第六讲：自动语音识别

端到端模型

王帅

2025 年 9 月 29 日

南京大学 NANJING UNIVERSITY

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

课程大纲

外地猜天气

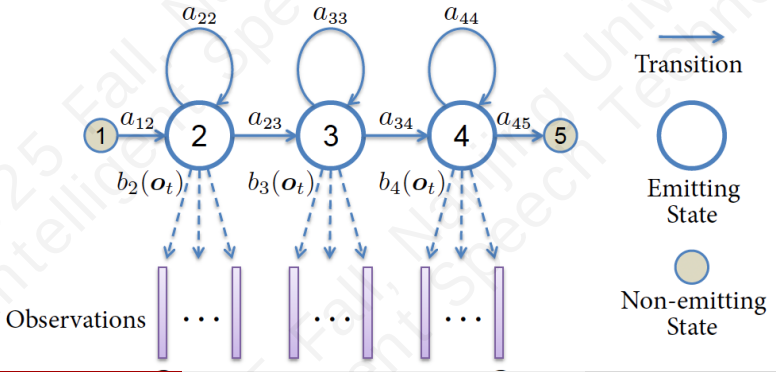
- 隐藏状态：
  天气的真实序列
  (晴 → 晴 → 雨)
- 观测序列：
  朋友的心情序列
  (开心, 开心, 难过)
- 我们的目标：
  根据心情序列，推断出最可能的天气序列。

语音识别系统

- 隐藏状态：
  构成单词的 音素序列
  (/h/ → /e/ → /l/ → /o/)
- 观测序列：
  从语音信号中提取的 声学特征 (MFCC) 序列 $(\vec{o}_1, \vec{o}_2, \vec{o}_3, \ldots, \vec{o}_T)$
- 我们的目标：
  根据声学特征序列，解码出最可能的音素序列，进而得到词序列。

HMM 成为了连接 不可见的语言单元（音素）和 可测量的声学信号之间的完美桥梁。

- 语音信号具有时序性和序列性，其特性随时间演进。
- HMM 能够优雅地对这种时变序列进行建模。
- 语音的观测值 (声学特征) 是连续的，但其背后的本质状态是离散的 (如音素)，这与 HMM 的



一个 HMM 模型 λ 由五个要素定义：

- 状态集合 $S = \{s_1, \dots, s_N\}$：模型的内部状态，如音素的不同阶段。
- 观测序列 $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$：每一时刻的声学特征向量[1]。
- 转移概率矩阵 $A = [a_{ij}]$，其中 $a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$。
- 发射概率（观测概率）$B = \{b_j(\mathbf{x}_t)\}$，其中 $b_j(\mathbf{x}_t) = p(\mathbf{x}_t | q_t = s_j)$。
- 初始状态分布 $\pi = \{\pi_i\}$，其中 $\pi_i = P(q_1 = s_i)$。

$$\lambda = (A, B, \pi)$$

[1]注意此处的 $\mathbf{X}$ 可以认为是之前提到的语音观测序列 $\mathbf{O}$，为了跟更广泛的 HMM 的定义保持符号一致，后续将采用 $\mathbf{X}$

1. 评估 (Evaluation): 给定模型 λ，计算观测序列 $\mathbf{X}$ 出现的概率 $p(\mathbf{X}|\lambda)$。
   - 算法: 前向算法 (Forward Algorithm)。
2. 解码 (Decoding): 给定模型 λ 和观测序列 $\mathbf{X}$，找到最可能的状态序列 Q。
   - 算法: 维特比算法 (Viterbi Algorithm)。
3. 学习 (Learning): 给定观测序列 $\mathbf{X}$，调整模型参数 λ 使得 $p(\mathbf{X}|\lambda)$ 最大。
   - 算法: Baum-Welch 算法 (即 EM 算法在 HMM 中的特例)。

HMM 的优势与局限性

优势：

- 数学理论完备
- 有效的推理算法
- 可解释性强
- 训练相对简单
- 成熟的工程实现

局限性：

- 条件独立假设过强
- 状态数量需要预先确定
- 难以建模长距离依赖
- 特征工程需求高
- 对齐质量依赖人工标注

核心问题

HMM 需要帧级别的精确对齐，这要求训练数据必须有详细的时间边界标注，增加了数据准备的复杂度。

核心思想

各取所长：保留 HMM 强大的时序建模能力（状态转移），同时用一个强大的深度神经网络 **(DNN)** 来替换掉相对较弱的 GMM，用以计算发射概率。

$$\text{GMM-HMM: } \text{HMM}(\pi, A) + \text{GMM}(B)$$

$$\Downarrow$$

$$\text{DNN-HMM: } \text{HMM}(\pi, A) + \text{DNN}(B)$$

核心思想

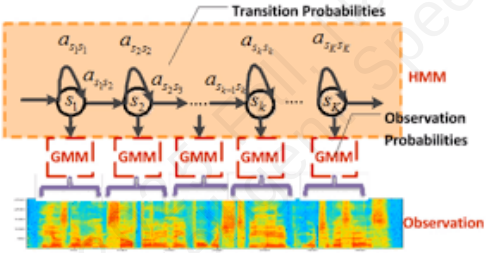
各取所长：保留 HMM 强大的时序建模能力（状态转移），同时用一个强大的深度神经网络 **(DNN)** 来替换掉相对较弱的 GMM，用以计算发射概率。

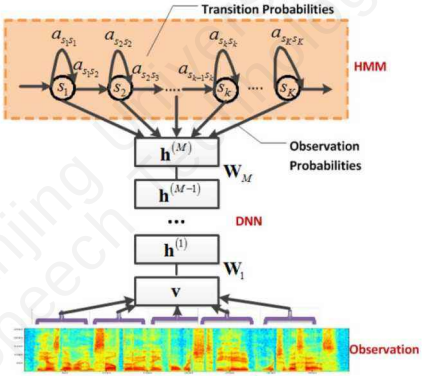
$$\text{GMM-HMM: HMM}(\pi, A) + \text{GMM}(B)$$

$$\Downarrow$$

$$\text{DNN-HMM: HMM}(\pi, A) + \text{DNN}(B)$$

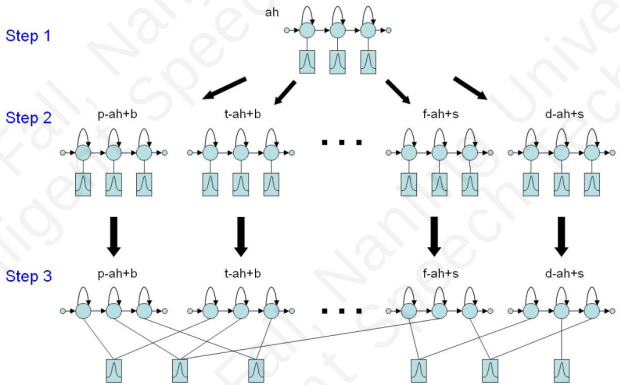
- 在这个混合系统中，DNN 的任务是：给定一个声学特征向量 $\mathbf{x}_t$，计算出它属于每一个 HMM 状态 s_j 的概率。
- HMM 的任务保持不变：使用维特比算法，在给定发射概率（由 DNN 提供）和转移概率的情况下，寻找最优的状态序列。


Transition Probabilities
HMM
Observation Probabilities
GMM
GMM
GMM
GMM
GMM
Observation


Transition Probabilities
HMM
Observation Probabilities
DNN
Observation

核心思想

将声学特性相似的三音素状态“绑定”在一起，共享 GMM 参数。



我们已经走了多远？——混合系统的时代

通过用 DNN 替代 GMM，DNN-HMM 混合系统极大地提升了识别性能，开启了深度学习的革命。然而，这套系统依然是一个由多个独立部分拼接而成的复杂流水线。

我们已经走了多远？——混合系统的时代

通过用 DNN 替代 GMM，DNN-HMM 混合系统极大地提升了识别性能，开启了深度学习的革命。然而，这套系统依然是一个由多个独立部分拼接而成的复杂流水线。

一个大胆的设想：能否一步到位？

构建一个单一的、统一的神经网络，直接完成从声学特征到文本的映射。

$$\text{声学特征} \xrightarrow{\text{一个神奇的黑箱模型}} \text{文本}$$

我们已经走了多远？——混合系统的时代

通过用 DNN 替代 GMM，DNN-HMM 混合系统极大地提升了识别性能，开启了深度学习的革命。然而，这套系统依然是一个由多个独立部分拼接而成的复杂流水线。

一个大胆的设想：能否一步到位？

构建一个单一的、统一的神经网络，直接完成从声学特征到文本的映射。

$$\text{声学特征} \xrightarrow{\text{一个神奇的黑箱模型}} \text{文本}$$

实现梦想的钥匙

为了实现这个目标，研究者们发明了多种革命性的架构，我们将深入探索其中的三大主流技术：CTC, Attention, RNN-T.

遗留问题

核心问题：GMM-HMM 系统打的标签为什么可以训练更好的 DNN-HMM 系统？

我们观察到一个现象：

系统 A: GMM-HMM

- 性能较差
- 词错误率 (WER) 较高

系统 B: DNN-HMM

- 性能更好
- 词错误率 (WER) 更低

然而，系统 B 的训练却依赖于系统 A 生成的对齐标签。

为什么？

要训练 DNN-HMM，我们需要知道每一帧语音对应哪个音素。这个过程被称为“强制对齐”。

- 输入：
  - 一段语音波形
  - 已知的、完全正确的文本转录

要训练 DNN-HMM，我们需要知道每一帧语音对应哪个音素。这个过程被称为“强制对齐”。

- 输入：
  - 一段语音波形
  - 已知的、完全正确的文本转录
- 工具：一个已经训练好的 GMM-HMM 系统。

要训练 DNN-HMM，我们需要知道每一帧语音对应哪个音素。这个过程被称为“强制对齐”。

- 输入：
  - 一段语音波形
  - 已知的、完全正确的文本转录
- 工具：一个已经训练好的 GMM-HMM 系统。
- 任务：GMM-HMM 系统被“强制”在给定的正确文本路径上，寻找最可能的音素与时间的对应关系。

要训练 DNN-HMM，我们需要知道每一帧语音对应哪个音素。这个过程被称为“强制对齐”。

- 输入：
  - 一段语音波形
  - 已知的、完全正确的文本转录
- 工具：一个已经训练好的 GMM-HMM 系统。
- 任务：GMM-HMM 系统被“强制”在给定的正确文本路径上，寻找最可能的音素与时间的对应关系。
- 输出：每一帧语音的音素状态标签。

GMM-HMM 的“差”主要体现在开放识别任务中。

- 声学建模能力弱:

GMM-HMM 的“差”主要体现在开放识别任务中。

- 声学建模能力弱：
  - GMM (高斯混合模型) 对声学特征分布的假设过于简单。

GMM-HMM 的“差”主要体现在开放识别任务中。

- 声学建模能力弱:
  - GMM (高斯混合模型) 对声学特征分布的假设过于简单。
  - 难以捕捉复杂的、非线性的语音变化（如不同说话人、口音、语速）。

GMM-HMM 的“差”主要体现在开放识别任务中。

- 声学建模能力弱:
  - GMM (高斯混合模型) 对声学特征分布的假设过于简单。
  - 难以捕捉复杂的、非线性的语音变化（如不同说话人、口音、语速）。
- 上下文信息有限:

悖论剖析 (1):

GMM-HMM 的“差”在哪里？

GMM-HMM 的“差”主要体现在开放识别任务中。

- 声学建模能力弱:
  - GMM (高斯混合模型) 对声学特征分布的假设过于简单。
  - 难以捕捉复杂的、非线性的语音变化（如不同说话人、口音、语速）。
- 上下文信息有限:
  - 传统 GMM 通常只看当前一帧，对上下文的建模能力非常有限。

悖论剖析 (1):

GMM-HMM 的“差”在哪里？

GMM-HMM 的“差”主要体现在开放识别任务中。

- 声学建模能力弱:
  - GMM (高斯混合模型) 对声学特征分布的假设过于简单。
  - 难以捕捉复杂的、非线性的语音变化（如不同说话人、口音、语速）。
- 上下文信息有限:
  - 传统 GMM 通常只看当前一帧，对上下文的建模能力非常有限。
- 多个模块误差累积:

悖论剖析 (1):

GMM-HMM 的“差”在哪里？

GMM-HMM 的“差”主要体现在开放识别任务中。

- 声学建模能力弱:
  - GMM (高斯混合模型) 对声学特征分布的假设过于简单。
  - 难以捕捉复杂的、非线性的语音变化（如不同说话人、口音、语速）。
- 上下文信息有限:
  - 传统 GMM 通常只看当前一帧，对上下文的建模能力非常有限。
- 多个模块误差累积:
  - 在开放识别中，声学模型、语言模型、解码器的任何一点小错误都可能被放大，导致最终识别结果错误。

GMM-HMM 的“差”主要体现在开放识别任务中。

- 声学建模能力弱:
  - GMM (高斯混合模型) 对声学特征分布的假设过于简单。
  - 难以捕捉复杂的、非线性的语音变化（如不同说话人、口音、语速）。
- 上下文信息有限:
  - 传统 GMM 通常只看当前一帧，对上下文的建模能力非常有限。
- 多个模块误差累积:
  - 在开放识别中，声学模型、语言模型、解码器的任何一点小错误都可能被放大，导致最终识别结果错误。

结论：它的“差”是整个系统在无约束条件下的综合表现。

GMM-HMM 在强制对齐任务中提供的标签是“足够好”的。

- 任务大大简化:

GMM-HMM 在强制对齐任务中提供的标签是“足够好”的。

- 任务大大简化:
  - 由于文本已知，搜索空间被极大地限制了。模型不需要在成千上万的词中做选择，只需在给定的音素序列上找到最佳路径。

GMM-HMM 在强制对齐任务中提供的标签是“足够好”的。

- 任务大大简化:
  - 由于文本已知，搜索空间被极大地限制了。模型不需要在成千上万的词中做选择，只需在给定的音素序列上找到最佳路径。
  - 这就像做“完形填空”而不是“开放式问答”，难度天差地别。

GMM-HMM 在强制对齐任务中提供的标签是“足够好”的。

- 任务大大简化:
  - 由于文本已知，搜索空间被极大地限制了。模型不需要在成千上万的词中做选择，只需在给定的音素序列上找到最佳路径。
  - 这就像做“完形填空”而不是“开放式问答”，难度天差地别。
- 提供了强统计相关性:

GMM-HMM 在强制对齐任务中提供的标签是“足够好”的。

- 任务大大简化:
  - 由于文本已知，搜索空间被极大地限制了。模型不需要在成千上万的词中做选择，只需在给定的音素序列上找到最佳路径。
  - 这就像做“完形填空”而不是“开放式问答”，难度天差地别。
- 提供了强统计相关性:
  - 尽管音素边界的标注可能不是 100% 精确到毫秒，但它在宏观上是正确的。

悖论剖析 (2):

对齐标签的“好”在哪里？

GMM-HMM 在强制对齐任务中提供的标签是“足够好”的。

- 任务大大简化:
  - 由于文本已知，搜索空间被极大地限制了。模型不需要在成千上万的词中做选择，只需在给定的音素序列上找到最佳路径。
  - 这就像做“完形填空”而不是“开放式问答”，难度天差地别。
- 提供了强统计相关性:
  - 尽管音素边界的标注可能不是 100% 精确到毫秒，但它在宏观上是正确的。
  - 它告诉我们：“这块区域的语音大概率是音素 /a/”。

GMM-HMM 在强制对齐任务中提供的标签是“足够好”的。

- 任务大大简化:
  - 由于文本已知，搜索空间被极大地限制了。模型不需要在成千上万的词中做选择，只需在给定的音素序列上找到最佳路径。
  - 这就像做“完形填空”而不是“开放式问答”，难度天差地别。
- 提供了强统计相关性:
  - 尽管音素边界的标注可能不是 100% 精确到毫秒，但它在宏观上是正确的。
  - 它告诉我们：“这块区域的语音大概率是音素 /a/”。
- 为 **DNN** 提供了监督信号:

悖论剖析 (2):

对齐标签的“好”在哪里？

GMM-HMM 在强制对齐任务中提供的标签是“足够好”的。

- 任务大大简化:
  - 由于文本已知，搜索空间被极大地限制了。模型不需要在成千上万的词中做选择，只需在给定的音素序列上找到最佳路径。
  - 这就像做“完形填空”而不是“开放式问答”，难度天差地别。
- 提供了强统计相关性:
  - 尽管音素边界的标注可能不是 100% 精确到毫秒，但它在宏观上是正确的。
  - 它告诉我们：“这块区域的语音大概率是音素 /a/”。
- 为 **DNN** 提供了监督信号:
  - 这些（可能带有一点噪声的）帧级别标签，正是 DNN 模型进行监督学习所需要的。

悖论剖析 (2):

对齐标签的“好”在哪里？

GMM-HMM 在强制对齐任务中提供的标签是“足够好”的。

- 任务大大简化:
  - 由于文本已知，搜索空间被极大地限制了。模型不需要在成千上万的词中做选择，只需在给定的音素序列上找到最佳路径。
  - 这就像做“完形填空”而不是“开放式问答”，难度天差地别。
- 提供了强统计相关性:
  - 尽管音素边界的标注可能不是 100% 精确到毫秒，但它在宏观上是正确的。
  - 它告诉我们：“这块区域的语音大概率是音素 /a/”。
- 为 **DNN** 提供了监督信号:
  - 这些（可能带有一点噪声的）帧级别标签，正是 DNN 模型进行监督学习所需要的。

结论：对齐标签的“好”，在于它为 DNN 提供了一个大规模、基本正确、可用于训练的帧级别标注数据集。

DNN (深度神经网络) 是一个极其强大的分类器，它能超越“老师”GMM。

- 强大的建模能力:
  - DNN 可以学习到声学特征和音素标签之间高度非线性的复杂映射关系，这是 GMM 无法做到的。
- 利用上下文信息:
  - DNN 的输入可以是一个包含前后多帧的特征窗，使其能够捕捉到 GMM 忽略的动态变化和上下文信息。
- 对噪声的鲁棒性与泛化能力:
  - 即使 GMM 提供的标签有微小的边界错误（噪声），DNN 在海量数据的训练下，能够学习到本质的规律，并自动平滑掉这些噪声。
  - 它学到的是一个更通用、更鲁棒的声学表征。

总结：为何“差”老师能教出“好”学生

1. 任务不同：GMM-HMM 的“差”体现在开放识别任务，但我们用它做的是强制对齐任务。
2. 标签“足够好”：在已知文本的约束下，GMM-HMM 能生成大规模、基本正确的帧级别音素标签，这是宝贵的监督信号。
3. **DNN** 更强大：DNN 作为分类器，其建模能力、上下文利用能力、泛化能力远超 GMM。
4. 超越老师：DNN 从 GMM 提供的“不完美”数据中学习，最终建立了一个更精确、更鲁棒的声学模型。当用这个新模型替换掉旧的 GMM 后，整个 DNN-HMM 系统的性能自然就超越了原来的 GMM-HMM 系统。

所以，不是“差”老师教出了“好”学生，
而是“好”学生利用“差”老师提供的基础教材完成了自我超越！

传统系统的核心痛点

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。

端到端 (E2E) 模型：化繁为简的革命

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。

E2E 如何解决这些痛点？

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 $\mathbf{x} = (x_1, x_2, ..., x_T)$，长度为 T。
- 输出：目标文本序列 $\mathbf{y} = (y_1, y_2, ..., y_U)$，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！

CTC 的核心思想：引入 Blank (ϵ) 标签

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。

CTC 的核心思想：引入 Blank (ϵ) 标签

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。

CTC 的核心思想：引入 Blank (ϵ) 标签

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' → "helo"
- 'h-e-l-ϵ-l-o' → "hello"

CTC 的核心思想：引入 Blank (ϵ) 标签

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' → "helo"
- 'h-e-l-ϵ-l-o' → "hello"

通过引入 ϵ，CTC 将模糊的“对齐”问题，巧妙地转化为了一个清晰的“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 $\mathbf{y}$

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 $\mathbf{y}$

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’

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 $\mathbf{y}$

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 $\mathcal{B}(\pi)$

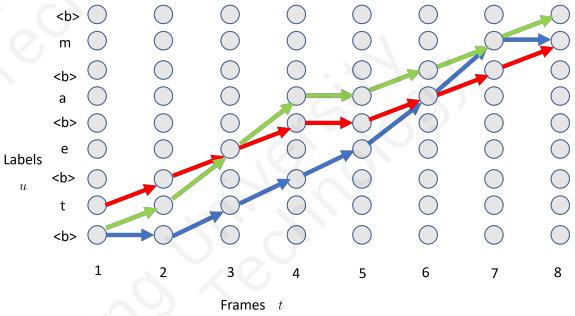
1. 合并连续重复的标签：‘h,h,e,l,ϵ,l,o,o,ϵ’ → ‘h,e,l,ϵ,l,o,ϵ’
2. 移除所有 **blank (ϵ)** 标签：‘h,e,l,ϵ,l,o,ϵ’ → ‘h,e,l,l,o’

最终输出正确：**"hello"**

CTC：路径与对齐

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。

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

CTC 的形式化定义

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 $\pi_t \in \mathcal{L}'$。

CTC 的形式化定义

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 $\mathbf{y}$。

示例：哪些路径可以生成"hello"？

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

- 路径 1: $(\mathsf{h}, \mathsf{e}, \mathsf{l}, \epsilon, \mathsf{l}, \mathsf{o})$ *(最简洁的形式)*
- 路径 2: $(\mathsf{h}, \mathsf{h}, \mathsf{e}, \mathsf{l}, \epsilon, \mathsf{l}, \mathsf{l}, \mathsf{o})$ *(包含字符重复)*
- 路径 3: $(\epsilon, \mathsf{h}, \mathsf{e}, \epsilon, \mathsf{l}, \epsilon, \mathsf{l}, \mathsf{o}, \epsilon)$ *(包含大量 blank)*
- 路径 4: $(\mathsf{h}, \mathsf{e}, \mathsf{e}, \mathsf{l}, \mathsf{l}, \epsilon, \epsilon, \mathsf{l}, \mathsf{o}, \mathsf{o})$ *(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。

CTC 的形式化定义

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 $\mathbf{y}$。

训练目标

训练模型（如 RNN），使其能够为正确的文本 $\mathbf{y}$ 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 $P(\mathbf{y}|\mathbf{x})$

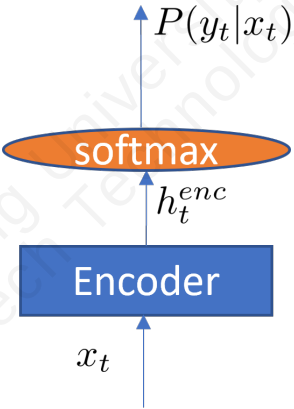
目标文本 $\mathbf{y}$ 的总概率是所有能“折叠”成 $\mathbf{y}$ 的路径 π 的概率之和。

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

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。

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

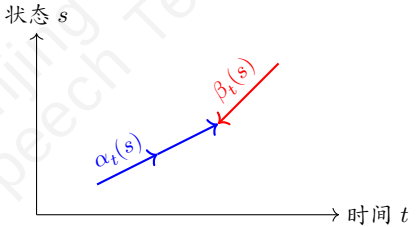


直接对所有可能的路径求和是不可行的，因为路径数量会随 T 指数级增长。

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。

CTC 与 HMM 的对比

对齐方式对比

HMM 对齐：

- 需要强制对齐 (Forced Alignment)
- 依赖预定义的音素边界
- 对齐质量影响识别性能
- 训练需要详细标注数据

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

HMM 需要精确的时间边界标注，而 CTC 可以从粗粒度的转录文本中自动学习对齐，大大降低了数据标注的成本。

CTC 与 HMM 的对比

建模能力对比

维度	**HMM**	**CTC**
时序建模	马尔可夫假设，短程依赖	可结合 RNN/Transformer，长程依赖
特征学习	需要手工特征工程	端到端特征学习
状态空间	预定义状态数量	动态状态空间
并行计算	序列化 Viterbi 解码	可并行训练和推理
可扩展性	状态数增长复杂度高	线性复杂度扩展
多模态融合	困难，需要复杂融合策略	自然支持多模态输入

理论复杂度分析：

- **HMM** 训练：$O(T \cdot N^2)$ (T 为序列长度，N 为状态数)
- **CTC** 训练：$O(T \cdot |L|^2)$ (|L| 为标签序列长度)
- **HMM** 解码：$O(T \cdot N^2)$ (Viterbi 算法)
- **CTC** 解码：$O(T \cdot |V|)$ (V 为词汇表大小，贪婪解码)

CTC 的优势与劣势

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？

从 CTC 到新范式：为什么要更进一步？

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。

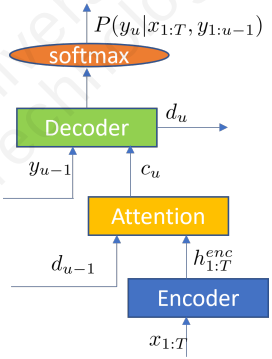
我们能否做得更好？

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

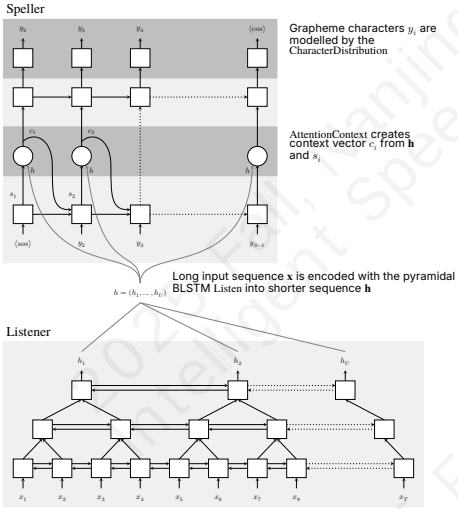
⇒

LAS 具体实现

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

LAS 的具体技术选择

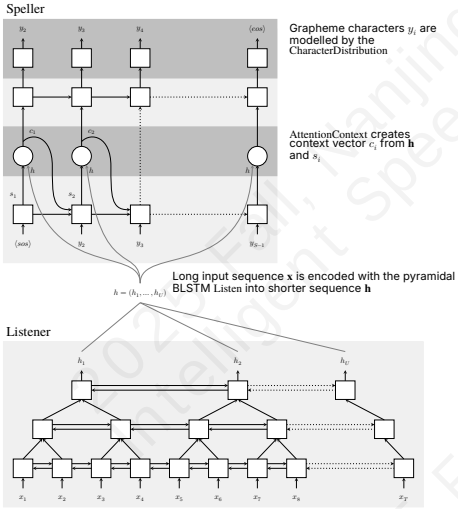
- **Listener (Encoder)** 实现：使用金字塔双向 **LSTM (pBLSTM)** 来处理长音频序列。
- **Speller (Decoder)** 实现：使用一个标准的单向 **LSTM** 来生成字符。
- **Attention** 实现：采用基于内容的点积注意力机制。



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 $\mathbf{h}$ 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

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

Listener 组件：金字塔 BLSTM 编码器

为什么需要金字塔结构？

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

金字塔 BLSTM 结构

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 机制详解

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

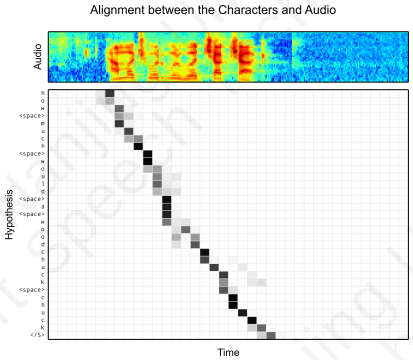
推理时，我们没有正确答案，只能将模型自己预测的上一个字符 $\hat{y}_{i-1}$ 作为输入。

- 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 会随着训练的进行而逐渐增加，让模型慢慢适应自己的错误。

注意力可视化



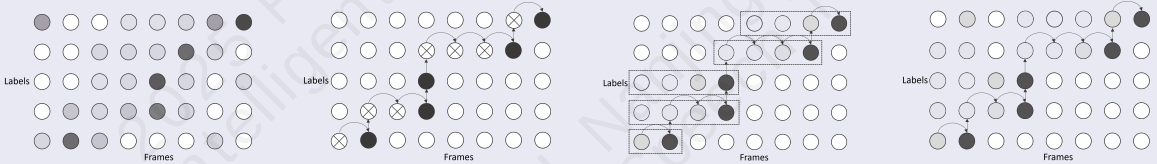
观察结果

- 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

AED 流式注意力方案对比

	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）

能否两全其美？

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。

能否两全其美？

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

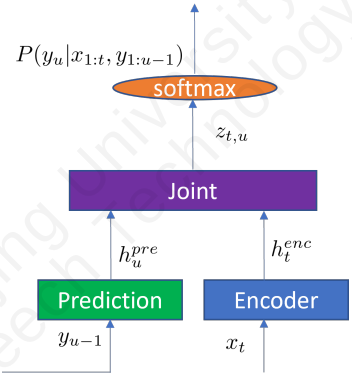
核心矛盾

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**

RNN-T 架构：三驾马车

- 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 核心机制：通过 Blank 解耦对齐

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，我认为可以再说一个字了。"
- 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

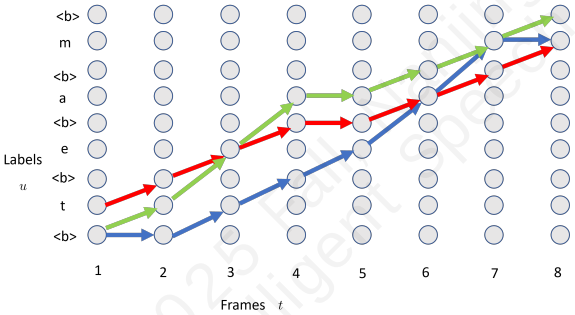


图: "team" 的 CTC 路径示例

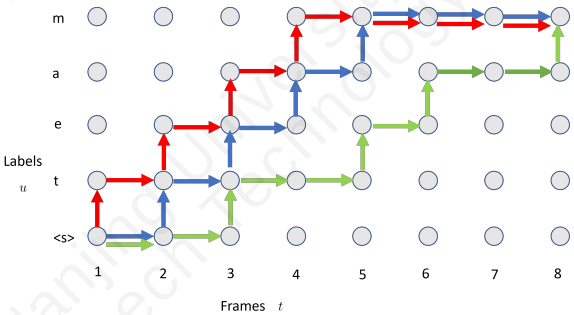
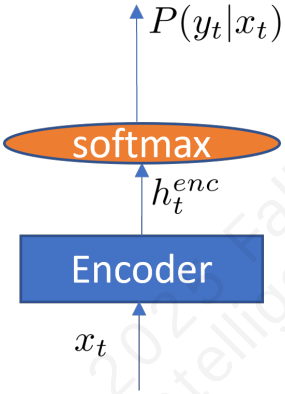


图: "team" 的 RNN-T 路径示意

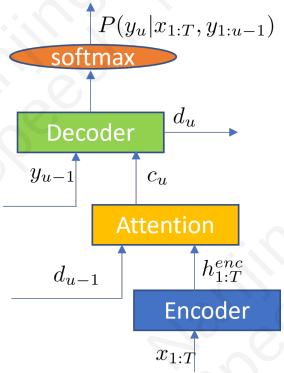
CTC vs. AED vs. RNN-T

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 $P(y_i \mid y_{<i})$)	有 (建模 $P(y_u \mid y_{<u})$)
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

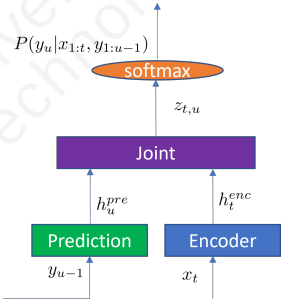
CTC vs. AED vs. RNN-T



(a) CTC



(b) AED

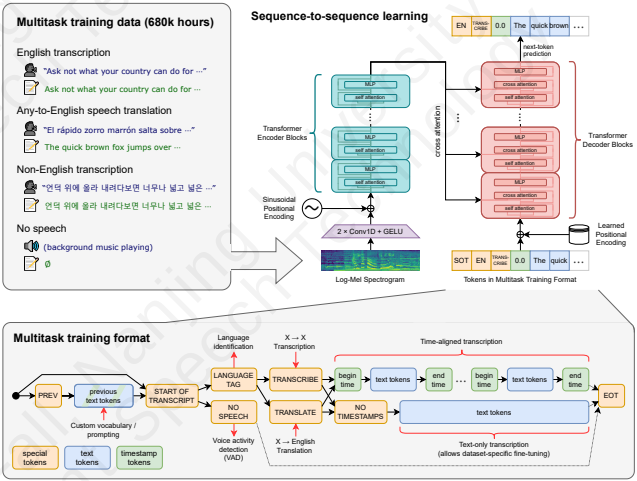


(c) RNN-T

- 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）

语音识别的挑战

- 传统 **ASR** 系统：
  - 需要针对特定数据集进行微调
  - 跨域鲁棒性有限
  - 复杂的多阶段流水线
- 主要问题：
  - 对分布偏移敏感
  - 在新域上泛化能力差
  - 计算资源需求高



核心思想

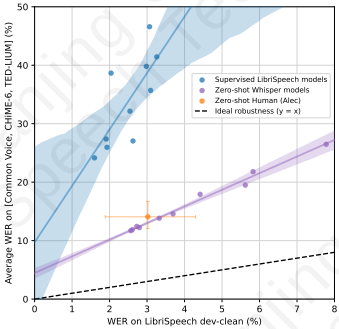
在大规模、多样化的弱监督语音-文本对齐数据上训练单一的编码器-解码器 Transformer，以获得跨域与多语言的强鲁棒性与零样本能力。

关键创新：

- 大规模多语言数据与多任务联合训练（识别/翻译/语言识别/VAD）
- 统一的序列到序列格式，用“特殊标记”指定任务与语言
- 字节级 BPE 分词（GPT-2 风格），对多语言与稀有字符友好
- 简洁的推理接口（CLI 与 Python），开箱即用

结果：

- 多域零样本条件下表现稳健，具备强鲁棒性
- 支持多语言识别、语言识别与语音翻译
- 提供多种模型规模，覆盖速度/准确率权衡
- 开源代码与权重（MIT License），无需任务特定微调即可使用



关键洞察

在单一数据集（如 LibriSpeech）上训练并达到极低 WER 的模型，常在分布外数据上显著退化；多样化弱监督预训练有助于弥合这一差距。

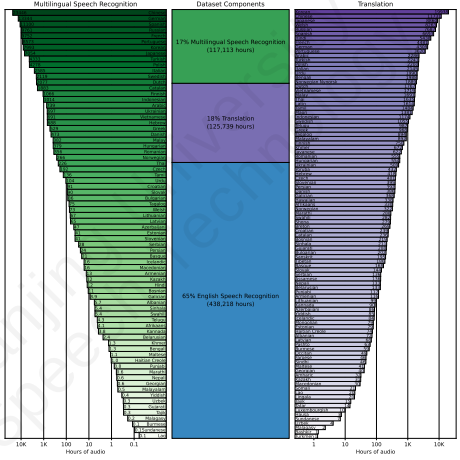
弱监督学习的规模化

先前工作：

- SpeechStew: 5,140 小时
- GigaSpeech: 10,000 小时
- People's Speech: 30,000 小时

Whisper 规模：

- 基于互联网大规模采集与质量过滤的多语言语音-文本对齐数据
- 覆盖语音识别与“非英语到英语”翻译等任务
- 数据多样性强，显著提升分布外鲁棒性



图：训练数据集统计

数据处理流水线

1. 网络规模收集：从互联网收集配对的音频与转录文本
2. 质量过滤：
   - 检测并移除机器生成的转录文本
   - 语言检测与匹配
   - 模糊去重
   - 对高错误率数据源进行人工检查
3. 分段：30 秒音频块与对应转录文本
4. 统一多任务序列：通过起始/语言/任务/时间戳/文本/结束等特殊标记，构造统一的目标序列

关键设计选择

最小化预处理与强质量过滤并重，让模型通过多任务标记自然学习不同任务与风格。

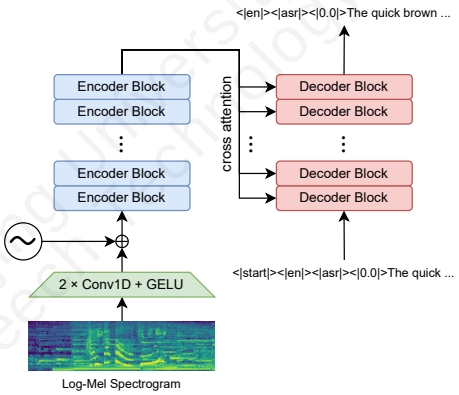
模型架构

架构：编码器-解码器 Transformer

- 编码器：接收 80 维对数 Mel 频谱
- 解码器：自回归语言模型，条件于音频与任务/语言标记
- 分词器：字节级 BPE（GPT-2 风格）

模型规模（**repo** 发布）：

- tiny（~39M），提供 tiny 与 tiny.en
- base（~74M），提供 base 与 base.en
- small（~244M），提供 small 与 small.en
- medium（~769M），提供 medium 与 medium.en
- large（~1550M），large-v2 / large-v3（多语言）
- turbo（~809M）：large-v3 的优化变体，推理更快，准确率略降



统一任务规范

所有任务都表示为标记序列：

1. ◁startoftranscript▷
2. 语言标记（多语言）
3. 任务标记：◁transcribe▷ 或 ◁translate▷
4. 时间戳标记：◁notimestamps▷ 或时间戳对
5. 输出文本
6. ◁endoftranscript▷

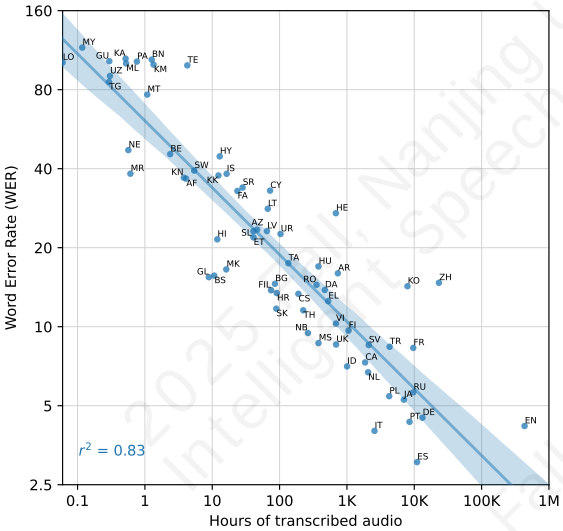
优势

- 单一模型统一处理识别、语言识别与翻译
- 通过标记自然实现任务与语言条件化
- 可输出时间戳，便于长音频切分与对齐

数据集	wav2vec 2.0 Large (no LM)	Whisper Large V2	相对错误率降低 (%)
LibriSpeech Clean	**2.7**	**2.7**	0.0
Artie	24.5	**6.2**	74.7
Common Voice	29.9	**9.0**	69.9
Fleurs En	14.6	**4.4**	69.9
Tedlium	10.5	**4.0**	61.9
CHiME6	65.8	**25.5**	61.2
平均	29.3	**12.8**	55.2

表: 在多域零样本条件下的示例对比（具体数值依赖评测设置与模型规模）

多语言性能



关键发现：

- 多语言性能与可用训练数据量强相关（示例中可观察到显著关联）
- 数据规模每显著增长一档，WER/CER 明显下降
- 文字系统差异与低资源数据稀缺带来挑战

基准测试结果：

- 覆盖 MLS、VoxPopuli、FLEURS、Common Voice 等多语种基准
- 语言列表见 repo `tokenizer.py`

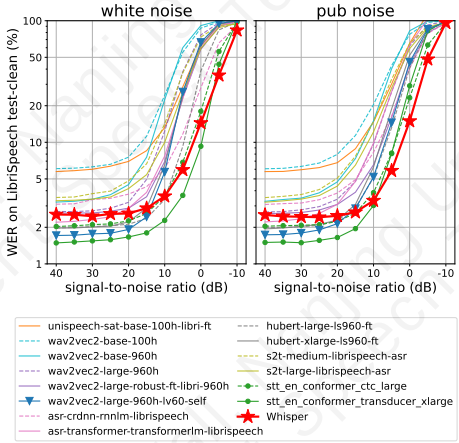
语音翻译性能

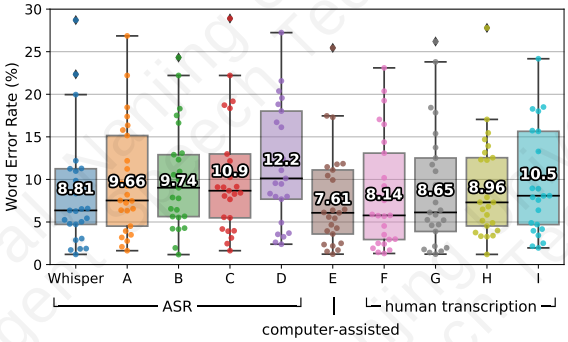
X → 英语	高资源	中资源	低资源	全部
XMEF-X	34.2	20.2	5.9	14.7
XLS-R (2B)	36.1	27.7	15.1	22.1
mSLAM-CTC (2B)	37.8	29.6	18.5	24.8
Maestro	**38.2**	31.3	18.4	25.2
零样本 Whisper	36.2	**32.6**	**25.2**	**29.1**

表: 在 CoVoST2 等基准上展现强零样本翻译能力（turbo 不支持翻译）

翻译洞察

- 训练数据与翻译性能的相关性弱于 ASR
- 在低资源语言上表现突出
- 执行 X→ 英语翻译时请选择 multilingual 模型（如 medium/large）


white noise
pub noise
WER on LibriSpeech test-clean (%)
100
50
20
10
5
2
1
40
30
20
10
0
-10
signal-to-noise ratio (dB)
unispeech-sat-base-100h-libri-ft
wav2vec2-base-100h
wav2vec2-base-960h
wav2vec2-large-960h
wav2vec2-large-robust-ft-libri-960h
wav2vec2-large-960h-lv60-self
asr-crdnn-rnnlm-librispeech
asr-transformer-transformerlm-librispeech
hubert-large-ls960-ft
hubert-xlarge-ls960-ft
s2t-medium-librispeech-asr
s2t-large-librispeech-asr
stt_en_conformer_ctc_large
stt_en_conformer_transducer_xlarge
Whisper



人类评估结果

- 多样化评测中接近专业人工转录准确度
- 具体数值受任务、口音、噪声与后处理影响

1. 大规模弱监督预训练覆盖多语言语音-文本对齐数据
2. 展示零样本鲁棒性在多域与多任务上稳定
3. 统一多任务方法用于识别、翻译与语言识别
4. 接近人类水平的多样化场景表现
5. 开放发布模型与代码（MIT License）

主要洞察

弱监督预训练在语音识别中的潜力被低估；多样且充足的数据规模使鲁棒的零样本泛化成为可能。

影响与未来方向

即时影响：

- 面向语音处理的通用基础模型
- 无需微调即可实现鲁棒 ASR
- 覆盖多语言识别、语音翻译与语言识别
- 与商业系统在多域场景中竞争

未来工作：

- 解码策略与时间戳稳定性优化
- 扩充低资源语言与多口音数据
- 微调与外部语言模型的系统化研究
- 轻量化与移动/边缘侧部署优化

代码与模型可用

https://github.com/openai/whisper

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