

Lecture 05: 高阶语音识别

端到端模型

王帅

南京大学智能科学与技术学院

2026 年 10 月 9 日

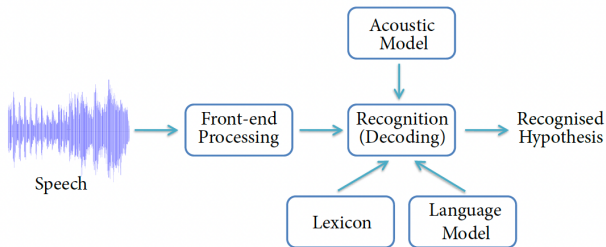


南京大學
NANJING UNIVERSITY



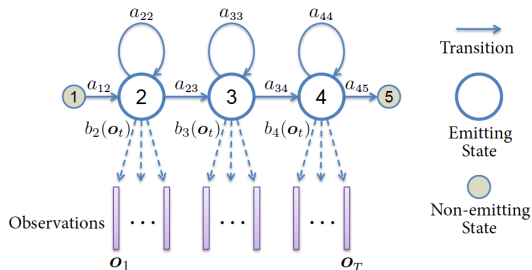
智能科学与技术学院
School of Intelligence Science and Technology

- 1 经典 ASR 系统与端到端建模的动机
- 2 端到端系统简介
- 3 CTC
- 4 AED
- 5 RNN-T



- **前端**：将波形转换为声学特征序列。
- **声学模型**：对特征与音素状态的对应关系打分。
- **词典与语言模型**：连接音素和词，并提供上下文偏好。
- **解码器**：结合各模块分数，搜索候选文字。

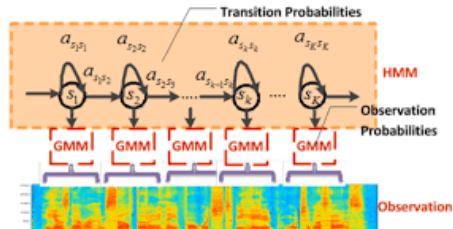
HMM: 声音与状态的时间对齐



- **观测**: 每一帧的声学特征。
- **隐藏状态**: 音素的不同发音阶段。
- **自环**: 在同一状态停留，容纳不同语速。
- **向前转移**: 按顺序进入后续状态。

核心思想

HMM 描述状态的先后和持续时间，GMM 或 DNN 提供声学分数。下一状态依赖当前状态，这就是一阶马尔可夫假设。



- 每个 HMM 状态对应一个声学特征分布。
- GMM 用多个高斯分量的加权和描述这个分布。
- 给定状态 s_j , GMM 计算当前特征的密度 $p(\mathbf{x}_t | s_j)$ 。
- HMM 结合声学分数与转移概率, 对整条状态路径打分。

分工

GMM 回答“这一帧与这个状态是否匹配”, HMM 回答“这些状态能否组成合理的时间序列”。

训练：估计模型参数

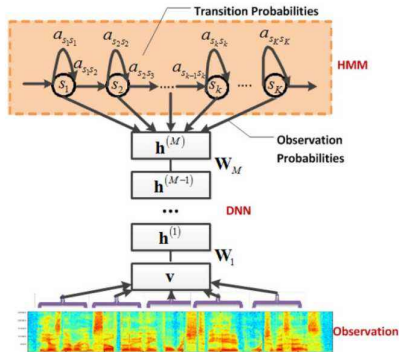
- 输入语音及对应文本，词典提供发音序列。
- 每一帧属于哪个状态通常未知。
- Baum-Welch / EM 交替估计状态占用与更新模型参数。

识别：搜索未知文字

- 测试时只有语音，需要比较候选文字及状态路径。
- Viterbi / 束搜索结合声学、词典和语言模型分数。
- 输出得分最高的候选结果。

对齐的来源

模型在训练中估计隐藏状态，通常不需要人工逐帧标注音素的时间边界。



- 用 DNN 替代 GMM，HMM 仍负责状态转移与时间对齐。
- DNN 输入当前帧及其上下文，输出状态后验 $P(s_j | \mathbf{x}_t)$ 。
- 对后验进行状态先验校正，得到供 HMM 解码使用的声学分数。
- 词典、语言模型和搜索解码器继续参与识别。

改变的是声学建模方式

神经网络可以学习更复杂的判别边界，并利用相邻帧信息。

- 1 训练 GMM-HMM: 得到初始声学模型。
- 2 生成强制对齐: 在已知文本对应的状态路径上, 为每帧估计 HMM 状态标签。
- 3 训练 DNN: 以声学特征为输入、状态标签为监督, 进行帧级分类训练。
- 4 组装混合系统: 使用 DNN 声学分数, 结合 HMM、词典和语言模型解码。

经典流程中的两种训练目标

GMM-HMM 训练提高观测似然, DNN 的帧级交叉熵训练提高状态分类能力。最终仍以文字序列的识别错误率评价整个系统。

强制对齐

已知信息
搜索范围

语音与正确文本
给定文本对应的发音和状态路
径

输出

帧级状态标签或时间边界

语音识别

语音
候选文本及对应状态路径

识别文字

为什么 GMM 标签可以用于训练 DNN

已知文本限制了对齐的搜索范围，GMM-HMM 可提供有用的帧级监督。DNN 用不同的模型结构和上下文特征学习这些标签，因此有机会改善声学建模。

对齐仍可能有误差，标签质量会影响 DNN 训练。

	GMM-HMM	DNN-HMM
声学模型	GMM 估计特征的条件密度	DNN 估计状态后验，再转换分数
时间结构	HMM 状态转移	HMM 状态转移
识别组件	发音词典、语言模型、解码器	发音词典、语言模型、解码器

- 三音素与状态绑定用于描述上下文并共享参数，涉及额外的模型设计。
- 声学建模、对齐、词典和解码相互关联，需要分阶段构建和调试。
- 各模块的局部目标与最终文字序列的识别错误率可能不一致。

从语音与文本配对数据直接训练

用神经序列模型估计 $P(\mathbf{y} | \mathbf{x})$ ，将声学建模与文字预测纳入统一的训练目标。

- 输出单元通常是字、子词或词，可以减少对独立发音词典和音素状态体系的依赖。
- 对齐问题仍然存在：语音帧数通常远多于文字单元数。
- 本讲关注 CTC、AED 与 RNN-T 如何处理对齐和输出依赖。

课后补充阅读

马尔可夫链、HMM 算法推导、GMM 参数更新和状态绑定的详细介绍，见《ASR 补充材料：马尔可夫链、HMM 与 GMM》。

这条“流水线”虽然有效，但存在诸多内在缺陷：

流程与优化问题

- **流程复杂**，维护成本高
- 优化目标**不一致**，导致次优解
- **误差累积**，逐级放大

依赖与门槛问题

- 高度依赖**专家知识**（发音词典）
- 训练过程**繁琐**（需要对齐）
- 难以处理**未登录词** (OOV)

核心矛盾

每个模块都追求局部最优，但其组合并非全局最优。

用一个**单一、统一**的神经网络，替代整个传统流水线。

核心思想：直接将**语音序列**映射到**文本序列**。

传统痛点

- 流程复杂，多模块
- 优化目标不一致
- 依赖专家发音词典
- 误差逐级累积
- 开发迭代缓慢

E2E 解决方案

- 架构极大简化
- 统一联合优化
- 自动学习发音
- 端到端联合训练
- 提升开发效率

对于一段语音和其对应的文本，我们面临一个核心难题：**对齐**。

- **输入**: 语音特征序列 $\mathbf{x} = (x_1, x_2, \dots, x_T)$ ，长度为 T 。
- **输出**: 目标文本序列 $\mathbf{y} = (y_1, y_2, \dots, y_U)$ ，长度为 U 。

核心挑战

- **长度不匹配**: 通常输入长度 T 远大于输出长度 U ($T \gg U$)。
- **边界不明确**: 我们不知道哪个音素或字符精确地对应到哪一帧音频。
- **预对齐成本高**: 人工为每一帧音频打上标签（例如音素）是极其昂贵且耗时的工作。

我们需要一种无需预对齐的端到端训练方法！

Connectionist Temporal Classification (CTC) 的核心创新是引入了一个特殊的空白标签“blank” (ϵ)。

Blank 标签的作用

- **填充输出**：模型在每个时间步 t 都必须输出一个标签。当不确定输出哪个字符时，就输出 ϵ 。

Connectionist Temporal Classification (CTC) 的核心创新是引入了一个特殊的空白标签“blank” (ϵ)。

Blank 标签的作用

- **填充输出**：模型在每个时间步 t 都必须输出一个标签。当不确定输出哪个字符时，就输出 ϵ 。
- **分隔重复字符**：如何区分“helo”和“hello”？CTC 规定，真正的重复字符之间必须由一个或多个 ϵ 分隔。

Connectionist Temporal Classification (CTC) 的核心创新是引入了一个特殊的空白标签“blank” (ϵ)。

Blank 标签的作用

- **填充输出**：模型在每个时间步 t 都必须输出一个标签。当不确定输出哪个字符时，就输出 ϵ 。
- **分隔重复字符**：如何区分“helo”和“hello”？CTC 规定，真正的重复字符之间必须由一个或多个 ϵ 分隔。

示例

- ‘h-e-l-o’ $\rightarrow$ “helo”
- ‘h-e-l- ϵ -l-o’ $\rightarrow$ “hello”

Connectionist Temporal Classification (CTC) 的核心创新是引入了一个特殊的**空白标签**“blank” (ϵ)。

Blank 标签的作用

- **填充输出**：模型在每个时间步 t 都必须输出一个标签。当不确定输出哪个字符时，就输出 ϵ 。
- **分隔重复字符**：如何区分“helo”和“hello”？CTC 规定，真正的重复字符之间必须由一个或多个 ϵ 分隔。

示例

- ‘h-e-l-o’ $\rightarrow$ “helo”
- ‘h-e-l- ϵ -l-o’ $\rightarrow$ “hello”

通过引入 ϵ ，CTC 将模糊的“对齐”问题，巧妙地转化为一个清晰的“路径解码”问题。

CTC 工作原理：一个直观的例子

模型在每个时间步 t 都会预测一个标签，形成一条长度为 T 的**中间路径** π 。

示例：从路径 π 到最终文本 y

时间帧	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
预测路径 π	h	h	e	l	ϵ	l	o	o	ϵ

CTC 工作原理：一个直观的例子

模型在每个时间步 t 都会预测一个标签，形成一条长度为 T 的**中间路径** π 。

示例：从路径 π 到最终文本 y

时间帧	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
预测路径 π	h	h	e	l	€	l	o	o	€

CTC 的“折叠”规则 $\mathcal{B}(\pi)$

- ① 合并连续重复的标签：‘h,h,e,l,€,l,o,o,€’ $\rightarrow$ ‘h,e,l,€,l,o,€’
- ② 移除所有 blank (€) 标签：‘h,e,l,€,l,o,€’ $\rightarrow$ ‘h,e,l,l,o’

CTC 工作原理：一个直观的例子

模型在每个时间步 t 都会预测一个标签，形成一条长度为 T 的**中间路径** π 。

示例：从路径 π 到最终文本 y

时间帧	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
预测路径 π	h	h	e	l	ε	l	o	o	ε

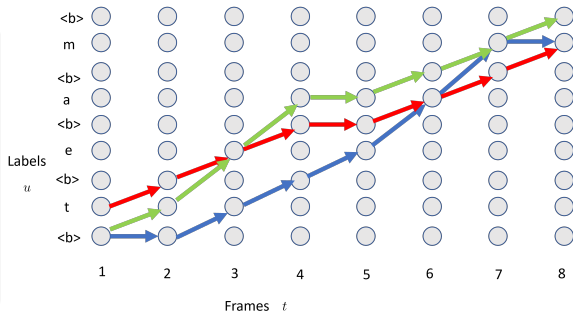
CTC 的“折叠”规则 $\mathcal{B}(\pi)$

- ① 合并连续重复的标签：‘h,h,e,l,ε,l,o,o,ε’ → ‘h,e,l,ε,l,o,ε’
- ② 移除所有 blank (ε) 标签：‘h,e,l,ε,l,o,ε’ → ‘h,e,l,l,o’

✓ 最终输出正确：“hello”

路径概念

- **CTC 路径**: 长度等于输入序列的标签序列
- **有效路径**: 可映射到目标标签序列的路径
- **空白标签**: 表示“无输出”或“继续”
- **重复标签**: 允许连续相同标签



路径规则

- 空白标签可以出现在任意位置
- 非空白标签可以重复
- 路径长度必须等于输入序列长度
- 去除空白和重复后得到目标序列

我们将直观的理解转化为数学语言。

- 扩展词汇表 $\mathcal{L}'$: 在原始字符集 $\mathcal{L}$ (如 a-z) 的基础上加入 blank 标签。

$$\mathcal{L}' = \mathcal{L} \cup \{\epsilon\}$$

我们将直观的理解转化为数学语言。

- **扩展词汇表** $\mathcal{L}'$: 在原始字符集 $\mathcal{L}$ (如 a-z) 的基础上加入 blank 标签。

$$\mathcal{L}' = \mathcal{L} \cup \{\epsilon\}$$

- **中间路径** π : 一个长度为 T 的序列, 其中每个元素 $\pi_t \in \mathcal{L}'$ 。

我们将直观的理解转化为数学语言。

- **扩展词汇表** $\mathcal{L}'$: 在原始字符集 $\mathcal{L}$ (如 a-z) 的基础上加入 blank 标签。

$$\mathcal{L}' = \mathcal{L} \cup \{\epsilon\}$$

- **中间路径** π : 一个长度为 T 的序列, 其中每个元素 $\pi_t \in \mathcal{L}'$ 。

- **多对一映射** $\mathcal{B}$: 存在大量不同的路径 π , 它们通过“折叠”规则 $\mathcal{B}$ 映射到最终文本 y 。

示例: 哪些路径可以生成“hello”?

假设目标文本 $y = (h, e, l, l, o)$ 。以下路径都是有效的:

- 路径 1: $(h, e, l, \epsilon, l, o)$ (最简洁的形式)
- 路径 2: $(h, h, e, l, \epsilon, l, l, o)$ (包含字符重复)
- 路径 3: $(\epsilon, h, e, \epsilon, l, \epsilon, l, o, \epsilon)$ (包含大量 blank)
- 路径 4: $(h, e, e, l, l, \epsilon, \epsilon, l, o, o)$ (更复杂的形式)

关键点: 两个 ‘l’ 之间必须至少有一个 ϵ 来分隔。

我们将直观的理解转化为数学语言。

- **扩展词汇表** $\mathcal{L}'$: 在原始字符集 $\mathcal{L}$ (如 a-z) 的基础上加入 blank 标签。

$$\mathcal{L}' = \mathcal{L} \cup \{\epsilon\}$$

- **中间路径** π : 一个长度为 T 的序列, 其中每个元素 $\pi_t \in \mathcal{L}'$ 。
- **多对一映射** $\mathcal{B}$: 存在大量不同的路径 π , 它们通过“折叠”规则 $\mathcal{B}$ 映射到最终文本 y 。

训练目标

训练模型 (如 RNN), 使其能够为正确的文本 y 对应的所有可能路径 π 分配高概率。

Step 1: 计算单条路径的概率 $P(\pi|\mathbf{x})$

关键的**条件独立性假设**：在给定输入 $\mathbf{x}$ 的情况下，每个时间步的输出是相互独立的。

$$P(\pi|\mathbf{x}) = \prod_{t=1}^T p_t(\pi_t|\mathbf{x})$$

其中 $p_t(k|\mathbf{x})$ 是模型在时间 t 输出标签 k 的概率 (通常来自 Softmax 层)。

Step 1: 计算单条路径的概率 $P(\pi|\mathbf{x})$

关键的**条件独立性假设**：在给定输入 $\mathbf{x}$ 的情况下，每个时间步的输出是相互独立的。

$$P(\pi|\mathbf{x}) = \prod_{t=1}^T p_t(\pi_t|\mathbf{x})$$

其中 $p_t(k|\mathbf{x})$ 是模型在时间 t 输出标签 k 的概率 (通常来自 Softmax 层)。

Step 2: 计算目标文本的概率 $P(\mathbf{y}|\mathbf{x})$

目标文本 $\mathbf{y}$ 的总概率是所有能“折叠”成 $\mathbf{y}$ 的路径 π 的概率之和。

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{y})} P(\pi|\mathbf{x})$$

Step 1: 计算单条路径的概率 $P(\pi|\mathbf{x})$

关键的**条件独立性假设**：在给定输入 $\mathbf{x}$ 的情况下，每个时间步的输出是相互独立的。

$$P(\pi|\mathbf{x}) = \prod_{t=1}^T p_t(\pi_t|\mathbf{x})$$

其中 $p_t(k|\mathbf{x})$ 是模型在时间 t 输出标签 k 的概率 (通常来自 Softmax 层)。

Step 2: 计算目标文本的概率 $P(\mathbf{y}|\mathbf{x})$

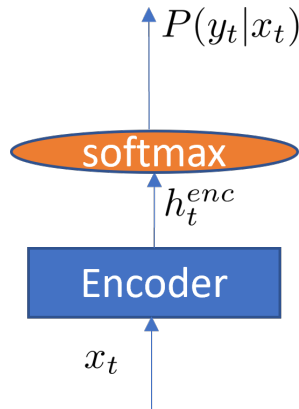
目标文本 $\mathbf{y}$ 的总概率是所有能“折叠”成 $\mathbf{y}$ 的路径 π 的概率之和。

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{y})} P(\pi|\mathbf{x})$$

Step 3: CTC 损失函数

训练的目标是最大化正确文本的对数似然概率，即最小化其负对数。

$$\mathcal{L}_{\text{CTC}} = -\ln P(\mathbf{y}|\mathbf{x})$$



直接对所有可能的路径求和是不可行的，因为路径数量会随 T 指数级增长。

解决方案：动态规划

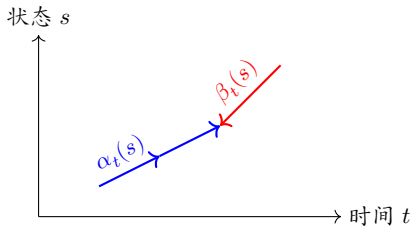
CTC 借鉴了 HMM 中的思想，使用 **前向-后向 (Forward-Backward) 算法** 来高效地计算 $P(\mathbf{y}|\mathbf{x})$ 及其梯度。

- **前向变量** $\alpha_t(s)$:

在时刻 t 结束时，所有能够折叠成 $\mathbf{y}$ 的前 s 个字符的路径的总概率。

- **后向变量** $\beta_t(s)$:

从时刻 t 开始，完成剩余字符序列的路径的总概率。



这个算法是 CTC 能够被实际应用的关键，它避免了暴力枚举，使得训练成为可能。

HMM 对齐:

- 状态路径是隐藏变量
- GMM-HMM 可在训练中估计对齐
- 经典 DNN-HMM 常用强制对齐标签
- 无需人工逐帧标注音素边界

CTC 对齐:

- 隐式学习对齐关系
- 不需要预先确定边界
- 对齐在训练中自动优化
- 只需要序列级别标签

关键差异

两者都可使用整句转录训练。CTC 对所有有效标签路径求和，省去了经典 DNN-HMM 中单独生成帧级状态标签的步骤。

CTC 与 HMM 的对比

声学建模与对齐结构



维度	经典 HMM 系统	CTC 系统
对齐单元 声学模型	音素状态与状态转移 GMM 或 DNN, 提供状态分数	输出标签与 blank 神经编码器, 提供逐帧标签概率
有效路径 训练计算 典型监督	状态拓扑与发音序列约束 对隐藏状态路径求和 转录文本, 混合系统可生成状态标签	折叠后得到目标文字的路径 对有效标签路径求和 转录文本, 无需预先提供帧标签

区分两个层面

长距离声学上下文由声学模型决定, DNN-HMM 也能使用神经网络。CTC 的区别主要在于输出单元、路径约束和序列训练目标。

优势

- **无需预对齐**：实现了真正的端到端训练。
- **简单高效**：模型结构简单，训练和推理速度快。
- **适合流式识别**：天然支持在线、实时的语音识别场景。

劣势

- **条件独立性假设**：这是 CTC 最强的假设，也是其主要瓶颈。它忽略了输出标签之间的依赖关系（即没有内置的语言模型）。
- **尖峰分布**：模型倾向于在极少数帧上输出非 blank 标签，大部分时间输出 blank，可能导致泛化能力问题。

结论

CTC 是一个里程碑式的工作，它优雅地解决了语音识别中的对齐问题。尽管其性能可能不如更复杂的模型（如 Attention, RNN-T），但它的思想和简洁性至今仍有深远影响。

- Graves et al., “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks,” ICML 2006.
- Graves, “Sequence Transduction with Recurrent Neural Networks,” arXiv:1211.3711.
- Hannun et al., “Deep Speech,” arXiv:1412.5567.
- Watanabe et al., “Hybrid CTC/Attention Architecture for End-to-End Speech Recognition,” IEEE JSTSP 2017.
- PyTorch CTCLoss 文档与官方示例。

CTC 的核心局限：条件独立性假设

CTC 假设在给定音频 $\mathbf{x}$ 的情况下，每个时间步的输出是相互独立的。

$$P(\pi|\mathbf{x}) = \prod_{t=1}^T p_t(\pi_t|\mathbf{x})$$

这忽略了输出标签之间的强相关性（例如，'q' 后面几乎总是'u'）。它没有内置的语言模型功能。

CTC 的核心局限：条件独立性假设

CTC 假设在给定音频 $\mathbf{x}$ 的情况下，每个时间步的输出是相互独立的。

$$P(\pi|\mathbf{x}) = \prod_{t=1}^T p_t(\pi_t|\mathbf{x})$$

这忽略了输出标签之间的强相关性（例如，'q' 后面几乎总是'u'）。它没有内置的语言模型功能。

我们能否做得更好？

是否可以构建一个模型，既能端到端训练，又能像人类一样，在生成一个词时考虑到前面已经说过的词？

CTC 的核心局限：条件独立性假设

CTC 假设在给定音频 $\mathbf{x}$ 的情况下，每个时间步的输出是相互独立的。

$$P(\pi|\mathbf{x}) = \prod_{t=1}^T p_t(\pi_t|\mathbf{x})$$

这忽略了输出标签之间的强相关性（例如，'q' 后面几乎总是'u'）。它没有内置的语言模型功能。

我们能否做得更好？

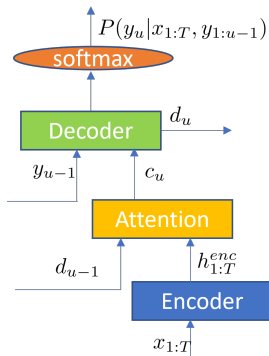
是否可以构建一个模型，既能端到端训练，又能像人类一样，在生成一个词时考虑到前面已经说过的词？

答案是肯定的：引入注意力编码器-解码器框架

AED (Attentional Encoder-Decoder) 是一个用于解决序列到序列问题的**通用框架**。

- **Encoder (编码器)**: 读取整个输入序列 $\mathbf{x}$, 将其转化为一组高级的上下文表示 $\mathbf{h}$ 。
- **Decoder (解码器)**: **自回归地** (一次一个) 生成输出序列 $\mathbf{y}$ 。
- **Attention (注意力)**: 允许 Decoder 在生成每一步输出时, 都能动态地“关注”到 Encoder 输出的最相关部分。
- 训练目标:

$$P(\mathbf{y}|\mathbf{x}) = \prod_u P(y_u|\mathbf{x}, \mathbf{y}_{1:u-1})$$



LAS 模型是 **AED** 框架在语音识别领域的一个经典实现。它非常形象地将 AED 的组件映射到了人类的听写行为上。

AED 通用组件

- Encoder
- Attention
- Decoder

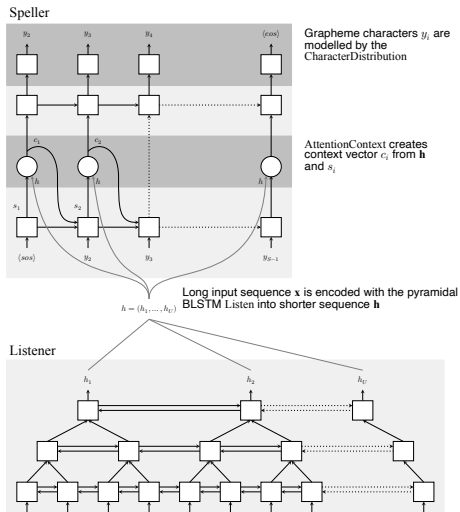


LAS 具体实现

- **Listen** (听)
- **Attend** (关注)
- **Spell** (拼写)

LAS 的具体技术选择

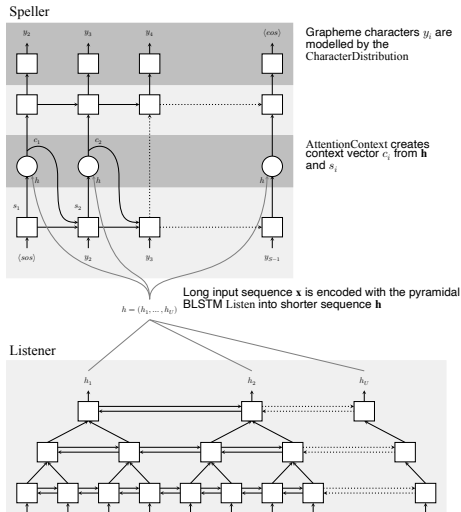
- **Listener (Encoder) 实现**: 使用 **金字塔双向 LSTM (pBLSTM)** 来处理长音频序列。
- **Speller (Decoder) 实现**: 使用一个标准的 **单向 LSTM** 来生成字符。
- **Attention 实现**: 采用基于内容的点积注意力机制。



核心组件：Listener

使用**金字塔循环网络 (pBLSTM)**，在编码的同时对序列进行降采样。

pBLSTM 编码器将音频序列转换为高级特征，逐层将相邻时间步的特征拼接，从而将序列长度减半，同时提取更高级的语义信息。



核心组件：Speller

Speller 是一个自回归的 RNN，它在生成第 i 个字符 y_i 时，依赖于：

- **已生成的内容**：上一个生成的字符 y_{i-1} 。
- **历史记忆**：解码器自身之前的状态 s_{i-1} 。
- **当前应该听什么**：一个通过 Attention 机制从 h 中动态提取的**上下文向量** c_i 。

Attention 机制

在每一步，模型如何知道应该“关注”音频的哪个部分？这正是 Attention 机制的作用，它负责计算出这个关键的上下文向量 c_i 。

问题定义

- 输入: $\mathbf{x} = (x_1, \dots, x_T)$ - 滤波器组频谱特征
- 输出: $\mathbf{y} = (\langle \text{sos} \rangle, y_1, \dots, y_S, \langle \text{eos} \rangle)$ - 字符序列
- 目标: 建模条件概率 $P(\mathbf{y}|\mathbf{x})$

链式法则分解

$$P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|\mathbf{x}, y_{<i})$$

LAS 模型

$$\mathbf{h} = \text{Listen}(\mathbf{x})$$
$$P(\mathbf{y}|\mathbf{x}) = \text{AttendAndSpell}(\mathbf{h}, \mathbf{y})$$

问题

- 音频序列长度：数百到数千帧
- 直接使用 BLSTM：收敛缓慢
- 注意力机制：难以从大量时间步中提取信息

解决方案

- 金字塔 BLSTM (pBLSTM)
- 每层时间分辨率减半
- 3 层 pBLSTM：总压缩比 $2^3 = 8$

计算复杂度：注意力机制复杂度为 $O(US)$ ，减少 U 可显著加速训练和推理

标准 BLSTM

$$h_i^j = \text{BLSTM}(h_{i-1}^j, h_i^{j-1})$$

金字塔 BLSTM (pBLSTM)

$$h_i^j = \text{pBLSTM}(h_{i-1}^j, [h_{2i}^{j-1}, h_{2i+1}^{j-1}])$$

关键特点

- 连续时间步输出拼接
- 每层时间分辨率减半
- 保持双向信息流

优势

- 减少计算复杂度
- 提高训练效率
- 便于注意力机制工作

核心操作

$$c_i = \text{AttentionContext}(s_i, \mathbf{h})$$

$$s_i = \text{RNN}(s_{i-1}, y_{i-1}, c_{i-1})$$

$$P(y_i | \mathbf{x}, y_{<i}) = \text{CharacterDistribution}(s_i, c_i)$$

组件说明

- s_i : 解码器状态
- c_i : 上下文向量
- RNN: 2 层 LSTM
- CharacterDistribution: MLP + Softmax

注意力机制

- 内容注意力
- 基于 s_i 和 h_u 的匹配
- 生成注意力权重 α_i

Attention 让 Speller 在每一步都能动态地决定“焦点”。

计算上下文向量 c_i 的三步：

- 1 计算相关性分数 (Score): 用解码器当前状态 s_{i-1} (作为 Query) 去和编码器所有输出 h_u (作为 Keys) 计算一个匹配分数 $e_{i,u}$ 。

$$e_{i,u} = \text{score}(s_{i-1}, h_u)$$

Attention 让 Speller 在每一步都能动态地决定“焦点”。

计算上下文向量 c_i 的三步：

- 1 **计算相关性分数 (Score):** 用解码器当前状态 s_{i-1} (作为 Query) 去和编码器所有输出 h_u (作为 Keys) 计算一个匹配分数 $e_{i,u}$ 。

$$e_{i,u} = \text{score}(s_{i-1}, h_u)$$

- 2 **计算注意力权重 (Weight):** 将分数通过 Softmax 归一化, 得到注意力权重 $\alpha_{i,u}$ 。权重高的部分表示是当前最应该关注的音频区域。

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_k \exp(e_{i,k})}$$

Attention 让 Speller 在每一步都能动态地决定“焦点”。

计算上下文向量 c_i 的三步：

- 1 计算相关性分数 (Score): 用解码器当前状态 s_{i-1} (作为 Query) 去和编码器所有输出 h_u (作为 Keys) 计算一个匹配分数 $e_{i,u}$ 。

$$e_{i,u} = \text{score}(s_{i-1}, h_u)$$

- 2 计算注意力权重 (Weight): 将分数通过 Softmax 归一化, 得到注意力权重 $\alpha_{i,u}$ 。权重高的部分表示是当前最应该关注的音频区域。

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_k \exp(e_{i,k})}$$

- 3 计算上下文向量 (Context): 对编码器所有输出 h_u 加权求和, 得到上下文向量 c_i 。

$$c_i = \sum_u \alpha_{i,u} h_u$$

Attention 让 Speller 在每一步都能动态地决定“焦点”。

计算上下文向量 c_i 的三步：

- 1 计算相关性分数 (Score): 用解码器当前状态 s_{i-1} (作为 Query) 去和编码器所有输出 h_u (作为 Keys) 计算一个匹配分数 $e_{i,u}$ 。

$$e_{i,u} = \text{score}(s_{i-1}, h_u)$$

- 2 计算注意力权重 (Weight): 将分数通过 Softmax 归一化, 得到注意力权重 $\alpha_{i,u}$ 。权重高的部分表示是当前最应该关注的音频区域。

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_k \exp(e_{i,k})}$$

- 3 计算上下文向量 (Context): 对编码器所有输出 h_u 加权求和, 得到上下文向量 c_i 。

$$c_i = \sum_u \alpha_{i,u} h_u$$

这个 c_i 就是为生成当前字符 y_i 而“量身定制”的声学信息摘要。

训练目标

$$\max_{\theta} \sum_i \log P(y_i | \mathbf{x}, y_{<i}^*; \theta)$$

其中 $y_{<i}^*$ 是真实的前序字符

训练-推理不匹配问题

- **训练时**：使用真实前序字符
- **推理时**：使用模型预测的前序字符
- **问题**：模型未学会处理预测错误

训练时：Teacher Forcing

为了加速收敛和稳定训练，我们总是将**正确的**上一个字符 y_{i-1}^* 作为输入，来预测当前字符 y_i 。

- **优点**：训练稳定，并行度高。
- **缺点**：造成了训练和推理之间的不匹配。

推理时：Autoregressive

推理时，我们没有正确答案，只能将模型**自己预测的**上一个字符 $\hat{y}_{i-1}$ 作为输入。

- **问题**：一旦预测错一个，错误可能会累积，导致后续输出完全错误。这就是**暴露偏差 (Exposure Bias)**。

训练时：Teacher Forcing

为了加速收敛和稳定训练，我们总是将**正确的**上一个字符 y_{i-1}^* 作为输入，来预测当前字符 y_i 。

- **优点**：训练稳定，并行度高。
- **缺点**：造成了训练和推理之间的不匹配。

推理时：Autoregressive

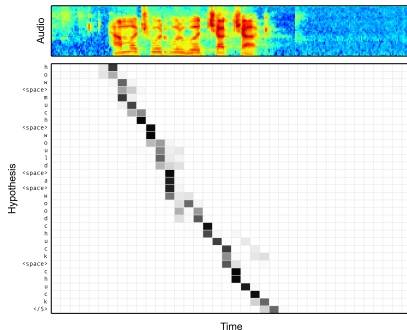
推理时，我们没有正确答案，只能将模型**自己预测的**上一个字符 $\hat{y}_{i-1}$ 作为输入。

- **问题**：一旦预测错一个，错误可能会累积，导致后续输出完全错误。这就是**暴露偏差 (Exposure Bias)**。

解决方案：Scheduled Sampling (计划采样)

在训练过程中，以一定的概率 p 选择模型自己的预测作为下一步的输入，而不是永远使用正确答案。这个概率 p 会随着训练的进程而逐渐增加，让模型慢慢适应自己的错误。

Alignment between the Characters and Audio



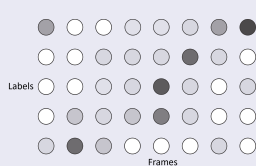
观察结果

- 注意力分布基本单调，无需位置先验
- 能够识别语句开始和结束
- 内容注意力机制有效工作

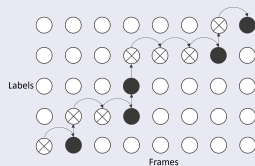
流式处理挑战

- 全注意力需要完整语音
- 流式场景下延迟较高
- 需要基于块的注意力策略

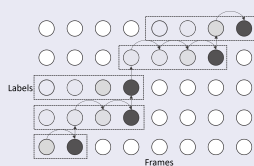
流式解决方案



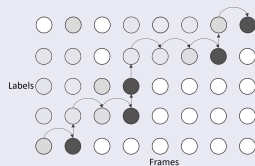
(a) 全注意力



(b) 单调注意力



(c) MoChA



(d) MILK

	Global	Monotonic	MoChA	MILK
是否流式	否 (需看全局/未来)	是 (只向前扫描)	是 (有限右视窗)	是 (仅回看历史)
未来依赖	强, 需要未来帧	无	有限: chunk 大小 w	无
上下文范围	全序列 (过去 + 未来)	极窄 (通常单点)	触发点附近局部块 (大小 w)	无限回看: 从起点到触发点
触发/对齐	软注意力, 自由对齐	硬触发, 单调前进	单调硬触发 + 块内软注意力	单调硬触发 + 历史段软注意力
延迟	高 (离线/大缓冲)	低 (触发即输出)	低-中 ($\approx w$ 帧额外延迟)	低 (无未来窗口)
计算复杂度	$O(T_{enc}T_{dec})$	近线性 (扫描式)	$O(T_{enc} + T_{dec} \cdot w)$	需要前缀和实现, 增量近线性
优点	精度高、全局依赖强	天然流式、对齐可解释	兼顾流式与鲁棒性, 性能接近离线	严格流式且能利用长历史
局限	不可严格在线、内存大	上下文太窄, 易脆弱	需选好 w ; 仍假设单调	实现更复杂, 仍受单调假设限制
典型场景	离线最佳精度	极低延迟、实现简单	低延迟线上常用折中	严格在线且需长历史支持

- 单调/MoChA/MILK 训练常用期望化注意力, 推理用硬触发以实现流式。

- MoChA 延迟主要由 chunk 大小 w 决定; MILK 需用累计/前缀技巧做高效回看。

- Bahdanau et al., “Neural Machine Translation by Jointly Learning to Align and Translate,” ICLR 2015. (注意力机制雏形)
- Chan et al., “Listen, Attend and Spell,” ICASSP 2016. (LAS, 端到端语音的经典 AED)
- Chorowski et al., “Attention-Based Models for Speech Recognition,” NIPS 2015.
- Watanabe et al., “Hybrid CTC/Attention Architecture for End-to-End ASR,” IEEE JSTSP 2017. (CTC+Attention 多任务)
- Chiu and Raffel, “Monotonic Chunkwise Attention,” ICLR 2018. (MoChA, 流式 AED)
- Raffel et al., “Online and Linear-Time Attention by Enforcing Monotonic Alignments,” ICML 2017. (Monotonic Attention)
- Arivazhagan et al., “Monotonic Infinite Lookback for Simultaneous Translation,” ACL 2019. (MILK, 可用于流式 AED)
- Gulati et al., “Conformer: Convolution-augmented Transformer for Speech Recognition,” Interspeech 2020. (常用作 AED 编码器)

我们已经看到了两种主流的端到端范式：

CTC

- 👍 支持流式识别
- 👎 条件独立假设太强

AED (Attention)

- 👍 建模输出依赖
- 👎 依赖全局信息，难于流式

核心矛盾

我们想要一个既能像 AED 一样考虑上下文（打破独立性假设），又能像 CTC 一样进行流式解码（无需等待整句话说完）的模型。

我们已经看到了两种主流的端到端范式：

CTC

支持流式识别
条件独立假设太强

AED (Attention)

建模输出依赖
依赖全局信息，难于流式

核心矛盾

我们想要一个既能像 AED 一样考虑上下文（打破独立性假设），又能像 CTC 一样进行流式解码（无需等待整句话说完）的模型。

答案：RNN-Transducer (RNN-T) 模型

- **音频编码器 (Encoder):** 与 CTC/LAS 类似，处理音频输入 $\mathbf{x}$ ，输出声学特征 h_t 。
- **标签预测网络 (Predictor):** 一个自回归 RNN，根据**之前已生成的字符**，输出一个代表历史文本信息的特征 h_u 。
- **联合网络 (Joint Net):** 一个前馈网络，将 h_t 和 h_u 融合，预测**下一个字符**的概率分布。

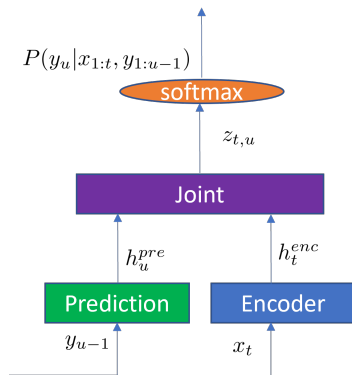


图: RNN-T 架构

RNN-T 的天才之处在于，它在每一步都可以在两个动作中选择一个：

在 (t, u) 时刻，联合网络输出一个分布，包含词汇表和 ϵ (blank)

- **动作 1：输出一个字符 (Emit)**: 如果模型预测出一个非 blank 字符 y_u 。
 - 我们就输出这个字符，然后时间步 t **保持不变**，标签步 u 前进到 $u + 1$ 。
 - 含义：“基于当前的音频信息，我认为可以说一个字了。”
- **动作 2：输出一个 Blank (Consume)**: 如果模型预测出 ϵ 。
 - 我们**不输出任何字符**，然后时间步 t 前进到 $t + 1$ ，标签步 u 保持不变。
 - 含义：“当前音频信息还不足以生成新字符，我需要再听一点。”

关键优势

这种机制使得 RNN-T 的对齐非常灵活。它既不像 CTC 那样强制每帧都有输出，也不像 AED 那样需要看完全局，从而实现了**真正的流式识别**。

CTC 和 RNN-T 的有效路径对比

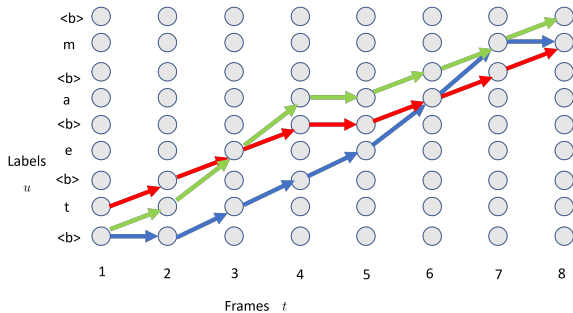


图: "team" 的 CTC 路径示例

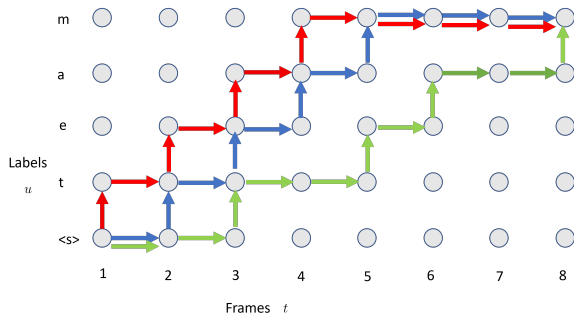
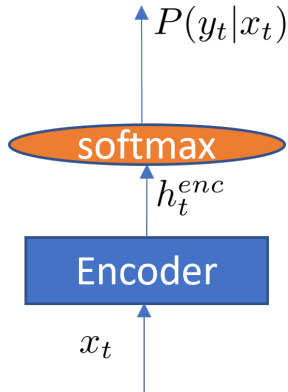
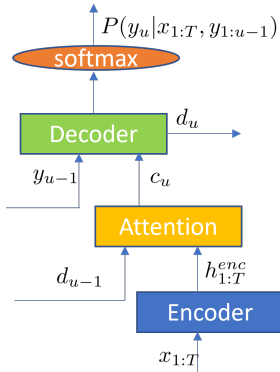


图: "team" 的 RNN-T 路径示意

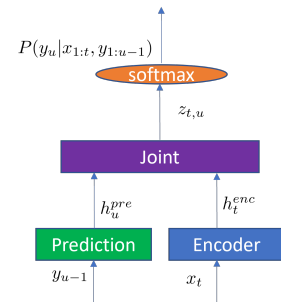
特性	CTC	AED (Attention)	RNN-T
核心思想	条件独立, 最大化所有有效路径概率	全局上下文自回归生成	局部上下文自回归生成
对齐方式	隐式 (通过 Blank 强制对齐)	显式 (通过 Attention 软对齐)	隐式 (通过 Blank 解耦对齐)
流式能力	天然支持	困难 (需特殊改造)	天然支持
输出依赖	无 (条件独立)	有 (建模 $P(y_i y_{<i})$)	有 (建模 $P(y_u y_{<u})$)
优点	速度快, 结构简单	精度高, 能利用全局信息	集两者之长: 流式且精度高
缺点	精度相对较低	计算复杂, 延迟高, 非流式	实现相对复杂, 训练不稳定
工业应用	早期主流, 现用于一些低资源场景	学术研究多, 部分离线任务	当前工业界主流选择



(a) CTC



(b) AED



(c) RNN-T

- Graves, “Sequence Transduction with Recurrent Neural Networks,” arXiv:1211.3711. (提出 RNN-T 框架)
- Graves et al., “Towards End-to-End Speech Recognition with Recurrent Neural Networks,” ICML 2014 Workshop. (早期端到端与 RNN-T 相关)
- Rao et al., “Exploring Architectures, Data and Units for Streaming End-to-End Speech Recognition with RNN-T,” ASRU 2017. (流式 ASR 里程碑)
- Prabhavalkar et al., “A Comparison of Sequence-to-Sequence Models for Speech Recognition,” Interspeech 2017. (RNN-T vs AED/CTC)
- Kannan et al., “An Analysis of Incorporating an External Language Model into a RNN-T for ASR,” ICASSP 2018. (LM 融合)
- Mahadeokar et al., “Streaming End-to-End Speech Recognition with the Transformer Transducer,” ICASSP 2021. (T-T, Transformer 化的 RNN-T)
- WeNet、NeMo、ESPnet 文档与示例 (RNN-T/T-T 实现与训练细节)

推荐阅读

- Wenet 社区杨超 E2E ASR 教程
- CMU Shinji Watanabe E2E ASR 报告
- Google Deepmind Tara Sainath E2E ASR 报告
- Jinyu Li: Recent Advances in End-to-End Automatic Speech Recognition