

# 语音识别技术精炼

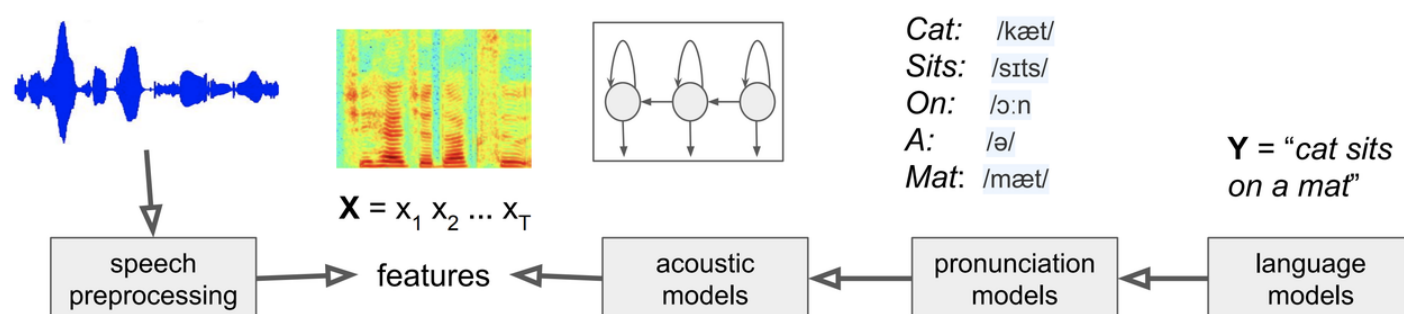
## 1. 传统语音识别系统

在了解End-to-End模型之前，先看看经典的语音识别系统是怎样工作的以及为什么需要End-to-End模型。

经典语音识别系统通常有四个组成部分：特征提取，如利用输入的wav提取MFCC特征，然后再经过三个独立的模型再求得它们概率的乘积得到总的概率。

💡 一个特征提取+三个models

1. Acoustic model：声学模型，即根据之前提取的特征预测每个对应的音素(phoneme)，传统上用HMM-GMM（隐马尔可夫-高斯混合模型）。**模型训练时需要做对齐操作（alignment），生成每一帧特征对应的标签，如果标签的颗粒度为音素，那就需要知道每帧特征对应哪个音素。**
2. Pronunciation model：词典模型，即根据音素组合成词语的发音, 传统上用一些pronunciation table。
3. Language model：语言模型，即根据发音预测对应的文本, 传统上用一些n-gram model。



神经网络技术发展后，特征提取又可用CNN来做，其他部分也可用DNN或一些RNN结构如LSTM来做。

🦄 存在问题：

1. 模块较多，pipeline较长，容易造成级联损失；
2. 需要做对齐操作，边界处存在对齐错误，导致信息损失。

## 2. 新质语音识别系统

### 2.1 CTC

对齐

 CTC全称是Connectionist Temporal Classification，主要解决前面讲的对齐alignment的问题。

### CTC loss

- 为什么会提出该loss？解决什么问题？
  - 针对输入和输出序列长度不一致的问题，如手写识别、语音识别
  - $X = [x_1, x_2, \dots, x_T] \rightarrow Y = [y_1, y_2, \dots, y_U]$

t h e q u i c k b r o w n f o x



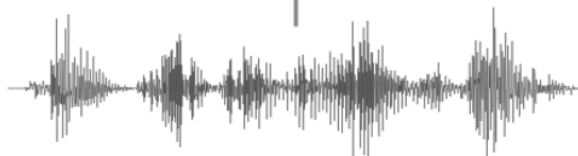
The quick brown fox

手写识别

输入：一张图像，像素序列或坐标序列

输出：字符串序列

j u m p s o v e r t h e l a z y d o g



语音识别


输入：一段语音，幅度值序列

输出：字符串序列

假设输入语音的特征训练为  $X = x_1, x_2 \dots x_T$

对应的label序列为  $Y = y_1, y_2, \dots, y_U$

对于ASR，通常  $U \leq T$

 CTC只是取消了显示对齐步骤，内部还是有对齐操作的，实在计算loss时，目标函数 $P(Y|X)$ 采用输入和输出的所有对齐概率累和。

为了说明CTC对齐的具体形式，首先考虑一个简单的方法。让我们用一个例子来说明。假设输入长度为6，且 $Y=[c,a,t]$ 。一种对齐 $X$ 和 $Y$ 的方法是将每个输入步骤分配一个输出字符，并合并重复项。

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	input ( $X$ )
c	c	a	a	a	t	alignment
c		a		t		output ( $Y$ )

### 这种方法存在两个问题：

(1) 强迫每个输入步骤都与某个输出对齐是没有意义的。例如，在语音识别中，输入可能包含没有对应输出的静音段落。

(2) 我们没有办法产生连续多个字符的输出。考虑对齐 [h, h, e, l, l, l, o]。合并重复项将会产生“helo”而不是“hello”。

$\epsilon$

为了解决这些问题，CTC引入了一个新的标记  $\epsilon$  (blank)。 $\epsilon$  标记不对应任何东西，只是从输出中移除。

如果Y中有两个相同的字符，那么一个有效的对齐必须在它们之间有一个  $\epsilon$ 。有了这个规则，我们可以区分折叠为“hello”和那些折叠为“helo”的对齐。

h h e  $\epsilon$   $\epsilon$  l l l  $\epsilon$  l l o

First, merge repeat characters.

h e  $\epsilon$  l  $\epsilon$  l o

Then, remove any  $\epsilon$  tokens.

h e l l o

The remaining characters are the output.

h e l l o

重新回到cat例子，输入长度为六。这里有一些有效和无效的对齐的例子。

### Valid Alignments

€ c c € a t

c c a a t t

c a € € € t

### Invalid Alignments

c € c € a t

c c a a t   

c € € € | t t

corresponds to  
 $Y = [c, c, a, t]$

has length 5

missing the 'a'

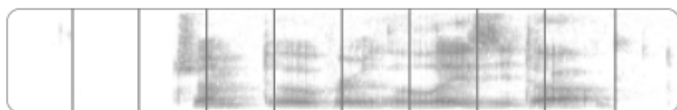


CTC对齐有一些显著的特性：

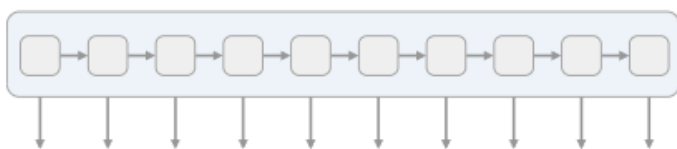
1. X和Y之间的对齐是单调的。如果我们推进到下一个输入，我们可以保持相应的输出不变，或者推进到下一个。
2. X到Y的对齐是多对一的。一个或多个输入元素可以对齐到单个输出元素，但反之则不行。
3. Y的长度不能大于X的长度。

## 损失函数

CTC对齐为我们提供了一种自然的方式来从每个时间步的概率转换到输出序列的概率。



We start with an input sequence, like a spectrogram of audio.



The input is fed into an RNN, for example.

h	h	h	h	h	h	h	h	h	h
e	e	e	e	e	e	e	e	e	e
l	l	l	l	l	l	l	l	l	l
o	o	o	o	o	o	o	o	o	o
€	€	€	€	€	€	€	€	€	€

The network gives  $p_t(a | X)$ , a distribution over the outputs  $\{h, e, l, o, \epsilon\}$  for each input step.

h	e	€	l	l	€	l	l	o	o
h	h	e	l	l	€	€	l	€	o
€	e	€	l	l	€	€	l	o	o

With the per time-step output distribution, we compute the probability of different sequences

h	e	l	l	o
e	l	l	o	
h	e	l	o	

By marginalizing over alignments, we get a distribution over outputs.

CTC目标对于单个(X,Y)对是：

$$p(Y | X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^T p_t(a_t | X)$$

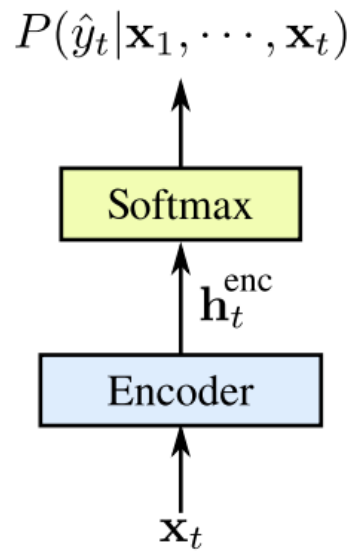
The CTC conditional **probability**

**marginalizes** over the set of valid alignments

computing the **probability** for a single alignment step-by-step.

为什么采用累和概率而不是最大概率？ 一种近似解。

CTC方案主要是在损失函数上的创新，模型架构跟传统方案是一致的，可用下图表示：



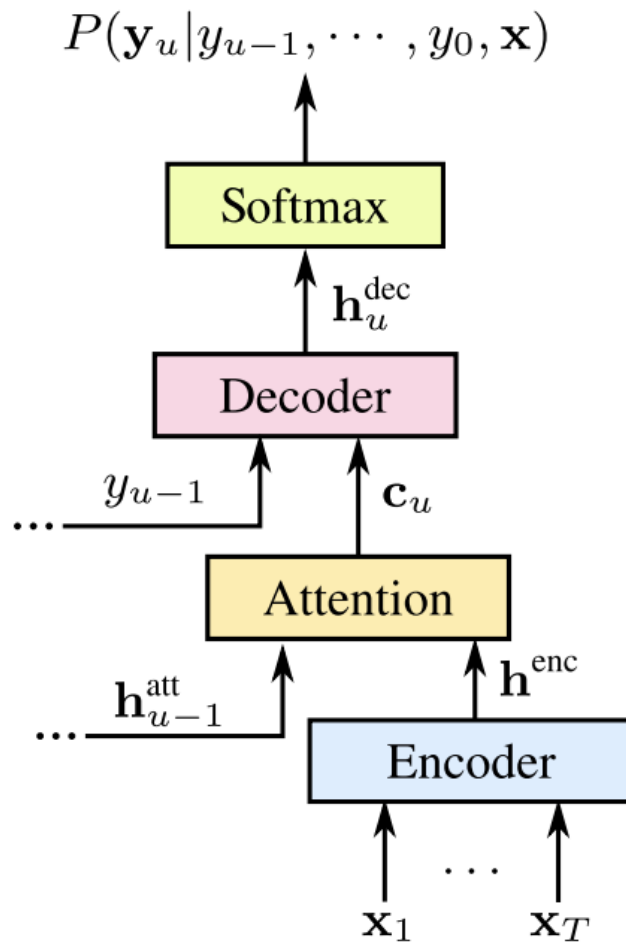
## 2.2 LAS

LAS，全称Listen Attend and Spell，它利用了attention机制来进行有效的alignment。

LAS模型主要由两大部分组成：

1. Listener即Encoder，利用多层RNN从输入序列提取隐藏特征。
2. Attend and Spell，即Attention用来得到context vector，decoder利用context vector以及之前的输出来产生相应的最终的输出。

其模型结构如下：

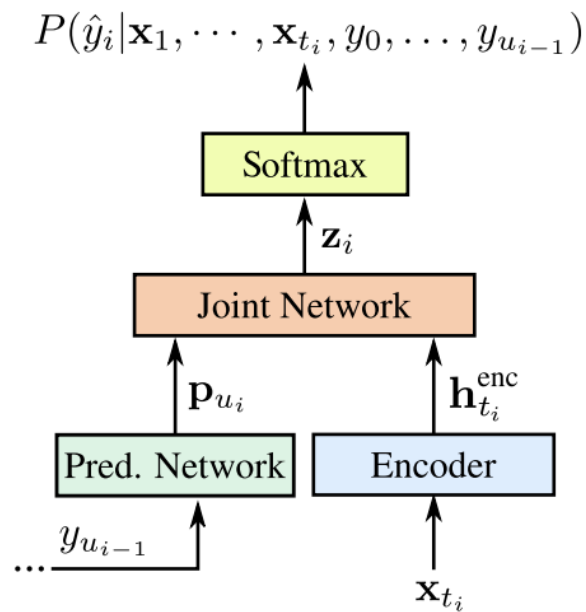


LAS模型由于考虑了上下文的所有信息，所以它的精确度可能较其他模型略高，但是同时由于它需要上下文的信息所以没法进行streaming的ASR，另外输入的语音长度对于模型的准确度也有较大的影响。

## 2.3 RNN-T

RNN-T全称是Recurrent Neural Network Transducer，是在CTC的基础上改进的。**CTC的缺点是它没有考虑输出之间的dependency，即  $y_t$  与之前帧的  $y_j, j < t$  没有任何关联**，而RNN-T则在CTC模型的Encoder基础上，又加入了将之前的输出作为输入的一个RNN，称为Prediction Network，再将其输出的隐藏向量 $p_u$ 与encoder得到的  $h_{\text{enc}}$  放到一个joint network中，得到输出logit再将其传到softmax layer得到对应的class的概率。

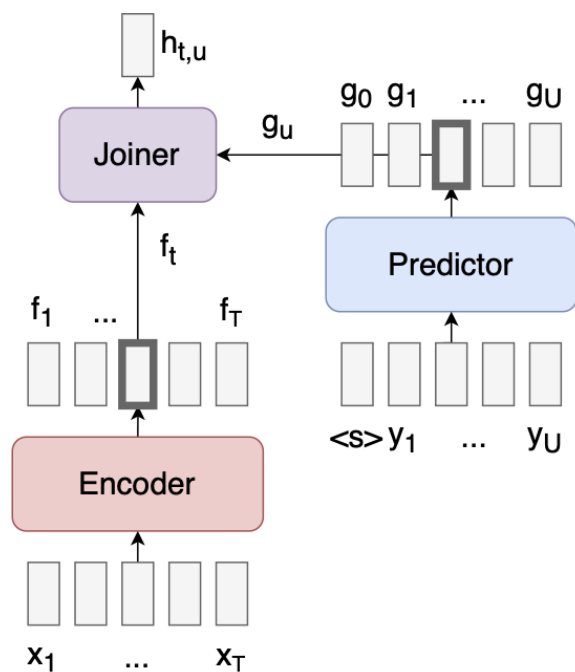
整体模型结构如下



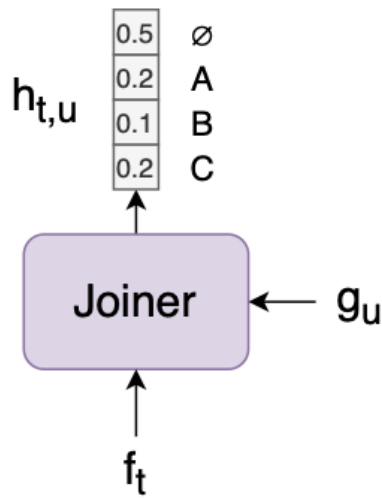
### 两大优点：

1. 可以做流式语音识别，模型无需看完整条输入音频即可输出；
2. Predictor网络输入为文本，可以用纯文本数据进行预训练。

接下来讲下具体细节





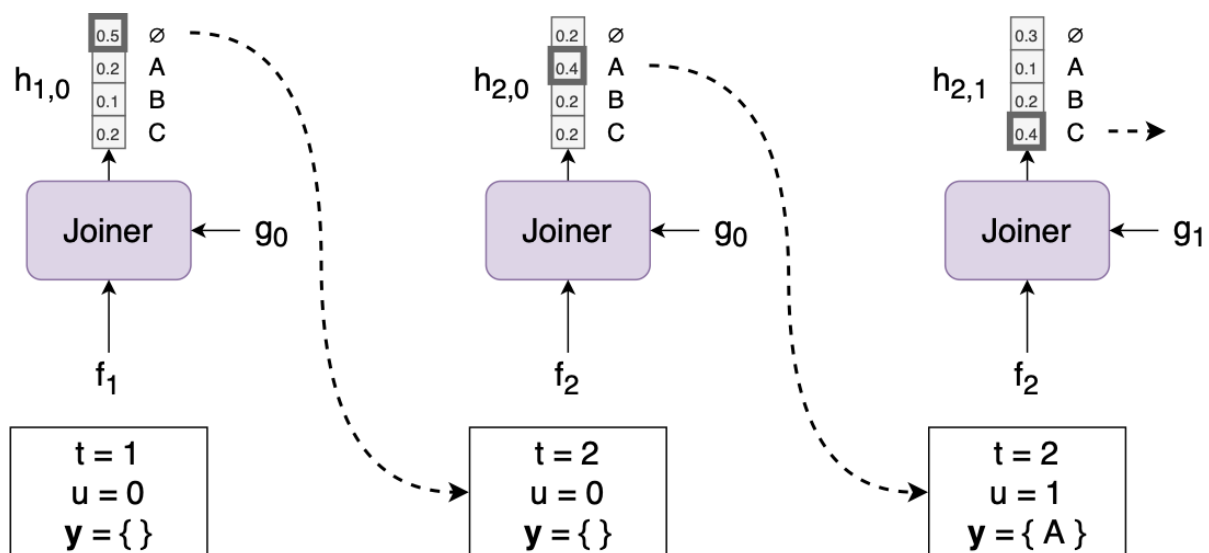


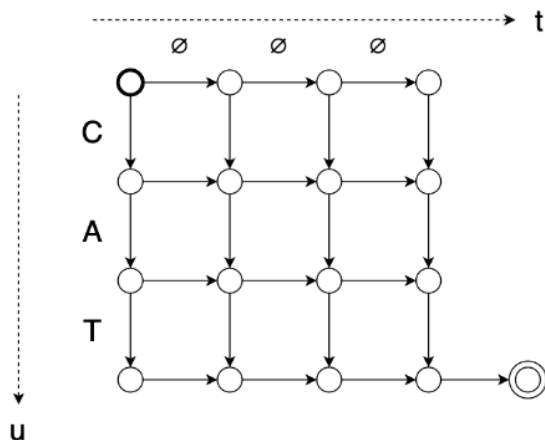
Given an input sequence  $x$ , generating an output sequence  $y$  can be done using a simple **greedy search** algorithm:

1. Start by setting  $t:=1$ ,  $u:=0$ , and  $y:=$  an empty list.
2. Compute  $f_t$  using  $x$  and  $g_u$  using  $y$ .
3. Compute  $h_{t,u}$  using  $f_t$  and  $g_u$ .
4. If the argmax of  $h_{t,u}$  **is a label**, **set  $u:=u+1$** , and output the label (append it to  $y$  and feed it back into the predictor).

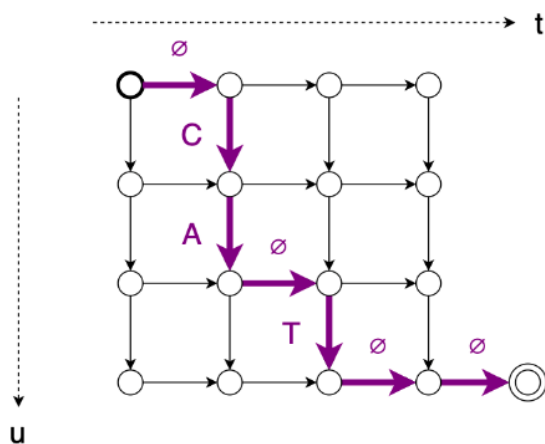
If the argmax of  $h_{t,u}$  **is  $\emptyset$** , **set  $t:=t+1$**  (in other words, just move to the next input timestep and output nothing).

5. If  $t=T+1$ , we're done. Else, go back to step 2.





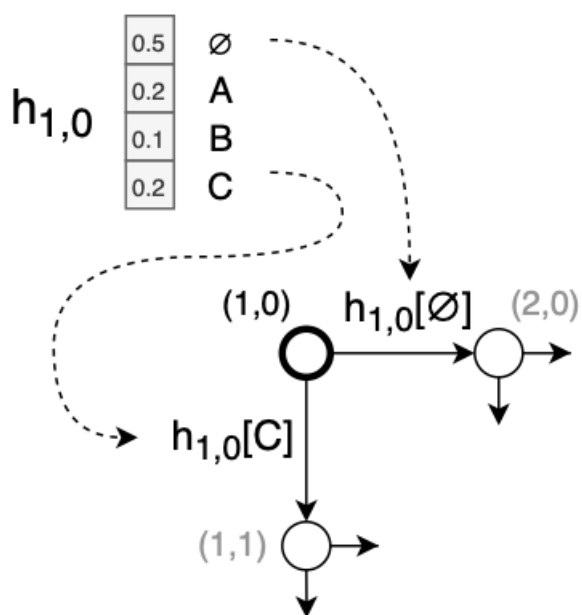
Here's one alignment:  $\mathbf{z} = \emptyset, C, A, \emptyset, T, \emptyset, \emptyset$



$\mathbf{z} = \emptyset, C, A, \emptyset, T, \emptyset, \emptyset$

对齐概率  $\downarrow p(\mathbf{z}|\mathbf{x}) = h_{1,0}[\emptyset] \cdot h_{2,0}[C] \cdot h_{2,1}[A] \cdot h_{2,2}[\emptyset] \cdot h_{3,2}[T] \cdot h_{3,3}[\emptyset] \cdot h_{4,3}[\emptyset]$

下图边的概率对应  $h_{t,u}$



## 训练

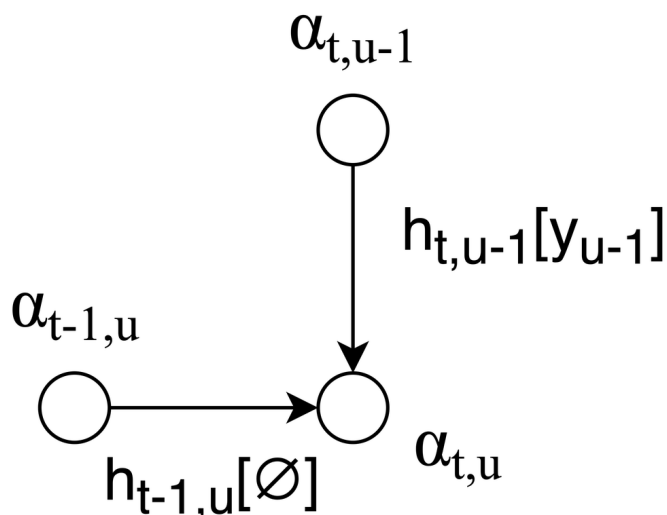
将  $p(\mathbf{y}|\mathbf{x})$  定义为所有对齐概率的累和，模型目标是 最小化损失函数  $-\log p(\mathbf{y}|\mathbf{x})$

采用前后向算法计算  $\alpha_{t,u}$

$$1 \leq t \leq T, 0 \leq u \leq U$$

$$\alpha_{t,u} = \alpha_{t-1,u} \cdot h_{t-1,u}[\emptyset] + \alpha_{t,u-1} \cdot h_{t,u-1}[y_{u-1}]$$

将这一计算过程可视化如下图



$$p(y|x) = \alpha_{T,U} \cdot h_{T,U}[\emptyset]$$

转换到log域：

$$\log \alpha_{t,u} = \log \text{sumexp}([\log \alpha_{t-1,u} + \log h_{t-1,u}[\emptyset], \log \alpha_{t,u-1} + \log h_{t,u-1}[y_{u-1}]])$$

$$\log p(y|x) = \log \alpha_{T,U} + \log h_{T,U}[\emptyset]$$

前面介绍的是前向算法，类似的可以采用后向算法。

## 内存空间

一般来说，Transducer模型看起来是个好主意。但这里有一个问题（也可能是Transducer直到最近才流行起来的未明说的原因）：

假设  $T=1000$ ， $U=100$ ， $L=1000$  个标签，以及批处理大小  $B=32$ 。那么为了存储所有  $(t,u)$  的  $h_{t,u}$  来运行前向-后向算法，我们需要一个大小为  $B \times T \times U \times L=3,200,000,000$  或者如果使用单精度浮点数的话是12.8GB的张量。而这仅仅是输出张量，还有连接网络的隐藏单元激活，其大小为  $B \times T \times U \times d_{\text{joiner}}$ 。

## 空间查找

我们之前提到过，可以使用贪婪搜索来预测  $y$ ，总是选择  $h_{t,u}$  的最高输出。但是，使用Beam Search可以获得更好的结果，即保持多个  $y$  的假设列表，并在每个输入时间步更新它们。

SpeechBrain工具包里有实现。

## 2.4 总结

这里仅仅简单的介绍与比较了几种常见的end-to-end的ASR模型，实际应用中还有很多的trick，比如建模单元的选择，选择wordpiece还是grapheme（对于英文来说，grapheme指的是26个英文字母+1空格+12常用标点。对于中文来说，grapheme指的是3755一级汉字+3008二级汉字+16标点符号），要不要和external language模型做fusion，如何考虑user context，如何考虑最新名词实体等（音乐、电视剧）等。

## 2.5 代码实战

<https://github.com/chinabing/RNN-Transducer>

### 参考资料：

<https://distill.pub/2017/ctc/>

<https://zhuanlan.zhihu.com/p/62836549>

<https://lorenlugosch.github.io/posts/2020/11/transducer/>