算式识别开题报告

项目背景

算式识别任务是从一张图片中识别算式文字信息，属于文字 OCR 识别。

下表是 18 年以来主流会议中 OCR 识别取得的进展，IC 是 ICDAR（国际文档分析和识别大会）的缩写[1]。主流算法在自然场景中的识别率已突破 90%，OCR 识别技术已经发展的十分成熟，我们的项目相对而言场景比较简单，所以应该是可以解决的。

Conf.	Date	Title	IC03	IC13
'18-AAAI	18/01/04	Char-Net: A Character-Aware Neural Network for Distorted Scene Text Recognition	0.915	0.908
'18-AAAI	18/01/04	SqueezedText: A Real-time Scene Text Recognition by Binary Convolutional Encoder-decoder Network	0.931	0.929
'18-CVPR	18/05/09	Edit Probability for Scene Text Recognition	0.946	0.944
'18-TPAMI	18/06/25	ASTER: An Attentional Scene Text Recognizer with Flexible Rectification	0.945	0.918
'18-ECCV	18/09/08	Synthetically Supervised Feature Learning for Scene Text Recognition	0.947	0.94
'19-AAAI	18/09/18	Scene Text Recognition from Two-Dimensional Perspective		0.914
'19-CVPR	18/12/14	ESIR: End-to-end Scene Text Recognition via Iterative Image Rectification		0.913
'19-PR	19/01/10	MORAN: A Multi-Object Rectified Attention Network for Scene Text Recognition	0.950	0.924

个人对计算机视觉有兴趣，并且身边工作中也有这样的需求，因此想借此机会进一步了解 OCR 的相关技术。

问题描述

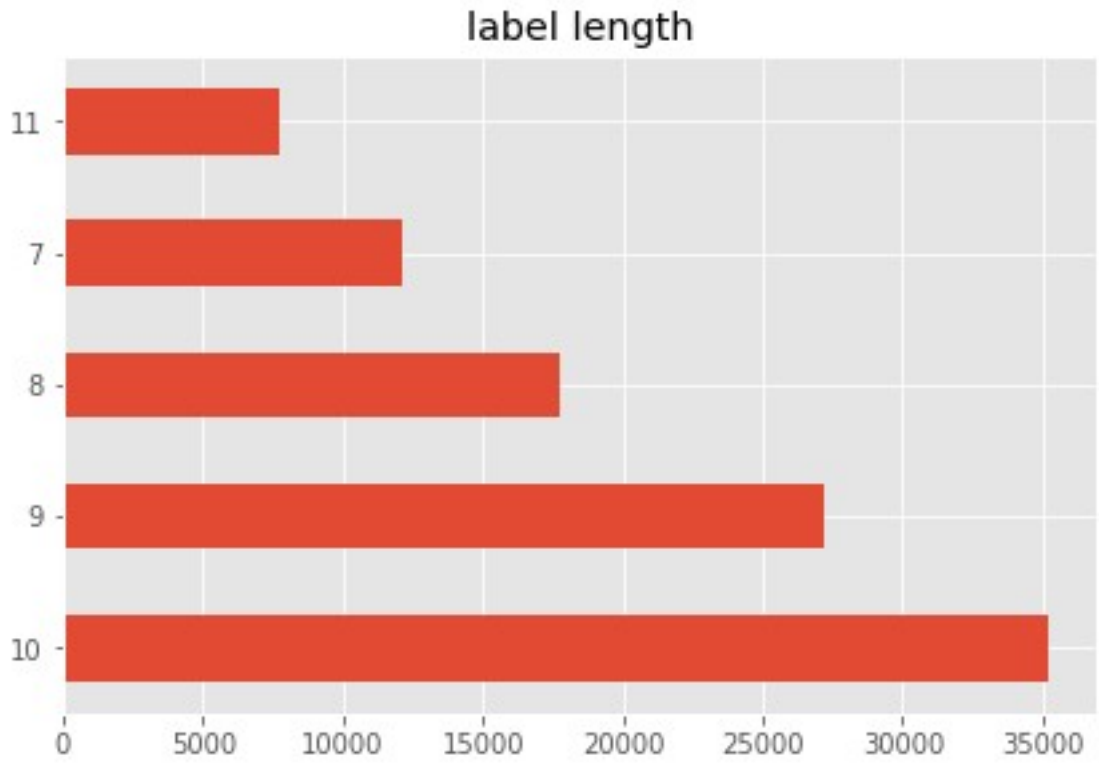
任务目标是：输入一张彩色图片，识别出图片中的算式文本。
这是一种经典的验证码识别场景，对于人来说很简单，程序而言需要准确顺序识别每个字符才算识别成功。

比如下图，考虑上下文后应识别为(0+0)+9=9。

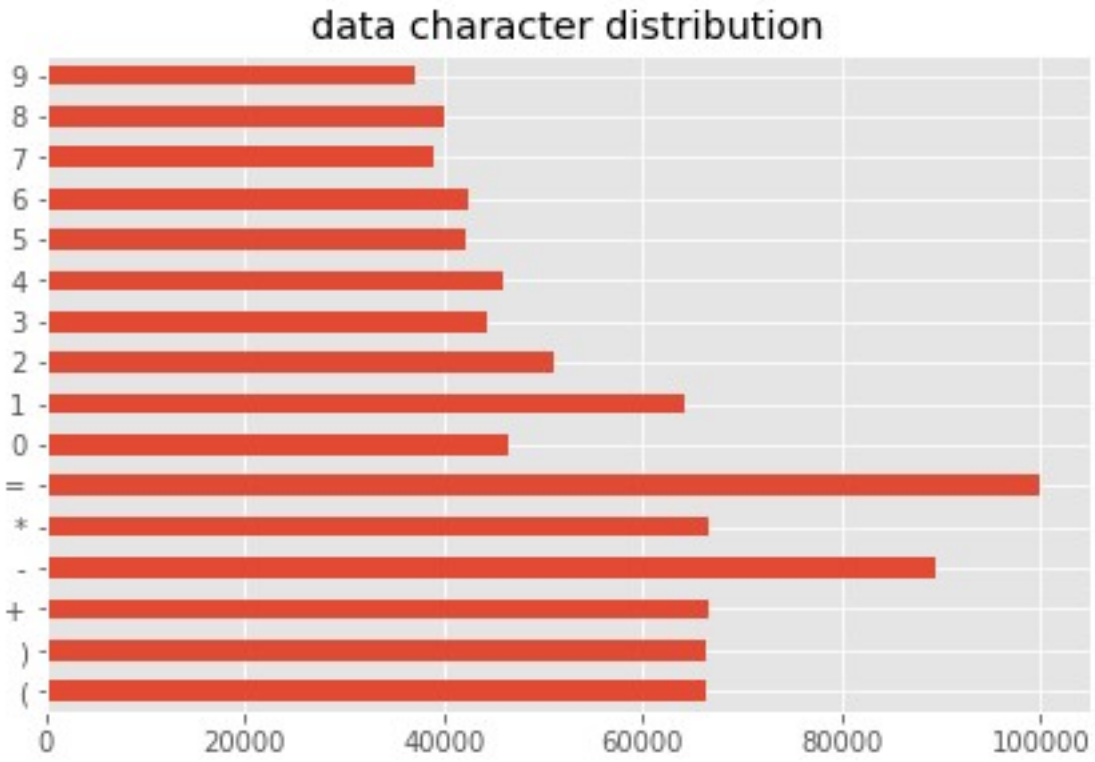
数据或输入

任务提供的数据集共 100,000 张图片，每张图片分辨率均为 300*64。算式由 3 个数字、2 个运算符、等号及运算结果组成，运算结果位数补丁，运算中还有括号，因此文本的长度不是固定的。需要注意的是，乘号只会以“*”的形式出现，所以十字符号只能是“+”。这个数据集将是本次任务中主要使用的数据集，首先会预留一部分用以作为测试集，查看程序的准确率；剩下的数据作为训练数据，并拟以 k 折的方式使用到训练中。

下面两图是对数据集的可视化统计。
Label length 直方图说明算式长度不定，分布从 7 到 11 都有，以 10 长度的数量最多，对模型的变长序列识别能力有较高要求。



data character distribution 直方图体现了整个数据集的字符分布情况，共 16 中字符，基于字典的算法是可以考虑的。算式中结果中“十几”运算结果会多一些，因此“1”的出现次数会多一些，到 60.000 多次，其他数字的分布差不多；运算符号中由于负数的存在，所以“-”的分布多一些，其他符号基本出现次数相同；“=”共 100,000，和数据集总数量一致，这是符合算式的特点的；“(”和“)”数量相等，这也是符合算式的特点的。从分布来看数据比较干净，呈均匀分布，但是也不排除其他的异常情况，这需要在训练调试过程中识别了。



对具体的图片进行观察后，发现图片有三个特点很可能影响模型的识别：1.字符不规律的倾斜，大小不一致；2.有噪点；3.由于文本长度不固定，右侧有不同程度的留白。在模型选择及预处理的时候应重点考虑这些特点。

评估标准

这里采用准确率来度量模型的好坏程度，即识别文字与图片中文字全部匹配时，算作识别正确，那么准确率就可以定义为：

$$准确率 = \frac{识别正确的样本个数}{总样本个数}$$

在算式识别这个任务中，我们关注的就是模型正确识别图片中文本的能力，并且数据集的观察结果没有明显的不均匀现象，因此采用准确率是可以有效度量模型的好坏的。

基准模型

本身模型会要求 99%的准确率。

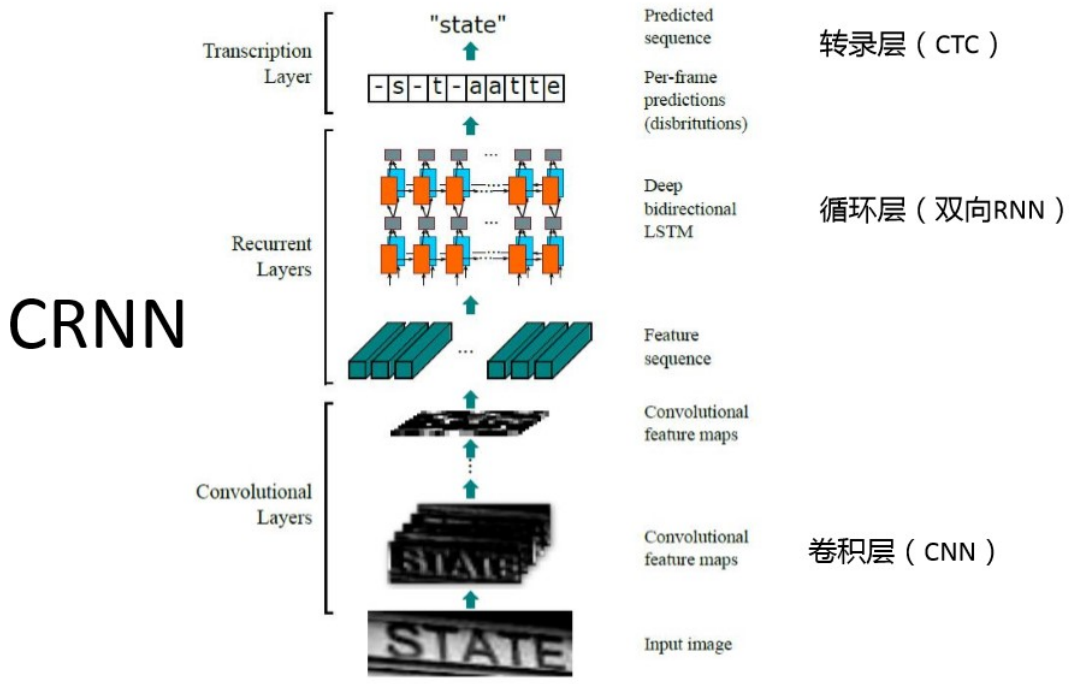
另外会对比 tesseract 在相同测试集上的识别准确率。Tesseract 是一个 OCR 库,目前由 Google 赞助(Google 也是一家以 OCR 和机器学习技术闻名于世的公司)。Tesseract 是目前公认最优秀、最精确的开源 OCR 系统，除了极高的精确度，Tesseract 也具有很高的灵活性。它可以通过训练识别出任何字体，也可以识别出任何 Unicode 字符。

项目设计

由于变长、不规则倾斜、大小不一的情况，对齐不易，先切割然后识别的策略不会优先考虑。总体上会主要参考 CRNN 的结构[2]，即卷积层、循环层、转录层的三层结构：1.卷积层提取图像特征；2.循环层将特征处理为连续帧的序列结果；3.转录层将序列结果处理合并为最终识别结果。

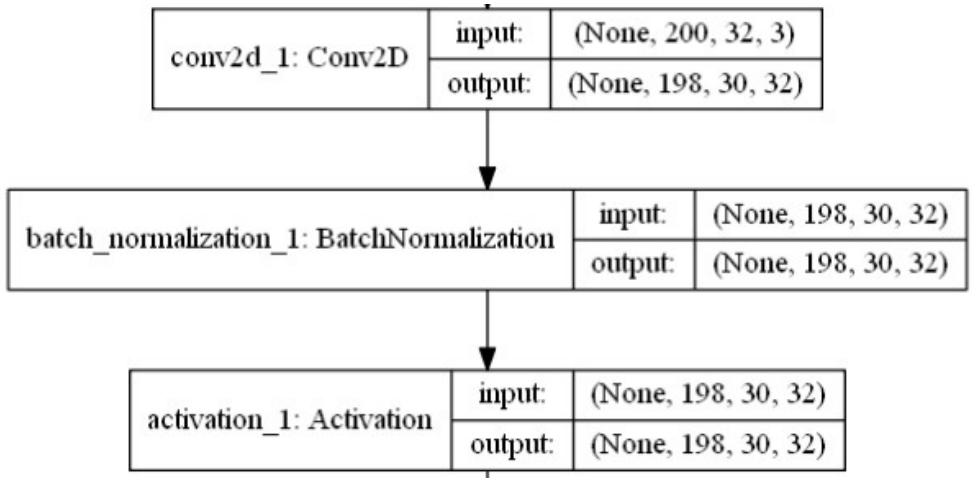
采取这种结构主要有 2 点好处：1.现有的数据集可以直接训练学习，不需要详细的标注（例如字符），模型训练对文本长度也无约束；2.在场景识别中表现具竞争力[2]。

具体网络设计中，会根据识别结果好坏，考虑调整为 densenet[3]+blstm[4]或其他结构[5]。



基于这个思路，整个项目将具体分为以下几步：

1. 图片上有一些噪点，可以使用开闭运算、高斯模糊等方法，消除或者减少这些噪点对识别的影响。在使用训练数据集时，计划将数据分为 5 份：4 份用作训练，1 分用来测试，并根据识别结果，考虑使用 K 折交叉验证。
2. 网络中卷积层部分考虑使用 3 层深度的 CNN，每层加入 BatchNormalization[6]加快收敛速度；循环层使用 GRU[7]。



对于 CRNN，比较关键的参数有：批量处理中每批次处理的样本数量，这个和显存有关，在正常训练的前提下，尽可能大；学习率使用 Adam 算法[8]自适应调整，Adam 中有一个步长参数控制调整学习率的幅度，训练分成两阶段设置这个参数，先取一个稍大的值，判断当前模型是否能收敛，并在能收敛的情况下快速收敛到一定程度，然后调小步长参数，获得最终的模型；迭代次数，预计 40 个迭代，视收敛情况进一步调整；试模型过拟合的情况，加入 dropout 层进行控制[9]。

3. 跳出模型的层面的话，还会使用融合来对识别性能进行强化，预计先使用简单的投票方式；数据集考虑使用额外的数据训练，增强模型的泛化能力。
4. 单纯从准确率结果和损失函数计算结果的收敛情况并不能直观看出模型的好坏，还会加入一些可视化的结果，从各种角度来展现模型的能力：学习曲线，主要看模型的收敛情况，以及拟合情况；对错分样本的展示，尝试总结错分原因，由此作为调整模型的依据；预处理的结果展示与对比。

引用

[1] hwalsuklee. https://github.com/hwalsuklee/awesome-deep-text-detection-recognition[Z], 2019

[2] Shi B , Bai X , Yao C . An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2015, 39(11):2298-2304.

[3] Iandola F, Moskewicz M, Karayev S, et al. Densenet: Implementing efficient convnet descriptor pyramids[J]. arXiv preprint arXiv:1404.1869, 2014.

[4] Ma X, Hovy E. End-to-end sequence labeling via bi-directional lstm-cnns-crf[J]. arXiv preprint arXiv:1603.01354, 2016.

[5] senlinuc . https://github.com/senlinuc/caffe_ocr[Z], 2017

[6] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[J]. arXiv preprint arXiv:1502.03167, 2015.

[7] Chung J, Gulcehre C, Cho K H, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. arXiv preprint arXiv:1412.3555, 2014.

[8] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[9] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The Journal of Machine Learning Research, 2014, 15(1): 1929-1958.