一文读懂CRNN+CTC文字识别

白裳

文字识别也是图像领域一个常见问题。然而，对于自然场景图像，首先要定位图像中的文字位置，然后才能进行识别。

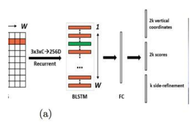
所以一般来说，从自然场景图片中进行文字识别，需要包括2个步骤：

- 文字检测：解决的问题是哪里有文字，文字的范围有多少
- 文字识别：对定位好的文字区域进行识别，主要解决的问题是每个文字是什么，将图像中的文字区域进转化为字符信息。

图1 文字识别的步骤

文字检测类似于目标检测，即用 box 标识出图像中所有文字位置。对于文字检测不了解的读者，请参考本专栏文章：

场景文字检测—CTPN原理与实现
zhuanlan.zhihu.com



本文的重点是如何对已经定位好的文字区域图片进行识别。假设之前已经文字检测算法已经定位图中的“subway”区域（红框），接下来就是文字识别。

图2 文字检测定位文字图像区域

基于RNN文字识别算法主要有两个框架：

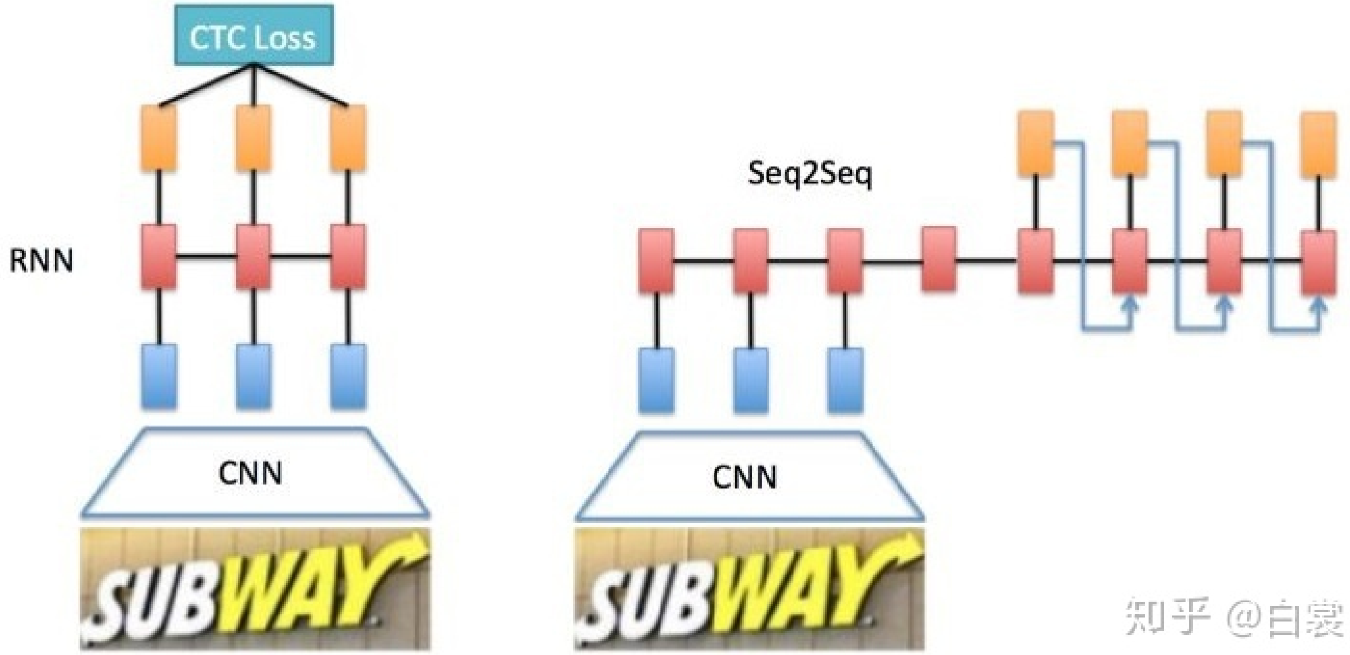


图3 基于RNN文字识别2种基本算法框架

1. CNN+RNN+CTC(CRNN+CTC)
2. CNN+Seq2Seq+Attention

本文主要介绍第一种框架CRNN+CTC，对应TensorFlow 1.15实现代码如下。本文介绍的CRNN网络结构都基于此代码。另外该代码已经支持不定长英文识别。

bai-shang/crnn_ctc_ocr_tf
github.com

需要说明该代码非常简单，只用于原理介绍，不保证泛化性等工程问题，也请勿提问。

CRNN基本网络结构

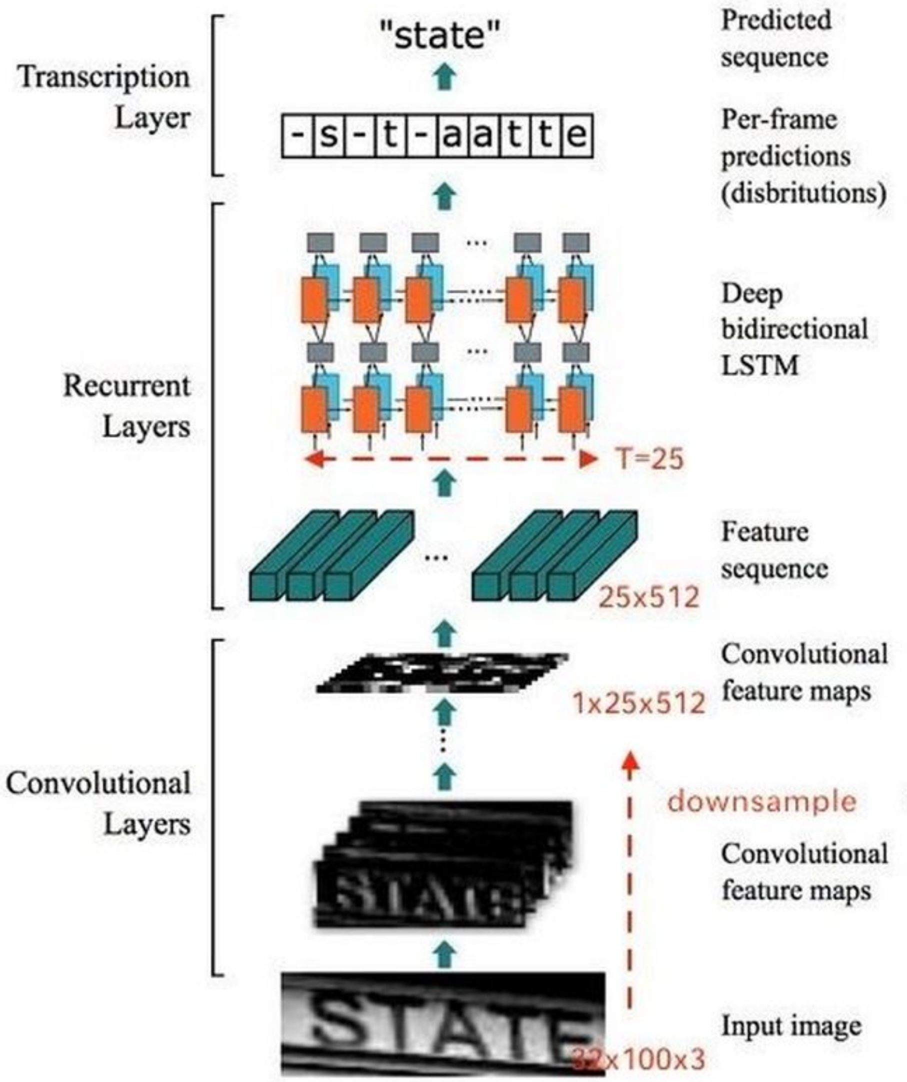


图4 CRNN网络结构（此图按照本文给出的github实现代码画的）

整个CRNN网络可以分为三个部分：

假设输入图像大小为 $(32, 100, 3)$，注意提及图像都是 $(\mathrm{Height}, \mathrm{Width}, \mathrm{Channel})$ 形式。

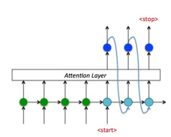
- Convlutional Layers

这里的卷积层就是一个普通的CNN网络，用于提取输入图像的Convolutional feature maps，即将大小为 $(32, 100, 3)$ 的图像转换为 $(1, 25, 512)$ 大小的卷积特征矩阵，网络细节请参考本文给出的实现代码。

- Recurrent Layers

这里的循环网络层是一个深层双向LSTM网络，在卷积特征的基础上继续提取文字序列特征。对RNN不了解的读者，建议参考：

完全解析RNN, Seq2Seq, Attention注意力机制
zhuanlan.zhihu.com



所谓深层RNN网络，是指超过两层的RNN网络。对于单层双向RNN网络，结构如下：

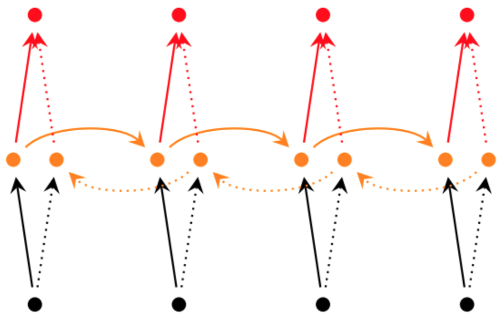


图5 单层双向RNN网络

而对于深层双向RNN网络，主要有2种不同的实现：

```
tf.nn.bidirectional_dynamic_rnn
```

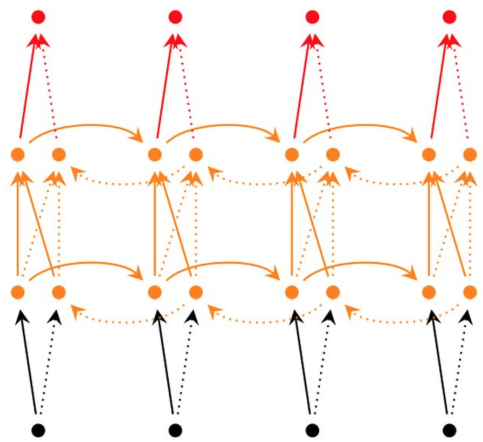


图6 深层双向RNN网络

```
tf.contrib.rnn.stack_bidirectional_dynamic_rnn
```

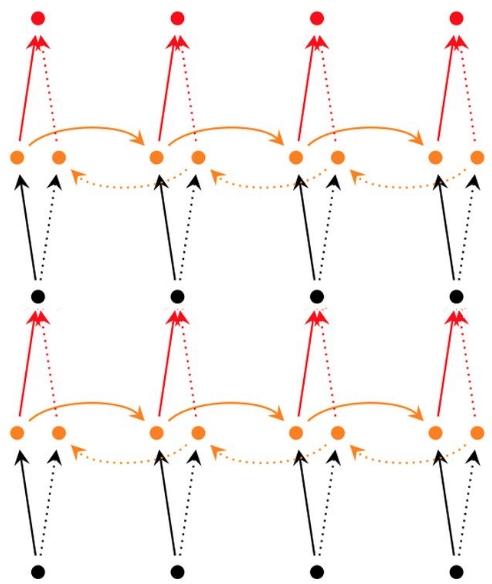


图7 stack形深层双向RNN网络

在CRNN中显然使用了第二种stack形深层双向结构。

由于CNN输出的Feature map是$(1, 25, 512)$大小，所以对于RNN最大时间长度 $T=25$ （即有25个时间输入，每个输入 x_t 列向量有 $D=512$ ）。

- Transcription Layers

将RNN输出做softmax后，为字符输出。

关于代码中输入图片大小的解释：

在本文给出的实现中，为了将特征输入到Recurrent Layers，做如下处理：

- 首先会将图像在固定长宽比的情况下缩放到 $32\times W\times 3$ 大小（ W 代表任意宽度）
- 然后经过CNN后变为 $1\times (W/4)\times 512$
- 针对LSTM设置 $T=(W/4)$ ，即可将特征输入LSTM。

所以在处理输入图像的时候，建议在保持长宽比的情况下将高缩放到 32，这样能够尽量不破坏图像中的文本细节（当然也可以将输入图像缩放到固定宽度，但是这样由于破坏文本的形状，肯定会造成性能下降）。

考虑训练Recurrent Layers时的一个问题：

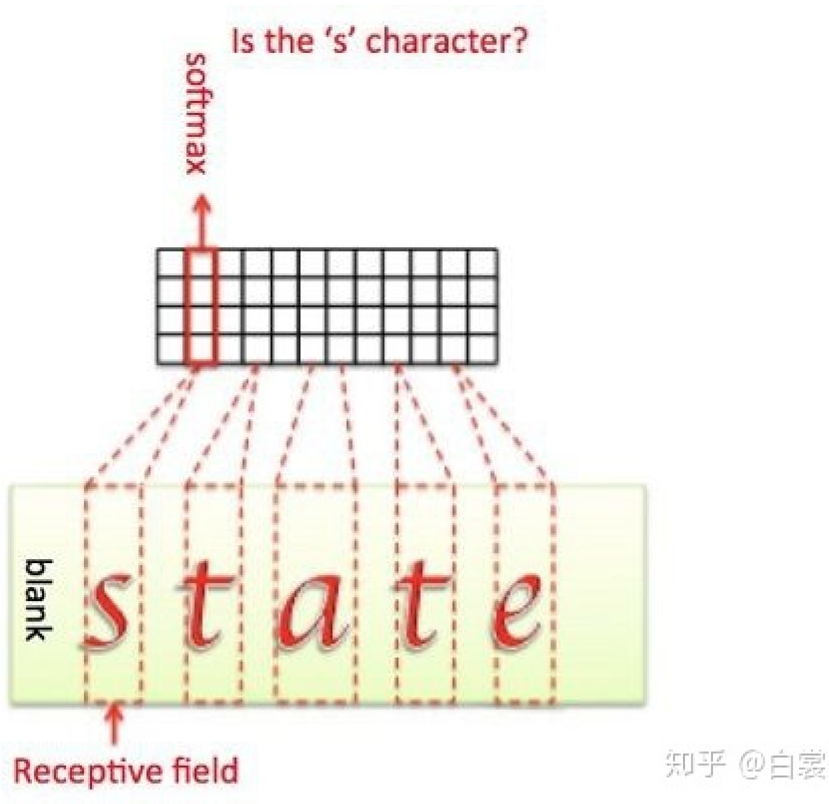


图8 感受野与RNN标签的关系

对于Recurrent Layers，如果使用常见的Softmax cross-entropy loss，则每一列输出都需要对应一个字符元素。那么训练时候每张样本图片都需要标记出每个字符在图片中的位置，再通过CNN感受野对齐到Feature map的每一列获取该列输出对应的Label才能进行训练，如图9。

在实际情况中，标记这种对齐样本非常困难（除了标记字符，还要标记每个字符的位置），工作量非常大。另外，由于每张样本的字符数量不同，字体样式不同，字体大小不同，导致每列输出并不一定能与每个字符一一对应。

当然这种问题同样存在于语音识别领域。例如有人说话快，有人说话慢，那么如何进行语音帧对齐，是一直以来困扰语音识别的巨大难题。

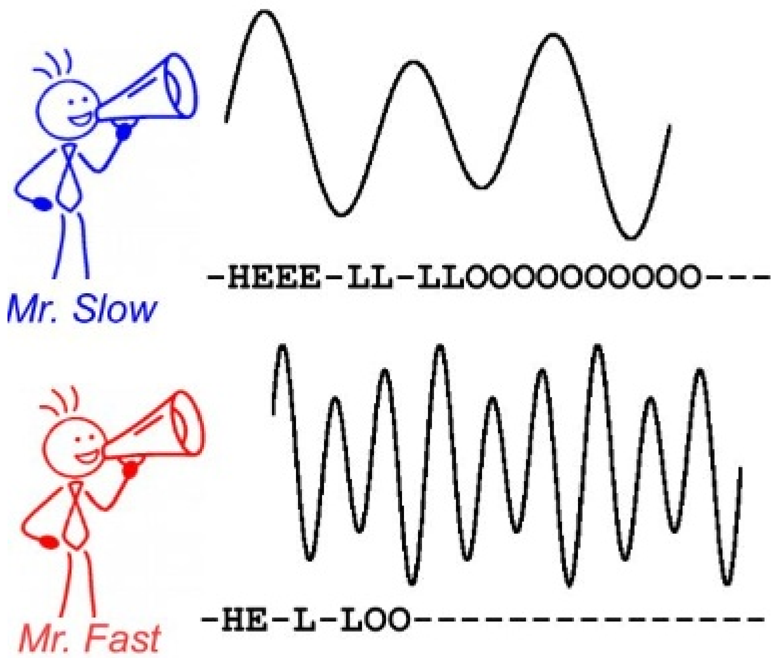


图9

所以CTC提出一种对不需要对齐的Loss计算方法，用于训练网络，被广泛应用于文本行识别和语音识别中。

Connectionist Temporal Classification(CTC)详解

在分析过程中尽量保持和原文符号一致。

Connectionist Temporal Classification: Labelling Unsegmented Sequence...
ftp.idsia.ch

整个CRNN的流程如图10。先通过CNN提取文本图片的Feature map，然后将每一个channel作为 $D=512$ 的时间序列输入到LSTM中。

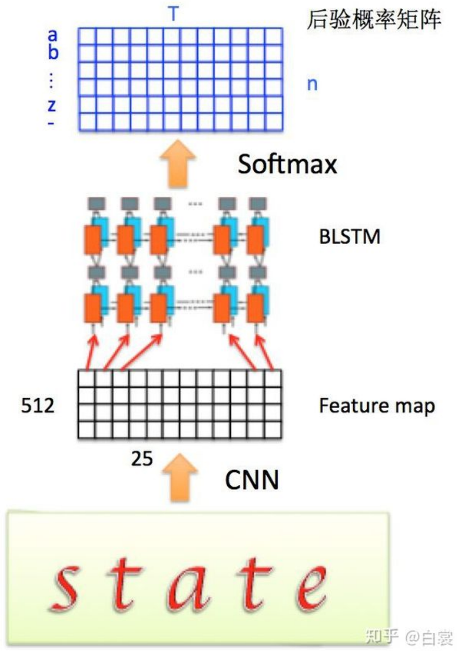


图10 CRNN+CTC框架

为了说明问题，我们定义：

- CNN Feature map

Feature map的每一列作为一个时间片输入到LSTM中。设Feature map大小为 $m\cdot T$ （图11中 $m=512$ ， $T=25$ ）。下文中的时间序列 t 都从 $t=1$ 开始，即 $1\leq t\leq T$ 。

定义为：

$$x=(x^1,x^2,\ldots,x^T)$$

其中 x 每一列 x^t 为：

$$x^t=(x_1^t,x_2^t,\ldots,x_m^t)$$

- LSTM

LSTM的每一个时间片后接softmax，输出 y 是一个后验概率矩阵，定义为：

$$y=(y^1,y^2,\ldots,y^t,\ldots,y^T)$$

其中， y 的每一列 y^t 为：

$$y^t=(y_1^t,y_2^t,\ldots,y_n^t)$$

其中 n 代表需要识别的字符集合长度。由于 y_i^t 是概率，所以服从概率假设： $\sum_k y_k^t=1$

对 y 每一列进行 $\mathbf{argmax}()$ 操作，即可获得每一列输出字符的类别。

那么LSTM可以表示为：

$$y = \mathtt{NET}_w(x)$$

其中 w 代表LSTM的参数。LSTM在输入和输出间做了如下变换：

$$\mathtt{NET}_w : (R^m)^T \rightarrow (R^n)^T$$

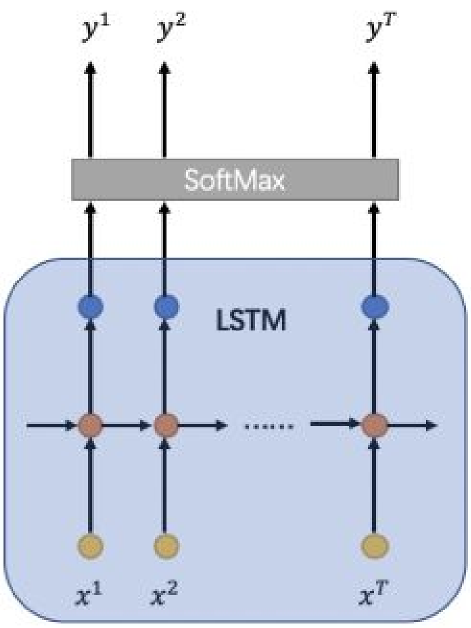


图11

- 空白blank符号

如果要进行 $L = \{a, b, c, \ldots, x, y, z\}$ 的26个英文字符识别，考虑到有的位置没有字符，定义插入blank的字符集合：

$$L' = L \cup \{\text{blank}\}$$

其中blank表示当前列对应的图像位置没有字符（下文以 — 符号表示blank）。

- 关于 B 变换

定义变换 B 如下（原文是大写的 β ，知乎没这个符号）：

$$B : L'^T \rightarrow L^{\leq T}$$

其中 L' 是上述加入blank的长度为 T 的字符集合，经过 B 变换后得到原始 L ，显然对于L 的最大长度有 $|L| \leq T$ 。

举例说明，当 $T = 12$ 时：

$$B(\pi_1) = B(--stta-t--e) = state$$

$$B(\pi_2) = B(sst-aaa-tee-) = state$$

$$B(\pi_3) = B(--sttaa-tee-) = state$$

$$B(\pi_4) = B(sst-aa-t---e) = state$$

对于字符间有blank符号的则不合并：

$$B(\pi_5) = B(-\ s\ t\ a-a\ t\ t\ e-\ \ e-) = staatee$$

当获得LSTM输出 y 后进行 B 变换，即可获得输出结果。显然 B 变换**不是单对单映射**，例如对于不同的 $\pi_1 \sim \pi_4$ 都可获得英文单词state。同时 $|L| = |state| = 5 \leq 12 = T$ 成立。

那么CTC怎么做?

对于LSTM给定输入 x 的情况下，输出为 l 的概率为：

$$p(l|x) = \sum_{\pi \in B^{-1}(l)} p(\pi|x)$$

其中 $\pi \in B^{-1}(l)$ 代表所有经过 B 变换后是 l 的路径 π 。

其中，对于任意一条路径 π 有：

$$p(\pi|x) = \prod_{t=1}^{T} y^t_{\pi_t}\ ,\ \forall \pi \in L'^T$$

注意这里的 $y^t_{\pi_t}$ 中的 π_t ，下标 t 表示 π 路径的每一个时刻；而上面 $\pi_1 \sim \pi_4$ 的下标表示不同的路径。两个下标含义不同注意区分。

***注意上式 $p(\pi|x) = \prod_{t=1}^{T} y^t_{\pi_t}$ 成立有条件，此项不做进一步讨论，有兴趣的读者请自行研究。**

如对于 $T=12$ 的路径 π_1 来说：

$$\pi_1 = (- - s\ t\ t\ a - t - - - e)$$

$$p(\pi_1|x) = y_-^1 \cdot y_-^2 \cdot y_s^3 \cdot y_t^4 \cdot y_t^5 \cdot y_a^6 \cdot y_-^7 \cdot y_t^8 \cdot y_-^9 \cdot y_-^{10} \cdot y_-^{11} \cdot y_e^{12}$$

实际情况中一般手工设置 $T \geq 20$ ，所以有非常多条 $\pi \in B^{-1}(l)$ 路径，即 $|B^{-1}(l)|$ 非常大，无法逐条求和直接计算 $p(l|x)$ 。所以需要一种快速计算方法。

CTC的训练目标

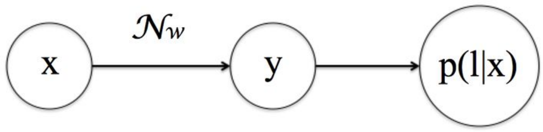


图14

CTC的训练过程，本质上是通过梯度 $\dfrac{\partial p(l|x)}{\partial w}$ 调整LSTM的参数 w ，使得对于输入样本为 $\pi \in B^{-1}(l)$ 时使得 $p(l|x)$ 取得最大。

例如下面图14的训练样本，目标都是使得 $l = state$ 时的输出 $p(l = \text{state}|x)$ 变大。

图14

CTC借用了HMM的“向前—向后”(forward-backward)算法来计算 $p(l|x)$

要计算 $p(l|x)$ ，由于有blank的存在，定义路径 l' 为在路径 l 每两个元素以及头尾插入blank。那么对于任意的 l'_i 都有 $l'_i \in L'$ （其中 $L' = L \cup \{blank\}$ ）。如：

$$l = state$$

$$l' = -s - t - a - t - e-$$

显然 $|l'| = 2|l| + 1$ ，其中 $|l|$ 是路径的最大长度，如上述例子中 $|l| = |state| = 5$ 。

定义所有经 B 变换后结果是 l 且在 t 时刻结果为 l_k （记为 $\pi_t = l_k$ ）的路径集合为 $\{\pi|\pi \in B^{-1}(l), \pi_t = l_k\}$ 。

求导：

$$\frac{\partial p(l|x)}{\partial y_k^t} = \frac{\partial \sum_{\pi \in B^{-1}(l)} p(\pi|x)}{\partial y_k^t} = \frac{\partial \sum_{\pi \in B^{-1}(l), \pi_t = l_k} p(\pi|x)}{\partial y_k^t} + \frac{\partial \sum_{\pi \in B^{-1}(l), \pi_t \neq l_k} p(\pi|x)}{\partial y_k^t}$$

注意上式中第二项与 y_k^t 无关，所以：

$$\frac{\partial p(l|x)}{\partial y_k^t} = \frac{\partial \sum_{\pi \in B^{-1}(l), \pi_t = l_k} p(\pi|x)}{\partial y_k^t}$$

而上述 $\frac{\partial p(l|x)}{\partial y_k^t}$ 就是恰好与概率 y_k^t 相关的路径，即 t 时刻都经过 l_k（$\pi_t = l_k$ ）。

举例说明，还是看上面的例子 π_1, π_2 （这里的下标 $1, 2$ 代表不同的路径）：

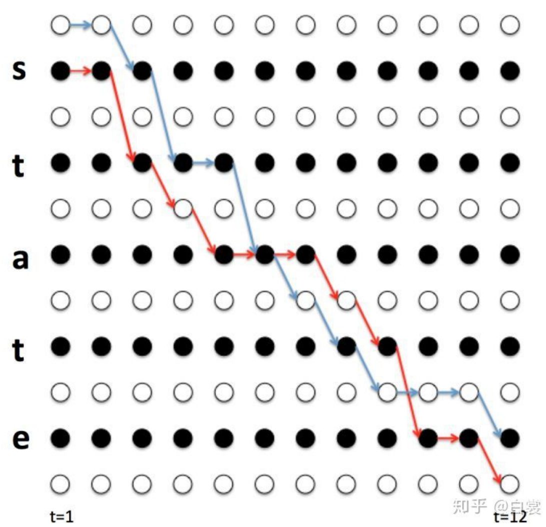


图15

蓝色路径 π_1 ：

$$B(\pi_1) = B(- - s\ t\ t\ a - t - - - e) = state$$

$$p(\pi_1|x) = y_-^1 \cdot y_-^2 \cdot y_s^3 \cdot y_t^4 \cdot y_t^5 \cdot y_a^6 \cdot y_-^7 \cdot y_t^8 \cdot y_-^9 \cdot y_-^{10} \cdot y_-^{11} \cdot y_e^{12}$$

红色路径 π_1 ：

$$B(\pi_2) = B(s\ s\ t - a\ a\ a - t\ e\ e-) = state$$

$$p(\pi_2|x) = y_s^1 \cdot y_s^2 \cdot y_t^3 \cdot y_-^4 \cdot y_a^5 \cdot y_a^6 \cdot y_a^7 \cdot y_-^8 \cdot y_t^9 \cdot y_e^{10} \cdot y_e^{11} \cdot y_-^{12}$$

还有 π_3, π_4 没有画出来。

而 $\pi_1, \pi_2, \pi_3, \pi_4$ 在 $t = 6$ 时恰好都经过 $\pi_6 = a$ （此处下标代表路径 π 的 t 时刻的字符）。所有类似于 $\pi_1, \pi_2, \pi_3, \pi_4$ 经过 B 变换后结果是 $l = state$ 且在 $\pi_6 = a$ 的路径集合表示为 $\{\pi|\pi \in B^{-1}(l), \pi_6 = a\}$ 。

观察 $\pi_1, \pi_2, \pi_3, \pi_4$ 。记 π_1 蓝色为 $b(blue)$ ， π_2 红色路径为 $r(red)$ ， π_1, π_2 可以表示：

$$\pi_1 = b = b_{1:5} + a_6 + b_{7:12}$$

$$\pi_2 = r = r_{1:5} + a_6 + r_{7:12}$$

那么 π_3, π_4 可以表示为：

$$\pi_3 = b_{1:5} + a_6 + r_{7:12}$$

$$\pi_4 = r_{1:5} + a_6 + b_{7:12}$$

计算：

$$\frac{\partial p(l|x)}{\partial y_a^6} = \frac{\partial \sum_{\pi \in B^{-1}(l), \pi_6 = a} p(\pi|x)}{\partial y_a^6} = \frac{\partial p(\pi_1|x) + \partial p(\pi_2|x) + \partial p(\pi_3|x) + \partial p(\pi_4|x) + \dots}{\partial y_a^6}$$

为了观察规律，单独计算 $p(\pi_1|x) + p(\pi_2|x) + p(\pi_3|x) + p(\pi_4|x)$ 。

$$\begin{aligned}
& p(\pi_1|x) + p(\pi_2|x) + p(\pi_3|x) + p(\pi_4|x) \\
& = y_-^1 \cdot y_-^2 \cdot y_s^3 \cdot y_t^4 \cdot y_t^5 \cdot y_a^6 \cdot y_-^7 \cdot y_t^8 \cdot y_-^9 \cdot y_-^{10} \cdot y_-^{11} \cdot y_e^{12} \\
& \quad + y_s^1 \cdot y_s^2 \cdot y_t^3 \cdot y_-^4 \cdot y_a^5 \cdot y_a^6 \cdot y_a^7 \cdot y_-^8 \cdot y_t^9 \cdot y_e^{10} \cdot y_e^{11} \cdot y_-^{12} \\
& \quad + y_-^1 \cdot y_-^2 \cdot y_s^3 \cdot y_t^4 \cdot y_t^5 \cdot y_a^6 \cdot y_a^7 \cdot y_-^8 \cdot y_t^9 \cdot y_e^{10} \cdot y_e^{11} \cdot y_-^{12} \\
& \quad + y_s^1 \cdot y_s^2 \cdot y_t^3 \cdot y_-^4 \cdot y_a^5 \cdot y_a^6 \cdot y_-^7 \cdot y_t^8 \cdot y_-^9 \cdot y_-^{10} \cdot y_-^{11} \cdot y_e^{12}
\end{aligned}$$

不妨令：

$$\texttt{forward} = p(b_{1:5} + r_{1:5}|x) = y_-^1 \cdot y_-^2 \cdot y_s^3 \cdot y_t^4 \cdot y_t^5 + y_s^1 \cdot y_s^2 \cdot y_t^3 \cdot y_-^4 \cdot y_a^5$$

$$\texttt{backward} = p(b_{7:12} + r_{7:12}|x) = y_-^7 \cdot y_t^8 \cdot y_-^9 \cdot y_-^{10} \cdot y_-^{11} \cdot y_e^{12} + y_a^7 \cdot y_-^8 \cdot y_t^9 \cdot y_e^{10} \cdot y_e^{11} \cdot y_-^{12}$$

那么 $p(\pi_1|x) + p(\pi_2|x) + p(\pi_3|x) + p(\pi_4|x)$ 可以表示为：

$$p(\pi_1|x) + p(\pi_2|x) + p(\pi_3|x) + p(\pi_4|x) = \texttt{forward} \cdot y_a^t \cdot \texttt{backward}$$

推广一下，所有经过 B 变换为 l 且 $\pi_6 = a$ 的路径（即 $\{\pi|\pi \in B^{-1}(l), \pi_6 = a\}$ ）可以写成如下形式：

$$\sum_{\pi \in B^{-1}(l), \pi_6 = a} p(\pi|x) = \texttt{forward} \cdot y_a^t \cdot \texttt{backward}$$

进一步推广，所有经过 B 变换为 l 且 $\pi_t = l_k$ 的路径（即 $\{\pi|\pi \in B^{-1}(l), \pi_t = l_k\}$ ）也都可以写作：

$$\sum_{\pi \in B^{-1}(l), \pi_t = l_k} p(\pi|x) = \texttt{forward} \cdot y_{l_k}^t \cdot \texttt{backward}$$

所以，定义前向递推概率和 $\texttt{forward} = \alpha_t(s)$ ：

对于一个长度为 T 的路径 π ，其中 $\pi_{1:t}$ 代表该路径前 t 个字符， $\pi_{t:T}$ 代表后 $T-t$ 个字符。

$$\alpha_t(s) = \sum_{\pi \in B(\pi_{1:t}) = l_{1:s}} \prod_{t'=1}^{t} y_{\pi_{t'}}^{t'}$$

其中 $\pi \in B(\pi_{1:t}) = l_{1:s}$ 表示前 t 个字符 $\pi_{1:t}$ 经过 B 变换为的 $l_{1:s}$ 的前半段子路径。 $\alpha_t(s)$ 代表了 t 时刻经过 l_s 的路径概率中 $1 \sim t$ 概率之和，即前向递推概率和。

由于当 $t=1$ 时路径只能从blank或 l_1 开始，所以 $\alpha_t(s)$ 有如下性质：

$$\alpha_1(-) = y_-^1$$

$$\alpha_1(l_1) = y_{l_1}^1$$

$$\alpha_1(l_t) = 0 \,,\ \forall t > 1$$

如上面的例子中 $T=12$ ，$l=state$ ，$l_{1:3}=sta$ 。对于所有 $\pi\in B(\pi_{1:6})=l_{1:3}$ 路径，当 $t=1$ 时只能从blank和 s 字符开始。

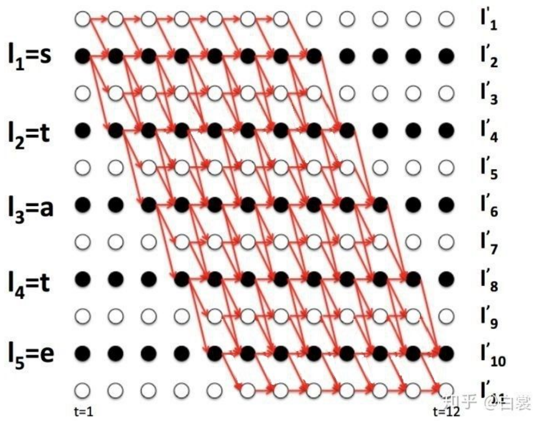


图16

图16是 $T=12$ 时经过压缩路径后能够变为 $l=state$ 的所有路径 $B^{-1}(l)$ 。观察图15会发现对于 $\alpha_6(a)$ 有如下递推关系：

$$\alpha_6(a)=(\alpha_5(a)+\alpha_5(t)+\alpha_5(-))\cdot y_a^6$$

也就是说，如果 $t=6$ 时刻是字符 a ，那么 $t=5$ 时刻只可能是字符 $a,t,blank$ 三选一，否则经过 B 变换后无法压缩成 $l=state$ 。

那么更一般的：

$$\alpha_t(l'_k)=(\alpha_{t-1}(l'_k)+\alpha_{t-1}(l'_{k-1})+\alpha_{t-1}(-))\cdot y_{l'_k}^t$$

同理，定义反向递推概率和 backword $=\beta_t(s)$ ：

$$\beta_t(s)=\sum_{\pi\in B(\pi_{t:T})=l_{s:|l|}}\prod_{t'=t}^{T}y_{\pi_{t'}}^{t'}$$

其中 $\pi\in B(\pi_{t:T})=l_{s:|l|}$ 表示后 $T-t$ 个字符 $\pi_{t:T}$ 经过 B 变换为的 $l_{s:|l|}$ 的后半段子路径。 $\beta_t(s)$ 代表了 t 时刻经过 l_s 的路径概率中 $t\sim T$ 概率之和，即反向递推概率和。

由于当 $t=T$ 时路径只能以blank或 $l_{|l'|}$ 结束，所以有如下性质：

$$\beta_T(-) = y_-^T$$

$$\beta_T(l'_{|l'|}) = y_{l'_{|l'|}}^T$$

$$\beta_T(l'_{|l'|-i}) = 0 \text{ , } \forall i > 0$$

如上面的例子中 $T = 12$ ，$l = state$ ，$|l| = |state| = 5$ ，$l_{3:5} = ate$ 。对于所有 $\pi \in B(\pi_{6:12}) = l_{3:6}$ 路径，当 $t = 12$ 时只能以 $l'_{|l'|} = l'_{11} = -$ （blank字符）或 $l'_{|l'|} = l'_{11} = e$ 字符结束。

观察图15会发现对于 $\beta_6(a)$ 有如下递推关系

$$\beta_6(a) = (\beta_7(a) + \beta_7(t) + \beta_7(-)) \cdot y_a^6$$

与 $\alpha_t(s)$ 同理，对于 $\beta_t(s)$ 有如下递推关系：

$$\beta_t(l'_k) = (\beta_{t+1}(l'_k) + \beta_{t+1}(l'_{k+1}) + \beta_{t+1}(-)) \cdot y_{l'_k}^t$$

那么forward和backward相乘有：

$$\alpha_t(l'_k)\beta_t(l'_k) = \sum_{\pi \in B^{-1}(l), \pi_t = l'_k} y_{l'_k}^t \prod_{t=1}^T y_{\pi_t}^t$$

或：

$$\alpha_t(l_k)\beta_t(l_k) = \sum_{\pi \in B^{-1}(l), \pi_t = l_k} y_{l_k}^t \prod_{t=1}^T y_{\pi_t}^t$$

注意，$y_{l_k}^t$ 与 $y_{l'_k}^t$ 可以通过图16的关系对应，如 $y_{l_1}^t = y_{l'_2}^t$ ，$y_{l_2}^t = y_{l'_4}^t$ 。

对比 $p(l|x)$ ：

$$p(l|x) = \sum_{\pi \in B^{-1}(l)} p(\pi|x) = \sum_{\pi \in B^{-1}(l)} \prod_{t=1}^T y_{\pi_t}^t$$

可以得到 $p(l|x)$ 与forward和backward递推公式之间的关系：

$$p(l|x) = \sum_{\pi \in B^{-1}(l), \pi_t = l_k} \frac{\alpha_t(l_k)\beta_t(l_k)}{y_{l_k}^t}$$

*** 为什么有上式 $\alpha_t(l_k)\beta_t(l_k) = \sum_{\pi \in B^{-1}(l), \pi_t = l_k} y_{l_k}^t \prod_{t=1}^{T} y_{\pi_t}^t$ 成立呢?**

回到图15，为了方便分析，假设只有 $\pi_1 \sim \pi_4$ 共4条在 $t = 6$ 时刻经过字符 a 且 B 变换为 l 的路径，即：

$$\pi_1 = b_{1:5} + a_6 + b_{7:12}$$

$$\pi_2 = r_{1:5} + a_6 + r_{7:12}$$

$$\pi_3 = b_{1:5} + a_6 + r_{7:12}$$

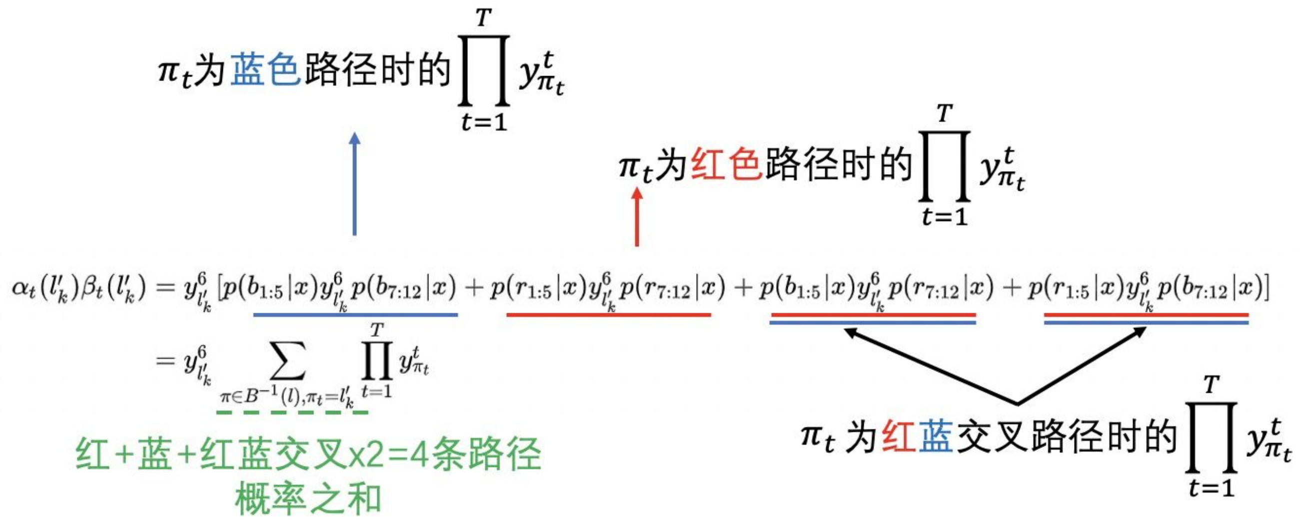
$$\pi_4 = r_{1:5} + a_6 + b_{7:12}$$

那么此时（注意虽然表示路径用 $\pi = \pi_1 + \pi_2$ 加法，但是由于 π_1 和 π_2 两件独立事情同时发生，所以 π 路径的概率 $p(\pi|x) = p(\pi_1|x)p(\pi_2|x)$ 是乘法）：

$$\alpha_t(l'_k) = p(b_{1:5}|x) \cdot y_{l'_k}^6 + p(r_{1:5}|x) \cdot y_{l'_k}^6$$

$$\beta_t(l'_k) = p(b_{7:12}|x) \cdot y_{l'_k}^6 + p(r_{7:12}|x) \cdot y_{l'_k}^6$$

则有：



训练CTC

对于LSTM，有训练集合 $S=\{(x_1,z_1),(x_1,z_1),\ldots,(x_N,z_N)\}$ ，其中 x 是图片经过CNN计算获得的Feature map， z 是图片对应的OCR字符label（label里面没有blank字符）。

现在我们要做的事情就是：通过梯度 $\frac{\partial p(l|x)}{\partial w}$ 调整LSTM的参数 w ，使得对于输入样本为 $\pi \in B^{-1}(z)$ 时有 $p(l|x)$ 取得最大。**所以如何计算梯度才是核心。**

单独来看CTC输入（即LSTM输出） y 矩阵中的某一个值 y_k^t （注意 y_k^t 与 $y_{l_k}^t$ 含义相同，都是在 t 时 $\pi_t = l_k$ 的概率）：

$$\frac{\partial p(l|x)}{\partial y_k^t} = \frac{\partial \sum_{\pi \in B^{-1}(l),\pi_t=l_k} \frac{\alpha_t(l_k)\beta_t(l_k)}{y_{l_k}^t}}{\partial y_{l_k}^t} = \frac{\sum_{\pi \in B^{-1}(l),\pi_t=l_k} \alpha_t(l_k)\beta_t(l_k)}{(y_{l_k}^t)^2}$$

上式中的 $\alpha_t(l_k)\beta_t(l_k)$ 是通过递推计算的常数，任何时候都可以通过递推快速获得，那么即可快速计算梯度 $\frac{\partial p(l|x)}{\partial y_k^t}$ ，之后梯度上升算法你懂的。

CTC编程接口

在Tensorflow中官方实现了CTC接口：

```
tf.nn.ctc_loss(
    labels,
    inputs,
    sequence_length,
    preprocess_collapse_repeated=False,
    ctc_merge_repeated=True,
    ignore_longer_outputs_than_inputs=False,
    time_major=True
)
```

在Pytorch中需要使用针对框架编译的warp-ctc：github.com/SeanNaren/wa...

2020.4更新，目前Pytorch已经有CTC接口：

```
torch.nn.CTCLoss(blank=0,reduction='mean',zero_infinity=False)
```

CTC总结

CTC是一种Loss计算方法，用CTC代替Softmax Loss，训练样本无需对齐。CTC特点：

- 引入blank字符，解决有些位置没有字符的问题

- 通过递推，快速计算梯度

看到这里你也应该大致了解MFCC+CTC在语音识别中的应用了（图17来源）。

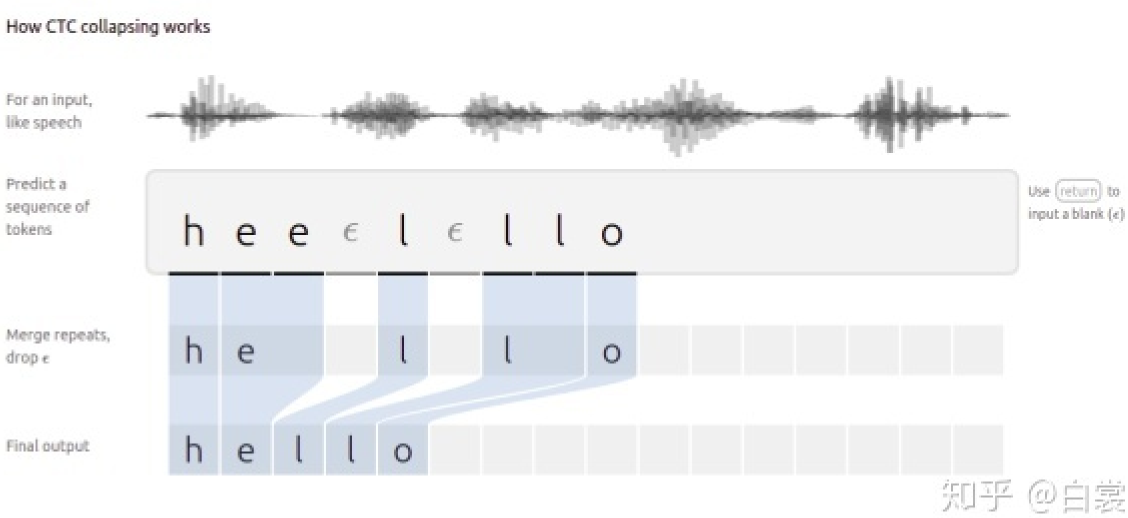


图17 MFCC+CTC在语音识别中的应用

CRNN+CTC总结

这篇文章的核心，就是将CNN/LSTM/CTC三种方法结合：

- 首先CNN提取图像卷积特征
- 然后LSTM进一步提取图像卷积特征中的序列特征
- 最后引入CTC解决训练时字符无法对齐的问题

即提供了一种end2end文字图片识别算法，也算是方向的简单入门。

特别说明

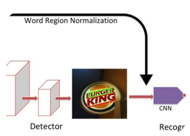
一般情况下对一张图像中的文字进行识别需要以下步骤

1. 定位文稿中的图片，表格，文字区域，区分文字段落（版面分析）
2. 进行文本行识别（识别）
3. 使用NLP相关算法对文字识别结果进行矫正（后处理）

本文介绍的CRNN框架只是步骤2的一种识别算法，其他非本文内容。CTC你学会(fei)了么？

想了解其他文字识别方法，请点这里：

文字识别方法整理
zhuanlan.zhihu.com



求求你们点个赞吧！

本文章只是介绍ctc原理，不包含1v1辅导服务。