端到端(E2E)语音识别

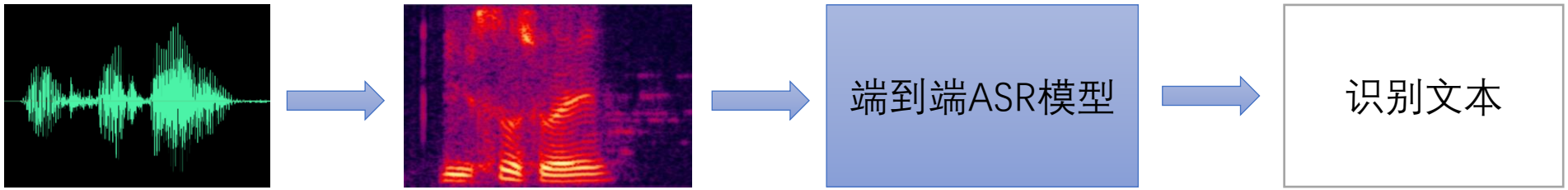
洪青阳

纲要

1. CTC
2. RNN-T
3. Attention模型
4. Hybrid CTC/Attention
5. Transformer/Conformer
6. 本章小结

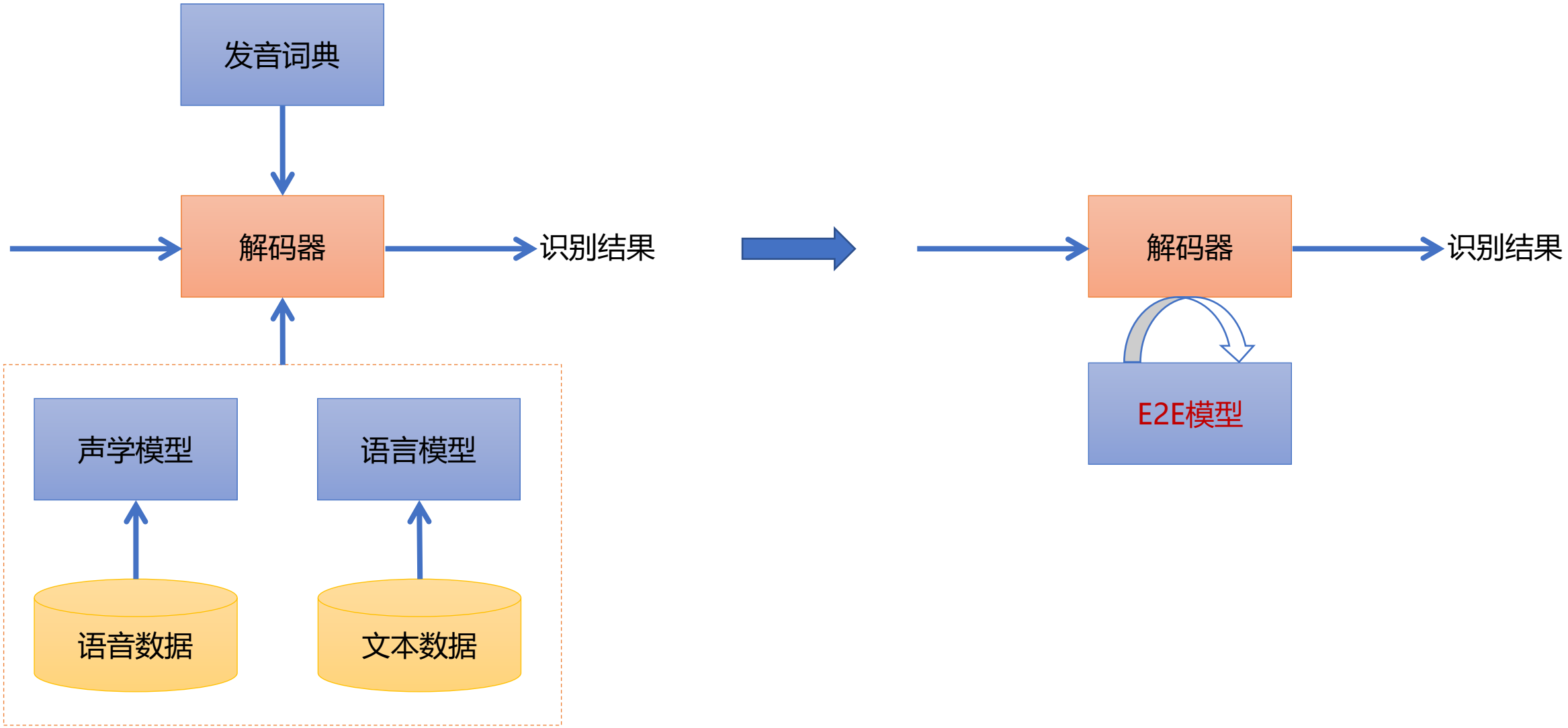
发展趋势

- 近年来，端到端（E2E）模型开始被应用于语音识别领域。不同于传统语音识别系统中的发音字典、声学模型和语言模型，E2E语音识别只分为输入端的语音特征和输出端的文本信息。



- 传统语音识别系统的三大组件被融合为一个E2E模型，直接实现输入语音到输出文本的转化，不需要单独训练声学和语言模型。

端到端（E2E）语音识别



E2E模型

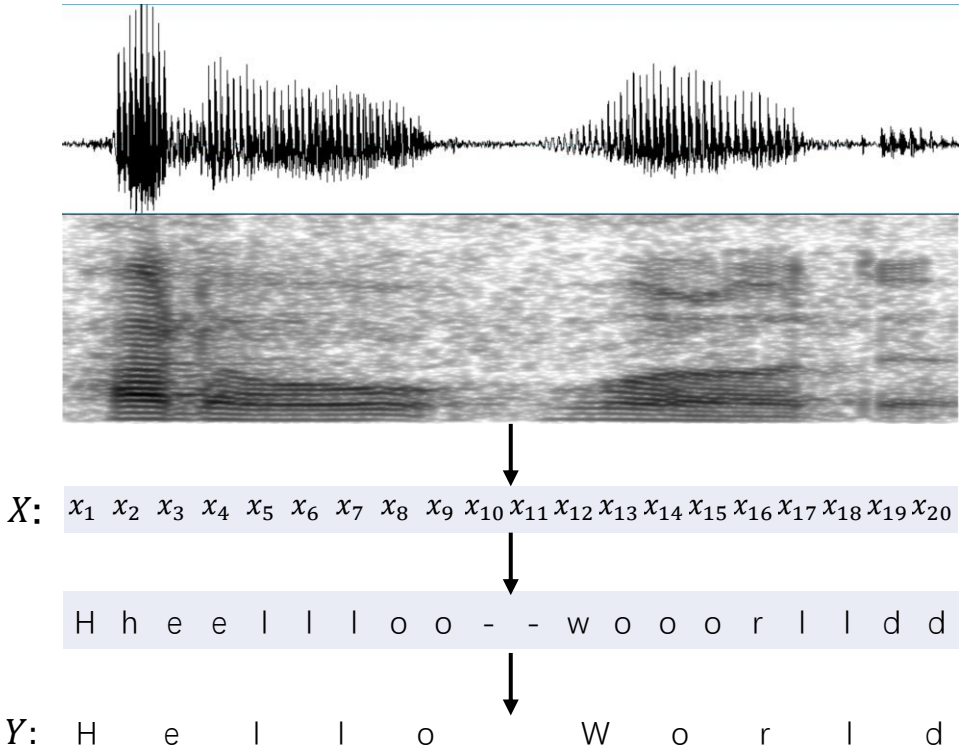
- CTC（Connectionist Temporal Classification）
- RNN-T
- Attention
- Transformer
- Conformer

CTC

- CTC全称连接时序分类。2006年，Graves等人在ICML 2006上首次提出CTC方法，直接自动对齐输出标签和输入序列，不再像DNN-HMM模型那样需要对齐标注。

- CTC假定输入符号是相互独立的，输出序列与输入序列是按时间顺序单调对齐的，然后通过动态规划来解决序列对齐问题。

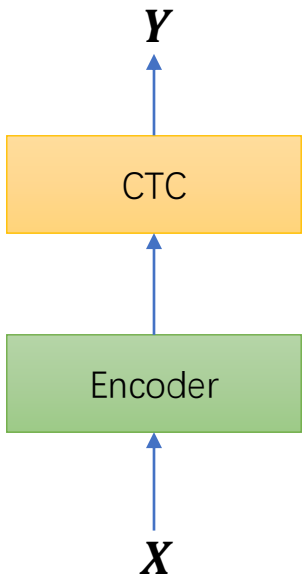
CTC



输出序列（“Hello World”）的字符个数和输入序列$\boldsymbol{X}$的长度并不相等，无法直接匹配，但通过中间的重复字符和空白字符（“-”），可建立跟输入序列的一一对应关系。

CTC

- 假设训练集为$\boldsymbol{S}$，每个样本$(\boldsymbol{X},\boldsymbol{Y})$由输入序列$\boldsymbol{X}=\{x_1,x_2,\cdots,x_T\}$和输出序列$\boldsymbol{Y}=\{y_1,y_2,\cdots,y_U\}$组成，其中$T$是输入序列的长度，$U$是输出序列的长度，$\boldsymbol{Y}$的每个标注来自于建模单元集$\boldsymbol{L}$。
- CTC的训练目标是使$\boldsymbol{X}$和$\boldsymbol{Y}$尽量匹配，即输出概率$P(\boldsymbol{Y}|\boldsymbol{X})$最大化。输入的序列经过Encoder之后，通过CTC衡量和真实的序列是否接近。



CTC

输出序列$\boldsymbol{Y}$可由各种CTC路径$A_{CTC}(\boldsymbol{X},\boldsymbol{Y})$生成，这些路径包含了标注重复与空白标签各种可能的组合，例如输出序列(a, b, c)可能来自于以下序列：

a	a	b	b	b	c
a	b	b	c	c	c
a	-	b	-	-	c
-	a	-	b	-	c
-	a	b	b	-	c

CTC

穷举所有可能的路径$A_{CTC}(\boldsymbol{X},\boldsymbol{Y})$，则$\boldsymbol{Y}$由$\boldsymbol{X}$生成的概率为：

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \sum_{\hat{\boldsymbol{Y}}\in A_{CTC}(\boldsymbol{X},\boldsymbol{Y})} P\left(\hat{\boldsymbol{Y}}\middle|\boldsymbol{X}\right)$$

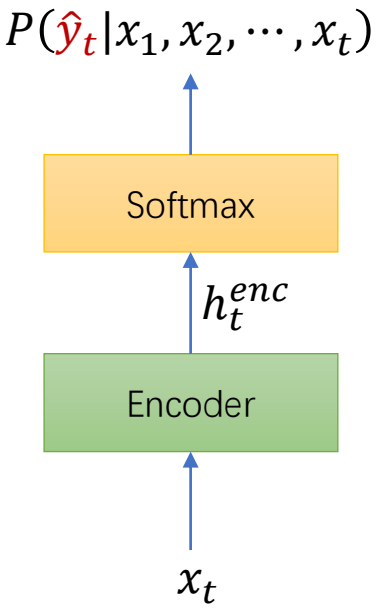
其中$\hat{\boldsymbol{Y}}$表示$\boldsymbol{X}$和$\boldsymbol{Y}$在CTC网络下某条对齐路径，其长度跟输入序列$\boldsymbol{X}$一致，即$\hat{\boldsymbol{Y}} = \{\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_T\}$。$\hat{\boldsymbol{Y}}$去除重复和空白标签后得到$\boldsymbol{Y}$。

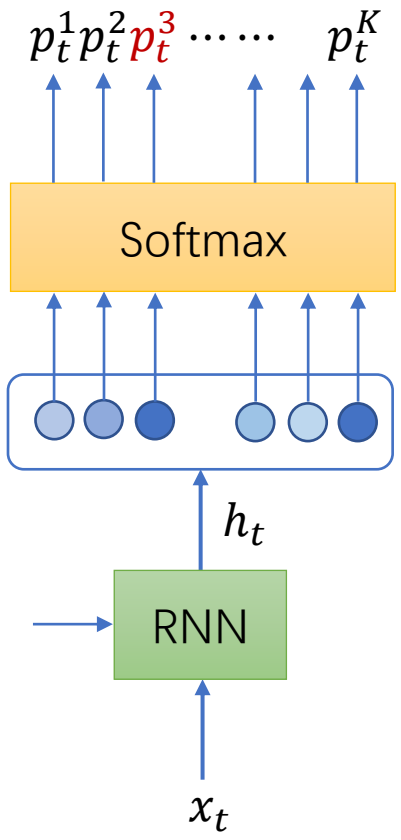
路径$\hat{\boldsymbol{Y}}$出现的概率是每个时刻的输出概率的乘积：

$$P\left(\hat{\boldsymbol{Y}}\middle|\boldsymbol{X}\right) = \prod_{t=1}^{T} P(\hat{y}_t|x_t), \forall\hat{\boldsymbol{Y}}\epsilon\boldsymbol{L}'^{T}$$

其中$\hat{y}_t$表示$\hat{\boldsymbol{Y}}$路径在t时刻的输出标签（$\boldsymbol{L}'$中一个），$P(\hat{y}_t|x_t)$是其对应的输出概率。

CTC

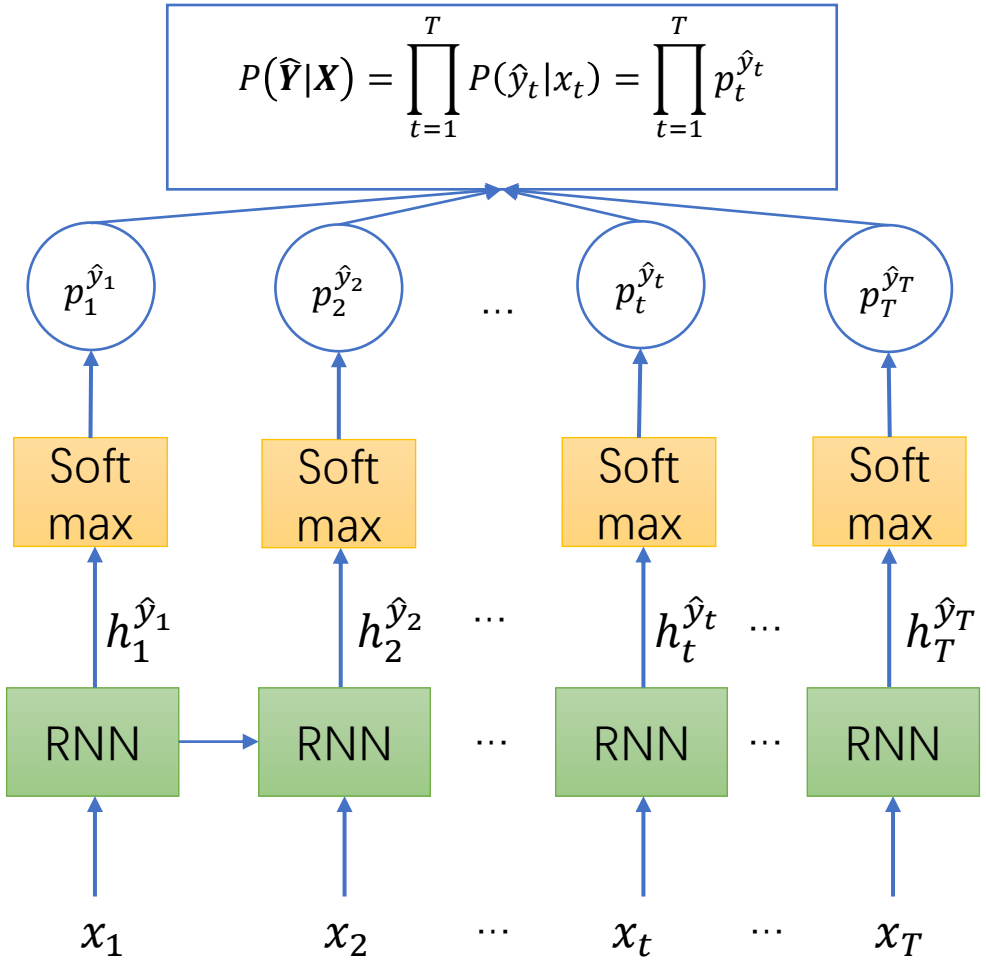




CTC从原始输入x_t到最后输出$\hat{y}_t$的过程

在某个时刻的输出$\hat{y}_t$根据其在建模单元集的索引选择K个输出值的其中一个，如对应的索引是第3个，则其值为p_t^3，即$P(\hat{y}_t|x_t) = p_t^3$。

CTC



输入序列$\boldsymbol{X} = \{x_1, x_2, \cdots, x_T\}$分别通过RNN得到隐藏层输出$\left\{h_1^{\hat{y}_1}, h_2^{\hat{y}_2}, \cdots, h_T^{\hat{y}_T}\right\}$，通过Softmax转化得到每个时刻的输出概率$p_t^{\hat{y}_t}$，这些概率再通过连乘得到$P(\hat{\boldsymbol{Y}}|\boldsymbol{X})$。

CTC损失函数

- 定义输入帧序列为$\boldsymbol{X}$，帧长为T，输出序列为$\boldsymbol{Y}$，该序列可由各种CTC路径$A_{CTC}(\boldsymbol{X},\boldsymbol{Y})$生成，则$\boldsymbol{Y}$由$\boldsymbol{X}$生成的概率为

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \sum_{\hat{\boldsymbol{Y}} \in A_{CTC}(\boldsymbol{X},\boldsymbol{Y})} P(\hat{\boldsymbol{Y}}|\boldsymbol{X})$$

- 设训练集为$\boldsymbol{S}$，CTC损失函数：

$$L(\boldsymbol{S}) = -\sum_{(\boldsymbol{X},\boldsymbol{Y}) \epsilon \boldsymbol{S}} \ln P(\boldsymbol{Y}|\boldsymbol{X})$$

- CTC训练优化的目标是使$L(\boldsymbol{S})$最小化，但计算$P(\boldsymbol{Y}|\boldsymbol{X})$的复杂度非常高，需要穷举所有路径，类似HMM状态的遍历过程。
- 为简化计算过程，我们可参照HMM的前向-后向算法来求解CTC的局部和全局概率。

前向算法

令$\boldsymbol{Y}'$为输出序列$\boldsymbol{Y}=\{y_1,y_2,\cdots,y_U\}$在句子开头和每个标签中间加上空白符，即$\boldsymbol{Y}'=\{b,y_1,b,y_2,b,\cdots,y_U,b\}$，其中$b$表示空白符。$\boldsymbol{Y}$的长度为$U$，则$\boldsymbol{Y}'$的长度为$2U+1$。

用$\alpha_t(s)$表示已经输出部分观察值$x_1,x_2,\cdots,x_t$，并且到达标签为s的概率：

$$\alpha_t(s)=P(x_1,x_2,\cdots,x_t,s)$$

前向算法按输入序列的时间顺序，从前向后递推计算输出概率。

1. 初始化

$$\alpha_1(1)=P(b|x_1)$$
$$\alpha_1(2)=P(y_1'|x_1)$$
$$\alpha_1(s)=0,\forall s>2$$

2. 迭代计算

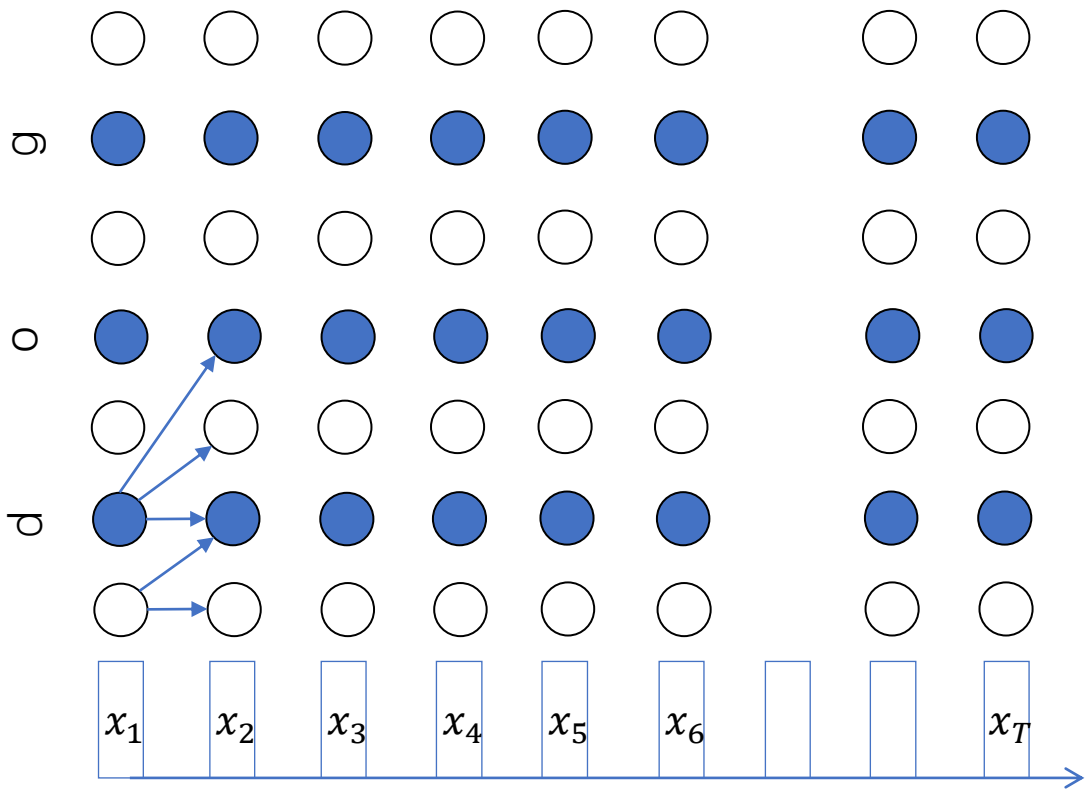
$$\alpha_t(s)=\begin{cases}\big(\alpha_{t-1}(s)+\alpha_{t-1}(s-1)\big)P(y_s'|x_t),\text{如果 } y_s'=b \text{ 或 } y_{s-2}'=y_s' \\ \big(\alpha_{t-1}(s)+\alpha_{t-1}(s-1)+\alpha_{t-1}(s-2)\big)P(y_s'|x_t),\text{其他}\end{cases}$$

apple

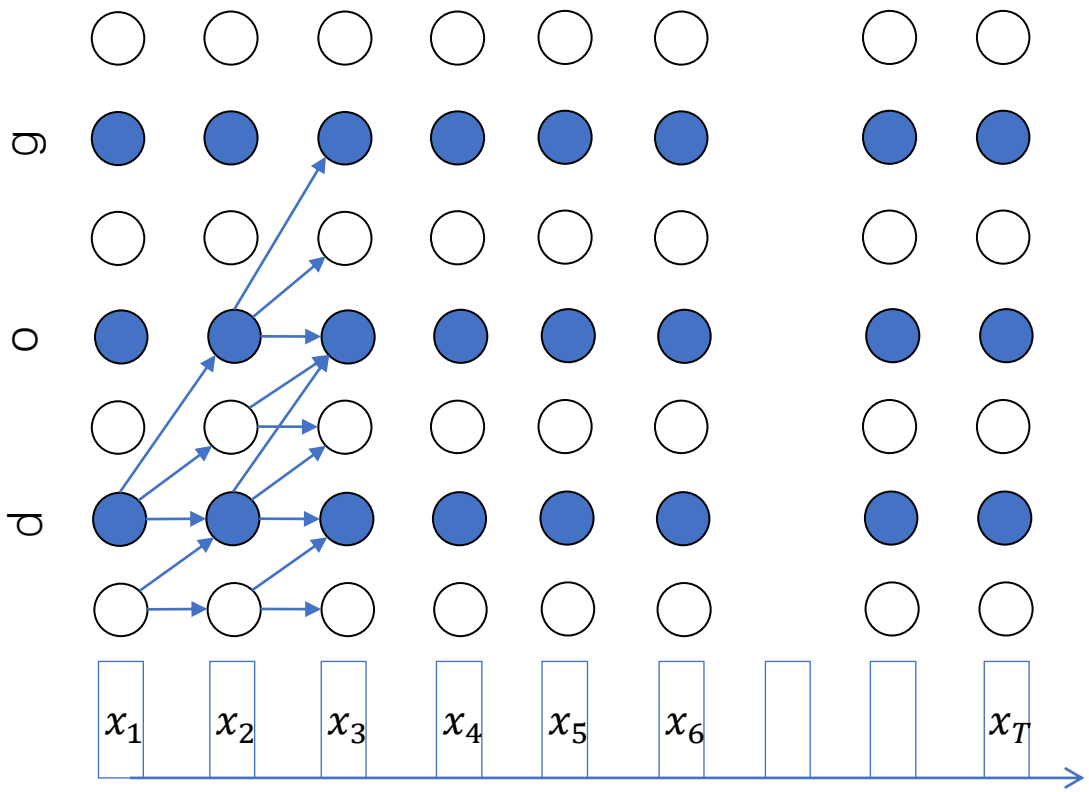
3. 终止计算

$$P(\boldsymbol{Y}|\boldsymbol{X})=\alpha_T(2U+1)+\alpha_T(2U)$$

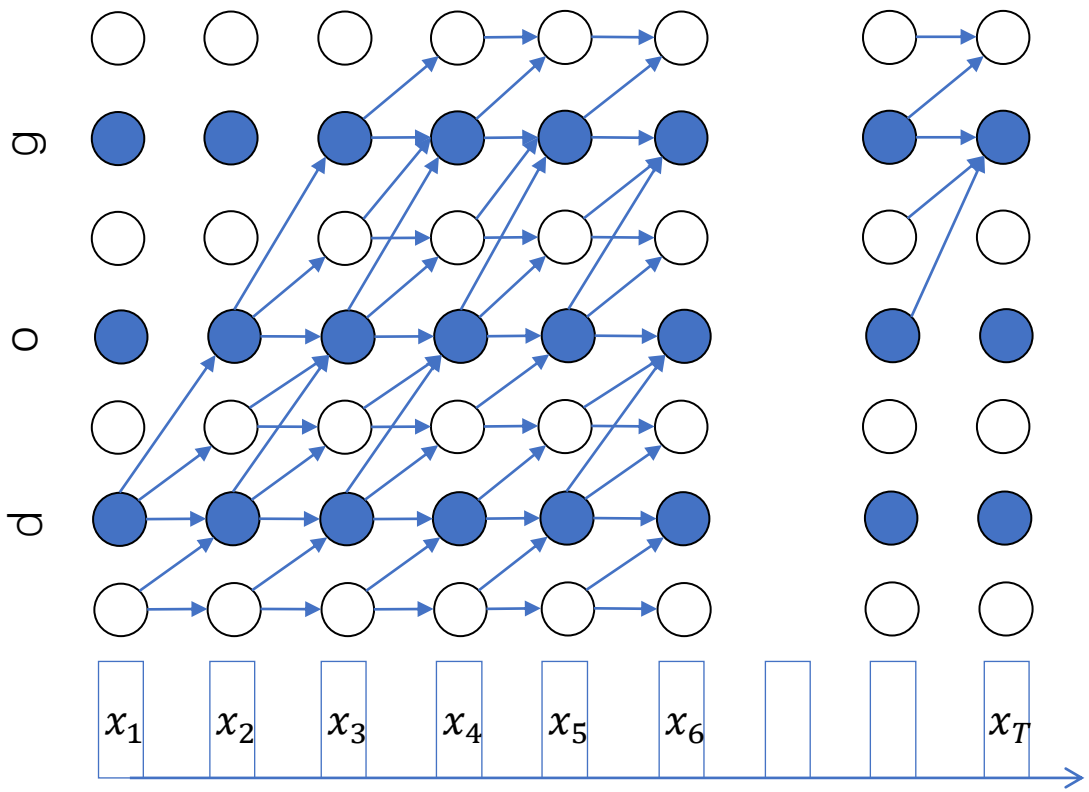
前向算法（t=2）



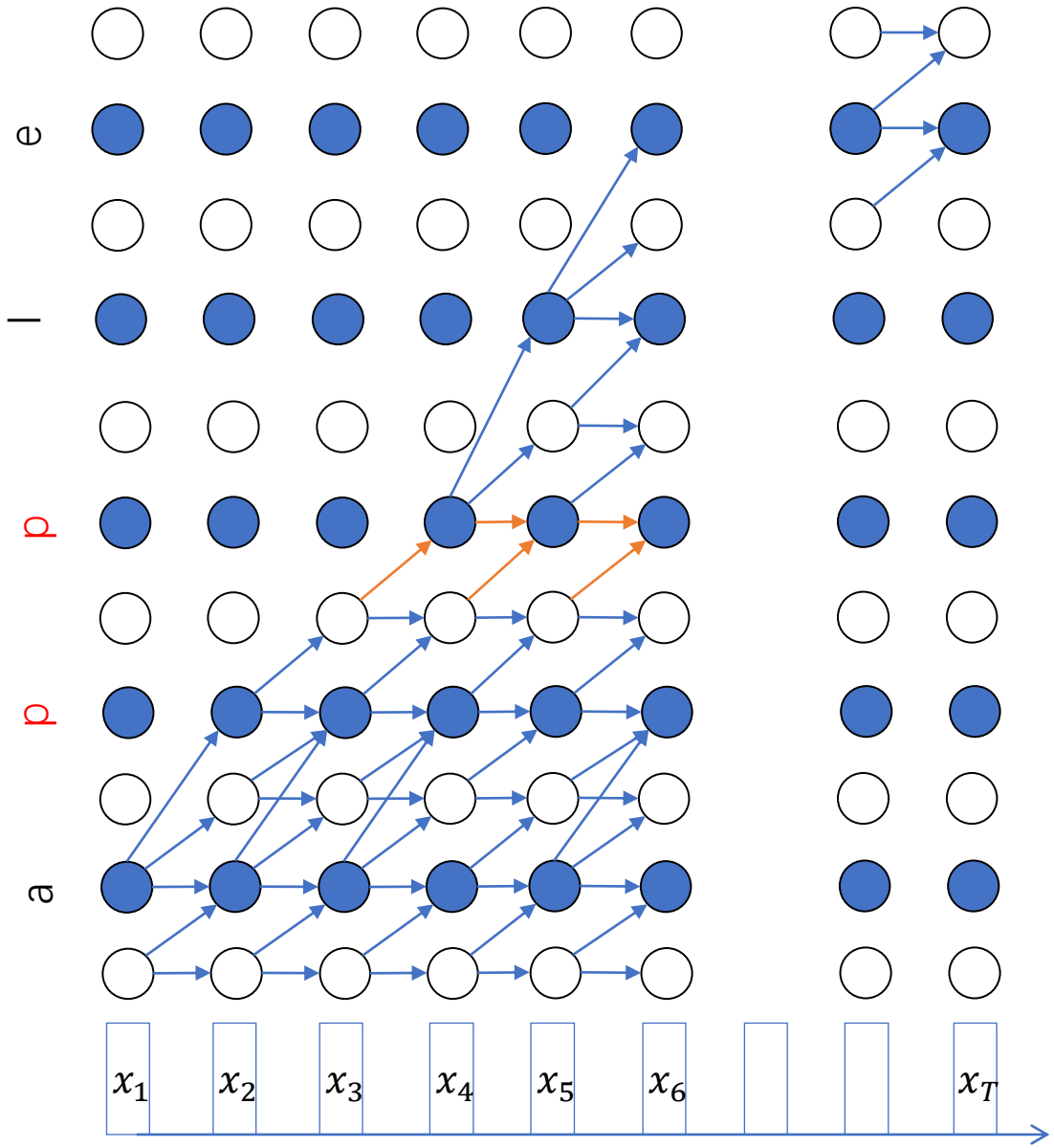
前向算法（t=3）



前向算法（t=T）



apple



后向算法

后向算法由后向前推算输出概率。用$\beta_t(s)$表示时刻t的标签为s，且输出观察值序列为$x_t x_{t+1}, \cdots, x_T$的概率：

$$\beta_t(s) = P(x_t, x_{t+1}, \cdots, x_T, s)$$

扩展输出序列$\boldsymbol{Y}' = \{b, y_1, b, y_2, b, \cdots, y_U, b\}$的长度为$2U+1$。

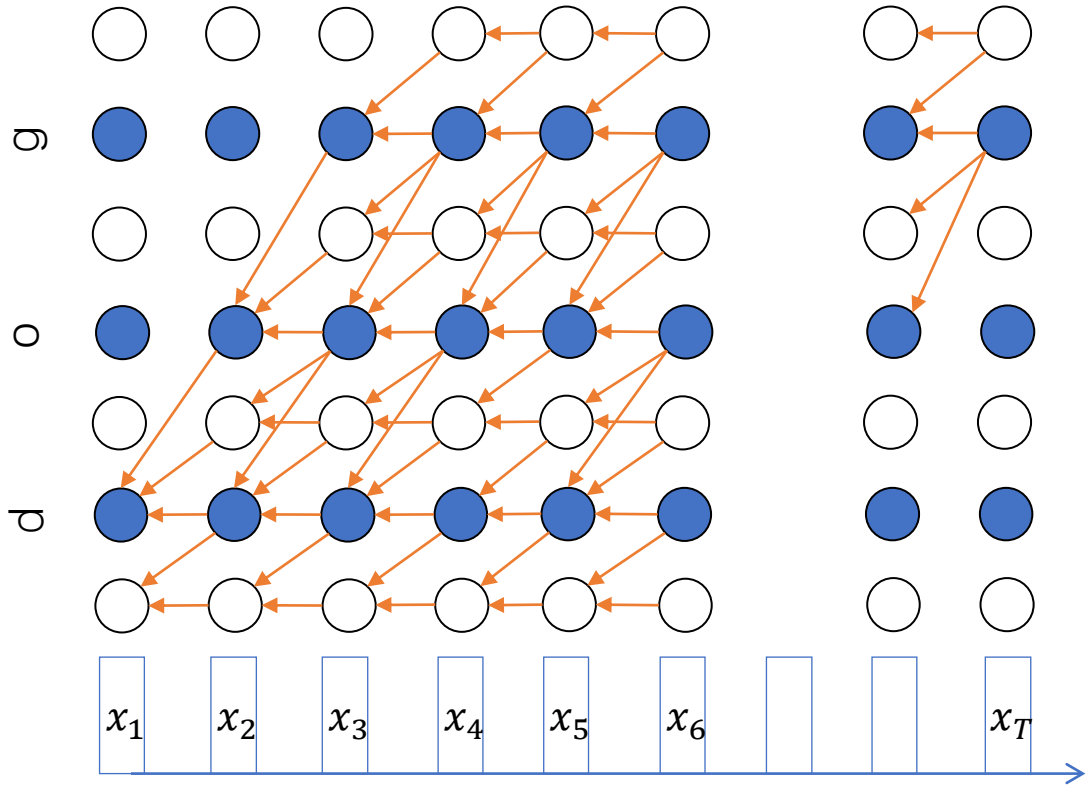
1. 初始化

$$\beta_T(2U+1) = P(y'_{2U+1}|x_T)$$
$$\beta_T(2U) = P(y'_{2U}|x_T)$$
$$\beta_T(s) = 0, \forall s < 2U$$

2. 迭代计算

$$\beta_t(s) = \begin{cases} \big(\beta_{t+1}(s) + \beta_{t+1}(s+1)\big)P(y'_s|x_t), if\ y'_s = b\ or\ y'_{s+2} = y'_s \\ \big(\beta_{t+1}(s) + \beta_{t+1}(s+1) + \beta_{t+1}(s+2)\big)P(y'_s|x_t), otherwise \end{cases}$$

后向算法（t=T）



求导过程

对于任意时刻t，利用前向概率和后向概率计算CTC的损失函数：

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \sum_{s=1}^{2U+1} \frac{\alpha_t(s)\beta_t(s)}{P(y_s'|x_t)}$$

$$L(\boldsymbol{S}) = -\sum_{(\boldsymbol{X},\boldsymbol{Y})\epsilon S} \ln P(\boldsymbol{Y}|\boldsymbol{X}) = -\sum_{(\boldsymbol{X},\boldsymbol{Y})\epsilon S} \ln\left(\sum_{s=1}^{2U+1} \frac{\alpha_t(s)\beta_t(s)}{P(y_s'|x_t)}\right)$$

求导过程

训练CTC网络的目标是使损失函数$L(\boldsymbol{S})$最小化。

令$P(k'|x_t) = p_t^{k'} = \frac{e^{h_t^{k'}}}{\sum_{k=1}^{K} e^{h_t^k}}$，对Encoder的输出$h_t^k$求偏导，其过程如下：

$$\frac{\partial L(\boldsymbol{S})}{\partial h_t^k} = -\sum_{(\boldsymbol{X},\boldsymbol{Y})\epsilon S} \frac{\partial \ln P(\boldsymbol{Y}|\boldsymbol{X})}{\partial h_t^k} = -\sum_{(\boldsymbol{X},\boldsymbol{Y})\epsilon S} \sum_{k'=1}^{K} \frac{\partial \ln P(\boldsymbol{Y}|\boldsymbol{X})}{\partial p_t^{k'}} \frac{\partial p_t^{k'}}{\partial h_t^k} \tag{11-18}$$

其中

$$\frac{\partial \ln P(\boldsymbol{Y}|\boldsymbol{X})}{\partial p_t^{k'}} = \frac{1}{P(\boldsymbol{Y}|\boldsymbol{X}) {p_t^{k'}}^2} \sum_{s\epsilon lab(\boldsymbol{Y}',k')} \alpha_t(s)\beta_t(s) \tag{11-19}$$

$s\epsilon lab(\boldsymbol{Y}',k')$表示$\boldsymbol{Y}'$序列中在$s$位置的输出为$k'$，即$y_s' = k'$。

$$\frac{\partial p_t^{k'}}{\partial h_t^k} = \frac{\partial \frac{e^{h_t^{k'}}}{\sum_{i=1}^{K} e^{h_t^i}}}{\partial h_t^k} = \begin{cases} p_t^{k'} - p_t^{k'} p_t^k, & if\ k' = k \\ -p_t^{k'} p_t^k, & if\ k' \neq k \end{cases} \tag{11-20}$$

把公式（11-19）和（11-20）代入（11-18），进一步整理得到：

$$\frac{\partial L(\boldsymbol{S})}{\partial h_t^k} = -\sum_{(\boldsymbol{X},\boldsymbol{Y})\epsilon S} \left(p_t^k - \frac{1}{P(\boldsymbol{Y}|\boldsymbol{X}) p_t^k} \sum_{s\epsilon lab(\boldsymbol{Y}',k)} \alpha_t(s)\beta_t(s) \right) \tag{11-21}$$

CTC训练过程

ALGORITHM 12.1: The CTC Training Algorithm

Step 1: 初始化

通过前向和后向算法分别计算$\alpha_t(s)$和$\beta_t(s)$;

Step 2: 求导

通过$\alpha_t(s)$和$\beta_t(s)$计算导数$\frac{\partial L(\boldsymbol{S})}{\partial h_t^k}$;

Step 3: 后向传播

通过反向传播对每层参数进行逐层优化。

Step 4: 迭代

重复2和3步骤，直到CTC损失代价收敛，即可完成端到端训练。

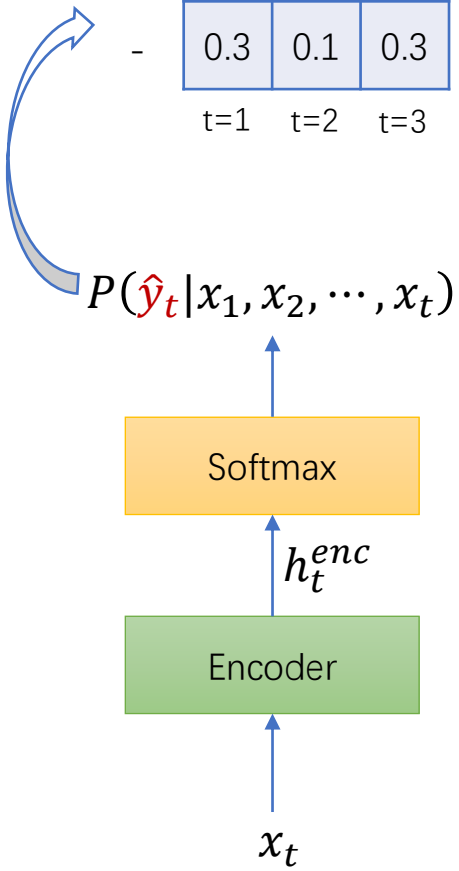
CTC解码

- 给定输入序列 $\boldsymbol{X}$， CTC的解码目标是找到概率最大的输出序列 $\boldsymbol{Y}^*$，即:

$$\boldsymbol{Y}^* = arg\max_{\boldsymbol{Y}}\{P(\boldsymbol{Y}|\boldsymbol{X})\}$$

CTC解码

	t=1	t=2	t=3
a	0.2	0.5	0.1
b	0.5	0.4	0.6
-	0.3	0.1	0.3



计算各种组合的概率：

$P(\boldsymbol{Y}=\text{blank}) = P(\text{---}) = 0.3*0.1*0.3 = 0.009$

$P(\boldsymbol{Y}=\text{a}) = P(\text{a--}) + P(\text{-a-}) + P(\text{--a}) + P(\text{aa-}) + P(\text{-aa}) + P(\text{aaa}) = 0.2*0.1*0.3 + 0.3*0.5*0.3 + 0.3*0.1*0.1 + 0.2*0.5*0.1 + 0.3*0.1*0.1 + 0.2*0.5*0.1 = 0.077$

$P(\boldsymbol{Y}=\text{b}) = P(\text{b--}) + P(\text{-b-}) + P(\text{--b}) + P(\text{bb-}) + P(\text{-bb}) + P(\text{bbb}) = 0.5*0.1*0.3 + 0.3*0.4*0.3 + 0.3*0.1*0.1 + 0.5*0.4*0.3 + 0.3*0.4*0.6 + 0.5*0.4*0.6 = 0.306$

$P(\boldsymbol{Y}=\text{aa}) = P(\text{a-a}) = 0.2*0.1*0.1 = 0.002$

$P(\boldsymbol{Y}=\text{ab}) = P(\text{ab-}) + P(\text{a-b}) + P(\text{-ab}) + P(\text{aab}) + P(\text{abb}) = 0.2*0.4*0.3 + 0.2*0.1*0.6 + 0.3*0.5*0.6 + 0.2*0.5*0.6 + 0.2*0.4*0.6 = 0.234$

$P(\boldsymbol{Y}=\text{ba}) = P(\text{ba-}) + P(\text{b-a}) + P(\text{-ba}) + P(\text{bba}) + P(\text{baa}) = 0.5*0.5*0.3 + 0.5*0.1*0.1 + 0.3*0.4*0.1 + 0.5*0.4*0.1 + 0.5*0.5*0.1 = 0.137$

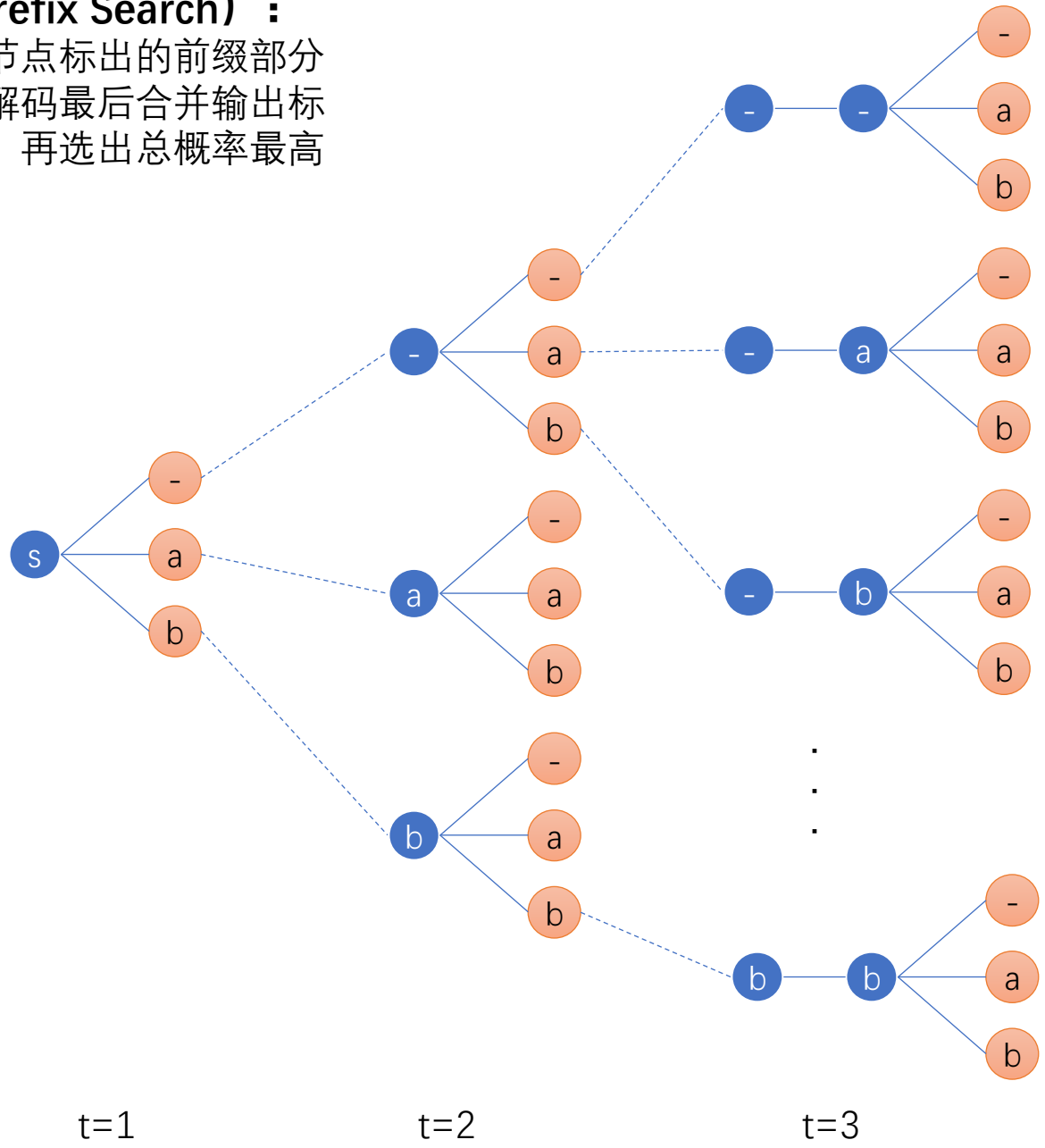
$P(\boldsymbol{Y}=\text{bb}) = P(\text{b-b}) = 0.5*0.1*0.6 = 0.03$

$P(\boldsymbol{Y}=\text{aba}) = 0.2*0.4*0.1 = 0.008$

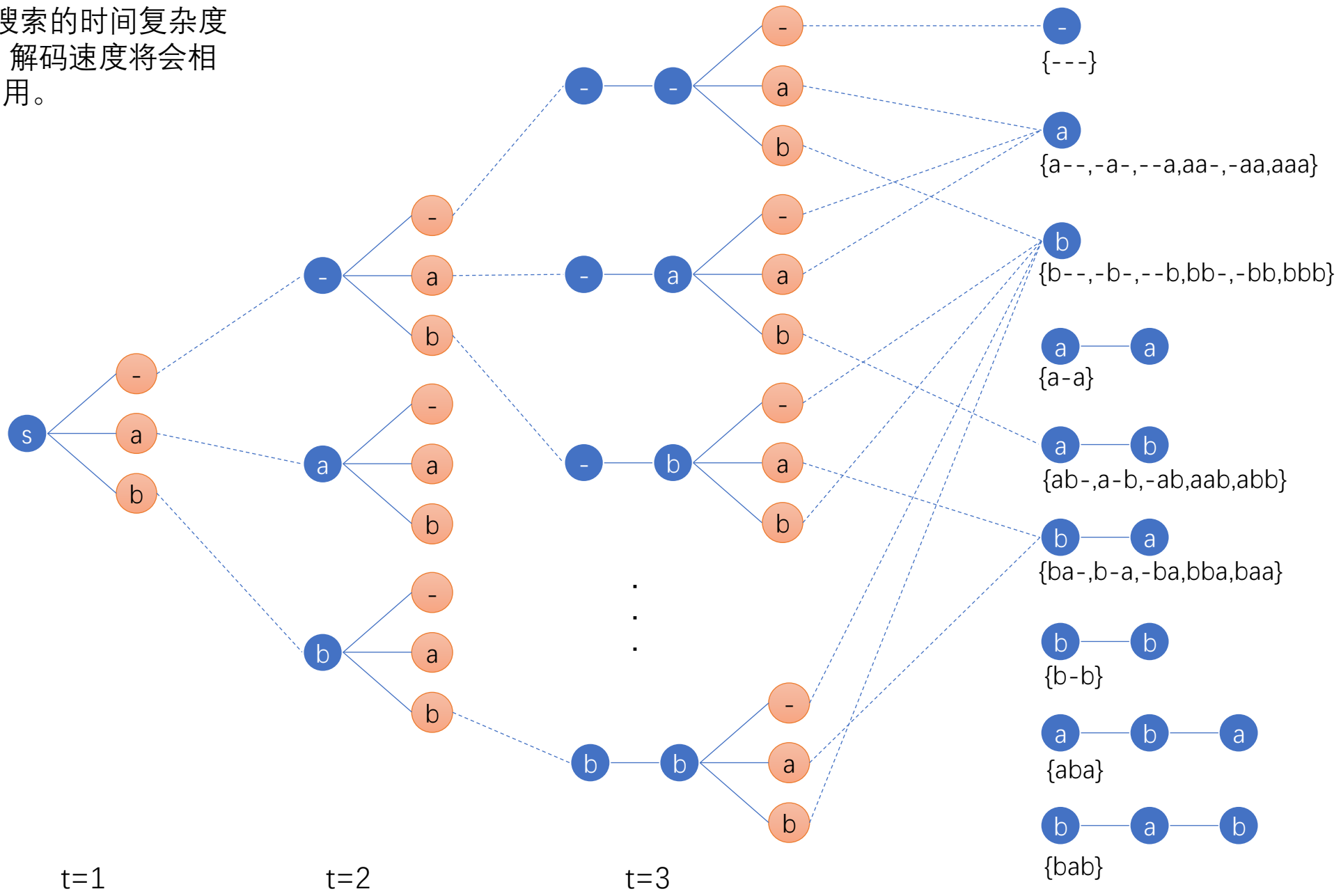
$P(\boldsymbol{Y}=\text{bab}) = 0.5*0.5*0.6 = 0.15$

总共有9种解码序列，其中输出概率最高的是$P(\text{Y=b})$，即最优解码序列$\boldsymbol{Y}^*$是{b}。

前缀搜索（Prefix Search）：
基于图中蓝色节点标出的前缀部分进一步扩展，解码最后合并输出标签一致的路径，再选出总概率最高的解码序列。



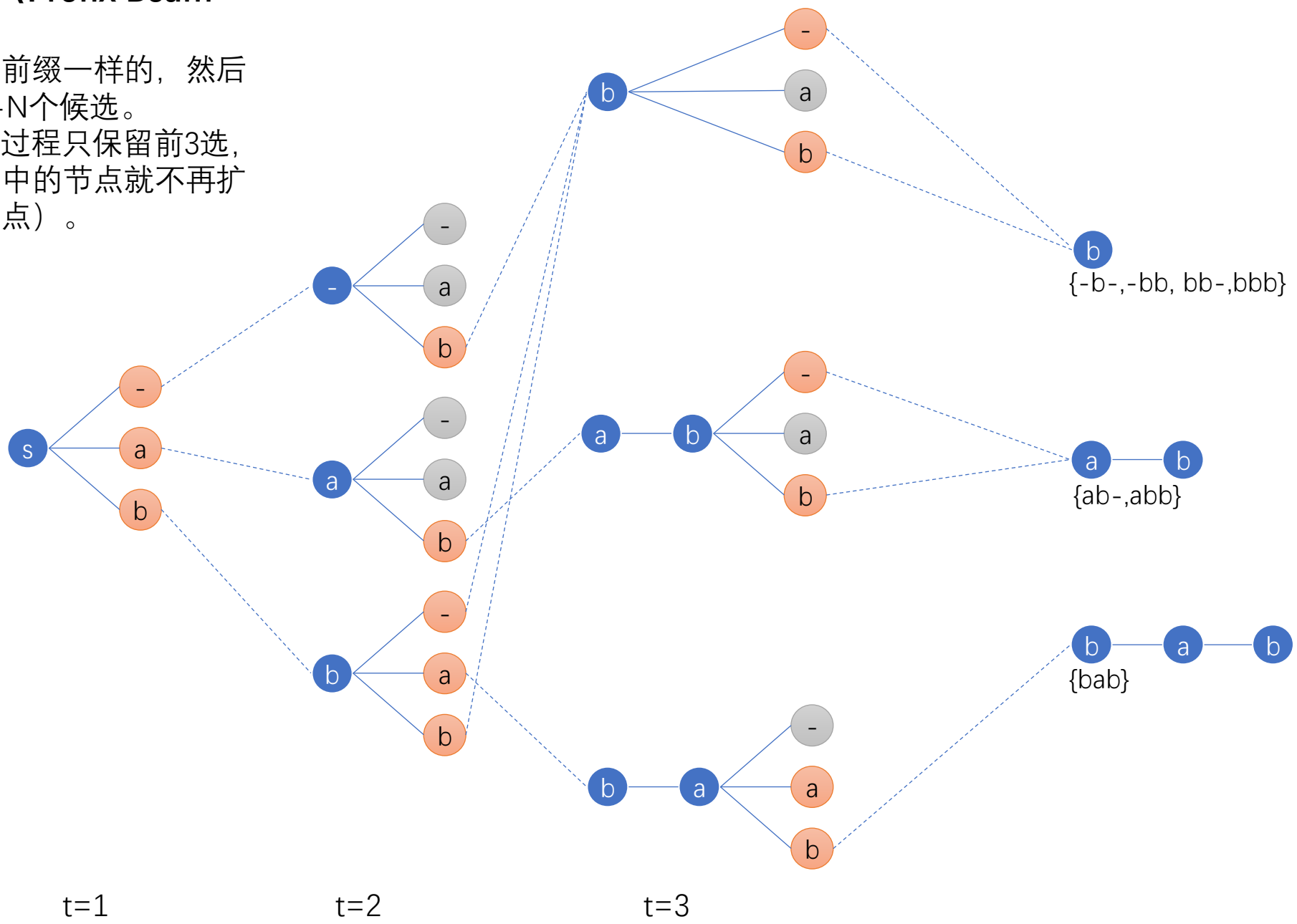
假如解码单元有K个，输入序列长度是T，则穷举搜索的时间复杂度为指数级的K^T，解码速度将会相当的慢，无法实用。



前缀剪枝搜索（Prefix Beam Search）：

每一时刻先合并前缀一样的，然后扩展只保留Top-N个候选。

如图所示的剪枝过程只保留前3选，每一时刻未被选中的节点就不再扩展（图中灰色节点）。



CTC集成语言模型

为提高准确率，可在解码过程中集成语言模型，求最优解修改如下：

$$\boldsymbol{Y}^* = arg\max_{\boldsymbol{Y}}\{P(\boldsymbol{Y}|\boldsymbol{X})P(\boldsymbol{Y})\}$$

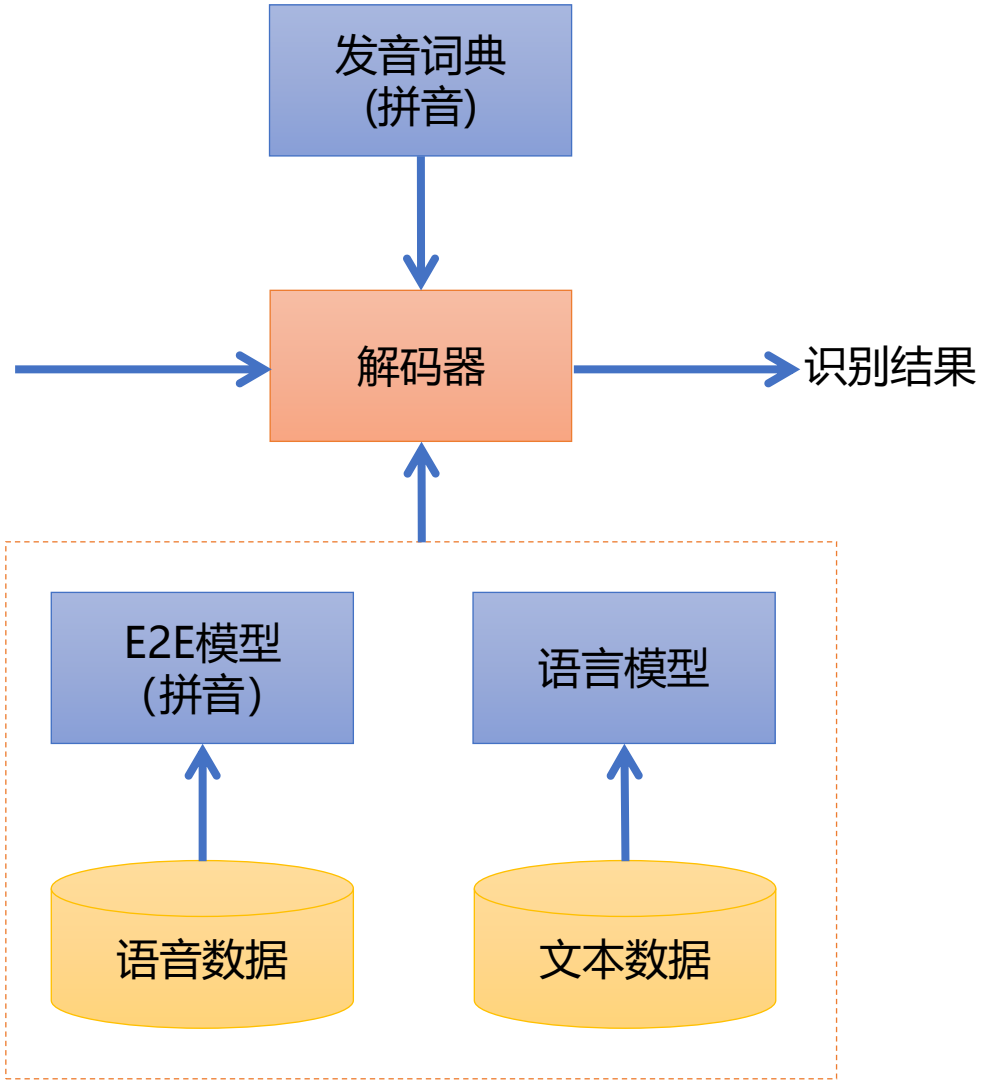
其中$P(\boldsymbol{Y})$代表语言模型。

CTC+WFST解码

- 分别编译子模块
  - 语言模型G.fst
  - 词典L.fst：音素或字符
  - 令牌T.fst：从CTC label到词典单元

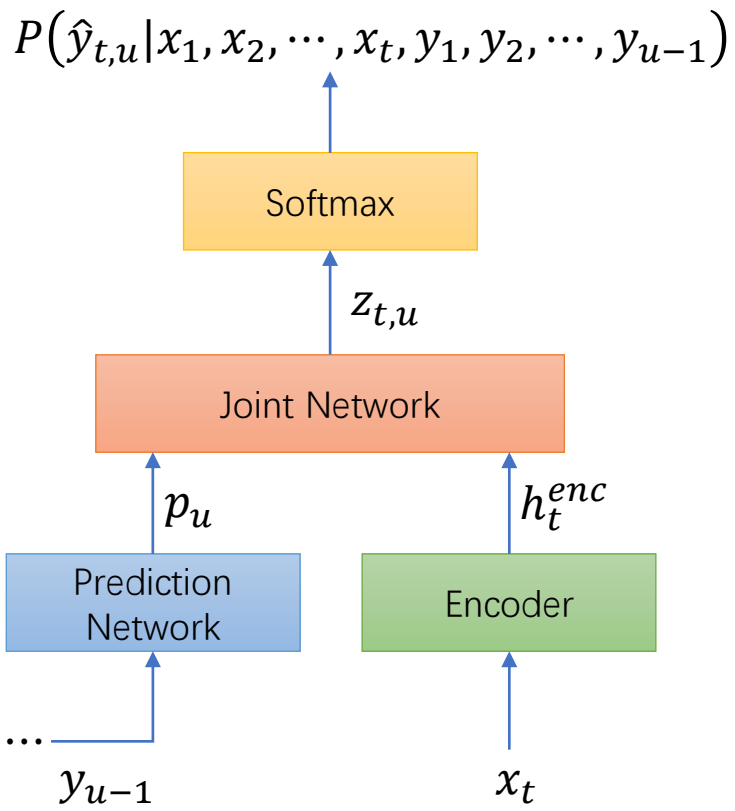
- 合并后得到搜索图

$$S = T \circ \min(\det(L \circ G))$$



RNN-T

- 为了联合优化声学模型与语言模型，Graves等人在2012年提出了RNN Transducer（RNN-T）。RNN-T能更好地对输出结果前后词之间的依赖关系进行建模。



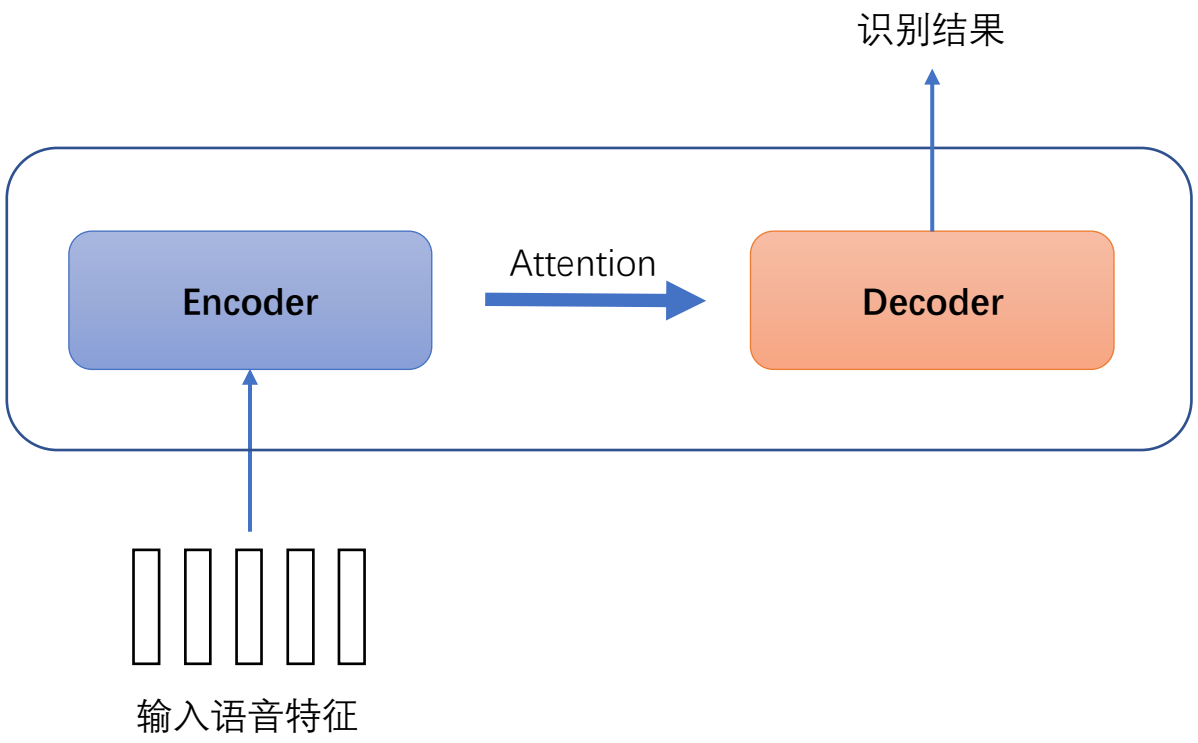
- Encoder：把输入特征序列$\boldsymbol{X} = \{x_1, x_2, \cdots, x_T\}$转为隐藏向量序列$\boldsymbol{h}^{enc} = \{h_1^{enc}, h_2^{enc}, \cdots, h_T^{enc}\}$，此部分相当于声学模型。
- Prediction Network：RNN把上一个输出标签y_{u-1}做为输入，并预测下一个标签y_u，此部分相当于语言模型。
- Joint Network：输入$\boldsymbol{h}^{enc}$和y_u，输出联合隐藏向量$\boldsymbol{Z} = \{z_1, z_2, \cdots, z_T\}$。

RNN-T

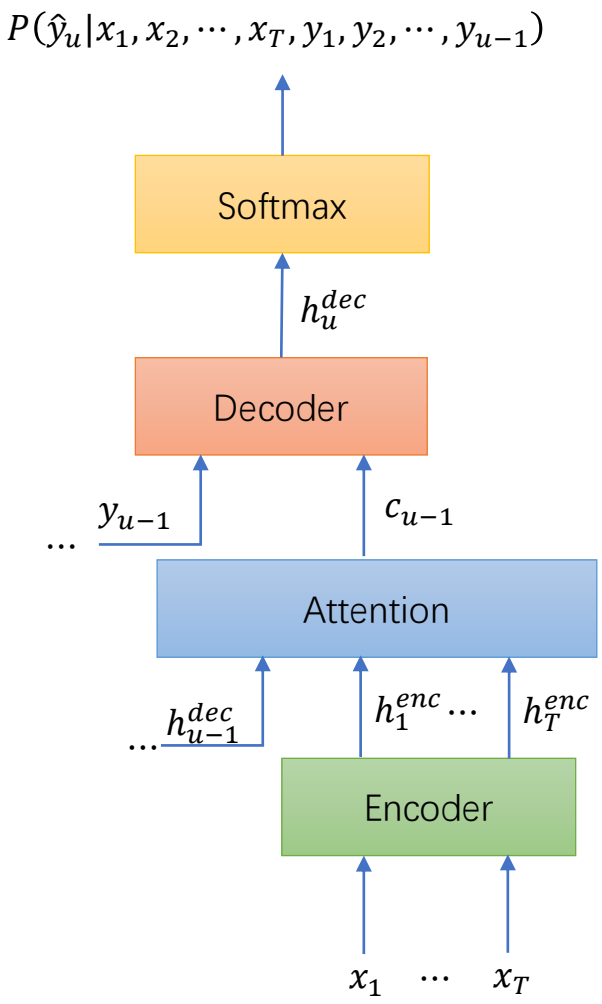
- RNN-T对声学模型和语言模型分别建模，同时又通过Joint Network联合优化，其损失函数跟CTC一致。因此，如果有足够的训练数据，RNN-T就不需要另外语言模型了，真正实现端到端的建模。

- 由于RNN-T是针对每帧输入特征进行预测输出，即不用等语音全部说完再出结果，因此可应用于流识别，特别是在嵌入式设备。

Attention模型

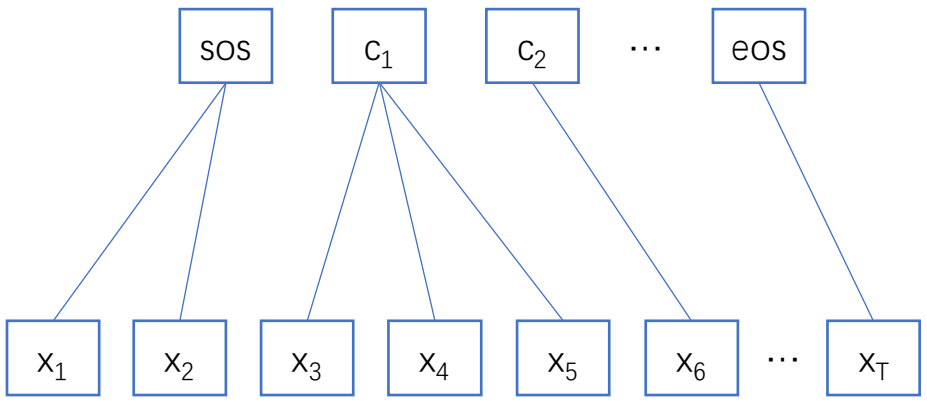


Attention-based Encoder-Decoder

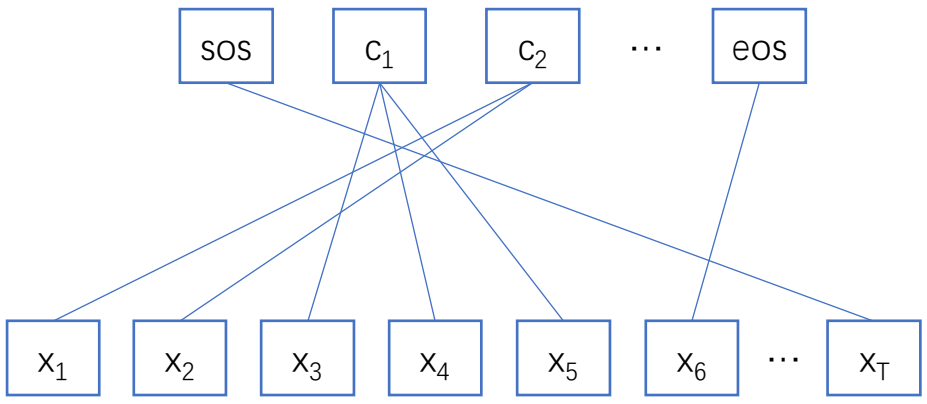


- Encoder：通过循环神经网络把输入特征序列$\boldsymbol{X} = \{x_1, x_2, \cdots, x_T\}$转为隐藏向量序列$\boldsymbol{h}^{enc} = \{h_1^{enc}, h_2^{enc}, \cdots, h_T^{enc}\}$。
- Decoder：计算基于之前预测标签和输入特征序列的概率分布。
- Attention：从Encoder输出所有帧计算注意力权重（可理解为重要性），并基于此权重构建Decoder网络的上下文向量。

Hybrid CTC/Attention



CTC对齐

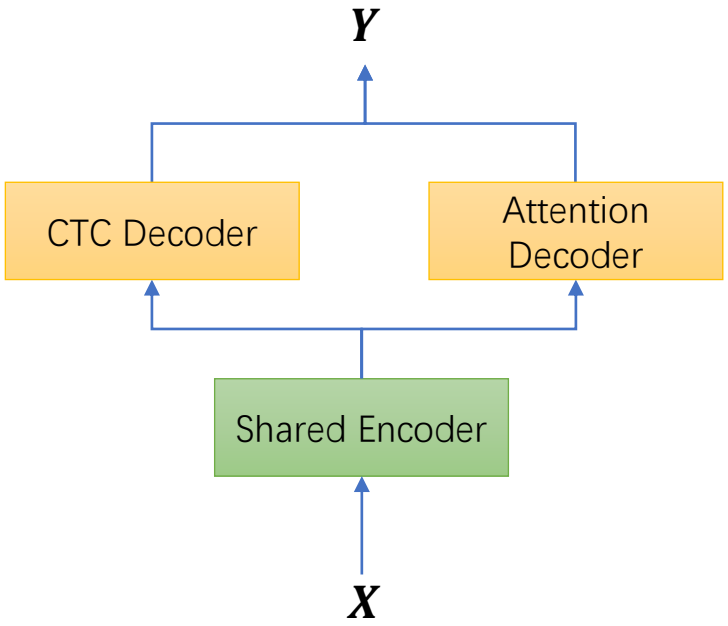


Attention对齐

CTC的前向后向算法可以引导输出序列与输入序列按时间顺序对齐，而Attention模型的对齐关系没有先后顺序的限制，完全靠数据驱动得到，难以训练。因此CTC和Attention模型各有优势，可把两者结合起来，构建Hybrid CTC/Attention模型，采用多任务学习，通过CTC避免对齐关系过于随机，以加快训练流程。

Hybrid CTC/Attention

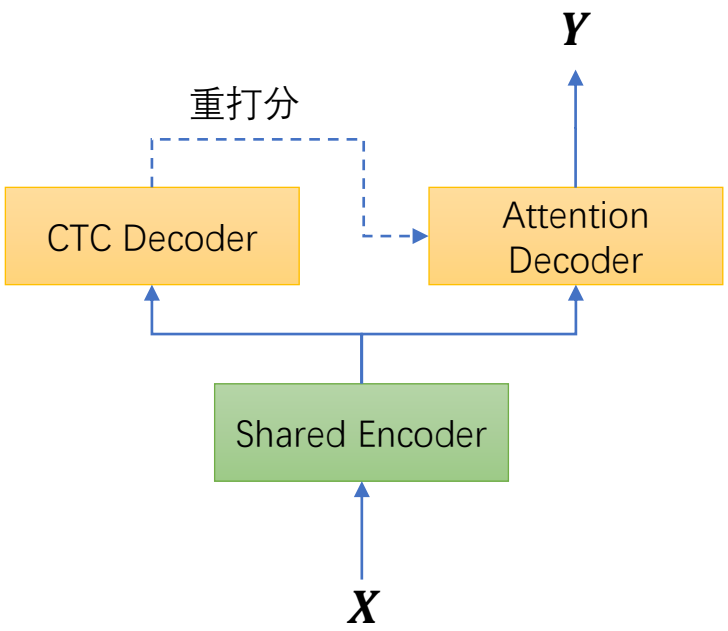
- 混合CTC/Attention模型的Loss计算是CTC-Loss与Attention-Loss做加权相加，其中Encoder部分CTC和Attention共用。



- 训练过程是多任务学习，将CTC与基于Attention机制的交叉熵$L(CTC)$和$L(Att)$相结合，如下式所示：

$$L = \lambda L(CTC) + (1 - \lambda)L(Att)$$

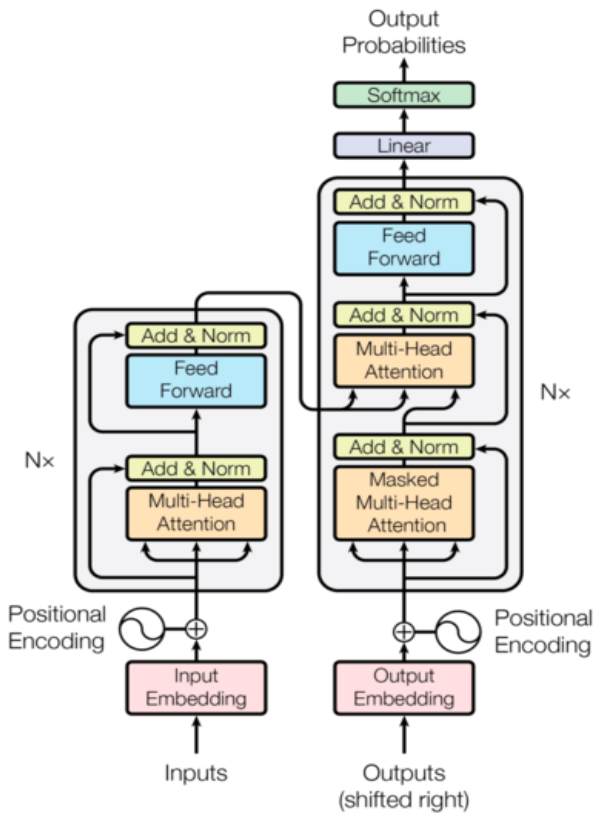
Two-pass解码

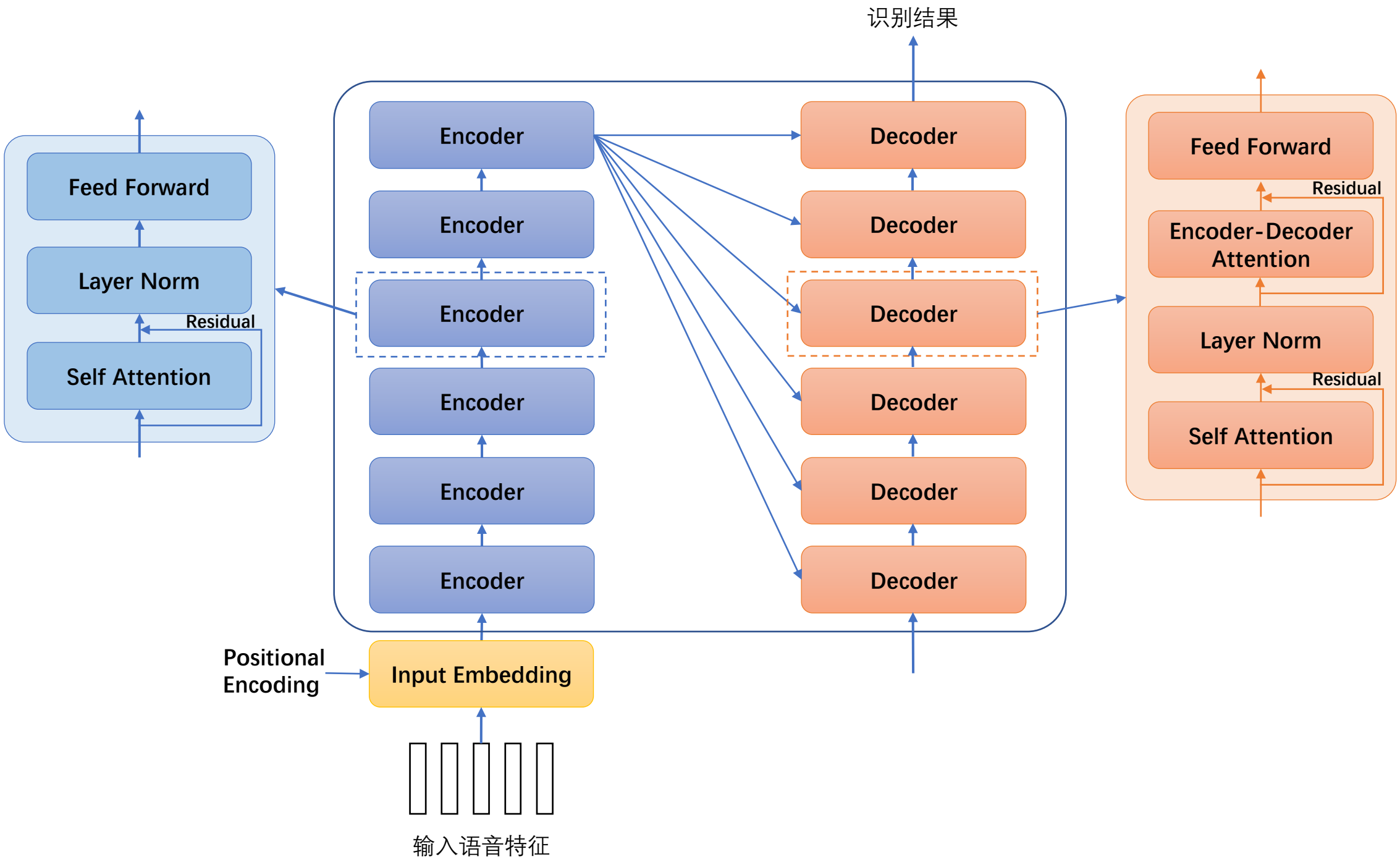


- First pass：采用CTC解码，并产生N-best结果；
- Second pass：利用Attention解码对N-best结果重打分。

Transformer

- 2017年，Google和多伦多大学在论文“Attention Is All you Need”中提出一种称为Transformer的全新架构，
- 这种架构在Decoder和Encoder均采用Attention机制，特别是在Encoder层，把传统的RNN完全用Attention替代，在机器翻译任务取得更优的结果，引起极大关注。




识别结果
Feed Forward
Layer Norm
Residual
Self Attention
Encoder
Encoder
Encoder
Encoder
Encoder
Encoder
Decoder
Decoder
Decoder
Decoder
Decoder
Decoder
Feed Forward
Residual
Encoder-Decoder Attention
Layer Norm
Residual
Self Attention
Positional Encoding
Input Embedding
输入语音特征

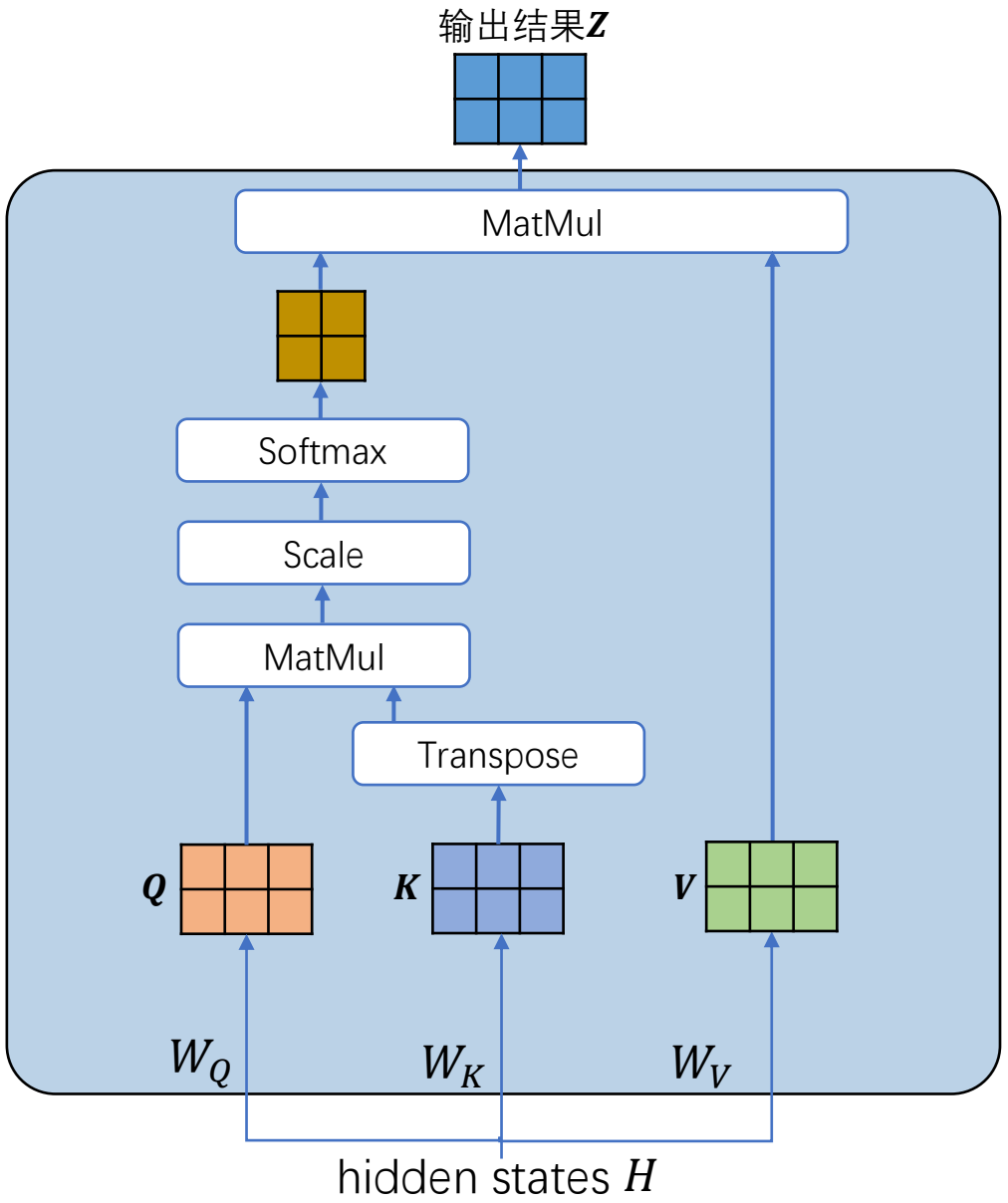
Transformer

- Encoder的每一层有三个操作，分别是Self-Attention、LayerNorm和FeedForward；

- Decoder的每一层有四个操作，分别是Self-Attention、LayerNorm、Encoder-Decoder Attention以及FeedForward操作。其中Attention层都有Residual残差连接，前一层的信息可直接传递到下一层。LayerNorm通过对层的激活值的归一化，可加速模型的训练过程，使其更快的收敛。

- Positional Encoding用来学习位置信息，并叠加到输入Embedding中。

Transformer

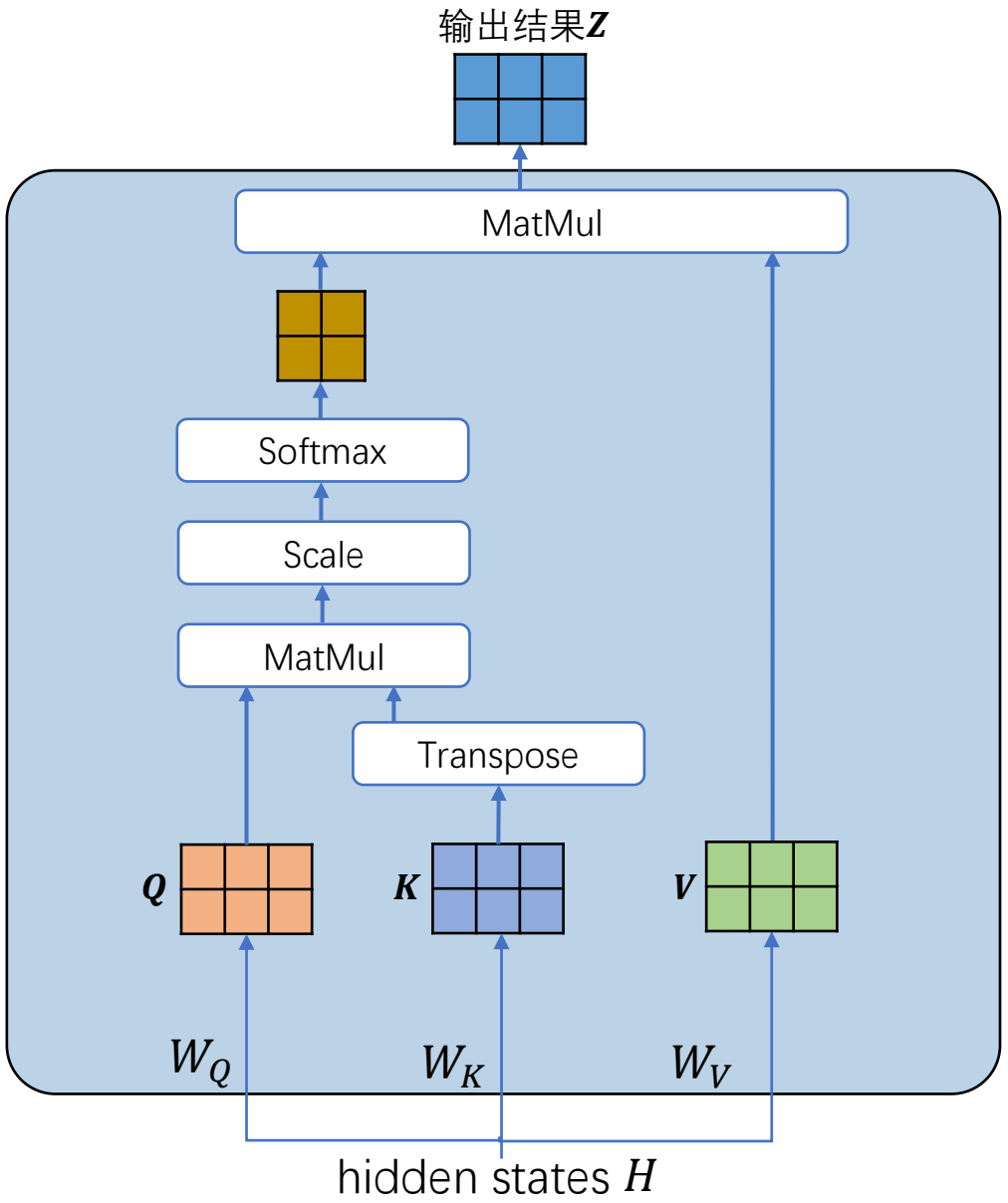


Transformer的核心模块是Self-Attention。

- 输入的H（来自于上一层的输出，最原始的输入来自于语音特征$\boldsymbol{X}$）与代表Query、Key、Value的三个矩阵W_Q、W_K、W_V相乘，分别得到$\boldsymbol{Q}$、$\boldsymbol{K}$和$\boldsymbol{V}$。图中$\boldsymbol{Q}$、$\boldsymbol{K}$和$\boldsymbol{V}$都是2*3矩阵。

- 在网络中，首先对矩阵$\boldsymbol{K}$通过Transpose操作进行转置，然后通过MatMul操作与矩阵$\boldsymbol{Q}$相乘，即$\boldsymbol{Q}\boldsymbol{K}^T$，注意$\boldsymbol{Q}$的列数要与$\boldsymbol{K}$的列数相等。$\boldsymbol{Q}\boldsymbol{K}^T$相乘结果通过Scale操作进行必要的缩放，一般除以矩阵$\boldsymbol{K}$的列数开平方，以避免值过大，导致Softmax函数梯度很小难以优化。

Transformer



经过Softmax层输出0~1分布的概率矩阵。这个概率矩阵再与矩阵$\boldsymbol{V}$相乘，得到最后的输出结果$\boldsymbol{Z}$（图中所示2*3矩阵），完整的计算公式如下：

$$\boldsymbol{Z} = Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}$$

其中d_k是矩阵$\boldsymbol{K}$的列数。

输出结果$\boldsymbol{Z}$的每一行Z_i代表一个位置的结果，这个位置对应输入语音特征序列$\boldsymbol{X}$的某一帧X_i，但这个位置输出结果还包含了其它帧X_j的信息，其计算过程如下：

$$Z_i = \sum_j softmax\left(\frac{Q_i \cdot K_j}{\sqrt{d_k}}\right)V_j$$

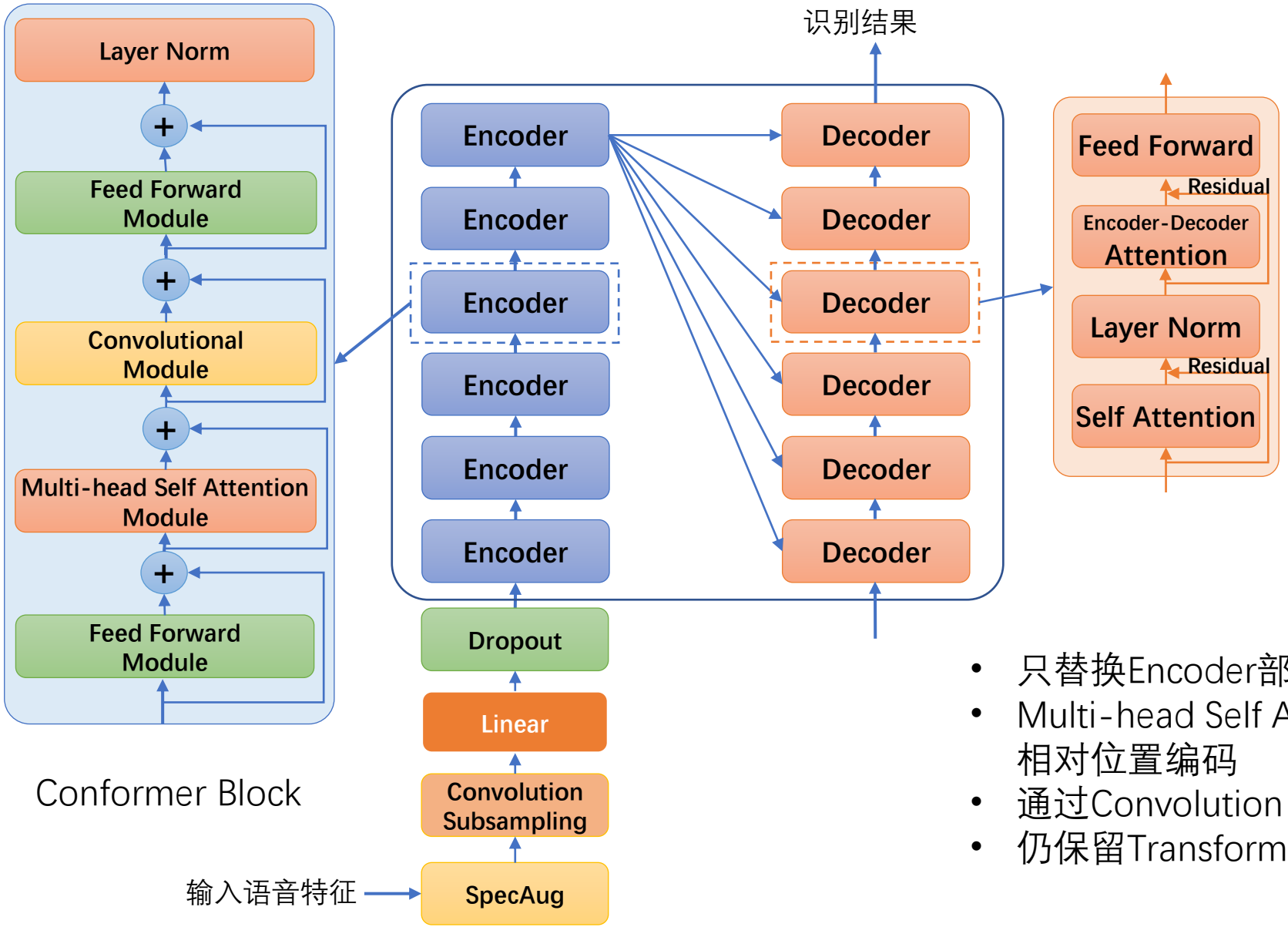
因此当前节点输出结果包含整个句子上下文信息，即不仅关注当前的帧，也能获取前后其它帧的信息，这些信息的重要性通过Attention来调节。

Transformer

- Transformer还可通过Multihead Attention机制，得到不同组的$\boldsymbol{Q}$、$\boldsymbol{K}$和$\boldsymbol{V}$表示，相当于不同空间的特征，最后再将结果结合起来。

- Transformer用Attention代替需要上下文递归处理的RNN，使得所有的计算都可以并发进行，而不像RNN那样需要依赖前一时刻的输出来进行计算，因而大大加快了运行速度。

Conformer



- 只替换Encoder部分
- Multi-head Self Attention Module使用相对位置编码
- 通过Convolution Module捕捉局部特征
- 仍保留Transformer的全局刻画能力

本章小结

- 端到端模型的创新和便捷之处在于它可以轻松地将之前传统系统的很多纷繁的步骤融合进一个模型里，而不需要再额外准备各种复杂的音素词典和声学/语言模型，这是语音行业的重大技术革新。

- 本章详细介绍了三种典型的E2E模型，包括CTC、RNN-T和AED，并具体介绍了Transformer和Conformer网络结构，其中Conformer效果较优。

- 为了提高E2E的性能，需要耗费更多的标注语料、计算资源和时间代价来对神经网络进行训练，如何调整模型结构、改进算法，成了E2E语音识别领域的研究热点。