49. 75%	0. 2675
MORN (DenseNet169) + ASRN (SE_ResNet50)	49. 22%	0. 272

GRCNN 为张培尧师姐负责的模型，此处不做展开。可以看到，相对了原来含 MORN 和 MORAN 模型，去掉 MORN 仅用 ASRN 部分会有 1% 左右的提升。进一步验证了 MORN 确定没能起到预期效果的猜想，跟可视化结果一致。

对此，我们通过分析模型代码，得出可能的原因其一是改进后的 CNN 依然没有足够的能力学习到整张图的每个部分的形变，其二是这个网络自身的局限，不适用于我们的比赛数据集。针对这两个猜想我们提出两个改进方案。其一是将原图尺寸增大一倍，给予网络更大的学习空间。其二是将 MORN 替换成 STN 网络进行对变形文本的矫正。本文主要记录第二个方案的实现。

(6.3) 将 MORN 替换成 STN

Spatial Transformer Networks

1) 论文地址：

<http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf>

2) 代码地址：https://github.com/WarBean/tps_stn_pytorch

3) 主要思路在于利用神经网络的强大学习能力学习弯曲文本的形变，进而根据形变在原图上进行采样，进而得到去除形变之后的更易于识别的图送到后面的识别网络中。弯曲文本纠正得越好，得到纠正后得图越容易识别，识别效果越好，正是因为有这样的联系，STN 通过识别网络的反馈进行弱监督。stn 思路跟 morn 一样，只是提取形变信息的方式略有不同。通过对比代码可以发现，stn 其实属于 morn 的一个简单的个例，其代表的形变类型局限于仿射变换等，没有 morn 直接学输入每个小像素块的形变所包含的多，但也正是因为 morn 需要学习的参数非常复杂，实验结果反而没有 stn 的好。stn 主要的框架如下图。

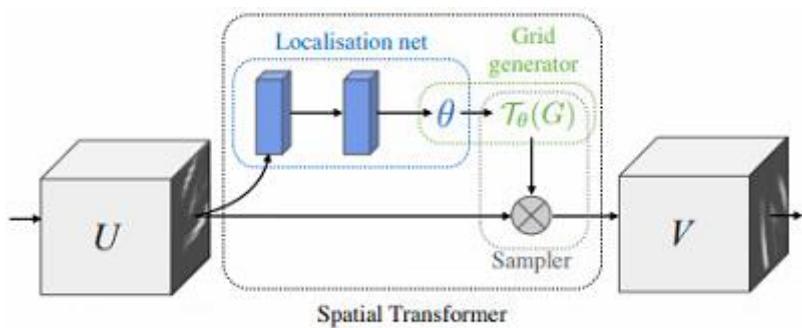


图 4. stn 的主要框架图

如图，Localisation net 为简单的回归网络，将输入的图片进行卷积操作，然后全连接回归出控制点的坐标。Grid generator 通过矩阵运算，计算出目标图 V 中的每个位置对应原图 U 中的坐标位置。此处用到了 tps 算法。最后 Sampler 根据 $T(G)$ 中的坐标信息，在原始图 U 中进行采样，将 U 中的像素复制到目标图 V 中。

4) 原代码用于 mnist 数据集上的 demo 效果如下图所示。

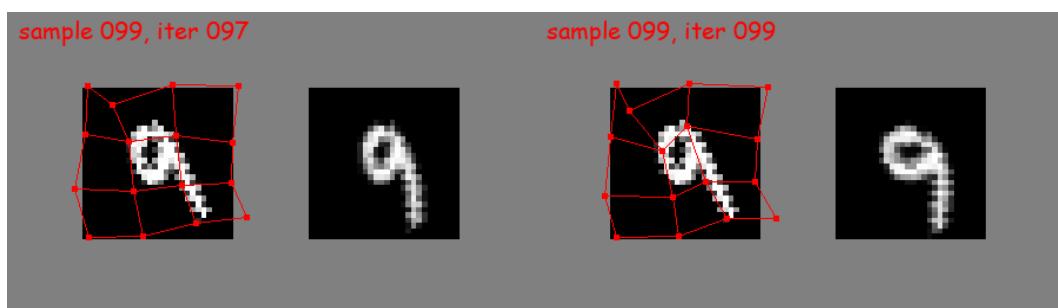


图 5. STN 在 mnist 上效果示意图

5.1) 直接将 demo 的 stn 输出送入 ASRN。此版本 stn 模型记为 stn0.

表 5. stn0 实验效果对比

模型	词准确率	编辑距离
MORAN_v6 (STN0+ASRN-se_resnet50)	49.77%	0.2666
MORAN_v6 (STN0+ASRN-se_resnet50)+分段=50epoches	49.68%	0.2843
MORAN_v6 (STN0+ASRN-se_resnet50)+5 倍数据增强+分段=50epoches	53.85%	0.2526
ASRN(SE_ResNet50)	49.75%	0.2675

可以看到，stn 提升效果并不十分明显。于是，将其输出以及学到的控制点进行可视化。

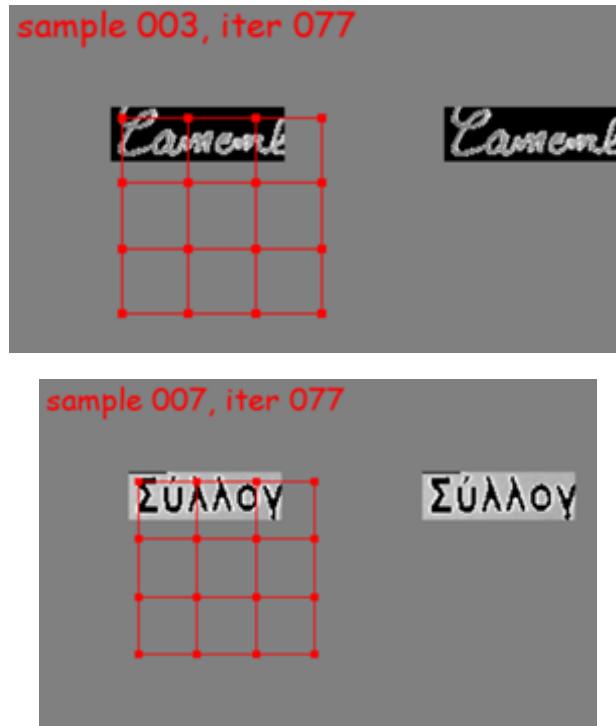


图 6. STN0 在 ArT 上效果示意图

5.2) 可视化发现，stn0 基本学不到东西，这也就是无法提升实验效果的原因。通过检查发现，stn0 在回归出控制点坐标的时候，原文为了保持其训练的稳定性，将最后一层 fc 的权重于 0 附近再渐渐学习不同的变换。原 demo 中 fc 层权重可在 0 附近波动，而代码移植后由于 python 版本不同等原因，

同样的写法，fc 层权重却一直为 0，没有更新。于是将置于 0 的条件去掉，再进行实验。



图 7. STN1 在 ArT 上效果示意图

可以通过可视化发现，去除 0 初始化以后，模型确实具有更大的学习形变能力，但是却很不稳定。于是进一步把 fc 权重除以 10^4 以减缓网络训练的不稳定性。此版本 stn 模型记为 stn1。

表 6. stn1 实验效果对比

模型	词准确率
MORAN_v6 (STN1+ASRN-se_resnet50)	50. 01%
ASRN (SE_ResNet50)	49. 75%

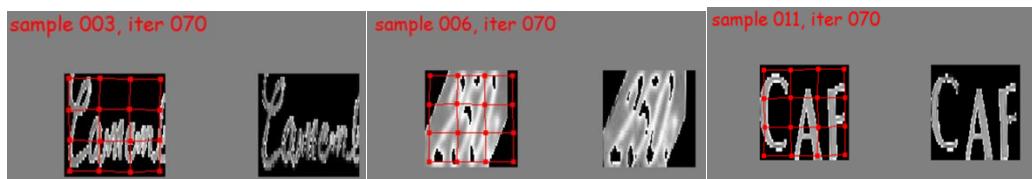


图 8. STN1 在 ArT 上效果示意图

可以看出，调整之后的 stn 具备轻微的调整能力，但是由于是弱监督，效果仍然十分不理想。

5.3) 为了改善 stn 的弱监督效果不够理想，进一步提取坐标点信息，直接给 stn 控制点坐标取代 Localisation net 的处理过程。



图 9. 给 STN2 输入的控制点在输入图上位置示意图

如上图，为剪切之后需要识别的图片。红色点为官方给出的坐标点，绿色点为解决红色的点个数不统一的情况，统一插值而得的 16 个点。点的大小仅代表给出的顺序。这部分实验没有全部完成，因出现网络不收敛情况而比赛时间接近尾声时间不足，没有继续深究。

(6.4) 针对 1stm 模型的探究

针对实验中出现的模型容易根据字符串的一部分字自己猜出整个字符串（但猜出来的实际是数据集中多现多次但跟目前的字符串不完全一样的错误序列）的情况，提出两种改进方案。

1) 减少 1stm 层数以缓解由于 1stm 模块过强语料学习能力导致的上述猜字情况。

表 7. 减少 1stm 层数实验效果

模型	词准确率
stn1_ase50_50_5_2	56. 59%
stn1_ase50_50_5_1	52. 70%

注：stn1_ase50_50_5_1 表示采用 stn1+ASRN 框架，ASRN 的 CNN 部分为 seresnet50，分段为 50epoches，前 50 训练 ASRN，后 50 训练 STN，之后合

一起训，进行 5 倍数据增强，LSTM 层数为 1。stn1_ase50_50_5_2 与 stn1_ase50_50_5_1 区别仅在于 LSTM 层数为 2。

2) 减少 lstm 隐含因子数量以降低其学习记忆的容量，希望可以通过减少其容量来迫使模型每次的预测都基于当前的输入合理进行，减少不必要的语义猜测。

表 8. 减少 lstm 隐含因子数量实验效果

模型	词准确率
stn1_ase5gori_50_12_art	56. 16%
stn1_ase5gori256_50_12_art	54. 90%
stn1_ase5g_50_12_art	58. 22%
stn1_ase5g256_50_12_art	56. 29%

注：stn1_ase5gori_50_12_art 表示采用 stn1+ASRN 框架，ASRN 的 CNN 部分为 seresnet50 和 GRCNN 融合，分段为 50epoches，前 50 训练 ASRN，后 50 训练 STN，之后合一起训，数据扩增对应编号 12 的数据构成。

stn1_ase5gori256_50_12_art 与 stn1_ase5gori_50_12_art 区别仅在于前者 LSTM 隐藏因子数量为 256，后者为默认的 512。stn1_ase5g_50_12_art 与 stn1_ase5gori_50_12_art 区别在于融合 seresnet50 和 GRCNN 时，前者为直接截取进行维度对齐，后者通过修改 seresnet50 的 padding 使之输出的尺寸跟 GRCNN 一致。

(6.5) 针对模块融合的实验

1) ASRN 的 CNN 部分，对不同的 CNN 提取的特征进行融合。

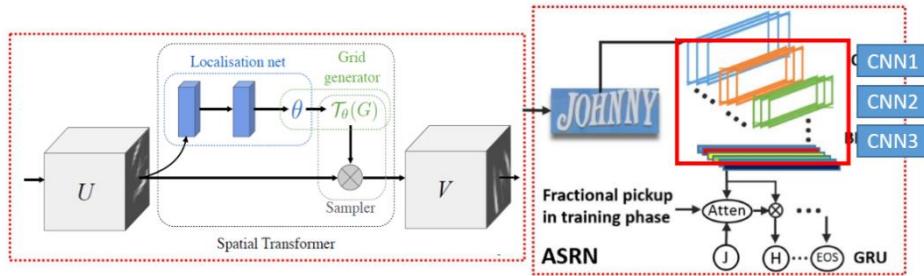


图 10. 融合 CNN 后得到的总的模型框架示意图

表 9. 融合 ASRN 部分多个 CNN 实验效果

模型	词准确率
stn1_ase50_50_12_art	59. 92%
stn1_am5_50_12_art	56. 09%
stn1_ase5g_50_12_art	58. 22%
stn1_am3_d3_50_12_art	56. 03%

注：stn1_ase5gori_50_12_art 表示采用 stn1+ASRN 框架，ASRN 的 CNN 部分为 seresnet50 和 GRCNN 融合，分段为 50epoches，前 50 训练 ASRN，后 50 训练 STN，之后合一起训，数据扩增对应编号 12 的数据构成。

stn1_am5_50_12_art 与之区别在于 ASRN 的 CNN 部分是效果最好的前五个 CNN 的融合，stn1_am3_d3_50_12_art 是效果最好的前三个 CNN 的融合，stn1_ase5g_50_12_art 是 seresnet50 和 GRCNN 两个 CNN 的融合。

2) 尝试将针对变形文本的纠正模块加入 GRCNN 中。

方案一：将 MORN 移植至 GRCNN 网络，从头开始训/用在 MORAN 中较好的模型的 MORN 部分参数进行 finetune

方案二：将 STN 移植至 GRCNN 网络

方案三：将 GRCNN 模块移植至 MORAN 网络，替换 ASRN 的 CNN 部分。

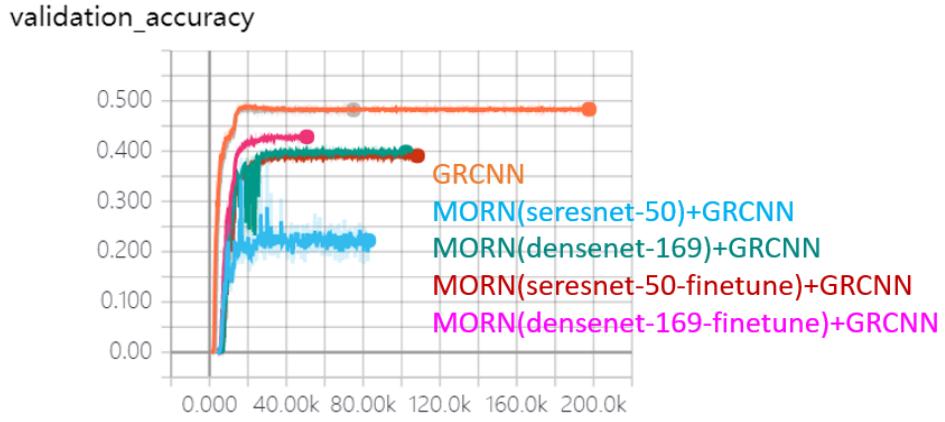


图 11. 三种方案部分实验效果对比图

(6.6) 针对训练方法的调优实验

1) 将原来的 3000 常用汉字 key 换成遍历 ArT 数据集之后含约 3850 个中文及特殊字符 key，让模型的学习空间更大。模型在预测时输出直接跟 key 相关，若 key 中没有需要的字符则无法正确预测。

表 10. 更新字典的实验效果

模型	使用的 key	词准确率	编辑距离
MORAN	3000 常用汉字加英文大小写和数字	45. 29%	0. 2941
MORAN	ArT 数据集的遍历	46. 25%	0. 294

2) 采用论文中提到的 curriculum learning strategy，先训练 ASRN 识别模块，再训练 MORN 纠正模块，最后一起训练。

表 11. 采用分段训练的实验效果

模型:	词准确率	编辑距 离
MORN(DenseNet169)+ASRN(SE_ResNet50)		
End-to-end	49. 22%	0. 272
3stages (boundary=5epoch)	49. 48%	0. 2708
3stages (boundary=20epoch)	49. 46%	0. 2717

3stages (boundary=50epoch)	53. 85%	0. 2526
----------------------------	---------	---------

3) 优化器及学习率

表 12. 采用不同优化器和学习率的实验效果

模型	词准确率
MORAN_v6 (STN1+ASRN-se_resnet50) +5 倍数据增强+分段=50+stn 优化器 =sgd	56. 27%
MORAN_v6 (STN1+ASRN-se_resnet50) +5 倍数据增强+分段=50+stn 优化器 =adadelta	56. 61%
MORAN_v6 (STN1+ASRN-GRCNN) +5 倍 数据增强+分段=50+stn 优化器=sgd	55. 42%
MORAN_v6 (STN1+ASRN-GRCNN) +5 倍 数据增强+分段=50+stn 优化器 =adadelta	55. 61%
MORAN_v6 (STN1+ASRN-GRCNN) +分段 =50+stn 优化器=adadelta	52. 29%
stn1_ase50_50_5_adam+学习率递减	50. 77%
stn1_ase50_50_5	56. 18%
stn1_ase50_50_5_adam	52. 68%
stn1_ase50_50_5_sgd	-

(6.7) 针对数据集的实验

加入收集到的数据集及生成的数据之后的实验结果如下。为了更好地分析不同的数据集对模型的效果，最后将最有效的数据集综合起来，做了一系列的实验。

表 13. 加入新的数据扩增的实验效果

数据集	数据集构成	分段 epoch	词准确率	数据集（数据集构成同左）	分段 epoch	词准确率
stn1_ase50_70_5_nword	5 倍数据增强	50	55. 61%	stn1_ase50_50_5_nword	70	52.44%
stn1_ase50_70_6_nword	同上+5 万生成地址	50	54. 16%	stn1_ase50_50_6_nword	70	55.96%
stn1_ase50_70_7_nword	同上+5 千小广告	50	56. 09%	stn1_ase50_50_7_nword	70	56.68%
stn1_ase50_70_8_nword	同 上 +icdar2013	50	56. 50%	stn1_ase50_50_8_nword	70	56.37%
stn1_ase50_70_9_nword	同 上 +icdar2015	50	55. 53%	stn1_ase50_50_9_nword	70	56. 53%
stn1_ase50_70_10_nword	同 上 +icdar2015-T RW	50	55. 22%	stn1_ase50_50_10_nword	70	57. 22%
stn1_ase50_70_11_nword	同上+ICPR	50	50. 16%	stn1_ase50_50_11_nword	70	54. 92%
stn1_ase50_70_12_nword	同上+RCTW	50	60. 76%	stn1_ase50_50_12_nword	70	61. 05%
stn1_ase50_70_13_nword	同上 + 横幅 第一批	50	53. 81%	stn1_ase50_50_13_nword	70	54. 61%
stn1_ase50_70_14_nword	同上 + 横幅 第二批	50	48. 33%	stn1_ase50_50_14_nword	70	44. 51%

				在 12 基础上加入 LSVT 数据	70	67.00%
--	--	--	--	--------------------	----	--------

注：nword 表示采用了综合 ArT 和 LSVT 数据遍历后的超大字典。

GRCNN 实验

第一阶段：用 LSVT 和 ArT 原数据集跑 baseline 实验，以下是不同算法模型的实验结果，实验评价指标包括全匹配准确率和归一化编辑距离。

Model	CNN	Training	Validation	Accuracy	Norm_ED
CRNN	7 层 CNN	ArT 45k	ArT 5k	40.70%	0.3784
CRNN	ResNet18	ArT 45k	ArT 5k	35.54%	0.433
SE-CRNN	SECNN	ArT 45k	ArT 5k	41.86%	0.3703
GRCNN	GRCL	ArT 45k	ArT 5k	44.92%	0.3436

表 1 Results of ArT dataset

Model	CNN	Training	Validation	Accuracy	Norm_ED
GRCNN	GRCL	LSVT 190k	LSVT 21.4k	52.78%	0.1865
SE-CRNN	SECNN	LSVT 190k	LSVT 21.4k	51.00%	0.1951

表 2 Results of LSVT dataset

从实验结果来看，ArT 数据集因为包含了大量任意朝向的形变文字且数据量小，数据集难度很大，baseline 的准确率都很低，LSVT 数据集相比 ArT 图片较规则，并且数据量较大，所以准确率整体水平比比 ArT 数据集的高。从 baseline 实验结果来看，两个数据集的难度都不小。

另外，从各个模型算法在实验中的表现来看，CNN 为 ResNet18 时性能最差，这也与之前参加 ICPR 竞赛时的结果一致，数据量较小时，ResNet 系列和 DenseNet 系列的网络因为层数较深、参数较多，没有训练充分而性能较差。SECNN 加入了特征重标定的策略，比 CRNN 的性能要好，GRCNN 在特征提取层加入了循环连接和门控机制，扩大了感受野，表现优异。

一开始是用 ArT 的数据集对模型算法做了对比实验，表现较好的模型是 GRCNN 和 SE-CRNN，所以 LSVT 数据集只用了这两个模型做实验。

第二阶段：对 LSVT 和 ArT 数据集进行数据增强，使用的是 Python 扩展工具包 ImgAug。经过多次试验，挑选出适合数据集的增强策略，以下是最后确定的增强策略。

LSVT：2 种数据增强的变换，包括模糊、转换等方法。

图 7 LSVT 的增强策略

ArT：12 种数据增强的变换，包括对比度变换、模糊、噪声、转换、颜色扰动等方法。



图 8 ArT 的增强策略

以下是不同算法模型的实验结果，实验评价指标包括全匹配准确率和归一化编辑距离。

Model	CNN	Training	Validation	Accuracy	Norm_ED
CRNN	7 层 CNN	ArT 45k	ArT 5k	47.17%	0.3354
CRNN	ResNet18	ArT 45k	ArT 5k	44.80%	0.3522
SE-CRNN	SECNN	ArT 45k	ArT 5k	48.40%	0.3267
GRCNN	GRCL	ArT 45k	ArT 5k	51.57%	0.2963

表 3 Results of ArT dataset with augmentation

Model	CNN	Training	Validation	Accuracy	Norm_ED
GRCNN	GRCL	LSVT 190k	LSVT 21.4k	53.22%	0.1808
SE-CRNN	SECNN	LSVT 190k	LSVT 21.4k	50.45%	0.1978
CRNN	7 层 CNN	LSVT 190k	LSVT 21.4k	49.02%	0.2062
CRNN	ResNet18	LSVT 190k	LSVT 21.4k	48.80%	0.2077

表 4 Results of LSVT dataset with augmentation

和第一阶段的实验相比，做了数据增强之后，同一模型同样配置的实验可以提升 6—7 个百分点，效果显著。一开始尝试了很多种增强方法，实验结果发现不是所有的增强方法都适合字符识别，最后挑选出来的都是对图片没有过多裁剪翻转操作，增强幅度较小的策略。

第三阶段，从两个方面入手来提升 GRCNN 的模型性能。首先是从算法模型入手，对 GRCNN 进行超参数的调整，主要包括优化器、学习率、时间步等，其次从数据入手，除了数据增强以外，在训练集额外加入新的数据集，包括 ICPR dataset(110k images)、ICDAR 2015-TRW(8k images)和 ICDAR 2017-RCTW(44k images)

下表是加入额外的训练数据之后的实验结果，实验评价指标包括全匹配准确率和归一化编辑距离。

Model	CNN	Training	Validation	Accuracy	Norm_ED
CRNN	7 层 CNN	ArT 200k	ArT 5k	50.45%	0.3069
SE-CRNN	SECNN	ArT 200k	ArT 5k	51.00%	0.3040

GRCNN	GRCL	ArT 200k	ArT 5k	51.81%	0.2927
-------	------	----------	--------	--------	--------

表 5 Results of ArT dataset with extra dataset

从实验结果可以看出，因为加入的额外训练数据集的分布与竞赛数据集的相似，都是中英文字符数据集，所以神经网络得到更充分的训练，实验性能均有所提升。之后对模型进行细致的参数调整，最后选定了效果最好的一组参数组合，实验结果如下表所示：

Model	CNN	Training	Validation	Accuracy	Norm_ED
GRCNN	GRCL	ArT 200k	ArT 5k	55.10%	0.2614
SE-CRNN	SECNN	ArT 200k	ArT 5k	52.26%	0.2921

表 6 Results of ArT dataset

这一阶段都是先在 ArT 数据集上进行实验，得到对实验效果有提升帮助的额外数据和参数设置，之后用 LSVT 数据集做进一步验证，LSVT 数据集的结果如下所示，加入了额外训练数据，并且使用了最好的参数组合。

Model	CNN	Training	Validation	Accuracy	Norm_ED
GRCNN	GRCL	LSVT 318k	LSVT 21.4k	55.22%	0.1751
SE-CRNN	SECNN	LSVT 318k	LSVT 21.4k	52.27%	0.1875

表 7 Results of LSVT dataset

以上就是竞赛期间算法的训练过程，基本上分为算法复现，数据扩充和模型调参这几部分工作。GRCNN 是性能较好的模型，也是最后竞赛提交结果采用的模型。

所发现的问题

在实验过程中，对模型的识别结果进行分析归类，可以帮助我们更好的对模型作出评价，发现问题并找寻新的思路来解决问题。下面是对 GRCNN 模型在 ICDAR2019 ArT 测试集上的表现，分为了四类，包括规则图片、弧形文字、多行文字、低分辨率/艺术字。每张图片下面有对应的 Ground Truth 和 Prediction，没有预测正确的字符标注成了红色。



图 9 规则图片



图 10 弧形文字

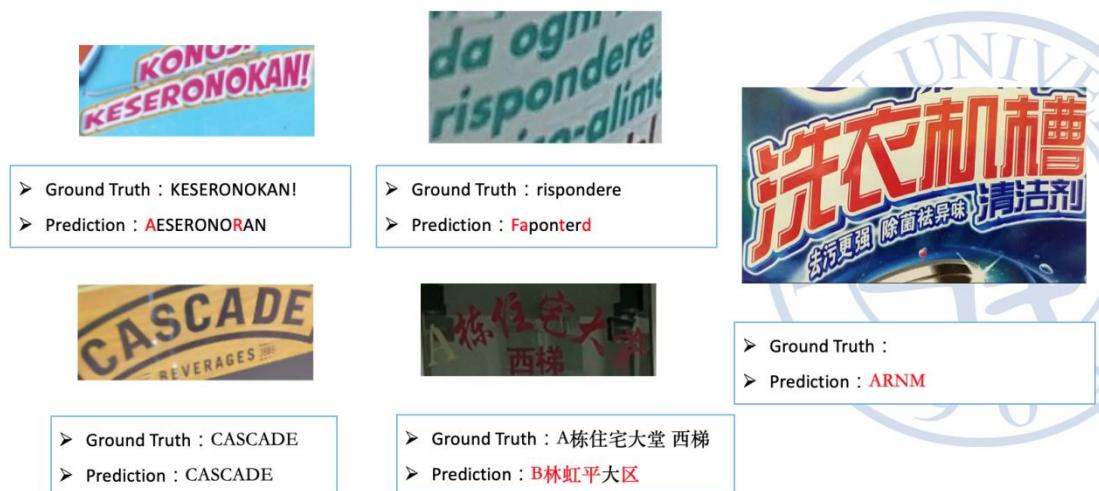


图 11 多行文字

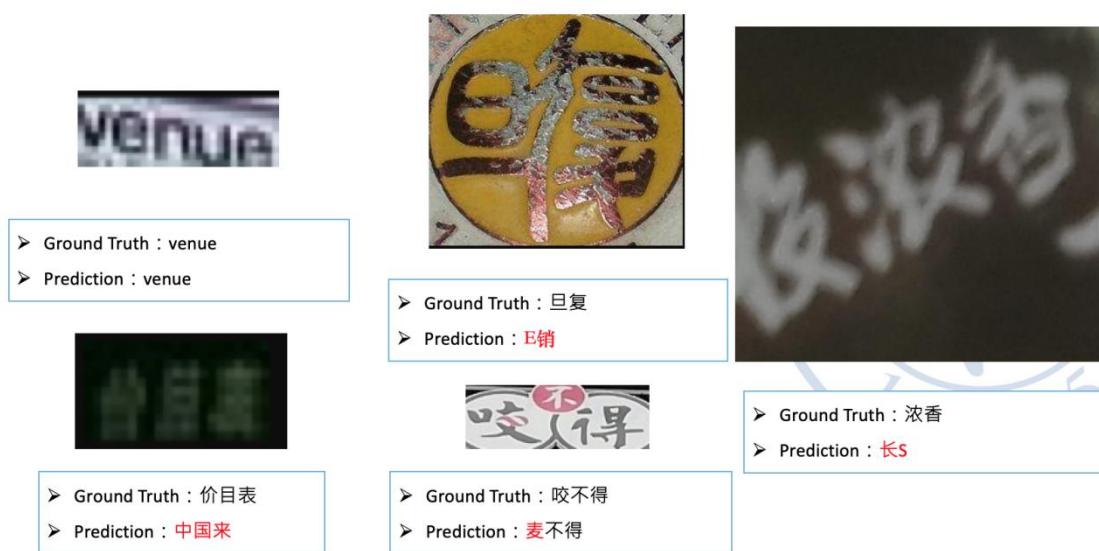


图 12 低分辨率/艺术字

从识别结果来看，模型对规则图片的识别结果较好，弧形文字的识别效果取决于形变幅度，幅度较小识别结果就不错，幅度较大的话，识别有很大的挑战。多行文字一般能识别出最中心的一行，低分辨率图片的识别结果还可以，艺术字一般很难识别正确。

总结来看，目前算法的不足之处有两个：

一是形变文字的识别；

二是语义信息太强，主要原因在于循环神经网络的过拟合。

之后可以针对这两个问题进行改进。

- 1) 各种调优和改进之后得到的 moran 模型由于没有用到 ctc-loss，对中文识别并不很理想。
- 2) 去除弯曲文本干扰部分还有很大的提升空间。

训练集异常竖排样本修复

在数据集处理中筛选出来的竖排文本中，存在不同的朝向，如图 13 所示，按照种类分为朝左竖排与朝右竖排。考虑到街景视觉效果，正常的街景文本是不会出现这类难被肉眼识别的文本图片，因此这类样本属于异常样本。这类样本在所有样本中占据少数部分，如果直接将此类文本送入模型训练，则训练效果不佳。因此我们试图寻找一种方法，识别出异常竖排图片的旋转角度，从训练集中筛选出此类样本，并进行复原处理。经过调研之后，我们发现目前还未有人做过相关课题。较为相近的模型为 RRPN，而 RRPN 虽然能识别出文本的旋转角，但是仅仅局限于识别小范围的角度，例如图 14 所示。



图 13 三种类型的竖排图片。左图为正常竖排文本，可以由 MORAN 识别。而中图为朝向为右的竖排文本，右图为朝向为左的竖排文本，属于数据集中的异常样本

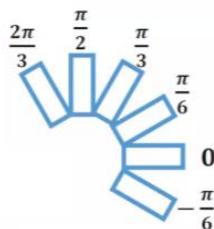


图 14 RRPN 模型能识别出的图片旋转角度

对于我们实验中要识别的+90 度与-90 度角，该模型未能很好地判别。因此我们从其他角度出发，构思了如下三种方法。在这之前，我们先对竖排文本中的每一张图片进行 90 度，

-90 度旋转。例如向右朝向的竖排图片经过两次旋转后分别得到的结果如图 15 所示。



图 15 向右朝向的竖排图片经过两次旋转后分别得到的结果

异常竖排文本在进行顺时针 90 度, 逆时针 90 度分别两次旋转之后, 会生成横排与颠倒横排文本。利用已经训练好的 MORAN 模型, 可以很好地识别横排文本, 如果将颠倒字符(如图 16 所示)也加入字符表, 进行训练, 理论上可以得到很好的实验结果。但是随着中文的加入, 原来字符表的规模可以达到 5000 字左右, 再为每一个字符生成颠倒字符代价过大, 因此不采取该方法。



图 16 中文字符“周”与其颠倒字符

对于较为清晰的街景文本图片, Python 的 CV2 库可以较好地分割出字符。之后我们尝试利用库中的 `match_template` 函数比较分割出的字符图与印刷体字符图的相似度(如图 17), 然而实验结果并不理想。



图 17 将分割的街景文本图片(左)与生成的正常、颠倒图片进行相似度比较

该方法的局限性在于, 街景图片的背景很容易被当成关键信息, 勾出的轮廓成为干扰项, 且不易排除。我们使用 CV2 的 `match_template` 函数进行实验, 颠倒的“周”与原图的相似度会比正常的“周”来得高。原因可能在于街景字体多样性, 并且有背景因素干扰。因此这种方法不可取。

已经训练好的 MORAN 模型已经对字符图片有较强的识别能力。我们使用 MORAN 对图 15 所示的两种横排图片进行识别, 并且对识别的结果进行编辑距离(ED)检测。理想情况下, 对于正常竖排文本(图 13 左), 实验结果中两个编辑距离的数值都较高, 但差距不明显。对于异常竖排文本(图 13 中, 右), 两次实验结果差距较大。只要设定一个阈值 `threshold`, 再将向左旋转判别结果 $ED_{turnleft}$ 与向右旋转判别结果 $ED_{turnright}$ 做差, 再与 `threshold` 作比较, 即可判断出异常样本类型。

对于左朝向竖排样本, 通常有

$$ED_{turnright} - ED_{turnleft} > threshold$$

而对于右朝向竖排样本，通常有

$$ED_{turnleft} - ED_{turnright} > threshold$$

由于该方法实现简便并且可靠，因此我们使用该方法用于挑选异常竖排文本。通过从训练集中筛选出异常竖排样本并进行复原，我们希望能够提升模型的学习效率，增强模型判别能力。

涉及的创新点

1. 替换 backbone：用 SENet 替换 CRNN 模型的标准 CNN；
2. 重新设计 GRCNN 的特征提取层：不同于论文，GRCL 的时间步设为 3，加入 batch normalization 层对数据归一化处理，优化器选择 Adam；
3. 数据增强和数据扩充，来增加数据样本的多样性。
4. 更换字典提 1 个点；
5. 将 asrn 部分的 cnn 换成 seresnet 或者 densenet 或者 grcnn 提 3-4 个点；
6. 将 morn 换成 stn 提 1 个点；
7. 分段训练提 4 个点；
8. 数据增强提 6-7 个点。

其他尝试过的模型：

(1) AON: Towards Arbitrarily-Oriented Text Recognition

1) 论文链接：

http://openaccess.thecvf.com/content_cvpr_2018/papers/Cheng_AON_Towards_Arbitrarily-Oriented_CVPR_2018_paper.pdf

2) 三方实现代码链接：<https://github.com/huizhang0110/AON>

3) 主要思路为对输入图片进行从上到下，从下到上，从左到右，从右到左四个方向提取特征进行融合，然后经过 blstm 和加入 attention 机制的解码器进行识别。

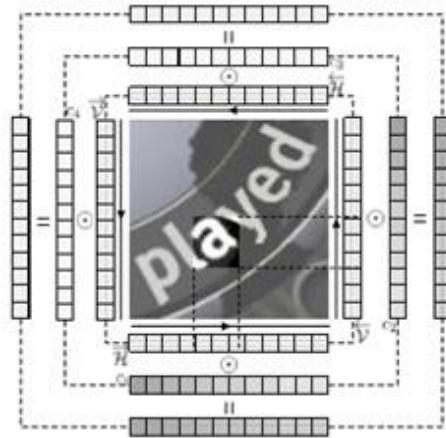


图 17. AON 识别主要思路示意图

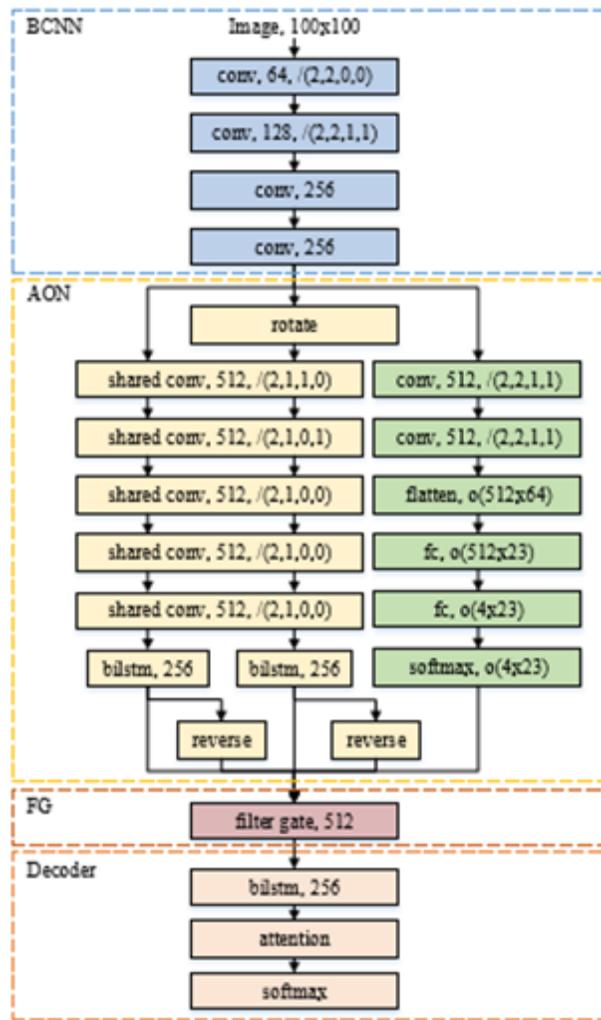


图 18. AON 识别主要网络框架示意图

4) 最大的亮点在于可以从多个方向对图片进行分析，尤其对于斜弯曲文本十分有效。

5) 局限性在于一是对相对英文文本更为复杂的中文识别能力不强，二是输入图片必须转换成 100*100 大小，对于长文本尤为不利，容易在 resize 的时候丢失大量特征。

6) 将代码调通并扩大类数和字典使之可用于比赛数据集后，在 ArT 上识别效果仅为 30% 左右，并且改进空间不大，因为没有继续改进而集中精力于 MORAN。

7) 将输入图片变形为 100*100 大小造成信息损失的部分样例示意图如下。



图 19. AON 将输入图片变形为 100*100 大小造成信息损失的部分样例示意图

(2) CRNN。An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition

此为文字识别最经典模型。主要思路为 CNN 提取简单特征，LSTM 提取语义信息，将字符串识别问题转换成序列识别问题，最后加入 CTC 得到最后的输出。

表 15. CRNN 实验效果

数据集	模型	词准确率	编辑距离
LSVT	CRNN(SE_Resnet18)	47.83%	0.2125
LSVT	CRNN(SE_Resnet34)	47.93%	0.2128
LSVT	CRNN(SE_Resnet50)	45.75%	0.2251

参赛人员名单及分工

指导老师

薛向阳 教授
李斌 研究员

检测组

邱泰儒（组长），苏上超，魏耀武，王浔彦，林少康，易子理，邓磊，徐沫霖

识别组

褚倩云（组长），张培尧，陈冠先，徐僖禧，陈竞晔

总负责

马建奇

特别鸣谢

杨华峰（参与讨论和模型提供）