arXiv:2305.04159v2 [eess.AS] 9 May 2023

关键时刻的前瞻：用于流式神经转导器的自适应非因果 Transformer

Grant P. Strimel [*1] **Yi Xie** [*1] **Brian King** [*1] **Martin Radfar** [1] **Ariya Rastrow** [1] **Athanasios Mouchtaris** [1]

摘要

流式语音识别架构用于低延迟的实时应用。这类架构通常以其因果性为特征。因果架构在每一帧输出词元，仅依赖当前和过去的信号；而非因果模型在每个步骤中都会接触到一个未来帧窗口，以提高预测准确率。这种二分法意味着实时自动语音识别（ASR）系统设计需要权衡：要么利用严格因果架构的低延迟优势，同时接受其预测性能的局限；要么发挥未来上下文模型的建模优势，同时承受更高的延迟惩罚。在这项工作中，我们放宽了这种选择的限制，并提出了自适应非因果注意力转导器（ANCAT）。从传统意义上讲，我们的架构是非因果的，但它通过在音频流中动态选择何时以及在多大程度上依赖未来上下文，以低延迟的流式方式运行。将这一机制与我们新颖的正则化算法结合后，即使显著降低延迟，也能达到与非因果配置相当的准确率，从而缩小与其因果对应架构之间的差距。我们通过实验展示了这一设计，并报告了在公开可用数据集和生产级语音助手数据集上的自动语音识别任务比较结果，其中包含准确率和延迟指标。

1. 引言

现代语音识别应用利用全神经网络架构的优势，推动了用户体验的提升。例如，这些架构使亚马逊 Alexa、Google Assistant 和 Apple Siri 等热门虚拟助手能够依靠稳健的远场语音识别，作为实时交互的主要媒介。同样，通话录音转录和视频字幕生成等离线服务也应用了这些架构。神经转导器、Listen-Attend-Spell 和 Transformer 等主流技术，本质上都是序列到序列模型，由音频编码器将声学输入映射为用于自回归解码的高层表示（Graves，2012；Chan 等，2016；Mohamed 等，2019；Chorowski 等，2015；Han 等，2020）。尽管每种神经网络自动语音识别架构都有各自的设计特点，但一般而言，它们都被归类或实现为因果模型或非因果模型。

因果自动语音识别模型是一种可流式处理的设计，会在音频信号中的每一帧到达时输出预测，而无需访问未来帧（He 等人，2019；Yu 等人，2021a；Radfar 等人，2022）。这些模型之所以称为因果模型，是因为每一帧的预测仅取决于其左侧（过去）的上下文。因果配置尤其适用于实时处理应用，在这类应用中，快速响应对于用户体验至关重要，例如虚拟助手或实时流媒体字幕生成。另一方面，非因果模型可以访问来自未来帧的信息。非因果模型既可以采用流式处理方式，仅使用数量有限的前瞻帧，也可以采用全上下文方式，在开始解码前访问完整的音频信号（Zhang 等人，2020；Moritz 等人，2020；Yeh 等人，2019；Tripathi 等人，2020）。显然，这些模型可以利用未来信息对预测进行整体消歧，从而显著提高准确率，优于相应的因果模型。然而，非因果处理带来的准确率提升通常伴随着实时应用中延迟大幅增加的代价，这可能妨碍此类模型在生产环境中的部署。

本文介绍了自适应非因果注意力转导器（ANCAT），这是一种神经网络自动语音识别架构，兼具非因果模型所展现的准确率提升，同时其延迟更接近流式因果架构。

[*]贡献相同 [1]亚马逊 Alexa，美国。通讯作者：Grant P. Strimel <gsstrime@amazon.com>、Yi Xie <yix-iey@amazon.com>。

论文集： 40th 国际机器学习会议，美国夏威夷州檀香山。PMLR 202，2023。版权所有 © 2023，作者所有。

ANCAT 通过赋予模型灵活性，使其能够在每一帧利用不同长度的未来上下文，从而实现这一提升。不过，模型经过训练后会有选择地决定是否以及何时利用未来上下文；它会同时考虑准确率和延迟方面的影响，仅在准确预测所必需时才查询未来上下文。因此，在整个帧序列上汇总后，模型自适应地摄取前瞻上下文所产生的延迟开销相对较小。我们的模型采用完全端到端的训练方式，在学习传统词元预测目标的同时，联合学习如何以及在何处应用非因果上下文。

在第 2 节概述相关的动机性工作后，本文余下部分将根据我们的贡献组织如下：第 3 节介绍 ANCAT设计及关键架构概念；第 4节推导出为训练我们的架构而开发的新型损失函数，并结合文献中常用的不同延迟概念，对未来上下文进行正则化；第 5 节展示实证结果，验证我们的设计要素，并展现 ANCAT 在开源和工业数据上的流式自动语音识别任务中的建模能力。

2. 相关工作

因果、非因果与可流式处理的自动语音识别。 许多先前研究都致力于兼顾非因果性的准确率优势与因果模型的低延迟部署能力。多项研究（Audhkhasi 等，2021；Moritz 等，2021；Yu 等，2021b）发现，利用非因果右侧上下文的蒸馏技术有助于训练完全因果模型。例如，“双模式”自动语音识别将单个端到端模型与共享权重的全上下文模式联合训练，从而提升因果自动语音识别的准确率（Yu 等，2021b）。

其他方法通过允许摄入有限量的逐帧前瞻上下文来提高流式处理的准确性，但代价是增加延迟。例如，在每一层使用固定的前瞻帧右侧上下文窗口十分常见（Zhang 等人，2020）。分块方法对于实现非因果性也很有效（Tsunoo 等人，2019；Dong 等人，2019；Shi 等人，2021；Chen 等人，2021）。采用分块时，流会被拆分为固定大小的组，每组由彼此相邻且不重叠的帧构成。在一个块内，所有帧都可以访问彼此，也可能访问前面块中的帧。与双模式训练类似，（Swi- etojanski 等人，2022）还将原位蒸馏扩展到可变大小的分块：训练一个模型，同时允许在推理时配置不同的块大小，以匹配硬件规格。与此同时，Scout网络（Wang 等人，2020）利用词边界确定块的位置和大小，并使用一个基于强制对齐数据训练的独立网络来预测边界推理时。

另一类方法通过训练流式编码器和全上下文编码器并将其堆叠起来，将因果和非因果上下文统一到一个系统中。（Narayanan 等人，2021）和（Li 等人，2021）提出了“级联”音频编码器：先使用因果第一遍编码器，再使用基于第一遍编码器输出运行的堆叠式非因果编码器。这些模块通过联合训练来提升准确率，但也接受了全上下文第二遍编码器带来的延迟影响。

自适应神经计算。自适应计算通常也称为可变计算或动态计算，是一种根据每个输入的具体情况，调整模型在推理期间执行的神经计算量的技术。该方法背后的直觉是：由于每个实例具有不同的特征，相应投入的计算量也应体现这种差异，仅在必要时配置更多资源并执行更多操作。研究人员已在各个机器学习领域广泛探索这些思想，包括自然语言处理（Grave，2016；Jernite 等人，2017；Dehghani 等人，2019；Elbayad 等人，2020）、计算机视觉（Bolukbasi 等人，2017；Figurnov 等人，2017）、语音（Macoskey 等人，2021a；b；Xie 等，2022；Peng 等人，2023）和推荐系统（Song 等人，2019）。关于全面的综述，读者可参考（Han 等人，2021）。尽管（Sukhbaatar 等人，2020；Chang 等人，2020）也提出了学习静态注意力跨度调整的方法，但我们的机制将针对流式处理，自适应地跨输入和帧改变跨度。此外，所有自适应技术实际上都会将其计算量与某个特定的成本指标绑定，例如浮点运算次数。对于我们的应用，我们将展示如何把动态计算机制与实时语音识别的关键延迟度量联系起来。

延迟度量。 评估流式自动语音识别架构的延迟存在诸多复杂之处，因此人们提出了多种指标，以概括其不同方面。用户感知延迟（UPL）、端点检测延迟、首个词元发射延迟、算法延迟、平均对齐延迟和部分识别延迟，都是文献中讨论的度量（Shangguan 等人，2021； Sainath 等人， 2020；Inaguma 等人， 2020；Yu 等人， 2021a ）。尽管每种定义都试图捕捉不同的延迟成因，但对于使用非因果上下文的模型，自然适合采用算法延迟。因此，我们也采用这一度量来指导自适应非因果流式架构的设计。算法延迟 表示处理输入并生成音频帧输出所需的时间，即帧长与所用前瞻帧数所对应时间之和（Shi 等人， 2021）。由于我们的算法具有非确定性，我们改为报告该指标的统计版本。我们

还在我们的设计中考虑了一项计算诱导的 *UPL* 指标，详见第4.3节。该指标综合考虑了处理速度以及非因果机制安排的额外计算量，从而得出它们对 UPL 的综合影响。

3. 自适应非因果设计

本文聚焦于经典的神经转导器架构，该架构由三个模块组成：编码器网络、预测网络和联合网络。编码器网络接收从音频信号中提取的一系列 T 特征帧向量（x_1，……，x_T），并将其映射为相应的一系列高级声学表示。预测网络对一系列 U 标签（y_1，……，y_U）进行自回归处理。联合网络结合编码器网络和预测网络的输出，以对下一个发出的标签的似然进行建模。

转导器架构是实时应用中的常见选择，因为编码器网络可以针对流式处理进行训练，使联合网络能够在第 i <style id='4}帧发出预测，同时仅依赖有限数量的未来帧；不过，在非流式场景中，可以采用全上下文编码器来提高准确率。我们的设计采用基于Transformer的编码器，它们由 L 个堆叠模块（又称层，这两个术语在本文中可互换使用）构成。每个模块接收前一个模块的输出，并生成帧 h_i^ℓ 和层 i ℓ对应的隐藏向量。基于Transformer的模块由多种子层组合构成，例如前馈层、层归一化层，以及 Conformer 中的卷积层（Gulati 等人，2020），但都包含多头自注意力（MHSA）子层。

3.1. 计算有向无环图与注意力掩码

基于 Transformer 的注意力架构为每一层的每个输出定义了自然的输入依赖计算图。在这种有向无环图（DAG）表示中，每个顶点 v_i^ℓ（或节点）表示对第 ℓ层的帧 i进行计算，而相邻层顶点之间的有向边 $(v_j^{\ell-1},\ v_i^\ell)$ 表示计算 v_i^ℓ时依赖 $v_j^{\ell-1}$ 的输出；也就是说，计算 h_i^ℓ时会对 $h_j^{\ell-1}$ （或其映射）进行注意力计算。

我们的 ANCAT 架构旨在动态且有策略地填补这一计算 DAG 中的边，同时兼顾准确率和延迟。为此，我们训练传统的编码器权重，同时在架构中引入可联合训练的调度器 $\mathcal{S}^\ell$，每一层 ℓ配备一个。每个调度器 $\mathcal{S}^\ell$ 接收前一层的结果 $h_i^{\ell-1}$，并确定要关注哪些未来的非因果输入，以进行计算 h_i^ℓ。因此，$\mathcal{S}^\ell(h_i^{\ell-1})$ 会预测从顶点 $v_j^{\ell-1}$ 到位置 $j > i$处顶点 v_i^ℓ 的非因果边。

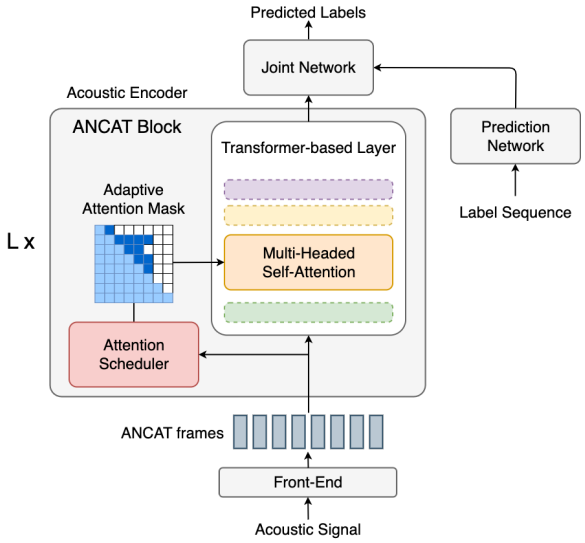

图 *1*。自适应非因果注意力转导器（ANCAT）。该架构是一种神经转导器，其声学编码器由 L 个基于 Transformer 的模块堆叠而成，每层均配有一个注意力调度器。每个调度器都会学习填充右侧上下文注意力连接，并将生成的掩码传递给多头自注意力。

训练期间，我们利用一系列注意力掩码$M^\ell \in [0,1]^{T\times T}$来表示这些边，其中$M_{i,j}^\ell = 1$表示边$(v_j^{\ell-1}, v_i^\ell)$完全存在，0 表示该边不存在：

$$M_{i,j}^\ell = \begin{cases} 1 & j \le i \\ \mathcal{S}^\ell(h_i^{\ell-1})_j & \text{otherwise} \end{cases} \qquad 1 \le i,j \le T$$

掩码应用于计算 MHSA 子层中各注意力头的注意力分数。具体而言，对于特定注意力头，基于查询矩阵 X_{query} 和键矩阵 X_{key} 计算得到的缩放点积值矩阵 P

$$P = \frac{X_{\text{query}} X_{\text{key}}^\top}{\sqrt{d}},$$

掩码后的注意力分数可计算为

$$A_{i,j} = \frac{\exp(P_{i,j}) M_{i,j}^\ell}{\sum_t^T \exp(P_{i,t}) M_{i,t}^\ell} \qquad 1 \le i,j \le T$$

对于层 ℓ 中的每个注意力头，或使用（Xie 等人，2022）的方法。

3.2. 学习式调度器

调度器的作用是填充其对应注意力掩码中向前（即主对角线右侧）的值。为简化学习过程，我们不逐一预测每个

连接各自独立，因此我们将每个调度器 $\mathcal{S}$ 视为针对特定输入，估计其所在层应采用的非因果注意力跨度长度（即需要考虑的未来帧数量）。我们采用一个平滑、可微的反向 S 形函数，该函数从 1 递减至 0，并带有可调参数 τ，用于控制曲线的陡峭程度。调度器学习在 K 个最大前瞻帧的范围内将该曲线的中心 o 移至何处。具体而言，$\mathcal{S}^\ell$ 的计算方式为

$$o_i^\ell = \sigma\left(\text{FFN}^\ell(h_i^{\ell-1})\right)(K+\epsilon)$$
$$\mathcal{S}^\ell(h_i^\ell)_j = \begin{cases} 1-\sigma\left(\left(j-i-o_i^\ell\right)/\tau\right) & j \le K \\ 0 & \text{otherwise,} \end{cases}$$

其中，σ 是标准 Sigmoid 函数，FFN^ℓ 是一个可学习的前馈网络，由两个线性变换组成，中间加入非线性激活函数，第二个变换将结果投影为单个标量。小常数 ϵ 用于使第 K 个前瞻帧有可能获得近乎完整的连接，同时在训练过程中保持数值稳定性。

请注意，该设计允许在学习过程中出现“软边界”，即掩码值介于 0 和 1 之间。不过，可以将 τ 逐渐退火至 0，使所有软边界逐步转变为明确的二值边界。我们的损失函数设计也将利用这一特性。

4. 正则化未来上下文

作为对 ANCAT架构设计的补充，我们设计了多个损失函数，用于对调度器的决策进行正则化，从而明确考虑流式系统中不同的延迟概念。

4.1. 朴素正则化

为了平衡神经转导器的主要训练目标与调度器摄入非因果帧所带来的影响，可以将训练损失函数修改为以下形式。

$$\mathcal{L} = \mathcal{L}_{\text{transducer}} + \lambda \mathcal{L}_{\text{sched}}$$

一种初步尝试是对所有前瞻连接进行正则化，直觉上，在各层中尽可能少地关注未来帧会带来更低的延迟。

$$\mathcal{L}_{\text{L1}} = \frac{1}{T}\sum_\ell \sum_{j>i} M_{i,j}^\ell$$

这种 L1 型正则化将作为我们实验的基线。然而，这种不加区分的正则化方式并未考虑连接对延迟的非局部影响。例如，较低

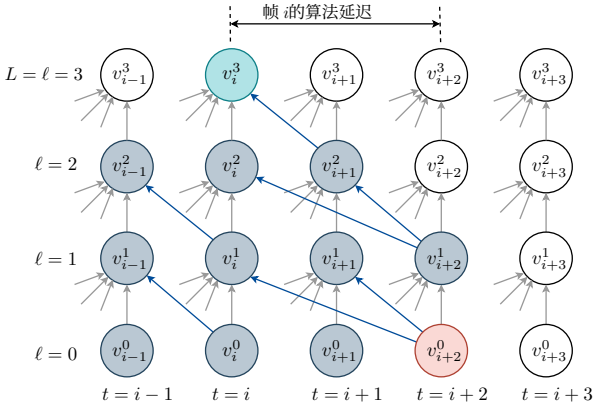

图 2。三层 ANCAT 的计算图示例。非因果边（蓝色）由学习式调度器提供。v_i^L（青色）的依赖节点（灰色）决定算法延迟。最终依赖帧（红色）与 v_i^L 之间的差值 $\delta(i)$ 决定帧 i 的算法延迟。

层中存在的前向边可能会由于逐层传播的效应，显著影响多个帧上的预测延迟。此外，上述简单的正则化并未与明确的延迟度量直接挂钩。

4.2. 算法延迟

这里，我们阐述如何根据算法延迟这一概念进行正则化。回顾一下，传统上，算法延迟指的是模型在某一帧发出符号之前必须等待的最短理论时间，这取决于预测时所依赖的非因果未来帧数量 $\delta(i)$ 以及每个时间步的预测 i 与帧长（该值为常数）。[1]对于标准架构，这一延迟是确定值。相比之下，在我们的 ANCAT架构中，算法延迟可能因帧而异，因此我们采用时间平均值的定义：

$$\text{algorithmic latency} = \frac{1}{T}\sum_i (\text{frame length})\delta(i)$$

下面我们推导一种算法，直接整合由调度器决策产生的算法延迟，并将结果汇总为训练期间应用的正则化损失。

为此，我们首先定义依赖函数的概念。我们定义 $D_{i,j}^\ell \in [0,1]$ 来表示在输入节点 v_j^0 上计算 h_i^ℓ 时的（分数）依赖关系。换言之，$D_{i,j}^\ell$ 表示 v_j^0 是否通过某条路径与 v_i^ℓ 相连

[1]在此计算中纳入“当前”第 i 帧十分常见；不过，为了使我们所采用的设定下的表述更清晰，我们将其省略。但由于这一帧只相当于一个加性常数，因此可以轻松地将任何结果转换为包含它的形式。

该路径经过计算有向无环图。依赖值可以是分数，因为训练期间边将是分数值。

对于 ANCAT 架构，我们提出以下记忆化动态规划公式，利用注意力掩码递归计算每一层的依赖矩阵

$$D^{\ell}_{i,j} = \begin{cases} M^{\ell}_{i,j} & \ell = 1 \\ \max\limits_{t} \; M^{\ell}_{i,t} \cdot D^{\ell-1}_{t,j} & \ell > 1 \end{cases}$$

该算法可视为计算输入帧与计算节点之间路径的最大分数强度。有趣的是，该算法可归约为一个经典的图最短路径问题，其中边权重取自 $v^{\ell-1}_{j}$ 和 v^{ℓ}_{i}，表示为 $-\ln M^{\ell}_{i,\,j}$，最终的$D^{\ell}_{i,\,j}$值则提取为 $\exp\{-$距离$(v^{0}_{j}, v^{\ell}_{i})\}$。

重要的是，对于我们的应用，该算法为每一层生成依赖矩阵 ℓ，其中$D^{\ell}_{i,\,j}$随 j单调递减。以下归纳论证证明这一事实：基础情形显然成立，因为$D^{1}_{i,\,j} = M^{1}_{i,\,j}$，且调度器函数具有单调性。现在令$D^{\ell}_{i,\,j} = M^{\ell}_{i,\,t} \cdot D^{\ell-1}_{t,\,j}$，其中某个 t。由于$D^{\ell-1}_{t,\,j-1} \geq D^{\ell-1}_{t\,j}$，根据归纳假设，至少存在一个 t'，即 $t' = t$，在其上取最大值，使得$D^{\ell}_{i,\,j-1} \geq M^{\ell}_{i,\,t'} \cdot D^{\ell-1}_{t',\,j-1} \geq M^{\ell}_{i,\,t} \cdot D^{\ell-1}_{t,\,j} = D^{\ell}_{i,\,j}$。

由于这种单调性，我们将 $D^{\ell}_{i,\,j}$ 视为在 j 上的近似逆累积分布函数，其中 $\hat{}P$（$v^{\ell}_{i} \leq t$的最后一个依存帧）$= 1 - D^{\ell}_{i,\,t}$。因此，相应的密度函数 $F^{\ell}_{i,\,j} = D^{\ell}_{i,\,j-1} - D^{\ell}_{i,\,j}$为计算节点 v^{ℓ}_{i}所需的最后一个前瞻帧的位置提供了自然的权重。

请注意，对于所有 i，ℓ，确实有 $\sum_{j} F^{\ell}_{i,\,j} = 1$；随着 τ逐渐退火，以产生来自调度器的二值连接，该分布会转变为严格的独热向量编码，指示 v^{ℓ}_{i}的最后一个帧依赖项的确切位置。

现在，我们可以使用此算法直接对话语中所有帧的平均（分数）算法延迟进行正则化

$$\mathcal{L}_{\text{Alg.Lat.}} = \frac{1}{T} \sum_{i} \hat{\delta}(i) = \frac{1}{T} \sum_{j>i} (j - i)\, F^{L}_{i,j}$$

使用 $F^{L}_{i,\,:}$来表示顶层 L在每一帧上最后一帧依赖关系的（权重）。

4.3. 计算诱导的 UPL

虽然算法延迟这一指标能让我们了解流式 ASR 模型的最短延迟，但它仅适用于……

……延迟，但它假设所有计算量都能瞬间完成，因此只能作为模型响应 UPL 的下限。为了更真实地建模

并针对 UPL 进行正则化，我们提出了一种同时考虑处理速度的损失函数。

我们将 μ 表示为有效处理器速度，即每秒可处理的计算节点（编码器层）数；将 ρ 表示为每秒帧数，即特征帧率。我们使用这些常数和第 4.2 节中介绍的算法，计算每个节点在哪个时间步变为可用，从而核算计算积压。最终帧上积压中剩余的工作量（待计算的节点数）将用于正则化 UPL。

令 q_i 表示在帧 i上输入依赖已满足的节点数，则有

$$q_i = \sum_{\ell} \sum_{t<i} F^{\ell}_{t,i}$$

即在 i时要添加到积压中的新增工作量。定义 b_i 为缓冲节点积压（在 i个时间步累积的延迟），采用与（Macoskey 等人，2021b）类似的方法，我们将计算引起的 UPL 损失表示为以秒为单位的响应延迟 $\mathcal{L}_{\text{UPL}} = b_T/\mu$，其中 b_i 通过递归方式计算如下

$$b_i = \begin{cases} 0 & i = 0 \\ \max\{b_{i-1} + q_i - \mu/\rho, 0\} & i > 0 \end{cases}$$

其中 μ/ρ 表示计算吞吐量，即每个时间步能够从积压中处理掉的节点数。图 3 展示了这一 UPL 计算的执行过程。

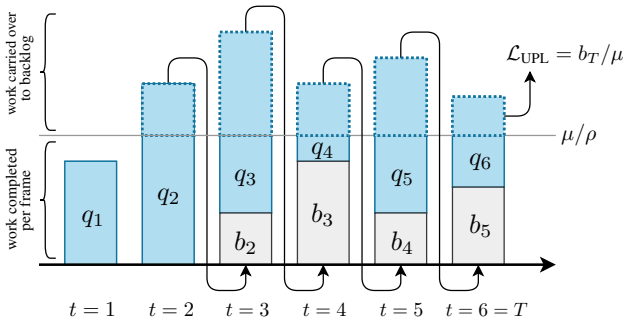

图 3。UPL 算法运行示例。在指定吞吐量下，当前帧无法完成的工作将转移到后续帧。话语结束时剩余的工作量即为 UPL。

5. 实验结果

本节将组织我们的结果，以展示 AN-CAT 在公开 ASR数据上的准确率和算法延迟综合改进，并辅以

表*1*。Conformer 和 T-T 编码器骨干网络在 LibriSpeech test“clean”上的结果。各模型在不同流式设置下进行训练，每层的最大前瞻帧数为 K；同时也报告了全上下文模型。逐层和分块分别表示标准前瞻注意力结构和既有文献中的分块感知注意力方法。采用 L1 正则化和我们的新型损失函数训练的 ANCAT 模型分别标记为 ANCAT-*L1* 和 ANCAT-*Alg.Lat.*。与算法延迟均值相近的最强竞争模型相比，ANCAT-*Alg.Lat.* 的词错误率显著降低；在第50和第90百分位数上也有同样的表现。

	Conformer													
K (# 前瞻)	0	2				5				10				Full
模型	*Causal*	*Layerwise*	*Chunked*	ANCAT-*L1*	ANCAT-*Alg.Lat.*	*Layerwise*	*Chunked*	ANCAT-*L1*	ANCAT-*Alg.Lat.*	*Layerwise*	*Chunked*	ANCAT-*L1*	ANCAT-*Alg.Lat.*	*Full Context*
词错误率 (%)	6.66	4.58	6.27	5.68	**5.31**	4.23	5.68	5.43	**5.13**	4.12	5.15	5.13	**4.76**	4.09
算法延迟（毫秒）	0.00	2240	63.2	60.5	60.1	3240	235	235	202	3520	518	495	498	-
算法延迟@50（毫秒）	0.00	2380	59.9	61.6	58.8	2940	236	223	237	2940	518	488	489	-
算法延迟@90（毫秒）	0.00	2920	62.4	86.4	76.7	5680	241	310	274	6650	541	705	623	-
ℓ_1-norm（帧）	0.00	27.1	6.94	1.78	8.12	65.3	27.3	7.04	16.5	122	60.3	16.2	59.1	-
		Transformer												
词错误率 (%)	6.90	4.88	6.56	5.92	**5.57**	4.63	5.74	5.67	**5.36**	4.42	5.65	5.27	**5.11**	4.55
算法延迟（毫秒）	0.00	2240	63.2	61.2	60.5	3240	235	232	229	3520	518	512	496	-
算法延迟@50（毫秒）	0.00	2380	59.9	60.4	55.8	2940	236	230	228	2940	518	508	484	-
算法延迟@90（毫秒）	0.00	2920	62.4	88.6	80.2	5680	241	269	305	6650	541	701	613	-
ℓ_1-范数（帧）	0.00	27.1	6.94	1.89	7.93	65.3	27.3	5.47	21.5	122	60.3	18.5	36.3	-

对行业数据、模型动态和计算诱导的 UPL 研究结果的总结；完整结果请参阅相应附录。

5.1. LibriSpeech 实验设置

我们使用 LibriSpeech 语料库（Panayotov 等人，2015）研究我们的架构。该语料库包含从有声读物朗读音频中收集的 960 小时训练数据。评估采用相应的“clean”测试数据。音频片段经过预处理，使用 64 维对数滤波器组能量特征提取器提取特征；随后以步长 2 堆叠这些特征向量，并进行 3 倍下采样，再输入一个小型卷积前端进行处理，生成 120 毫秒帧，作为基于 Transformer 的模块的输入。

我们在两种流行的基于 Transformer 的架构上开展实验：Conformer (Gulati 等人，2020) 和 Transformer-Transducer (T-T) (Zhang 等人，2020)。对于 Conformer 和 T-T，我们均采用由 14 个堆叠模块组成的编码器、具有 640 个单元的单层 LSTM 预测网络，以及具有 512 个单元的前馈联合网络。对于 ANCAT 变体，我们为每个模块添加了隐藏单元维度为 64 的可学习调度器。详细模型配置列于附录 G。

对于 LibriSpeech，所有模型均使用基于 Transformer 的编码器骨干网络训练 150 个训练轮次，每轮 1k 步；编码器与调度器使用 Adam 优化器（Kingma 和 Lei Ba，2015），按照（Gulati 等人，2020）中指定的相同超参数设置进行端到端联合训练。在训练过程中，我们还应用了 SpecAugment（Park 等人，2019），掩码参数为 ($F = 27$)，并使用 20 个时间掩码，最大时间掩码比例为 ($p_S = 0.04$)，其中时间掩码的最大尺寸设为 p_S 话语长度的倍数。我们使用包含 4097 个词片段的词汇表，并采用标准 RNN-T 波束搜索解码，波束宽度为 16。模型配置和训练超参数的更多细节见附录 G。

5.2. 基线

我们使用因果和全上下文注意力为 Conformer 和 T-T 模型建立基线。我们还为先前研究提出的非因果流式机制建立基线。这些机制包括在每层接收 K 前瞻帧的流式注意力（Zhang 等人，2020），

此处记为 逐层，以及分块 Transformer 注意力（Shi 等人，2021；Chen 等人，2021），记为 分块，这是一种常用且高效的对比方法。此外，我们还展示了采用基础 L1 型正则化的 ANCAT 模型，该正则化针对未来帧的数量进行约束，如第 4.1 节所述，标记为 ANCAT-*L1*。

5.3. 算法延迟结果

我们的主要实证结果表明，采用新型正则化损失进行训练后，ANCAT 在延迟-准确率权衡曲线的每个工作点上都取得了显著改进，从而推动了现有方法的帕累托前沿。具体而言，对于每个固定的准确率目标，我们的 ANCAT 模型相比最接近的基线显著降低了算法延迟；反过来，对于给定的算法延迟，ANCAT 也能显著提升准确率。我们通过在不同的超参数设置下训练基线模型和 ANCAT模型来验证这一结果，其中最大每层前瞻跨度为 K （即 2、5 和 10）。我们报告测试语句的词错误率（WER）（%）和统计算法延迟（毫秒）。此外，我们还报告了算法延迟中位数和第90百分位数指标下的延迟，分别记为算法延迟@50和算法延迟@90。我们还纳入了注意力图中可访问的未来帧的ℓ_1-范数，用于表征前向注意力计算总量的额外增加（相对于因果注意力的额外计算）。聚合均值均取自测试集中的所有话语。

LibriSpeech 的结果列于表1中。对于给定的 K，非自适应模型的算法延迟值是固定的，但可以通过为正则化项 λ 设置不同的惩罚系数来调节 ANCAT模型的词错误率与延迟权衡。为清晰比较表1中的词错误率，我们选择 λ 相应的值，使得在每种设置下， K 算法延迟大致与效率最高的可比基线相匹配。可以看到，使用我们提出的 ANCAT，并采用新颖的算法延迟损失作为正则化项进行训练，ANCAT- 算法延迟在相同延迟工作点下始终比其他方法取得更低的词错误率。这一结论对 Conformer 和 T-T 编码器以及所有设置 K 均成立：与非自适应模型相比，平均词错误率提升了 11%（在较低延迟场景下最高可达 18%）。还可以观察到，与朴素的 L1 正则化相比，在给定算法延迟预算下，我们的 ANCAT-算法延迟模型能够基于 ℓ_1-范数度量实现更多的未来帧注意力。因此，尽管 ANCAT-算法延迟总体上使用了更高的计算量，但其计算分配策略更加合理，能够在匹配延迟的同时实现更低的词错误率。

为了进一步凸显权衡方面的改进， be-

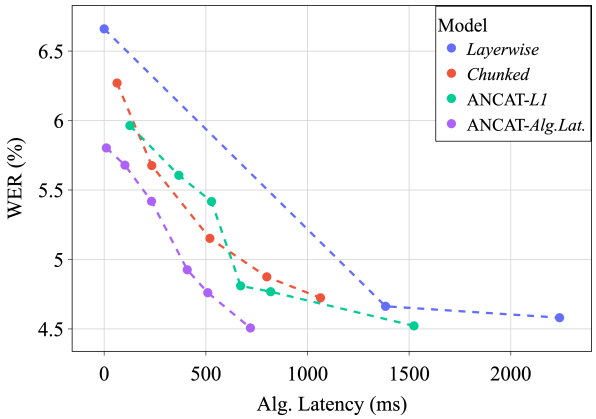

图 4。 在 LibriSpeech 测试集“clean”数据上，四种不同类型的 Conformer 模型的词错误率与算法延迟：逐层 （每层的右侧上下文为 K 个帧，分别为 0、1 和 2），分块 （K 个帧，分别为 2、5、10、15 和 20），ANCAT-*L1* （$K = 10$）改变 λ，以及 ANCAT-算法延迟。（ $K = 10$ ）改变 λ。ANCAT- 算法延迟。 相比其他模型提供了更优的准确率-延迟权衡。

WER 与算法延迟之间的权衡。我们在图 4 中记录了不同训练正则化程度下 Conformer ANCAT 模型的性能。图中显示，ANCAT- 算法延迟 在每个工作点上始终优于所有其他架构，因此完全定义了高效解决方案的帕累托前沿，为所考虑的模型提供了最优的权衡。此外，图中强调，仅靠非因果自适应性并不能 带来这种改进，因为 ANCAT-*L1* 的表现与静态 分块 性能相近；相反，ANCAT 自适应性与恰当选择我们提出的新型正则化方法相结合，对于该方法在延迟约束下取得成功至关重要。

为了进一步了解我们提出的算法延迟损失的行为及其对调度的影响，我们在图中可视化了测试集中的一段话语的注意力掩码5，并与静态方法进行比较。注意力掩码是在最终训练周期生成的，其中 $\tau = 1e-4$。较暗的区域表示没有注意力。我们展示了 Conformer 模型中第 3 个块和第 13 个块的映射。可以观察到，ANCAT-算法延迟 学习以步进模式构建其注意力结构，根据输入以不同大小和位置的块策略有效运行。相比之下，ANCAT-*L1*的注意力模式呈锯齿状，更强调调度器的局部决策，而不是 ANCAT- 算法延迟 所做的协调性更好的决策，后者考虑了对延迟的全局影响。我们还展示了一个采用更高延迟惩罚进行训练的 ANCAT 模型的掩码，以说明其对未来注意力连接的关闭切换更加明显，接近完全因果模型。

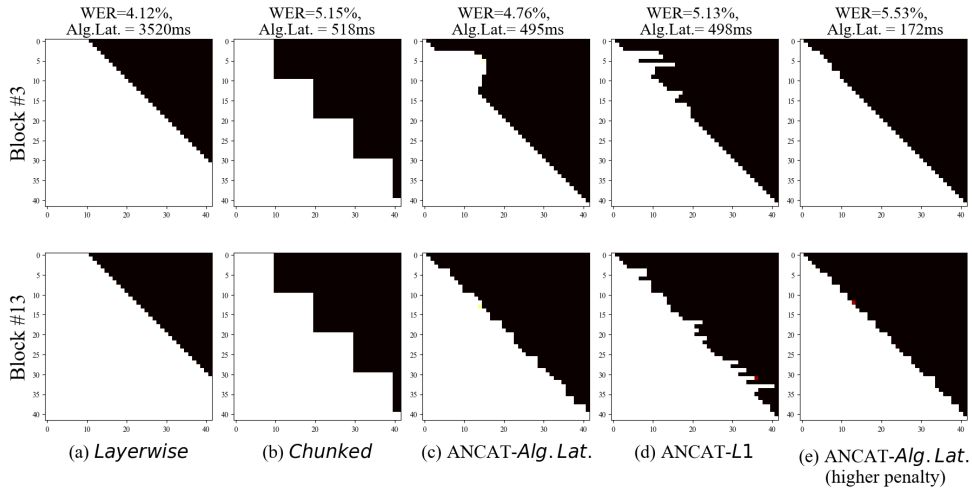

图 *5.* Conformer 模型第 3 个和第 13 个模块的注意力掩码可视化。所有注意力掩码均使用同一个测试示例语句生成。该语句经过前端下采样后共有 41 帧。最大前瞻帧数为 $K = 10$。较暗区域表示注意力已关闭。我们选择了算法延迟性能相近的 (a)、(b) 和 (c) 模型，而 (d) 展示的是一个经过优化、具有较高延迟惩罚的模型。请注意，与 L1 正则化产生的锯齿状调度 (d) 相比，ANCAT-*Alg.Lat.* 产生了刻意设计、结构化的阶梯式模式 (c,e)。

5.4. 工业语音助手

我们在一个去标识化的大型工业语音助手数据集上，针对 Conformer 和 T-T 重复进行了算法延迟实验，以验证我们的建模方法在不同架构和数据组成下的稳健性。我们发现，与 LibriSpeech 的结果类似，ANCAT- 算法延迟 在每个工作点上始终优于 逐层、分块以及 ANCAT-*L1* 模型，并且词错误率通常能相对改善超过 5% 。附录C 详细介绍了这一类似的实验设置及其结果。

5.5. 计算诱导的 UPL

除了将算法延迟损失用作正则化方法外，我们还针对计算诱导的 UPL 进行正则化实验，涵盖 Lib- riSpeech 和工业数据。结果见附录 D ，与算法延迟方面的发现一致：在第4.3 节介绍的采用积压延迟方法进行正则化的 ANCAT，表现优于基线。我们发现，在所有进行实验的计算吞吐量设置下，对于给定的 UPL 预算，ANCAT 都能达到更高的准确率；通常，ANCAT-*UPL* 相较于最接近的基线，使 LibriSpeech 上的词错误率相对降低 8% 以上，工业数据上降低 7%。

5.6. 补充发现与分析

我们在后续附录中提供了该方法的补充可视化结果和观察。附录A展示了与其他常见延迟指标进行比较时，ANCAT 相较于基线能带来更优的性能。附录B 表明 ANCAT 的表现也很好

(与基线相比，词错误率（WER）相对改善 10%）在使用 LibriSpeech“other”数据和加性噪声的更具挑战性的条件下也能实现。我们进一步展示了话语的难度与所采用的前瞻程度之间的相关性。附录 E 重点介绍并扩展了更多注意力图示例，而附录 F 则展示了 ANCAT 的特性如何通过温度退火在整个训练过程中不断演变。图中表明，我们的“软”注意力连接和延迟度量在采用二值边运行时延迟计算的情况下，能够在整个训练过程中平稳收敛。

6. 结论

在这项工作中，我们提出了一种用于流式语音识别的自适应非因果架构。该模型能够动态调整音频流中每一帧的未来上下文注意力跨度，兼顾准确率与延迟。我们在构建架构的同时，还设计了新颖的正则化损失函数，将模型的逐帧前瞻决策与语音应用中重要的延迟度量关联起来。与基线相比，该机制带来了更优的帕累托权衡前沿；在延迟相当的情况下，许多场景中的相对词错误率提升超过 15%。我们在公开数据集、大型生产数据集以及不同架构上开展的实验，进一步验证了该方法的稳健性和适用性。我们希望未来的工作能基于这些贡献，针对其他应用、延迟度量和计算模型提出自适应非因果方法。

作者谨此感谢 Gautam Tiwari 提供的专业指导和建议。

参考文献

Audhkhasi, K., Chen, T., Ramabhadran, B., 和 Moreno, P. J. 混合模型注意力：灵活的流式与非流式自动语音识别。国际语音通信协会年会（*INTERSPEECH*）论文集，第 4096–4100 页，2021。

Bolukbasi, T., Wang, J., Dekel, O., and Saligrama, V. 用于高效推理的自适应神经网络。国际机器学习会议（*ICML*），第 812–821 页，2017 年。

Chan, W., Jaitly, N., Le, Q. V., and Vinyals, O. 听、关注并拼写。*IEEE*声学、语音与信号处理国际会议（*ICASSP*），第 4960–4964 页，2016 年。Chang, X., Subramanian, A. S., Guo, P., Watanabe, S., Fu- jita, Y., and Omachi, M. 使用自适应跨度自注意力的端到端自动语音识别。国际语音通信协会年会论文集（*INTERSPEECH*），第 3595–3599 页，2020 年。Chen, X., Wu, Y., Wang, Z., Liu, S., and Li, J. 开发用于大规模数据集语音识别的实时流式 Transformer 转导器。*IEEE*声学、语音与信号处理国际会议（*ICASSP*），第 5904–5908 页，2021 年。Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. 基于注意力的语音识别模型。神经信息处理系统进展（*NeurIPS*），第 577–585 页，2015 年。Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, Ł. 通用 Transformer。国际学习表征会议（*ICLR*），第 1–23 页，2019 年。Dong, L., Wang, F., and Xu, B. 自注意力对齐器：一种使用自注意力网络和块跳跃的低延迟端到端自动语音识别模型。*IEEE*声学、语音与信号处理国际会议（*ICASSP*），第 5656–5660 页，2019 年。Elbayad, M., Gu, J., Grave, E., and Auli, M. 深度自适应 Transformer。国际学习表征会议（*ICLR*），2020 年。Figurnov, M., Collins, M. D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., and Salakhutdinov, R. 残差网络的空间自适应计算时间。*IEEE*计算机视觉与模式识别会议（*CVPR*），第 1790–1799 页，2017 年。

Graves, A. 使用循环神经网络进行序列转导。国际机器学习会议（*ICML*），2012。Graves, A. 循环神经网络的自适应计算时间。*arXiv* 预印本 *arXiv:1603.08983*，2016。Gulati, A., Qin, J., Chiu, C. C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., 和 Pang, R. Con-former：用于语音识别的卷积增强型 Transformer。国际语音通信协会年会论文集（*INTERSPEECH*），第 5036–5040 页，2020。Han, W., Zhang, Z., Zhang, Y., Yu, J., Chiu, C. C., Qin, J., Gulati, A., Pang, R., 和 Wu, Y. ContextNet：利用全局上下文改进用于自动语音识别的卷积神经网络。国际语音通信协会年会论文集（*INTERSPEECH*），第 3610–3614 页，2020。Han, Y., Huang, G., Song, S., Yang, L., Wang, H., 和 Wang, Y. 动态神经网络：综述。*IEEE* 模式分析与机器智能汇刊，44 (11):7436–7456，2021。He, Y. 和 al., E. 面向移动设备的流式端到端语音识别。*IEEE*声学、语音与信号处理国际会议（*ICASSP*），第 6381–6385 页，2019。Inaguma, H., Gaur, Y., Lu, L., Li, J., 和 Gong, Y. 面向流式序列到序列自动语音识别的最小延迟训练策略。*IEEE*声学、语音与信号处理国际会议（*ICASSP*），第 6064–6068 页，2020。Jernite, Y., Grave, E., Joulin, A., 和 Mikolov, T. 循环神经网络中的可变计算。国际学习表征会议（*ICLR*），第 1–12 页，2017。Kingma, D. P. 和 Lei Ba, J. Adam：一种随机优化方法。国际学习表征会议（*ICLR*），2015。Li, B., Gulati, A., Yu, J., Sainath, T. N., Chiu, C. C., Narayanan, A., Chang, S. Y., Pang, R., He, Y., Qin, J., Han, W., Liang, Q., Zhang, Y., Strohman, T., 和 Wu, Y. 一种更好、更快的流式自动语音识别端到端模型。*IEEE*声学、语音与信号处理国际会议（*ICASSP*），第 5634–5638 页，2021。Macoskey, J., Strimel, G. P., 和 Rastrow, A. 双焦点神经自动语音识别：利用关键词检测优化推理。*IEEE*声学、语音与信号处理国际会议（*ICASSP*），第 5999–6003 页，2021a。

Macoskey, J., Strimel, G. P., Su, J., 和 Rastrow, A. 面向低延迟语音识别的摊销神经网络。国际语音通信协会年会论文集（*INTERSPEECH*），第 1868–1872 页，2021b。

Mohamed, A., Okhonko, D., andZettlemoyer, L. 用于自动语音识别的卷积上下文 Transformer。*arXiv* 预印本 *arXiv:1904.11660*，2019。

Moritz, N., Hori, T., and Roux, J. L. 使用 Transformer 模型进行流式自动语音识别。*IEEE*声学、语音与信号处理国际会议 *(ICASSP)*，第 6074–6078 页，2020。

Moritz, N., Hori, T. 和 Le Roux, J. 用于流式端到端语音识别的双重因果/非因果自注意力。国际语音通信协会年会（*INTERSPEECH*）论文集，第 4116–4120 页，2021。Narayanan, A., Sainath, T. N., Pang, R., Yu, J., Chiu, C. C., Prabhavalkar, R., Variani, E. 和 Strohman, T. 用于统一流式与非流式自动语音识别的级联编码器。*IEEE*声学、语音与信号处理国际会议（*ICASSP*），第 5629–5633 页，2021。Panayotov, V., Chen, G., Povey, D. 和 Khudanpur, S. LibriSpeech：基于公有领域有声读物的自动语音识别语料库。*IEEE*声学、语音与信号处理国际会议（*ICASSP*），第 5206–5210 页，2015。Park, D. S., Chan, W., Zhang, Y., Chiu, C.-c., Zoph, B., Cubuk, E. D. 和 Le, Q. V. Specaugment：一种用于自动语音识别的简单数据增强方法。*arXiv* 预印本 *arXiv:1904.08779*，2019。Peng, Y., Lee, J. 和 Watanabe, S. I3D：具有输入依赖动态深度的语音识别 Transformer 架构。*arXiv* 预印本 *arXiv:2303.07624*，2023。Radfar, M., Barnwal, R., Swaminathan, R. V., Chang, F. J., Strimel, G. P., Susanj, N. 和 Mouchtaris, A. ConvRNN-T：用于流式语音识别的卷积增强循环神经网络转导器。国际语音通信协会年会（*INTERSPEECH*）论文集，第 4431–4435 页，2022。Sainath, T. N. 等. 一种流式端侧端到端模型，其质量和延迟均超越服务器端传统模型。*IEEE*声学、语音与信号处理国际会议（*ICASSP*），第 6059–6063 页，2020。

Shangguan, Y., Prabhavalkar, R., Su, H., Mahadeokar, J., Shi, Y., Zhou, J., Wu, C., Le, D., Kalinli, O., Fuegen, C., 和 Seltzer, M. L. 剖析端侧端到端语音识别的用户感知延迟。*arXiv* 预印本 *arXiv:2104.02207*，2021年。

Shi, Y., Wang, Y., Wu, C., Yeh, C.-F., Chan, J., Zhang, F., Le, D., and Seltzer, M. Emformer：用于低延迟流式语音识别的高效基于记忆 Transformer 的声学模型。*IEEE*声学、语音与信号处理国际会议 *(ICASSP)*，第 6783–6787 页，2021。

Song, W., Charlin, L., Xiao, Z., Zhang, M., Wang, Y., and Tang, J. 通过动态图注意力网络进行基于会话的社交推荐。第十二届*ACM*网页搜索与数据挖掘国际会议（*WSDM*），第 555–563 页，2019。Sukhbaatar, S., Grave, E., Bojanowski, P., and Joulin, A. Transformer 中的自适应注意力跨度。计算语言学协会年会（*ACL*），第 331–335 页，2020。Swietojanski, P., Braun, S., Can, D., da Silva, T. F., Ghoshal, A., Hori, T., Hsiao, R., Mason, H., McDermott, E., Silovsky, H., Travadi, R., and Zhuang, X. 用于可配置变换器转导器语音识别的可变注意力掩码。*arXiv* 预印本 *arXiv:2211.01438*，2022。Tripathi, A., Kim, J., Zhang, Q., Lu, H., and Sak, H. Transformer 转导器：统一流式与非流式语音识别的单一模型。*arXiv* 预印本 *arXiv:2010.03192*，2020。Tsunoo, E., Kashiwagi, Y., Kumakura, T., and Watanabe, S. 探索在线端到端变换器自动语音识别。*arXiv* 预印本 *arXiv:1910.11871*，2019。Wang, C., Wu, Y., Lu, L., Liu, S., Li, J., Ye, G., and Zhou, M. 利用侦察网络实现低延迟端到端流式语音识别。国际语音通信协会年会（*INTERSPEECH*），第 2112–2116 页，2020。Xie, Y., Macoskey, J., Radfar, M., Chang, F. J., King, B., Rastrow, A., Mouchtaris, A., and Strimel, G. P. 用于流式自动语音识别的计算成本摊销变换器。国际语音通信协会年会（*INTERSPEECH*），第 3043–3047 页，2022。Yeh, C.-F., Mahadeokar, J., Kalgaonkar, K., Wang, Y., Le, D., Jain, M., Schubert, K., Fuegen, C., and Seltzer, M. L. Transformer-转导器：利用自注意力实现端到端语音识别。*arXiv* 预印本 *arXiv:1910.12977*，2019。

Yu, J., Chiu, C.-C., Li, B., Chang, S.-y., Sainath, T. N., He, Y., Narayanan, A., Han, W., Gulati, A., Wu, Y. 和 Pang, R. FastEmit：采用序列级发射正则化的低延迟流式自动语音识别。*IEEE* 国际声学、语音与信号处理会议（*ICASSP*），第 6004–6008 页，2021a。

Yu, J.、Han, W.、Gulati, A.、Chiu, C.-C.、Li, B.、Sainath, T. N.、Wu, Y. 和 Pang, R. 双模式自动语音识别：通过全上下文建模统一并改进流式自动语音识别。国际学习表征会议 *(ICLR)*，2021b。

Zhang, Q.、Lu, H.、Sak, H.、Tripathi, A.、Mcdermott, E.、Koo, S. 和Kumar, S. Transformer 转导器：一种采用 Transformer 编码器和 RNN-T 损失的流式语音识别模型。*IEEE*声学、语音与信号处理国际会议 *(ICASSP)*，第 7829–7833 页，2020 年。

A. 发射延迟和端点检测延迟

为了更全面地研究 ANCAT 架构所带来的延迟性能提升，我们在本节中对额外的发射/转录（EM）延迟和端点检测（EP）延迟进行了测量。EM.Latency 指解码生成某个词元的帧（计入未来前瞻）与根据对齐数据确定的该词元实际发音时间之间的平均差值。EP.Latency 特指语音结束词元发射时间与最后一个词实际发音时间之间的差值（同样计入未来前瞻）。在表 2 中，数值表明 EM.Latency 和 EP.Latency 达到了与最佳基线相当（或更好）的水平，而我们提出的 ANCAT-算法延迟的词错误率更优。

表2. Conformer 在 LibriSpeech 测试集“clean”上的结果。本表中的模型与论文正文表1中的模型相同。本表报告了平均发射/转录（EM）延迟和端点检测延迟（EP）。

	Conformer												
K (# 前瞻)	0	2				5				10			
模型	*Causal*	*Layerwise*	*Chunked*	ANCAT-*L1*	ANCAT-*Alg.Lat.*	*Layerwise*	*Chunked*	ANCAT-*L1*	ANCAT-*Alg.Lat.*	*Layerwise*	*Chunked*	ANCAT-*L1*	ANCAT-*Alg.Lat.*
词错误率（%）	6.66	4.58	5.68	6.27	**5.31**	4.23	5.68	5.43	**5.13**	4.12	5.15	5.13	**4.76**
EM.延迟（毫秒）	235	2130	305	296	**290**	3125	395	315	**282**	3418	622	560	**551**
EP.延迟（毫秒）	74	1992	141	116	**116**	2992	252	172	**139**	3286	490	402	**408**

B. LibriSpeech“其他”数据与噪声的影响

LibriSpeech 语料库还提供了更具挑战性的测试集“other”，其中包含从更具挑战性的说话人群体中选取的 5.1 小时语音信号。我们的 ANCAT 和基线模型的结果如图 6 所示。除了所有模型的错误率自然更高之外，我们还观察到，其图表与测试集“clean”的图表非常相似，其中 ANCAT-算法延迟提供了最优的准确率-延迟权衡。同样，ANCAT-算法延迟 在所考察的模型中，界定了算法延迟与词错误率之间的帕累托前沿，同时在若干工作点上实现了高达 10% 以上的相对词错误率改善。

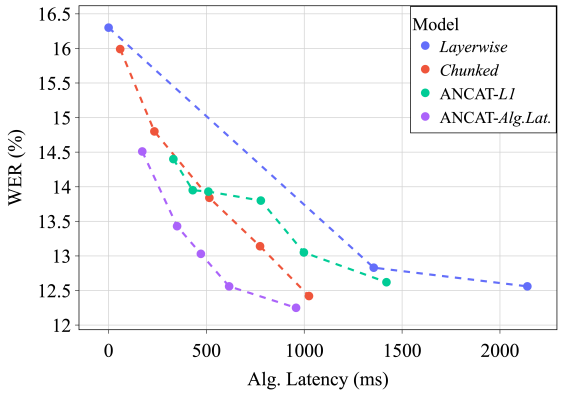

图 6。在 LibriSpeech 测试“其他”数据上，不同 Conformer 模型类型的词错误率与算法延迟的关系：逐层 （每层的右侧上下文为 K0、1 和 2 帧）、分块 （K 2、5、10、15 和 20 帧）、ANCAT-*L1* （$K = 10$）改变 λ，以及 ANCAT-算法延迟。 （$K = 10$）改变 λ。与其他模型相比，ANCAT-算法延迟。 实现了更优的准确率-延迟权衡。

可以观察到，虽然测试集“other”数据的词错误率有所上升，算法延迟也显著增加。因此，我们考察了给定 ANCAT 模型的词错误率与算法延迟之间的关系。自然地，更难的话语（产生更高的词错误率）是否与 ANCAT 调度器导致的算法延迟增加相关？图7（左）对此给出了肯定的回答。该可视化通过将测试集“clean”和“other”中所有单条话语的词错误率划分为五个百分位数组来构建。随后，将每组的词错误率与其平均延迟进行比较，以观察两者之间的正相关关系。这一发现表明，对于更具挑战性的话语，ANCAT 确实会采用更多前瞻注意力。我们还在图7 （右）展示了一项简单实验：对 LibriSpeech 测试集“clean”数据添加不同程度的噪声，并呈现延迟与词错误率/噪声之间的平滑趋势。这一发现支持了这样一种观点：在更具挑战性的条件下，ANCAT-算法延迟 模型会选择采用更多前瞻注意力，以弥补增加的模糊性。

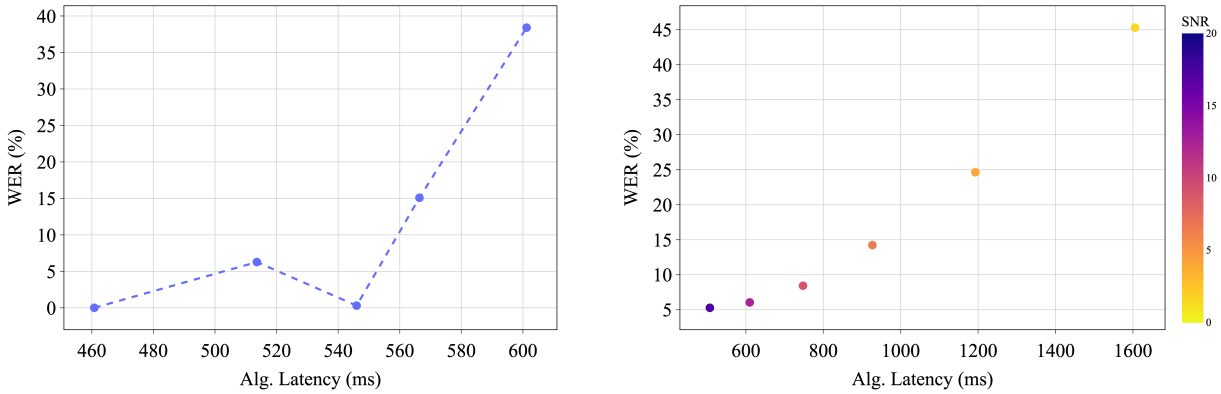

图 7。单个 ANCAT-算法延迟 模型的词错误率与算法延迟。左侧子图将 LibriSpeech 测试集中的“clean”和“other”话语按各自的词错误率分为五个百分位数组，并绘制每组的汇总词错误率与平均延迟的关系。右侧子图展示同一模型在测试集“clean”数据上的表现，其中音频信号叠加了不同强度的白噪声。话语难度与所采用的前瞻量之间呈现明显趋势：越具挑战性的样例，延迟越高。

C. 语音助手数据

本节讨论语音助手数据的实验设置，以及与 LibriSpeech 结果相对应的算法延迟结果。后续附录将展示更多发现和图表，涵盖 LibriSpeech 和语音助手实验。

C.1. 实验设置

我们的内部数据集由去标识化的远场语音助手话语组成。训练数据集包含约 15 万小时的转录音频。我们使用的模型架构与为 LibriSpeech 构建的架构相同，只在配置上做了几处调整，使其更适合语音助手数据。具体来说，我们使用了稍大的词片段分词模型，大小为 4k；两层卷积前端的步幅分别为 2 和 1（得到 60 毫秒的 ANCAT 帧），而 LibriSpeech 使用的步幅分别为 2 和 2。对于内部数据，我们报告的所有准确率指标均为相对于基线全因果模型（绝对词错误率低于 7.5%）的词错误率相对值（WERR）。

C.2. 算法延迟

以下各图展示了与算法延迟相关的 Conformer 结果。我们还在附录H 中附上了最终表格，以供参考。图8 展示了不同类型的模型和不同正则化因子下，词错误率与算法延迟的变化情况。与图4的处理方式相同，我们调整静态模型逐层和分块的前瞻跨度，同时将 K 固定为 ANCAT模型每层 10 帧，并调整其正则化惩罚。这些结果与 ANCAT-算法延迟在 LibriSpeech 上的结果非常吻合，后者提供了最优的准确率—延迟工作点。在许多情况下，对于相同的延迟预算，ANCAT-算法延迟在词错误率上比最接近的竞争对手相对低 5%。同样可以发现，ANCAT-*L1* 的表现与分块相当，并且

这表明，与更朴素的正则化方案相比，新颖的延迟正则化算法对自适应方法的成功至关重要。

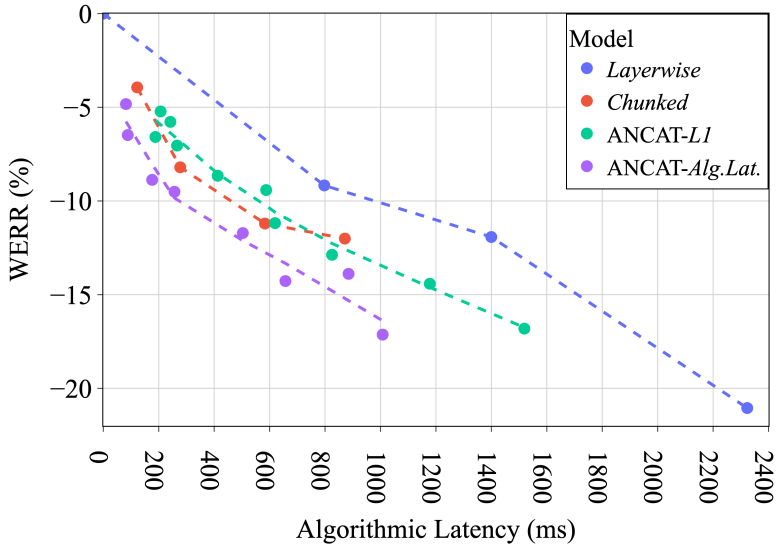

图 *8.* 四种不同类型 Conformer 模型在语音助手数据上的 WERR 与算法延迟的关系：1）逐层 （K 每层右侧上下文分别为 0、1、2 和 10 帧），分块 （K 分别为 5、10、20 和 30 帧），ANCAT-*L1* （$K = 10$改变正则化项 λ），以及 ANCAT-算法延迟 （$K = 10$改变 λ）。

为了补充上文的图 8 并突出 ANCAT 的可调性，图 9 展示了 WER 和延迟随正则化常数 λ 在 Conformer ANCAT-算法延迟 模型在语音助手数据上的变化情况。从图中可以看出，WERR 和延迟曲线随权重变化相对平滑，因此很容易选择所需的准确率-延迟工作点。最后请注意，正则化程度从零到较高的 ANCAT 模型确实接近预期的工作点：高正则化时延迟接近于零，而低正则化时准确率提升显著，与最大右侧上下文模型的表现相近。

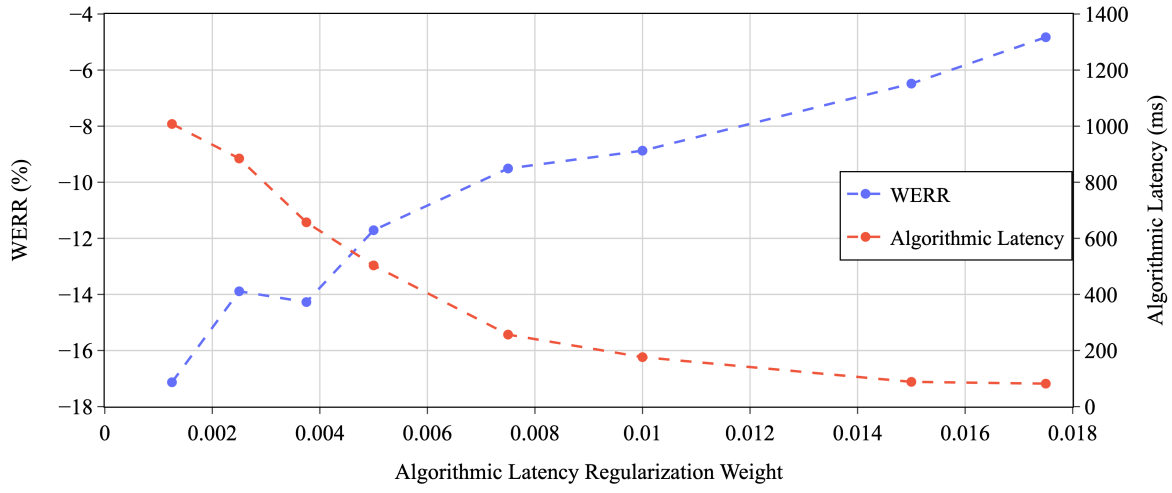

图 *9.* Conformer ANCAT-算法延迟 模型在语音助手数据上的 WERR 和算法延迟与 ANCAT 正则化权重 λ 的关系。

D. 计算诱导的 UPL 结果

章节 4.3 提出了一种新颖的正则化方法，同时考虑执行硬件的处理能力和调度器的决策，以降低应用在使用 ANCAT 模型时可能预期的近似 UPL。基于这一延迟概念，我们使用 LibriSpeech 和语音助手数据进行实验，构建经过 UPL 正则化的 ANCAT-*UPL* 模型。对于采用 14 个块的模型架构和帧率 ρ，实验考察了以下计算吞吐量值： μ 1.5、2 和 3（$\times 14\rho$）。这些数值表示计算 DAG 中每秒可处理的节点数，直观来说，相当于模型每帧进行 1.5、2 和 3 次前向传播的速度（即实时倍数）。

在图 10 和 11中，我们可以看到，不同正则化权重和计算吞吐量下，各模型的 WER 和计算 UPL 如何变化。对于固定的吞吐量，随着正则化减弱，UPL 平滑上升，这与我们的算法延迟结果类似。此外，我们发现，对于 LibriSpeech 和语音助手，ANCAT-*UPL* 在准确率与延迟之间实现了显著优于其他方法的权衡。ANCAT-*UPL*在每个工作点上都始终优于其他架构，并完整地定义了高效解的帕累托前沿。对于固定的 UPL 预算，ANCAT-*UPL* 通常能使 LibriSpeech 上的 WER 相较于最接近的竞争方法降低超过 8%，并使语音助手上的 WER 降低 7%。

这些图还表明，为 ANCAT-*UPL* 模型提供更多计算能力，自然能在 UPL 和词错误率之间实现更好的权衡。更高的吞吐量使模型能够更大程度地应用前瞻注意力，同时不会付出过高的延迟代价，因为一旦计算资源可用（所有 DAG 依赖关系均已满足），执行速度就会更快。此外，根据这一定义的延迟，只要每段话语最终帧上剩余的计算积压极少，模型就可以按需关注尽可能多的未来上下文，前提是能在话语结束前完成这部分工作。正如图中所示，模型获得的计算吞吐量越高，就越能利用这一特性来改善词错误率。

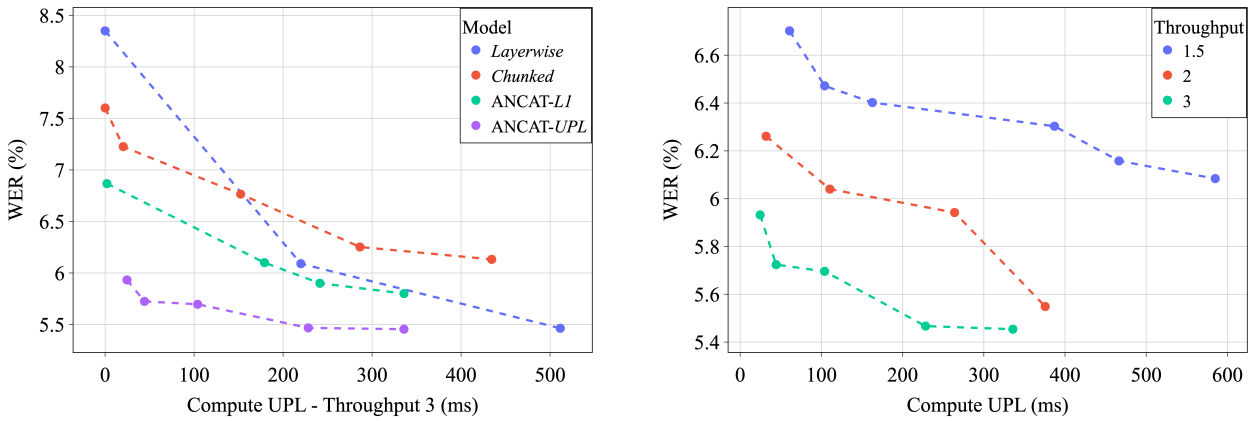

图 *10.* LibriSpeech 测试集“clean”：计算 UPL 与 WER。（左）在固定吞吐量下，根据计算积压得出的 UPL，对比四种不同类型的 Conformer 模型：1）逐层 注意力模型（K 每层右侧上下文为 0、1 和 2 帧），2）分块（K 每层右侧上下文为 2、5、10、15 和 20 帧）、ANCAT-*L1*，以及 4）ANCAT-*UPL*，其中 ANCAT 模型具有 $K = 10$，且正则化项 λ各不相同。就感知延迟而言，ANCAT-*UPL* 实现了更优的准确率-延迟权衡。（右）ANCAT-*UPL* 模型在固定正则化因子下训练，但计算吞吐量各异。更快的计算能力为模型提供了更大的前瞻灵活性，从而使 ANCAT-*UPL* 模型获得更优的准确率-延迟工作点。根据我们的实证观察，UPL 损失对时间掩码模式十分敏感，且容易在小型数据集上过拟合。因此，对于 UPL 结果，我们仅对 SpecAugmentation 应用频率掩码，这使得 WER 略高于我们其他 LibriSpeech 实验中的结果。

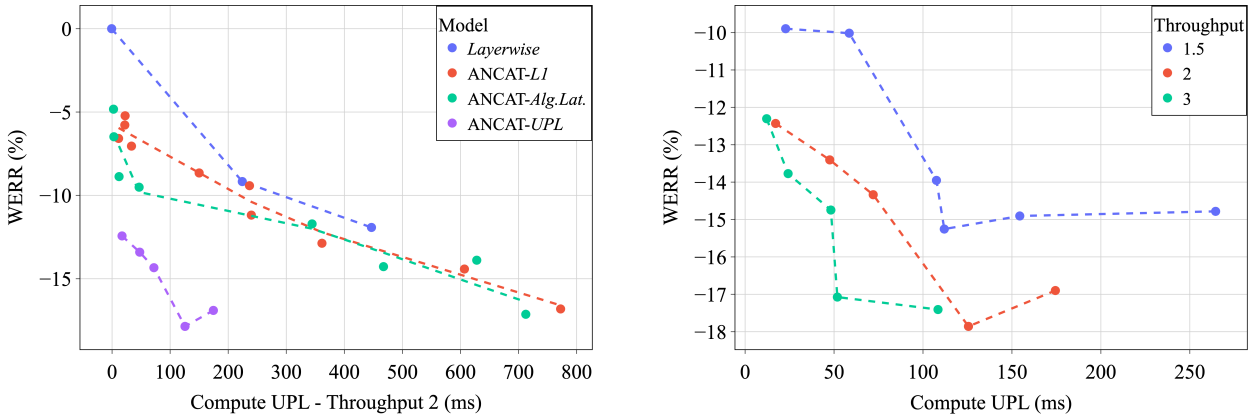

图 *11*。语音助手：计算量 UPL 与 WERR。（左）在固定吞吐量下，基于计算积压的 UPL，对比四种不同类型的 Conformer 模型：1 K 0, 1, 2 2 *1*,) 3 4 $K = 10$ λ 注意力模型（每层具有 at 和帧的右侧上下文），) ANCAT-*L*) ANCAT-算法延迟，以及) ANCAT-*UPL*，其中 ANCAT 模型的正则化项各不相同。就感知延迟而言，ANCAT-*UPL* 提供了更优的准确率-延迟权衡。（右）ANCAT-*UPL* 模型采用固定的正则化因子训练，但吞吐量各不相同。更快的计算能力为模型提供了更大的前瞻灵活性，从而使 ANCAT-*UPL* 模型取得更好的准确率-延迟工作点。

E. 注意力动态

图 12展示了一个示例话语的逐帧延迟度量，同时，图 13 和 14也展示了 LibriSpeech 和语音助手数据示例中所有层的完整注意力掩码。图 12显示，话语中每一帧的算法延迟都可能有很大差异；在模型置信度较低、需要更多未来上下文的位置，会出现峰值。我们还观察到，在整个话语中，计算积压的累积与消减也出现了类似的峰值。不过请注意，积压的上升和下降过程始终较为平稳，因为随着中间层节点的依赖关系得到满足，节点完成进度可以更稳定地推进。还要注意，在这个包含 46 帧（时长 2.76 秒）的示例中，由于话语结束后不应再有前向注意力，因此在话语边界及其之后，算法延迟理应为 0。同样，由于计算积压表示剩余工作量，且最终帧处的积压不为零，因此该话语会产生正的 UPL。我们的 UPL 损失算法旨在最小化话语边界处（帧索引 45）的积压。

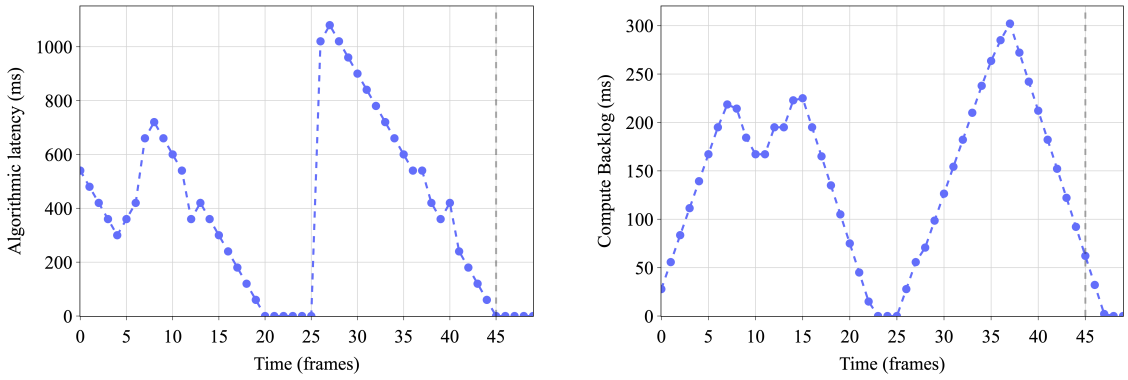

图 *12*。语音助手 Conformer ANCAT 模型逐帧算法延迟（左），以及话语中每一帧的计算积压（右）。该话语长度为 46 帧，评估计算积压时使用的吞吐量为 2。

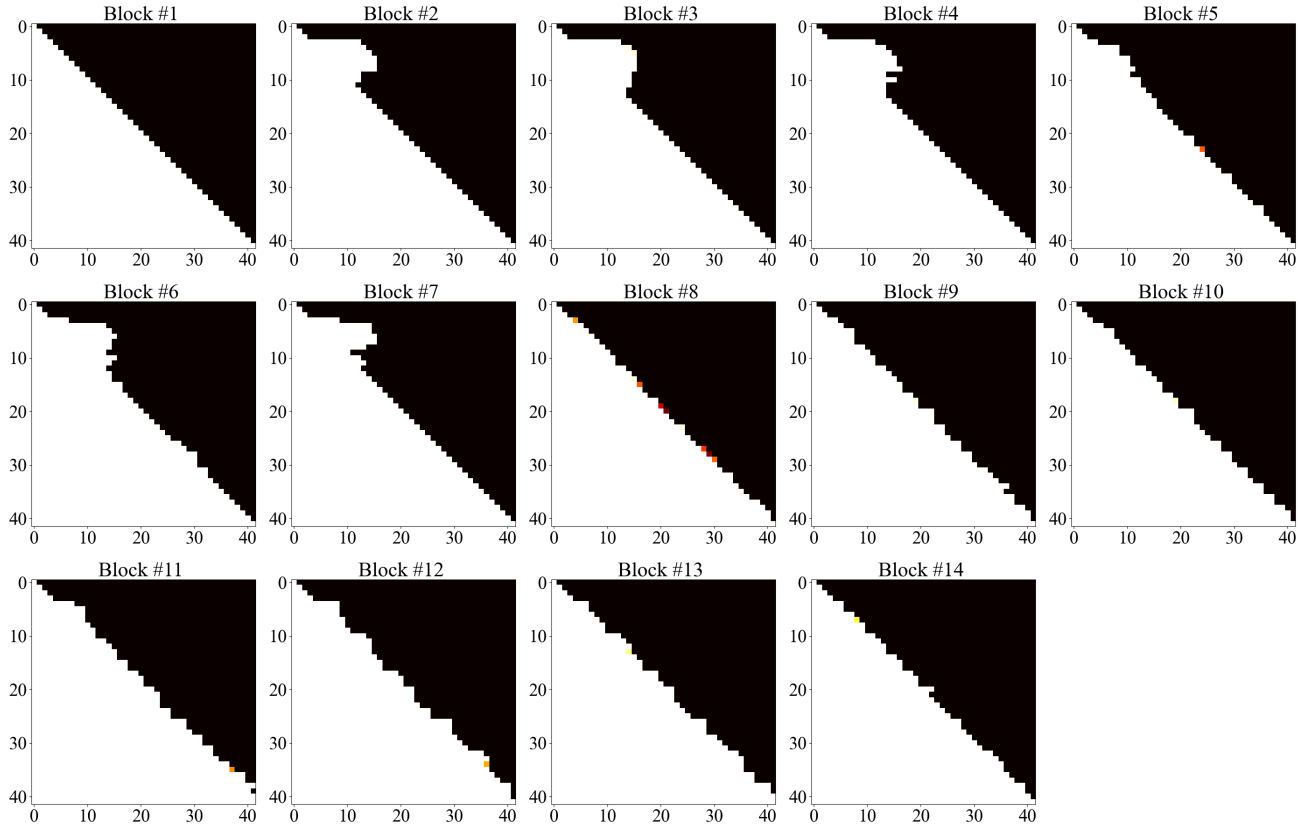

图 *13.* LibriSpeech 示例话语的 Conformer ANCAT-*Alg.Lat.* 模型分块注意力掩码。可以观察到 ANCAT-*Alg.Lat.* 的独特阶梯状模式，其中话语中的特定片段包含大量聚集的前瞻。

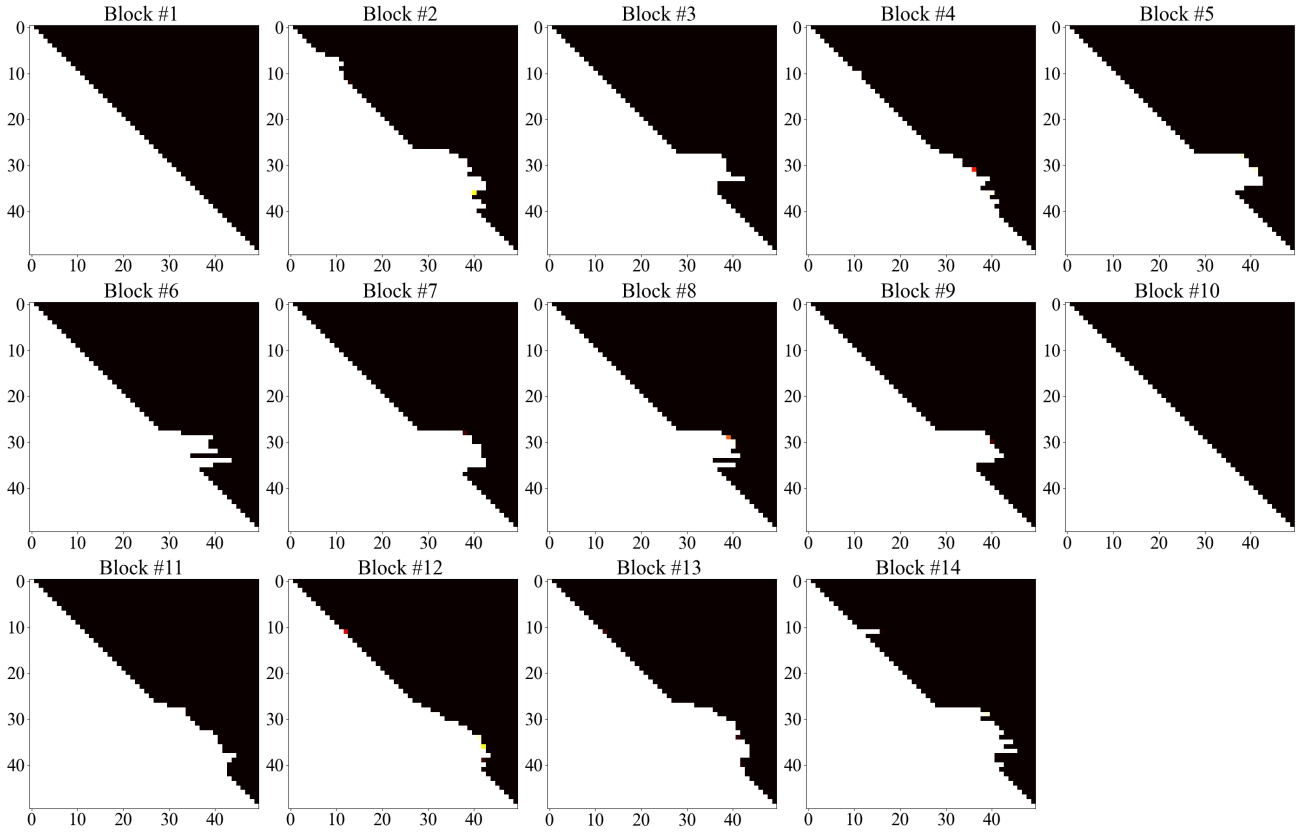

图 *14*。语音助手示例话语的 Conformer ANCAT-*Alg.Lat.* 模型分块注意力掩码。可以观察到 ANCAT-*Alg.Lat.* 的独特阶梯式模式：话语中的特定片段包含大量聚集的前瞻。

F. 训练特性

在整个训练过程中，温度参数 τ 从 1 逐渐退火至接近 0，使软DAG边（以及注意力掩码）转变为二值形式。本附录展示了训练过程中 τ 的变化如何影响 ANCAT架构的各个组件。

图15 展示了不同温度下未来帧调度器的各种掩码值。请注意，曲线如何从反向的“S”形逐渐变尖，最终变成阶跃函数。

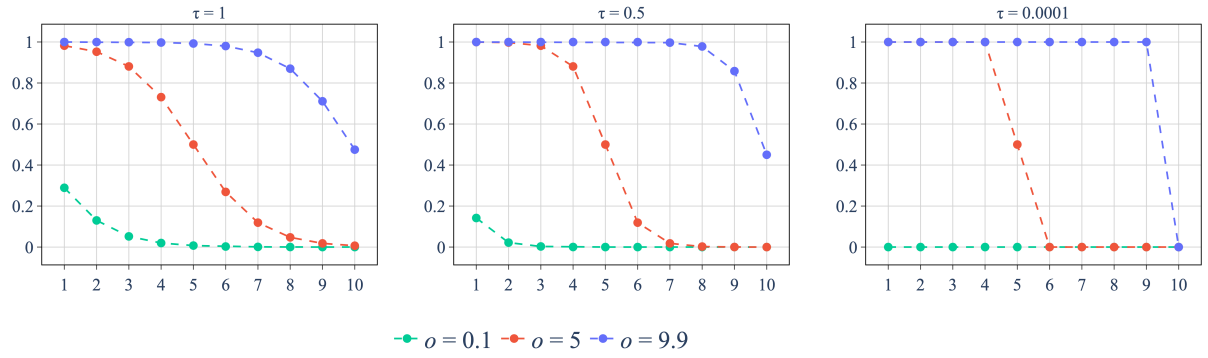

图 *15.* 不同预测曲线中心 o 在最大跨度为 $K = 10$ 时、不同温度下的调度器取值。这些取值对前向注意力位置进行掩码，并表示计算 DAG 中前向边的存在。随着温度降低，这些边会逐渐二值化。

在训练过程中，ANCAT 具有软边界。因此，损失函数会计算相应的“软延迟”。与此同时，“硬延迟”将所有非零边界值视为 1，用于评估测试时真实的运行时延迟。自然，在训练初期，这两种度量有所不同。不过，我们在图 16 中展示，随着温度逐渐退火，硬延迟和软延迟度量会平滑地收敛至最终的算法延迟，且硬延迟始终是软延迟的上界。

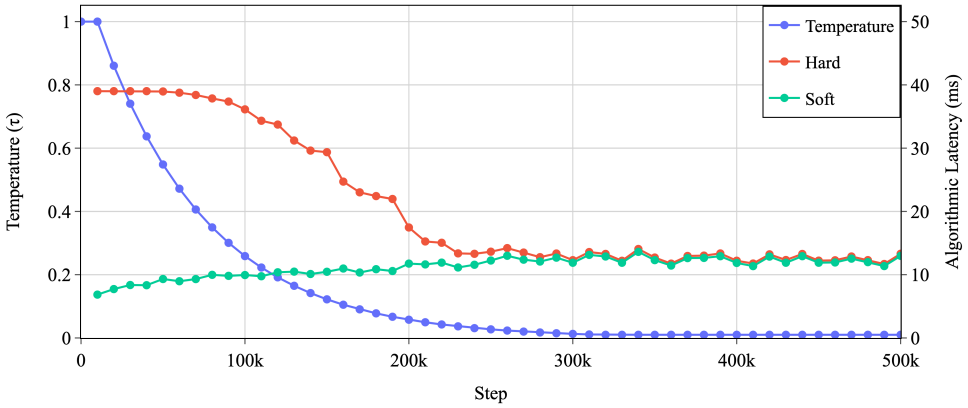

图 *16.* 训练过程中的温度、软延迟和硬延迟。使用 Conformer ANCAT-*Alg.Lat.* 模型每隔 10k 步评估一次模型检查点。

图 17 和 18 展示了同一段来自测试集的 46 帧话语在训练期间三个检查点处，ANCAT 模型的逐帧注意力、算法延迟和计算积压情况。图 17 展示了该话语的前瞻注意力掩码值如何随训练变化并最终达到稳定位置。可以看到，训练初期有许多分数形式的注意力掩码值，但随着温度退火，这些值变得更加两极化（更接近 0 或 1）。在最后一个检查点，温度 τ 为 0.01，注意力掩码几乎是二值的。

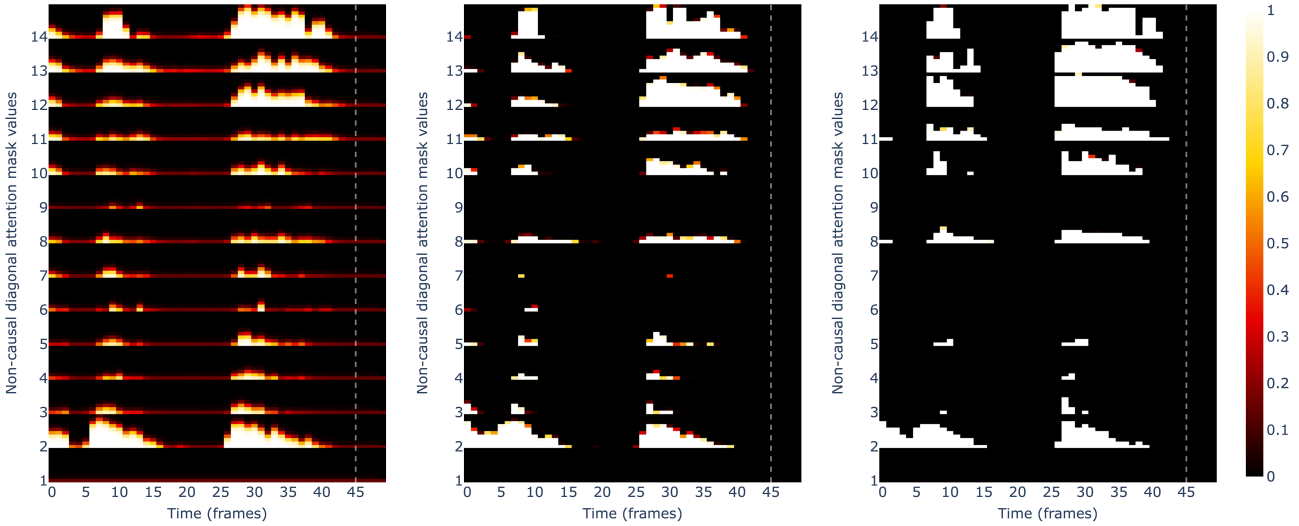

图 *17*。 Conformer ANCAT-*UPL* 模型中，一段话语的逐帧非因果注意力掩码值。这些掩码值分别来自以下检查点评估结果：第 50k 步、温度为 0.548（左）；第 160k 步、温度为 0.105（中）；以及第 500k 步（最终步）、温度为 0.01（右）。该话语由一位远场用户说出：“[语音助手], 播放圣诞音乐。”每层的最大前向跨度 K 为 10，图中仅显示完整掩码主对角线右侧的 10 个前向掩码值。这 10 个前向掩码值从底部向顶部递增。层标注在左侧纵轴上。在整个训练过程中，注意力掩码从分数值逐渐转变为近似二值。随着模型收敛并适应退火温度，大部分注意力结构得以保留，但也出现了明确差异。

接下来，我们在图 18 中展示训练期间逐帧软延迟和硬延迟的特征。首先请注意，尽管在较早的训练轮次中，逐帧软延迟和硬延迟模式差异很大，但正如预期，它们在最终检查点处收敛到几乎相同的值。还可以看到，在第一个检查点处，硬算法延迟达到最大值，这与全上下文注意力模型的预期相符；同样，在该检查点处，随着每一帧不断加入实时无法处理的额外工作，积压会随时间逐渐增加。

接下来，比较不同检查点之间的逐帧延迟，我们可以看到软算法延迟保持着相似的形状，但其幅度在训练过程中略有增加。这表明，在训练早期，模型能够学习将重要的注意力权重保持在较小但非零的水平，以最小化软惩罚项。然而，随着温度降低，这些非零权重变得越来越两极化，其中如今更强的加权前向连接会导致更高的算法延迟。这与图16的发现一致。

最后，我们讨论逐帧计算积压。与算法延迟的趋势相似，我们看到软延迟和硬延迟在训练轮次中逐渐趋同。软积压保持相同的形状，但幅度有所增加。硬积压从最大值开始，随后逐渐降低，并与软积压计算结果趋同。仔细观察第一个检查点中的硬积压，可以发现，由于硬算法延迟较大，积压会随着帧数逐帧增加。进一步仔细观察可发现，每经过 10 帧，斜率都会略微下降。这是因为经过 10 帧后，第一层的节点开始变得可用，因为每层最大右侧上下文为 10 帧。经过 20 帧后，第二层开始变得可用，此后每隔 10 帧便有一层变得可用。如果话语足够长，我们会看到硬计算积压在 140 帧（14 层 $\times$ 10前瞻）时达到最大值。

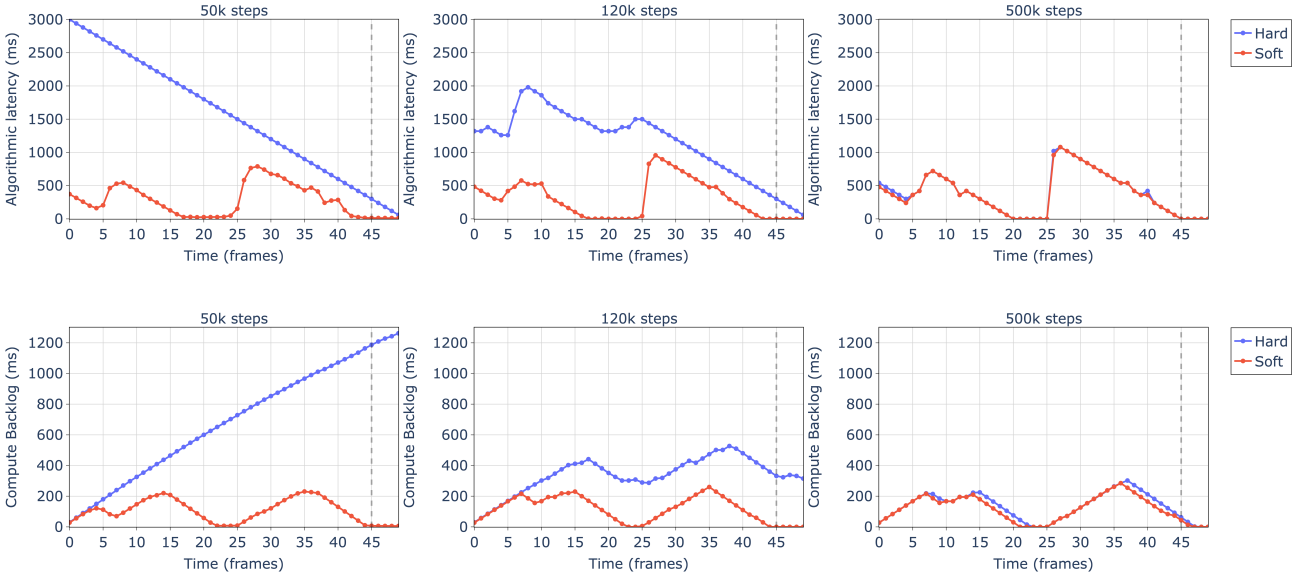

图 *18*。语音助手 Conformer ANCAT 模型逐帧算法延迟（上）以及各帧存在的计算积压（下）。该话语在训练期间的三个检查点进行评估：步数 50k（温度为 0.548，左）、步数 160k（温度为 0.105，中）和最终步数 500k（温度为 0.01，右）。计算积压评估使用的吞吐量为 2。

G. 训练配置细节

本文中展示的所有模型均使用 LibriSpeech 和语音助手数据进行训练，训练所用设备为 6×8 NVIDIA Tesla V100 GPU，每个核心的分桶批次大小为 $[32, 16, 12, 4]$，并根据不同的话语长度进行设置。对于 LibriSpeech ANCAT模型，我们最初将调度器的温度设为 $\tau = 1$，并将其退火至 $1\mathrm{e}{-4}$，指数衰减率为 0.99996，从第 40 个训练轮次开始。语音助手采用更长的退火过程，在 32 万步内将温度降至 0.01。我们还在表 3 中列出了 Conformer 模型的详细编码器配置。实验中使用的 T-T 架构具有相同的超参数，但卷积模型的超参数除外，因为这些参数不适用。调度器网络总共包含 0.2M 个参数，与我们的主要 Transformer 编码器相比，规模相当轻量。

表 *3*。 Conformer 编码器模型超参数。

Conformer 编码器设置	
参数数量（百万）	27.5
编码器层	14
编码器维度	256
注意力头数	4
前馈维度	1024
逐点卷积 1 的卷积核大小	(512, 256, 1, 1)
逐通道卷积核大小	(32, 1, 1, 1)
第二个逐点卷积核大小	(256, 512, 1, 1)
输出尺寸	512

H. 语音助手结果表

为求完整，我们还 像处理 LibriSpeech 一样，为 Conformer 和 T- T
基线及 ANCAT 模 型构建了语音助手数据表。我们提供模型标签名称以便查阅。

标签	注意力（右侧上下文）帧)	正则化 Term	Reg. 权重	吞吐量	WERR	Alg. 延迟 (ms)	UPL (ms)
基线	因果	N/A	N/A	2.0	0.0	0.0	0.0
第1层	逐层（1）	N/A	N/A	2.0	-9.2	797.3	224.2
第2层	逐层（2）	N/A	N/A	2.0	-11.9	1399.7	446.7
第10层	逐层（10）	N/A	N/A	2.0	-21.1	2323.0	1160.8
分块5	分块（5）	N/A	N/A	2.0	-3.9	122.1	82.8
分块10	分块（10）	N/A	N/A	2.0	-8.2	277.5	194.7
分块20	分块（20）	N/A	N/A	2.0	-11.2	583.2	402.3
分块30	分块（30）	N/A	N/A	2.0	-12.0	871.4	681.6
L1-A	ANCAT（10）	L1	1.00e-04	2.0	-16.8	1519.0	772.5
L1-B	ANCAT（10）	L1	2.50e-04	2.0	-14.4	1177.9	606.8
L1-C	ANCAT（10）	L1	5.00e-04	2.0	-12.9	825.2	361.4
L1-D	ANCAT（10）	L1	7.50e-04	2.0	-11.2	619.4	239.8
L1-E	ANCAT（10）	L1	1.00e-03	2.0	-9.4	587.5	236.7
L1-F	ANCAT（10）	L1	1.25e-03	2.0	-8.7	412.7	150.0
L1-G	ANCAT（10）	L1	1.50e-03	2.0	-7.0	266.0	33.4
L1-H	ANCAT（10）	L1	1.75e-03	2.0	-5.8	242.1	21.9
L1-I	ANCAT（10）	L1	2.00e-03	2.0	-6.6	188.1	11.2
L1-J	ANCAT（10）	L1	2.50e-03	2.0	-5.2	206.8	22.4
AL-A	ANCAT（10）	算法延迟	1.25e-03	2.0	-17.1	1007.9	712.5
AL-B	ANCAT（10）	算法延迟	2.50e-03	2.0	-13.9	884.8	628.0
AL-C	ANCAT（10）	算法延迟	3.75e-03	2.0	-14.3	657.1	467.4
AL-D	ANCAT（10）	算法延迟	5.00e-03	2.0	-11.7	503.4	344.4
AL-E	ANCAT（10）	算法延迟	7.50e-03	2.0	-9.5	256.7	46.6
AL-F	ANCAT（10）	算法延迟	1.00e-02	2.0	-8.9	176.3	12.0
AL-G	ANCAT（10）	算法延迟	1.50e-02	2.0	-6.5	88.2	3.1
AL-H	ANCAT（10）	算法延迟	1.75e-02	2.0	-4.8	81.8	2.5
CB01.5-A	ANCAT（10）	UPL	8.75e-05	1.5	-14.8	888.1	264.5
CB01.5-B	ANCAT（10）	UPL	1.30e-04	1.5	-14.9	725.2	154.4
CB01.5-C	ANCAT（10）	UPL	1.93e-04	1.5	-14.0	611.8	107.6
CB01.5-D	ANCAT（10）	UPL	4.38e-04	1.5	-10.0	526.5	58.5
CB01.5-E	ANCAT（10）	UPL	9.77e-04	1.5	-9.9	455.9	22.8
CB2-A	ANCAT（10）	UPL	1.17e-04	2.0	-16.9	814.9	174.4
CB2-B	ANCAT（10）	UPL	1.73e-04	2.0	-17.9	796.0	125.6
CB2-C	ANCAT（10）	UPL	2.57e-04	2.0	-14.3	642.2	72.0
CB2-D	ANCAT（10）	UPL	5.83e-04	2.0	-13.4	585.2	47.6
CB2-E	ANCAT（10）	UPL	1.30e-03	2.0	-12.4	482.2	17.1
CB3-A	ANCAT（10）	UPL	1.75e-04	3.0	-17.4	890.0	108.4
CB3-B	ANCAT（10）	UPL	2.59e-04	3.0	-17.1	722.6	51.9
CB3-C	ANCAT（10）	UPL	3.85e-04	3.0	-14.7	673.6	48.2
CB3-D	ANCAT（10）	UPL	8.75e-04	3.0	-13.8	594.5	24.2
CB3-E	ANCAT（10）	UPL	1.95e-03	3.0	-12.3	550.6	12.2

表*4*。 语音助手 Conformer 实验结果。

标签	注意力（右侧上下文）帧）	正则化 Term	Reg. 权重	吞吐量	WERR	Alg. 延迟 (ms)	UPL (ms)
基线	因果	N/A	N/A	2.0	0.0	0.0	0.0
第10层	逐层（10）	N/A	N/A	2.0	-18.1	2322.7	1160.9
分块10	分块（10）	N/A	N/A	2.0	-5.5	277.5	194.7
分块20	分块（20）	N/A	N/A	2.0	-9.5	583.3	402.4
L1	ANCAT（10）	L1	5.00e-04	2.0	-15.6	967.6	310.5
AL	ANCAT（10）	算法延迟	5.00e-03	2.0	-16.4	563.8	292.0

表*5*。 语音助手 Transformer-Transducer 实验结果。