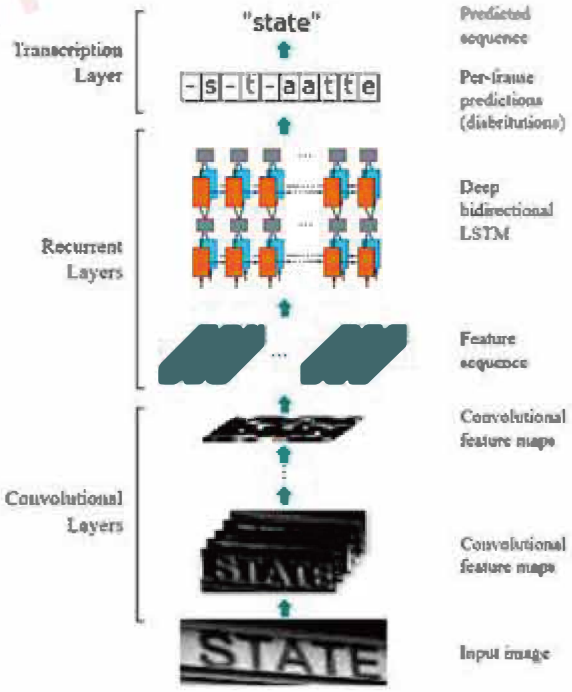
OCR之CRNN论文解读

前 言

01

网络结构

CRNN是一种卷积循环神经网络结构，用于解决基于图像的序列识别问题，特别是场景文字识别问题。CRNN网络结构如下图：



网络结构包含三部分，从下到上依次为：

- 卷积层，作用是从输入图像中提取特征序列;

- 循环层，作用是预测从卷积层获取的特征序列的标签（真实值）分布;
- 转录层，作用是把从循环层获取的标签分布通过去重整合等操作转换成最终的识别结果;

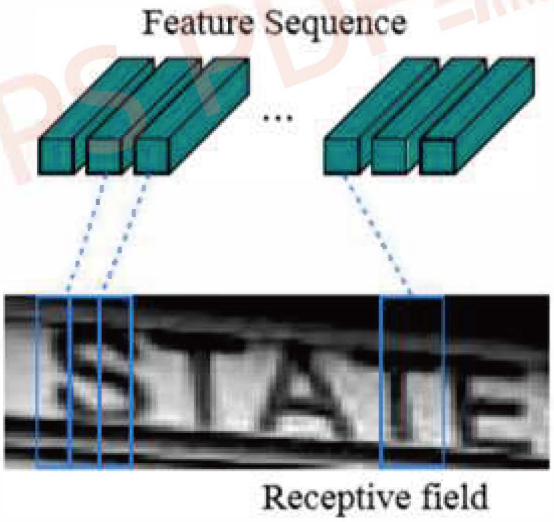
02

卷积层

CRNN卷积层由标准的CNN模型中的卷积层和最大池化层组成，自动提取出输入图像的特征序列。

与普通CNN网络不同的是，CRNN在训练之前，先把输入图像缩放到相同高度（图像宽度维持原样），论文中使用的高度值是32。

提取的特征序列中的向量是从特征图上从左到右按照顺序生成的，每个特征向量表示了图像上一定宽度上的特征，论文中使用的这个宽度是1,就是单个像素。



特别强调序列的顺序是因为在之后的循环层中，先后顺序是LSTM训练中的一个重要参考量。

03

循环层

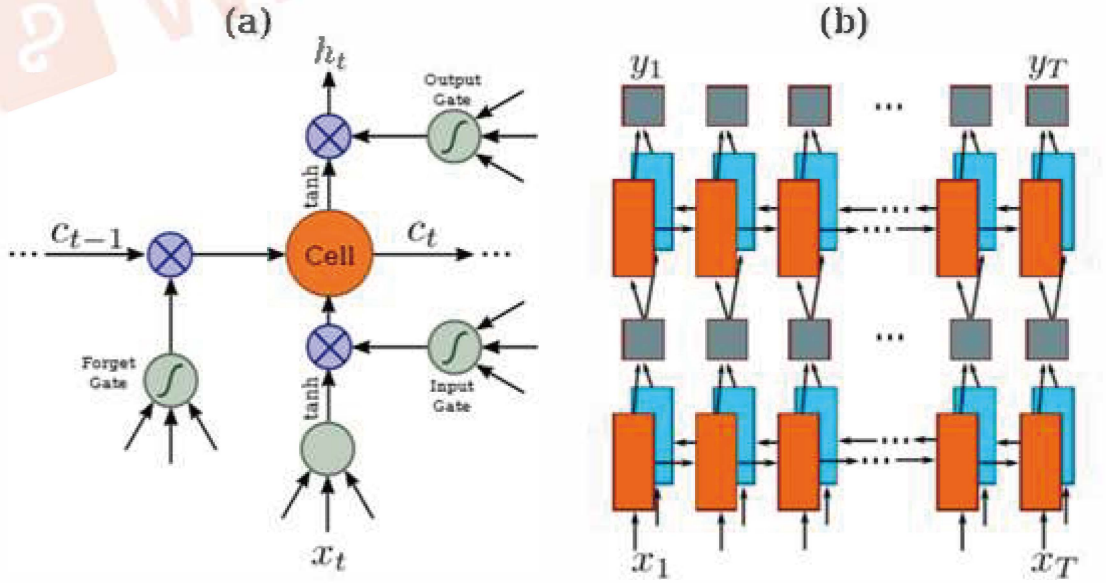
循环层由一个双向LSTM循环神经网络构成，预测特征序列中的每一个特征向量的标签分布（真实结果的概率列表），循环层的误差被反向传播，最后会转换成特征序列，再把特征序

列反馈到卷积层，这个转换操作由论文中定义的“Map-to-Sequence”自定义网络层完成，作为卷积层和循环层之间连接的桥梁。

在卷积层的上部建立一个深度双向递归神经网络，称为递归层。递归层对特征序列x = x1.....xT中每帧的标签分布进行预测。递归层的优点有三方面。首先,RNN具有很强的序列上下文信息捕获能力。使用上下文线索进行基于图像的序列识别比单独处理每个符号更加稳定和有用。以场景文本识别为例，宽字符可能需要几个连续帧来充分描述。此外，有些歧义字在观察其上下文时更容易区分，例如“il”通过字高对比比单独识别更容易识别。

其次，RNN可以将误差微分反向传播到它的输入，即卷积层，让我们在一个统一的网络中共同训练递归层和卷积层。第三，RNN能够对任意长度的序列进行操作，从开始到结束进行遍历。

传统的RNN单元在输入层和输出层之间有一个自连接的隐含层。每当它在序列中接收到一个帧 xt 时，它就用一个非线性函数更新它的内部状态（或称隐藏状态）ht，这个函数接受当前输入xt和上一个内部状态ht-1: ht=g(xt,ht-1)。然后，基于ht进行预测yt。这样就捕获了过去的上下文{xt'}t'<t并用于预测。传统的RNN单元存在消失梯度问题，限制了其可以存储的上下文范围，增加了训练过程的负担。LSTM是一种RNN单元，专门为解决这个问题而设计。一个LSTM(如下图所示)由一个存储单元和三个乘法门，即输入门、输出门和遗忘门组成。从概念上讲，记忆单元存储过去的上下文，并且输入和输出门允许单元长时间存储上下文。同时，通过遗忘门可以清除单元中的内存。LSTM的特殊设计允许捕获长期依赖关系，这通常发生在基于图像的序列中。



LSTM是方向性的，它只使用过去的上下文。然而，在基于图像的序列中，来自两个方向的上下文是有用的，并且相互补充。因此，将两个向前和向后的LSTM合并为一个双向LSTM。此外，可以对多个双向LSTM进行叠加，得到如上图b所示的深双向LSTM。与浅层结构相比，深层结构允许更高层次的抽象，并在语音识别任务中取得了显著的性能改进。

在递归层中，误差差沿上图b所示箭头的相反方向传播，例如，通过时间反向传播(BPTT)。在递归层的底部，传播的微分序列被连接到映射中，反转了将特征映射转换为特征序列的操作，然后反馈到卷积层。在实践中，我们创建了一个称为“映射-序列”的自定义网络层，作为卷积层和循环层之间的桥梁。

04

转录层

网络结构简图：

Type	Configurations
Transcription	-
Bidirectional-LSTM	#hidden units:256
Bidirectional-LSTM	#hidden units:256
Map-to-Sequence	-
Convolution	#maps:512, k:2 × 2, s:1, p:0
MaxPooling	Window:1 × 2, s:2
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
MaxPooling	Window:1 × 2, s:2
Convolution	#maps:256, k:3 × 3, s:1, p:1
Convolution	#maps:256, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:128, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:64, k:3 × 3, s:1, p:1
Input	W × 32 gray-scale image

Transcription层是将lstm层的输出与label对应，采用的技术是CTC，可以执行端到端的训练，用来解决输入序列和输出序列难以一一对应的问题，不要求训练数据对齐和一一标注，直接输出不定长的序列结果。对于一段长度为T的序列来说，每个样本点t（t远大于T）在RNN网络的最后一层都会输出一个softmax向量，表示该样本点的预测概率，所有样本点的这些概率传输给CTC模型后，输出最可能的标签，再经过去除空格（blank）和去重操作，就可以得到最终的序列标签，CTC对齐输入输出是多对一的，例如he-l-lo-与hee-l-lo对应的都是“hello”。

基于词典的转录

基于字典的模式，其实是就是上面CTC的基础上，在获得结果时，又从字典查了一遍，来更加提高准确率，而没有字典的就只能取高概率的结果，少了从字典查这一步。

采用了由Graves等人提出的连接时序分类(Connectionist TemporalClassification CTC) 层中定义的条件概率。该概率定义为：基于每帧的预测y=y1,y2.....yT的标签序列 l,它忽略了 l 中每个标签的位置。因此，当我们以该概率的负对数作为训练网络的目标时，我们只需要图像及其对应的标签序列，避免了为个别字符标注位置的劳动。

条件概率的公式简述如下:输入是一个序列y=y1,y2.....yT,其中T为序列长度。其中，每一个

$$y_t \in \Re^{|\mathcal{L}'|}$$

是集合 L' =LU 上的概率分布，而 L 包含任务中的所有标签(例如所有英文字符)，以及表示为的“空白”标签。一个序列到序列的映射函数 β 定义在如下序列上：

$$\pi \in \mathcal{L}'^T$$

其中，T是长度， $\prod$是预测概率。 β 映射 $\prod$ 到标签序列 I 上，通过先去除重复的标签，再去除空白的标签。例如： β 映射 “--hh-e-l-ll-oo--”到“hello”（ “-”代表空格) 。之后，定义条件概率为所有预测概率 到标签序列 l 上的映射 β 的概率和:

$$p(\mathbf{l}|\mathbf{y}) = \sum_{\pi:\mathcal{B}(\pi)=\mathbf{l}} p(\pi|\mathbf{y}),$$

其中， $\prod$ 的概率定义为p($\prod$ |y)=

$$\prod_{t=1}^{T} y_{\pi_t}^t$$

其中， $y_{\pi_t}^t$ 是标签 $\prod_t$ 在时间戳t的概率。

在基于词典的模式中，每个测试示例都与一个词典 D 相关联。主要的，序列标签通过选择词典中定义的拥有最高的条件概率来被选择，例如，l*=

$$\arg\max_{\mathbf{l}\in\mathcal{D}} p(\mathbf{l}|\mathbf{y})$$

然而，对于较大的词典，对词典进行穷举搜索，即对词典中的所有序列计算条件概率，并选择概率最大的一个，将非常耗时。为了解决这个问题，通过无词典转录预测的标签序列，在编辑距离度量下往往接近于ground-truth。这表明可以将搜索限制为最近邻的候选对象 $\mathcal{N}_\delta(\mathbf{l}')$ ，其中 σ 是最大编辑距离，l'是为y在无词序模式下转录的序列:

$$\mathbf{l}^* = \arg\max_{\mathbf{l}\in\mathcal{N}_\delta(\mathbf{l}')} p(\mathbf{l}|\mathbf{y}).$$

候选对象 $\mathcal{N}_\delta(\mathbf{l}')$ 可以以bk树数据结构被有效找寻，bk树数据结是一种专门适用于离散度量空间的度量树。bk树的搜索时间复杂度为

$$O(\log|\mathcal{D}|)$$

因此，这个方案很容易扩展到非常大的词典。在本方法中，离线地为词典构造一个bk树。然后，通过查找小于或等于编辑距离 σ 的查询序列，用bk树执行快速在线搜索。

无词典的转录

以定义的条件概率最高的序列l*作为预测，

$$\mathrm{l}^* \approx \mathcal{B}(\arg\max_{\boldsymbol{\pi}} p(\boldsymbol{\pi}|\mathbf{y}))$$

即在每个时间戳 t上获取最可能的标签 ，并将结果序列映射到l*。

END