arXiv:2603.05727v1 [cs.CL] 5 Mar 2026

大语言模型的结构化多维表示学习

A. ELICHI*、K. JBILOU*、M. EL GUIDE † 和 F. DUFRENOIS‡

摘要。 Transformer 架构在广泛的模式识别和自然语言处理任务中实现了最先进的性能，但其规模扩展伴随着参数大幅增长，以及嵌入维度中的冗余。在本研究中，我们提出一种基于 $\mathscr{L}$-乘积的嵌入空间结构化谱分解方法，适用于三阶张量。通过将词元表示重塑为谱张量切片，并在变换域中执行注意力和前馈运算，我们得到一种 张量 *Transformer* 架构，将编码器分解为 p 个独立的谱子 Transformer，同时保留标准 Transformer 语义。我们证明，所提出的 $\mathscr{L}$-Transformer 在谱意义上等价于 p 个并行运行于降维嵌入上的 Transformer；在总嵌入大小固定的情况下，这使编码器参数减少 $\approx 1/p$ （不计偏置和归一化参数等低阶项）。采用实值离散余弦变换（DCT）实例化后，该方法仍完全可微，并兼容现有训练流水线。除压缩之外，谱分解还对嵌入频率引入归纳偏置，使依切片而异的频率缩放成为可能，从而提升泛化能力。在 IMDB 和 AG News 上的实验表明，所提出的模型可大幅减少编码器参数（p=4 时最高可达 75%），同时保持有竞争力的准确率。在 IMDB 上，张量化编码器在压缩条件下达到或超过标准基线；而在 AG News 上，采用中等宽度时，我们观察到准确率略有下降，换取了 4×的编码器削减；在 BERT-base 宽度（d=768）下，性能则恢复到持平水平。

关键词。 张量 Transformer， $\mathscr{L}$-积，变换域注意力，谱张量分解，离散余弦变换（DCT），结构化表示学习，参数高效 Transformer

1 引言 Transformer 架构 [33]已成为现代模式识别与自然语言处理中的主导范式，在文本分类、机器翻译和大规模语言建模方面取得了最先进的性能 [2, 31, 5, 16, 24]。然而，其成功伴随着参数量的大幅增长，尤其是在嵌入维度和前馈层中。随着模型宽度的增加，冗余与过参数化随之出现，这推动了关于结构化压缩和低秩近似的研究 [35, 27, 18], ， 包括蒸馏 [25, 8],、架构参数共享 [14], 以及结构化分解 [30]。

现有的大多数压缩策略通过剪枝、低秩矩阵分解或事后秩约简，直接在权重空间中进行操作。这些方法虽然有效，但只是对预训练模型进行近似，并未从根本上改变嵌入空间的表示几何。另一种视角是通过结构化谱分解对嵌入维度本身进行重新参数化，从而在训练之前施加代数结构。

张量代数为结构化多维表示提供了一个自然的框架 [13, 23,11]。具体而言，三阶张量的 $\mathscr{L}$-乘积通过沿指定模 [9]应用可逆线性变换来定义乘法。在变换域中，该运算会形成块对角结构，将张量切片解耦，同时通过逆变换保持全局一致性。

在本研究中，我们利用这一代数性质，提出了一种张量 Transformer 架构。词元嵌入被重塑为三阶张量，注意力和前馈运算则使用 $\mathscr{L}$-乘积重新表述。我们的主要理论结果表明，所得编码器在谱意义上等价于在变换域中对降维嵌入进行运算的 p 个独立 Transformer。这种等价性使编码器参数减少 $\approx 1/p$，同时保留标准自注意力在每个谱切片内的表达能力。

所提出的构造并不等同于对嵌入维度进行简单划分。正变换和逆变换通过全局线性算子耦合谱分量，使注意力和前馈更新之后能够在切片之间进行结构化混合。采用离散余弦变换 [1], 时，该框架仍为实值、完全可微，并且兼容标准优化流水线。

除了减少参数外，谱公式还引入了频域归纳偏置。通过分配依赖切片的位置缩放系数，模型可以强调低频分量，或在谱通道之间以调和方式分配注意力。这种结构化谱加权有助于提升任务相关性和稳定性，适用于中等规模的分类任务。

*法国加来海岸-奥帕尔大学´，LMPA，F. Buisson 街 50 号，62228 加来邮政信箱，加来，法国。
†FGSES，穆罕默德六世理工大学，拉巴特，摩洛哥
‡ LISIC，F. Buisson 街 50 号，加来海岸-奥帕尔大学，62228 加来邮政信箱，加来，法国。

在 IMDB 和 AG News 上的实证评估表明，张量 Transformer 能够大幅减少编码器参数。在某些配置下，相较于标准 Transformer 基线，谱加权进一步提升了预测性能。这些发现表明，结构化谱因子分解为基于注意力的模型中的扁平嵌入表示提供了一种有原则且计算高效的替代方案。

第 2 节回顾高效 Transformer、低秩适配、张量化参数化以及基于变换的混合等相关工作。第 3 节介绍后续内容所需的张量记法和代数运算。第 4 节定义 $\mathscr{L}$-变换和 $\mathscr{L}$-乘积，并回顾逐切片乘法和 $\mathscr{L}$-SVD 等关键性质。第 5 节介绍所提出的张量化 Transformer 架构：嵌入张量化、$\mathscr{L}$-多头注意力、$\mathscr{L}$-前馈网络、归一化，以及支撑 $\approx 1/p$ 编码器参数缩放的逐切片等价性结果。第 6 节报告在 IMDB 和 AG News 上进行的实证结果和消融实验，包括参数匹配基线、扩展至 $d = 768$，以及效率测量。最后，第 7 节总结全文并概述局限性与未来方向，附录则提供更多实现细节和补充分析。

2 相关工作 Transformer 将多头自注意力与逐位置前馈模块相结合，已成为序列建模的主流架构，使语言、视觉和多模态场景中的可扩展表示学习成为可能 [33]。这一架构的扩展也催生了基础语言模型，它们展现出出色的少样本性能和广泛的泛化能力[2, 31, 5, 16, 24]。与此同时，标准自注意力可能会大量占用内存，并存在结构性冗余（包括注意力头之间的冗余），因此，大量研究致力于在保持下游准确率的同时，近似词元交互并压缩参数 [35, 12, 34, 3, 4]。这些工作涵盖了改变词元交互方式的架构修改、训练后直接修改权重矩阵的干预措施，以及避免更新整个模型的参数高效微调策略。

一类重要方法通过低秩结构来消除冗余。在参数高效微调中，低秩适配器将可训练的秩约束更新注入预训练线性映射，以较低的更新成本实现较高的准确率 [7]。与之互补的训练后方法则无需额外训练即可压缩或修改预训练权重。层选择性秩约简会移除所选权重矩阵的高阶分量，无需增加参数或使用额外数据，便能提升下游推理能力 [27]。相关的以张量为中心的压缩思路也可直接应用于大型模型中的矩阵集合。TRAWL 将模型参数表示为张量格式，并通过结构化约简降低存储和计算成本 [19]。这些工作共同表明，应将相关矩阵组视为结构化对象，而非彼此独立的二维数组；在多头注意力中，这一点尤其自然，因为注意力头构成了一个具有显著相关性的额外模。其他互补的压缩路线还包括针对预训练 Transformer 的知识蒸馏 [25, 8], 、架构参数共享 [14], , 以及为 Transformer 块量身定制的结构化矩阵分解 [30]。

第二条研究路线通过利用多线性结构，发展张量分解和张量化网络参数化方法，以降低复杂度。综述研究已确立 CP分解、Tucker分解及相关分解在压缩和正则化高维模型方面的核心地位 [13, 32, 6]。张量列车表示因其良好的扩展性和稳定的代数性质，对于超高维张量尤其具有吸引力 [23], , 并已用于张量化神经网络层，使大型权重矩阵能够以紧凑的因子分解形式存储和运算 [22]。在Transformer的语境下，这些观点支持将注意力中的参数和激活值视为高阶张量，并通过多线性算子进行处理，从而捕捉因展平而被掩盖的模间依赖关系。

在张量代数中，一种颇具影响力的框架是基于变换的张量-张量积，它将三阶张量视为作用于矩阵的线性算子。t-积形式体系及其相关的张量奇异值分解，提供了矩阵分解在算子理论上的类比，十分适合对指定张量模 [11] 上的相关性进行建模。这一视角通过定义沿管维度应用可逆线性变换所诱导的张量-张量积得以推广，从而形成一类灵活的乘积，既涵盖傅里叶型构造，也允许采用其他正交变换 [9]。这类基于变换的乘积在学习场景中颇具吸引力，因为它们兼具两项有助于模型设计的特性：在变换域中，它们能逐切片保留类似矩阵的运算体系；同时，还能通过所选变换，以有原则的方式编码跨切片交互。

最后，谱混合也被用作显式注意力的一种高效替代方案：它以固定变换取代学习得到的词元间交互核，以较低开销在全局范围内混合信息。例如，FNet 以傅里叶混合层替代注意力，并在多个场景中取得了具有竞争力的结果 [15]。这与利用实正交变换（如离散余弦变换）的方法自然相连；这类变换能够实现类似的全局混合，同时保持纯实数值。综上，低秩适配与训练后秩缩减、张量化参数化与权重压缩、基于变换的张量-张量积，以及谱混合层，共同构成了一个连贯的基础：将注意力及其参数视为结构张量，并设计能够高效捕获跨头依赖的变换域算子。下一小节将总结这些方向，并阐明所提出的 L-乘积张量化在其中的定位。

所提出的 *L*-乘积张量化在高效 *Transformer* 方法中的定位 高效 Transformer 相关研究可根据两个维度进行分类：（i）被结构化的对象（权重还是表示），以及（ii）施加结构的时机（训练期间还是事后）。LoRA 等参数高效微调方法将低秩更新注入预训练线性映射，在保持基础模型不变的同时减少可训练参数的数量 [7]。与之互补的训练后方法（如层选择性秩约简）无需额外任务训练即可修改预训练权重 [27]。第三类方法通过张量格式或分解来表示矩阵集合，从而压缩权重；近期研究如 TRAWL 将结构化张量约简应用于大模型权重 [19], ，而经典基础方法包括 CP/Tucker分解和张量列车表示 [13, 32, 6, 23] ，以及更早用于神经网络的张量化策略 [22]。最后，谱混合层以固定变换取代学习得到的词元交互；FNet 使用傅里叶变换混合取代注意力 [15]。

本工作与之最主要的区别在于结构所处的位置：它通过由可逆变换 [9], 诱导的 L-乘积对嵌入和中间表示进行重新参数化，而不是近似单个权重矩阵。在变换域中，所得编码器在谱上等价于 p个独立的 Transformer，分别处理降维嵌入；在嵌入总规模固定的情况下，编码器参数可精确减少 $1/p$倍（而共享嵌入和任务头保持不变）。这种等价性还伴随着每个模块之后通过逆变换产生的全局耦合，因此该构造并非简单的特征划分。

3 预备知识与符号说明

3.1 张量、元素、切片和纤维 张量是一个多维数组。阶 （也称为模数或维度）是访问一个元素所需的指标数量。标量、向量和矩阵分别对应于 0 阶、1 阶和 2 阶张量。

设 $\mathscr{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times nN}$ 为一个 N 阶N 张量。其元素记为

$$a_{i_1 i_2 \cdots i_N}, \qquad 1 \le i_k \le n_k,\ k = 1, \ldots, N.$$

Frobenius 范数为

$$\|\mathscr{A}\|_F^2 = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_N=1}^{n_N} a_{i_1 i_2 \cdots i_N}^2.$$

在本文中，三阶张量是核心对象。对于 $\mathscr{A} \in \mathbb{R}^{n_1 \times n_2 \times n^3}$:

- 一个正面切片 是矩阵 $\mathscr{A}(:,:,k) \in \mathbb{R}_{n_1 \times n^2}$。
- 固定除一个指标以外的所有指标即可得到一条纤维 。
- 一条管 是一个模式 3 纤维，即 $\mathscr{A}(i,j,:) \in \mathbb{R}^{1 \times 1 \times_{n^3}}$。

3.2 矩阵化（展开） $\mathscr{A} \in \mathbb{R}^{n_1 \times \cdots \times n_N}$ 的模式n 矩阵化（展开）记作 $\mathscr{A}_{(n)}$ ，且属于

$$\mathscr{A}_{(n)} \in \mathbb{R}^{n_n \times \prod_{k \ne n} n_k}.$$

具体的索引映射见附录 A.1。

表2.1最接近方法比较：所提出的 $\mathscr{L}$-积张量化的定位。

方法类别	结构化对象	应用时机	精确与近似 mate	主要优势／典型局限
低秩参数高效微调（LoRA） [7]	权重更新	微调	近似于完整微调调优	可训练参数很少；基础编码器参数量保持不变。
训练后秩缩减 [27]	权重	事后	近似修改模型	无需重新训练；准确率取决于层的选择tion和秩选择。
张量化权重压缩-sion（TRAWL; CP/Tucker分解/张量列车分解） [19, 13, 23, 32, 6]	权重作为张量因子	训练或后处理 hoc	近似因子分解 tion	存储量/参数重新减少；可能影响优化和准确率。
谱混合（类似 FNet） [15]	令牌混合操作-tor	训练	精确变换（不同层类型）	以较低的参数成本实现全局混合；并非旨在针对精确的 $1/p$ 编码器缩放。
本工作（$\mathscr{L}$-乘积张量化-tion) [9]	**表示和张量算子**	**训练时重参数化-化**	**精确的逐切片等价性（变换域）**	**编码器参数规模为 $\approx 1/p$；注意力图存储年龄和 T^2d 注意力项保持不变；实际运行时间收益需要批量切片执行。**

3.3 n-模乘积 n模乘积定义了沿模 n 将张量与矩阵相乘的运算。

定义 3.1（n 模乘积）。设 $\mathscr{A} = [a_{i_1 i_2 \cdots i_N}] \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ 且 $X = [x_{j i_n}] \in \mathbb{R}^{J \times I_n}$。$n$ 模乘积 $\mathscr{A} \times_n X$ 是一个阶为 N、大小为

$$I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N,$$

其元素为

$$(\mathscr{A} \times_n X)_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} a_{i_1 \cdots i_N} x_{j i_n}.$$

等价地，用展开表示为，

$$\mathscr{B} = \mathscr{A} \times_n X \quad \Longleftrightarrow \quad \mathscr{B}_{(n)} = X \mathscr{A}_{(n)}.$$

类似地，沿模 n 与向量 $v \in \mathbb{R}^{I_n}$ 进行收缩，得到一个阶为（$N-1$）的张量，此处记为 $\mathscr{A} \bar{\times}_n v$ [1]，，定义为

$$(\mathscr{A} \bar{\times}_n v)_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} a_{i_1 \cdots i_N} v_{i_n}.$$

备注 1. 许多标准张量分解都可以通过 n 模乘积简洁地表示，包括 *CANDECOMP/PARAFAC（CP）分解 [6],* 、*Tucker* 分解 *[32],* 和张量列车（*TT*）分解 *[23]*。

[1]我们使用上划线明确表示降阶步骤

4 $\mathscr{L}$-乘积 我们回顾 $\mathscr{L}$-乘积（一种基于变换的张量乘法），它通过变换域中的逐切片运算统一了若干张量代数框架 [9]。

4.1 $\mathscr{L}$-变换与逐面乘法 定义 4.1（$\mathscr{L}$-变换）。设 $Z \in \mathbb{R}^{p\times p}$ 可逆。定义线性变换

$$\mathscr{L} : \mathbb{R}^{n\times m\times p} \to \mathbb{R}^{n\times m\times p}, \qquad \mathscr{L}(\mathscr{A}) = \mathscr{A} \times_3 Z,$$

其逆变换为

$$\mathscr{L}^{-1}(\mathscr{A}) = \mathscr{A} \times_3 Z^{-1}.$$

我们使用以下简写

$$\widehat{\mathscr{A}} := \mathscr{L}(\mathscr{A}), \qquad \widehat{A}^{(k)} := \widehat{\mathscr{A}}(:,:,k),\ k = 1,\dots,p.$$

定义 4.2（面乘积）。设 $\widehat{\mathscr{A}} \in \mathbb{R}^{m\times \ell\times p}$ 和 $\widehat{\mathscr{B}} \in \mathbb{R}^{\ell\times n\times p}$。它们的逐面乘法 $\widehat{\mathscr{C}} = \widehat{\mathscr{A}} \triangle \widehat{\mathscr{B}}$ 是 $\mathbb{R}^{m\times n\times p}$ 中的张量，其正面切片满足

$$\widehat{C}^{(k)} = \widehat{A}^{(k)}\,\widehat{B}^{(k)}, \qquad k = 1,\dots,p.$$

定义 4.3（$\mathscr{L}$-乘积）。设 $\mathscr{A} \in \mathbb{R}^{m\times \ell\times p}$ 和 $\mathscr{B} \in \mathbb{R}^{\ell\times n\times p}$。它们的 $\mathscr{L}$-乘积为

$$\mathscr{A} *_{\mathscr{L}} \mathscr{B} := \mathscr{L}^{-1}\Big(\widehat{\mathscr{A}} \triangle \widehat{\mathscr{B}}\Big) = \mathscr{L}^{-1}\Big(\mathscr{L}(\mathscr{A}) \triangle \mathscr{L}(\mathscr{B})\Big). \tag{4.1}$$

4.2 单位、转置、正交性与可逆性 定义 4.4（$\mathscr{L}$-单位张量）。单位张量 $\mathscr{I} \in \mathbb{R}^{m\times m\times p}$ 定义为满足

$$\widehat{\mathscr{I}}(:,:,k) = I_m, \qquad k = 1,\dots,p,$$

并设定 $\mathscr{I} := \mathscr{L}^{-1}(\widehat{\mathscr{I}})$。

定义 4.5 ($\mathscr{L}$-转置）。设 $\mathscr{A} \in \mathbb{R}^{m\times n\times p}$。其 $\mathscr{L}$-转置 $\mathscr{A}^T \in \mathbb{R}^{n\times m\times p}$ 定义为

$$\widehat{(\mathscr{A}^T)}(:,:,k) = \big(\widehat{\mathscr{A}}(:,:,k)\big)^H, \qquad k = 1,\dots,p,$$

其中 $(\cdot)^H$ 表示共轭转置（在实数值情形下可简化为转置）。

定义 4.6（$\mathscr{L}$-积下的结构张量）。方张量 $\mathscr{Q} \in \mathbb{R}^{m\times m\times p}$ 满足：

- $\mathscr{L}$-正交，如果

$$\mathscr{Q}^T *_{\mathscr{L}} \mathscr{Q} = \mathscr{Q} *_{\mathscr{L}} \mathscr{Q}^T = \mathscr{I}.$$

  等价地，每个正面切片 $\widehat{Q}^{(k)}$ 都是酉/正交的。
- f-对角张量当每个正面切片 $\widehat{Q}^{(k)}$ 是对角的。
- 可逆，如果存在 $\mathscr{Q}^{-1} \in \mathbb{R}m\times m\times p$ 使得

$$\mathscr{Q} *_{\mathscr{L}} \mathscr{Q}^{-1} = \mathscr{Q}^{-1} *_{\mathscr{L}} \mathscr{Q} = \mathscr{I}.$$

  等价地，每个 $\widehat{Q}^{(k)}$ 都是可逆的。

4.3 $\mathscr{L}$-SVD 与秩 定理 4.7（$\mathscr{L}$-SVD）。设 $\mathscr{A} \in \mathbb{R}^{m\times n\times p}$。则存在 $\mathscr{L}$-正交张量 $\mathscr{U} \in \mathbb{R}^{m\times m\times p}$ 和 $\mathscr{V} \in \mathbb{R}^{n\times n\times p}$，以及一个 f-对角张量 $\mathscr{S} \in \mathbb{R}^{m\times n\times p}$，使得

$$\mathscr{A} = \mathscr{U} *_{\mathscr{L}} \mathscr{S} *_{\mathscr{L}} \mathscr{V}^T.$$

对角管 $\mathscr{S}(i,i,:)$ 称为奇异管；它们的 ℓ_2-范数在 $\mathscr{L}$-框架 *[11, 10]*中起到奇异值的作用。

备注 2（计算）。在实践中，通常会计算 $\widehat{\mathscr{A}} = \mathscr{L}(\mathscr{A})$，执行 p 次独立的矩阵奇异值分解 $\widehat{A}^{(k)} = \widehat{U}^{(k)}\widehat{S}^{(k)}(\widehat{V}^{(k)})^H$，并通过 $\mathscr{L}^{-1}$ 映射回去，以得到 $\mathscr{U}$、$\mathscr{S}$、$\mathscr{V}$。

定义 4.8（$\mathscr{L}$-平均秩）。对于 $\mathscr{A} \in \mathbb{R}^{m\times n\times p}$， $\mathscr{L}$-平均秩为

$$\operatorname{rank}_a(\mathscr{A}) := \frac{1}{p}\sum_{k=1}^{p} \operatorname{rank}\big(\widehat{A}^{(k)}\big).$$

定义 4.9 ($\mathscr{L}$-管秩)。设 $\mathscr{A} = \mathscr{U} *_{\mathscr{L}} \mathscr{S} *_{\mathscr{L}} \mathscr{V}T$ 是一个 $\mathscr{L}$-SVD。$\mathscr{L}$-管秩是非零奇异管的数量:

$$\operatorname{rank}_T(\mathscr{A}) := \#\Big\{ i \in \{1,\dots,\min(m,n)\} \ : \ \|\mathscr{S}(i,i,:)\|_2 > 0 \Big\}.$$

（对于含噪数据，通常使用一个较小的阈值，而不是 0。）

4.4 截断 $\mathscr{L}$-SVD（谱截断）通常只需要少数几个占主导地位的奇异管。截断的 $\mathscr{L}$-SVD 保留最大的 k 个奇异管（按 ℓ_2-范数衡量），并得到

$$\mathscr{A} \approx \mathscr{U}_k *_{\mathscr{L}} \mathscr{S}_k *_{\mathscr{L}} \mathscr{V}_k^T, \qquad k \ll \min(m,n).$$

这种谱截断减少了有效自由度，提高了数值稳定性，并能大幅降低存储和计算成本。

4.5 特殊情形：t-乘积和基于 DCT 的乘积 几种经典张量乘积是 $\mathscr{L}$-乘积 [9] 的特殊情形:

- **t-乘积:** 选择 Z 作为（酉）离散傅里叶变换矩阵；则 $\mathscr{L}$ 对应于沿管应用快速傅里叶变换，从而得到复值变换切片。
- **基于 DCT 的乘积:** 选择Z 作为离散余弦变换矩阵 [1]；然后 $\mathscr{L}$沿管应用离散余弦变换，并且对于实值数据仍保持实值。

备注 3. 基于 *FFT* 的变换通常会产生复数系数（幅度和相位），而基于 *DCT* 的变换对于实值输入则完全为实数。此外，*DCT* 通常具有较强的能量聚集特性 *[1],*，这对于通过谱截断进行压缩、去噪以及低秩张量近似都可能十分有利。

5 $\mathscr{L}$-乘积框架下的张量大语言模型 张量大语言模型（*Tensor LLM*）是一种 Transformer 风格的神经网络，其中内部表示和参数被组织为高阶张量，并通过 $\mathscr{L}$-乘积代数进行组合。与依赖矩阵乘法的标准 Transformer 相比，张量大语言模型在由变换诱导的张量空间中运行，从而可以显式利用多线性结构。

5.1 词元嵌入的张量化 令$X \in \mathbb{R}^{T\times d}$ 为通常的输入序列（长度为T，嵌入维度为d）。取一个整数p，使得$p \mid d$，并定义切片宽度 $d_s := d/p$。 [2] 我们将X 重塑为一个三阶张量

$$\mathscr{X} = \operatorname{Ten}_p(X) \in \mathbb{R}^{T\times d_s\times p},$$

将特征维度拆分为p 个切片:

$$\mathscr{X}(t,j,k) := X\big(t,(k-1)d_s + j\big), \quad t = 1,\dots,T,\ j = 1,\dots,d_s,\ k = 1,\dots,p. \tag{5.1}$$

逆运算 Mat$_p(\cdot)$ 沿模 2 拼接p 个切片，并恢复为 $\mathbb{R}^{T\times d}$ 中的矩阵。下文中，上下文表示将以张量形式生成,

$$\mathscr{Z} \in \mathbb{R}^{T\times d_s\times p},$$

并可在需要时展开回标准矩阵表示$Z = \operatorname{Mat}_p(\mathscr{Z}) \in \mathbb{R}^{T\times d}$.

从矩阵到三阶张量。给定 $X \in \mathbb{R}^{T\times d}$ 和整数 p，使得 p $\mid$d，我们令 $d_s = d/p$，并将 X 重塑为一个三阶张量 $\mathscr{X} \in \mathbb{R}^{T\times d_s\times p}$，方法是将特征维度分组成 p 个连续的块，每块宽度为 d_s。具体而言，对于每个词元位置 t，向量 X(t, :) $\in \mathbb{R}^d$ 被视为一个 $d_s \times p$ 矩阵；第三模索引 k $\in \{1,\dots,p\}$ 选择切片/通道，而第二模索引 j $\in \{1,\dots,d_s\}$ 选择该切片内的坐标。按照张量术语，每个模式 3 纤维 $\mathscr{X}$(t, j, :) 都是一个长度为 p 的管，而 $\mathscr{L}$-变换则沿着该管维度应用。图 5.1 示意了这一折叠操作。

[2] 干净的张量化假设p 可整除d。若不满足，可（i）选择一个相近的可行p，或（ii）将嵌入维度填充至p 的下一个倍数，并在展开后截断回原维度；为避免符号说明过于繁杂，全文均令$p \mid d$。

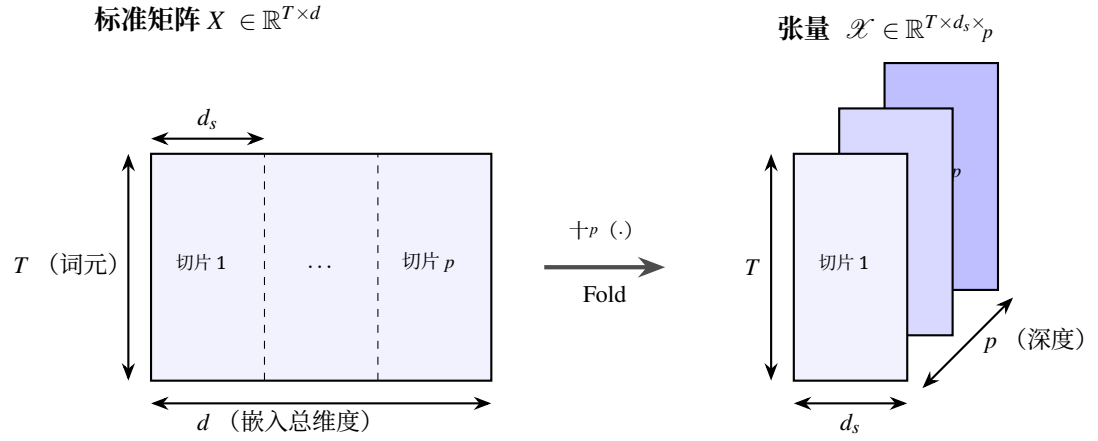


FIG. 5.1. 词元嵌入的张量化。将矩阵 $X \in \mathbb{R}^{T\times d}$ 重塑为 $\mathscr{X} \in \mathbb{R}^{T\times d_s\times p}$，维度为 $d_s = d/p$，方法是将特征维度拆分为 p 个宽度为 d_s的块。第三模（p）是 $\mathscr{L}$-积所用的管维度；沿模式 3 应用 $\mathscr{L}$会生成 p 个变换域前切片 $\widehat{X}^{(k)} \in \mathbb{R}^{T\times d_s}$。

5.2 $\mathscr{L}$-变换与谱切片 令 $Z \in \mathbb{R}^{p\times p}$ 为定义.....的可逆变换矩阵

$$\widehat{\mathscr{A}} = \mathscr{L}(\mathscr{A}) = \mathscr{A} \times_3 Z, \qquad \mathscr{L}^{-1}(\mathscr{A}) = \mathscr{A} \times_3 Z^{-1}.$$

我们将变换域中的正面切片记为

$$\widehat{A}^{(i)} := \widehat{\mathscr{A}}(:,:,i), \quad i = 1,\dots,p.$$

Thro在实现导向的部分（层归一化/前馈网络）中，我们聚焦于实正交变换 s（特别是 DCT 型变换），从而对于实数输入，变换域切片仍为实数。[3]

5.3 基本运算：Softmax 和跨头拼接 定义 5.1（逐行 softmax）。设 $S \in \mathbb{R}^{T\times T}$。逐行 *softmax* 定义为

$$\mathrm{Softmax}(S)_{t,u} = \frac{\exp(S_{t,u})}{\sum_{v=1}^{T}\exp(S_{t,v})}, \quad t,u = 1,\dots,T.$$

Softmax(S)的每一行之和均为1。

定义 5.2（切片内的注意力头拼接）。固定一个变换域切片索引 $i \in \{1,\dots,p\}$，并令 $\widehat{H}^{(1)}i$ …、 $\widehat{H}^{(h)}i \in \mathbb{R}^{T\times d_h}$为 h 个注意力头的输出（其中 $d_s = h\,d_h$）。定义

$$\mathrm{Concat}\big(\widehat{H}_i^{(1)},\dots,\widehat{H}_i^{(h)}\big) \in \mathbb{R}^{T\times d_s}$$

作为按列拼接。

5.4 张量位置编码初始化 设 $\mathscr{X} \in \mathbb{R}^{T\times d_s\times p}$ 为张量化输入。我们定义一种切片感知的正弦位置编码 $\mathscr{P} \in \mathbb{R}^{T\times d_s\times p}$，并设定

$$\mathscr{X}_{\mathrm{pos}} = \mathscr{X} + \mathscr{P}.$$

定义 5.3（切片感知正弦位置编码）。对于 $t = 1,\ .....,\ T,\ j = 1,\ .....,\ d_s,\ k = 1,\ .....,\ p$ 定义

$$\mathscr{P}(t,j,k) = \begin{cases} \sin\Big(\dfrac{t}{10000^{\frac{2\lfloor (j-1)/2\rfloor}{d_s}}}\ \alpha_k\Big), & \text{if } j \text{ is odd},\\ \cos\Big(\dfrac{t}{10000^{\frac{2\lfloor (j-1)/2\rfloor}{d_s}}}\ \alpha_k\Big), & \text{if } j \text{ is even}, \end{cases}$$

[3]在所有实验中，我们使用正交归一化的离散余弦变换（第二型离散余弦变换，采用正交归一化缩放），因此 $\mathscr{L}^{-1}$ 通过采用相同归一化的对应逆离散余弦变换实现。其他正交变换也可使用，而不改变切片级运算；傅里叶变换/离散傅里叶变换的选择仍然有效，但会产生复值切片（附录 A.4）。

其中 α_k 是切片相关频率缩放因子。

备注 4（频率缩放策略）。常见选择包括：
1. 线性（本文采用）： $\alpha_k = k/p$. *2.* 指数： $\alpha_k = 2^{(k-1)/(p-1)}$
（当 $\alpha_1 = 1$ $p = 1$ 时使用）。*3.* 调和： $\alpha_k = k$。

学习式绝对张量位置编码（“可学习”变体）。除了固定的 $\{\alpha_k\}$ 策略外，我们还考虑一种学习式位置编码，其中整个张量

$$\mathscr{P}_{\text{learn}} \in \mathbb{R}^{T \times d_s \times p}$$

被视为可训练参数，并在输入处加一次： $\mathscr{X}_{\text{pos}} = \mathscr{X} + \mathscr{P}_{\text{learn}}$。此位置编码在所有层之间共享（输入级绝对位置编码，如标准学习式位置嵌入）。在我们的实验中，它仅通过与模型其余部分相同的优化器设置（AdamW 权重衰减[17]）进行正则化，不施加其他约束（例如单调性）。

备注5。当 $p = 1$ 且 $\alpha_1 = 1$时，定义 *5.3* 化简为标准正弦位置编码。

5.5 $\mathscr{L}$**-多头注意力** 我们以一种使逐切片谱计算显式化的形式给出张量多头注意力。

参数。设 h 为每个切片内的注意力头数量，并令 $d_s = hd_h$。 [4] 对于每个注意力头 $j = 1,\dots,h$，我们使用张量值投影

$$\mathscr{W}_Q^{(j)}, \mathscr{W}_K^{(j)}, \mathscr{W}_V^{(j)} \in \mathbb{R}^{d_s \times d_h \times p}, \qquad \mathscr{W}_O \in \mathbb{R}^{d_s \times d_s \times p}.$$

$\widehat{\ }$它们的变换域版本为 $\mathscr{W}_\bullet = \mathscr{L}(\mathscr{W}_\bullet)$。

谱域计算。设 $\widehat{\mathscr{X}} = \mathscr{L}(\mathscr{X}_{\text{pos}})$，并记 $\widehat{X}^{(i)} = \widehat{\mathscr{X}}(:,:,i) \in \mathbb{R}^{T \times d_s}$。对于每个切片 $i = 1,\dots,p$ 以及注意力头 $j = 1,\dots,h$，定义

$$\widehat{Q}_i^{(j)} = \widehat{X}^{(i)} \widehat{W}_{Q,i}^{(j)}, \quad \widehat{K}_i^{(j)} = \widehat{X}^{(i)} \widehat{W}_{K,i}^{(j)}, \quad \widehat{V}_i^{(j)} = \widehat{X}^{(i)} \widehat{W}_{V,i}^{(j)},$$

其中 $\widehat{W}_{(j)Q,i} := \widehat{\mathscr{W}}_{(j)Q}(:,:,i)$（对开尔文,$V$同理）。切片 i内的缩放点积注意力为

$$\widehat{A}_i^{(j)} = \text{Softmax}\left(\frac{\widehat{Q}_i^{(j)} (\widehat{K}_i^{(j)})^\top}{\sqrt{d_h}} \right) \in \mathbb{R}^{T \times T}, \qquad \widehat{\text{head}}_i^{(j)} = \widehat{A}_i^{(j)} \widehat{V}_i^{(j)} \in \mathbb{R}^{T \times d_h}.$$

拼接注意力头（定义5.2）以得到 $\widehat{H}_i \in \mathbb{R}^{T \times d_s}$，并在切片内应用输出投影 i:

$$\widehat{Y}_i = \widehat{H}_i \widehat{W}_{O,i}, \qquad \widehat{W}_{O,i} := \widehat{\mathscr{W}_O}(:,:,i).$$

将 $\widehat{Y}1,\dots,\widehat{Y}_p$堆叠为正面切片，得到 $\widehat{\mathscr{Y}} \in \mathbb{R}^{T \times d_s \times p}$，张量输出为

$$\text{MHA}_{\mathscr{L}}(\mathscr{X}_{\text{pos}}) = \mathscr{L}^{-1}(\widehat{\mathscr{Y}}) \in \mathbb{R}^{T \times d_s \times p}.$$

$\widehat{\ }$实现细节 由于Z是固定的，我们直接将变换后的权重 $\mathscr{W}_\bullet = \mathscr{W}_\bullet \times_3 Z$ 存储为可训练参数，并避免在每次前向传播时重新变换权重；仅对激活值进行变换。 [5]

备注 6（切片循环的实用并行实现）。虽然算法 *1–2* 采用*“for i in parallel”*的循环形式编写，但在实际应用中，我们使用批量 内核执行逐切片运算。在 *PyTorch/JAX* 中，标准方法是通过重塑 $\mathscr{X}$，将切片索引视为批次维度，使其形状为 (p, T, d_s)（并融合头索引，使有效批次大小为 $p \cdot h$），然后通过一次调用使用批量矩阵乘法/注意力内核。采用这种批处理方式时，所有 p 个切片都会在图形处理器上并发执行；若使用显式 *Python* 循环实现，实际运行时间可能会随 p 近乎线性增长，从而显著影响计时结果。附录 A.2 给出了完整展开到切片/头级别的实现。

[4] 与总宽度为d、具有H 个注意力头的标准 Transformer 相比时，我们分配注意力头，使得$H = ph$，且每个注意力头的宽度相匹配：$d_h = d/H = d_s/h$。这使得标准编码器与张量化编码器之间的注意力粒度具有可比性。[5] 在我们的实现中，激活值在每个编码器块中变换一次（在逐切片注意力/前馈网络之前），并在该块之后映射回来。也可以让激活值在连续的逐切片子层中保持在变换域，以避免重复的$\mathscr{L}/\mathscr{L}^{-1}$ 调用；这是一种工程选择，并不改变注意力/前馈网络核心的逐切片等价性陈述。

算法 1 $\mathscr{L}$-多头注意力（谱实现）
要求： $\mathscr{X} \in \mathbb{R}^{T\times d_s\times p}$, $\mathscr{P}$, $\{\mathscr{W}_{(j)Q}, \mathscr{W}_{(j)K}, \mathscr{W}_{(j)V}\}_{j=1}^{h}$, $\mathscr{W}_O$, 变换 Z_T

Ensure: $\mathscr{Y} \in \mathbb{R}^{\times d_s\times p}$
$\mathscr{X}_{\text{pos}} \leftarrow \mathscr{X} + \mathscr{P}$
$\widehat{\mathscr{X}} \leftarrow \mathscr{X}_{\text{pos}} \times_3 Z$
Transform all weights: $\widehat{\mathscr{W}_\bullet} \leftarrow \mathscr{W}_\bullet \times_3 Z$
for $i = 1$ to p **in parallel do**
  $\widehat{X}^{(i)} \leftarrow \widehat{\mathscr{X}}(:,:,i)$
  for $j = 1$ to h **do**
    $\widehat{Q}_i^{(j)} \leftarrow \widehat{X}^{(i)}\widehat{W}_{Q,i}^{(j)}$
    $\widehat{K}_i^{(j)} \leftarrow \widehat{X}^{(i)}\widehat{W}_{K,i}^{(j)}$
    $\widehat{V}_i^{(j)} \leftarrow \widehat{X}^{(i)}\widehat{W}_{V,i}^{(j)}$
    $\widehat{S}_i^{(j)} \leftarrow \frac{1}{\sqrt{d_h}}\widehat{Q}_i^{(j)}(\widehat{K}_i^{(j)})^\top$
    $\widehat{\text{head}}_i^{(j)} \leftarrow \text{Softmax}(\widehat{S}_i^{(j)})\widehat{V}_i^{(j)}$
  end for
  $\widehat{H}_i \leftarrow \text{Concat}(\widehat{\text{head}}_i^{(1)}, \ldots, \widehat{\text{head}}_i^{(h)})$
  $\widehat{Y}_i \leftarrow \widehat{H}_i\widehat{W}_{O,i}$
end for
$\widehat{\mathscr{Y}} \leftarrow \text{stack}(\widehat{Y}_1, \ldots, \widehat{Y}_p)$
$\mathscr{Y} \leftarrow \widehat{\mathscr{Y}} \times_3 Z^{-1}$
return $\mathscr{Y}$

5.6 $\mathscr{L}$-前馈网络 定义 5.4（标准前馈网络（矩阵））。设 $X \in \mathbb{R}^{T\times d}$。标准前馈网络定义为

$$\text{FFN}(X) = \sigma(XW_1 + b_1)W_2 + b_2,$$

其中 $W_1 \in \mathbb{R}^{d\times d_{ff}}$、$W_2 \in \mathbb{R}^{d_{ff}\times d}$，且通常有 $d_{ff} = 4d$。

定义 5.5（$\mathscr{L}$-前馈网络（张量））。设 $\mathscr{X} \in \mathbb{R}^{T\times d_s\times p}$，并令 dff, $s = 4d_s$。定义张量权重和偏置

$$\mathscr{W}_1 \in \mathbb{R}^{d_s\times d_{ff,s}\times p}, \quad \mathscr{W}_2 \in \mathbb{R}^{d_{ff,s}\times d_s\times p}, \quad \mathscr{B}_1 \in \mathbb{R}^{1\times d_{ff,s}\times p}, \quad \mathscr{B}_2 \in \mathbb{R}^{1\times d_s\times p}.$$

Then

$$\text{TFFN}(\mathscr{X}) = \sigma\big(\mathscr{X} *_{\mathscr{L}} \mathscr{W}_1 + \mathscr{B}_1\big) *_{\mathscr{L}} \mathscr{W}_2 + \mathscr{B}_2, \tag{5.2}$$

其中 σ 逐元素作用（例如 *ReLU* 或 *GELU*）。

定理 5.6（$\mathscr{L}$变换域中的前馈网络）。设 $\mathscr{X} = \mathscr{L}(\mathscr{X})$，参数同理。于是对于每个切片 $i = 1, \ldots, p$,

$$\widehat{H}_i = \widehat{X}_i\widehat{W}_{1,i} + \widehat{b}_{1,i}, \quad \widehat{G}_i = \sigma(\widehat{H}_i), \quad \widehat{Y}_i = \widehat{G}_i\widehat{W}_{2,i} + \widehat{b}_{2,i},$$

这恰好是对每个切片独立应用的标准前馈网络。

5.7 张量层归一化 定义 5.7（张量层归一化）。令 $\mathscr{X} \in \mathbb{R}^{T\times d_s\times p}$。沿特征模态（模-2）为每个位置 t和切片 k定义均值和方差：

$$\mu_{t,k} = \frac{1}{d_s}\sum_{j=1}^{d_s}\mathscr{X}(t,j,k), \qquad \sigma_{t,k}^2 = \frac{1}{d_s}\sum_{j=1}^{d_s}\big(\mathscr{X}(t,j,k) - \mu_{t,k}\big)^2.$$

具有可学习的参数 $\Gamma, \mathscr{B} \in \mathbb{R}^{1\times d_s\times p}$,

$$\text{TLN}(\mathscr{X})(t,:,k) = \Gamma(:,:,k) \odot \frac{\mathscr{X}(t,:,k) - \mu_{t,k}}{\sqrt{\sigma_{t,k}^2 + \varepsilon}} + \mathscr{B}(:,:,k).$$

备注 7. 在我们的实现中，层归一化是在原始张量域中逐切片应用的（每个切片内进行模态 2 归一化）。我们不依赖于 $\mathscr{L}$ 下的不变性。

5.8 $\mathscr{L}$-编码器层定义 5.8（张量编码器层）。令 $\mathscr{X} \in \mathbb{R}^{T \times d_s \times p}$。张量编码器层为

$$\mathscr{X}' = \mathrm{TLN}\big(\mathscr{X} + \mathrm{MHA}_{\mathscr{L}}(\mathscr{X})\big), \tag{5.3}$$
$$\mathscr{Y} = \mathrm{TLN}\big(\mathscr{X}' + \mathrm{TFFN}(\mathscr{X}')\big). \tag{5.4}$$

定理 5.9（ $\mathscr{L}$-注意力与 $\mathscr{L}$-前馈网络核心的逐切片等价性）。令 $\widehat{\mathscr{X}} = \mathscr{L}(\mathscr{X})$并记 $\widehat{X}_i = \widehat{\mathscr{X}}(:,:,i) \in \mathbb{R}^{T \times d_s}$。那么，对于经 $\mathscr{L}$变换的固定参数，*(i)* 多头注意力$_{\mathscr{L}}(\cdot)$和 *(ii)* 张量前馈网络$(\cdot)$的计算在变换域中是逐切片的：对于每个 $i \in \{1,\dots,p\}$，相应的输出切片仅取决于 X_i 和第 i 个切片的变换后参数，并且等同于宽度为 d_s的标准注意力/前馈网络计算。

备注8（层归一化位置与“精确等价性”）。在我们的实现中，层归一化是在映射回原始张量域之后应用的（并沿模 2 逐切片计算）。由于层归一化是非线性的，且在一般的 $\mathscr{L}$ 变换下不具有不变性，因此完整编码器层在变换域中并非严格可分离。因此，定理 *5.9* 将精确性声明限定为实际实现的内容：注意力/前馈网络核心满足逐切片等价性，而跨切片耦合则通过交替的 $\mathscr{L}^{-1}$ 混合及后续归一化产生。重要的是，根据逐切片核心推导出的编码器参数缩放以及投影/前馈网络 *FLOP* 降低仍然成立。

5.9 $\mathscr{L}$-解码器层（掩码自注意力 -注意力 +可选交叉 -注意力） 对于解码器-only 大语言模型（GPT-style），仅使用掩码自注意力。为完整起见，我们写出编码器–解码器形式。

定义 5.10（张量解码器层）。设 $\mathscr{Y} \in \mathbb{R}^{T_t \times d_s \times p}$（解码器流）和 $\mathscr{Z}_{enc} \in \mathbb{R}^{T_e \times d_s \times p}$（编码器流）。

$$\mathscr{Y}' = \mathrm{TLN}\big(\mathscr{Y} + \mathrm{MHA}^{masked}_{\mathscr{L}}(\mathscr{Y})\big), \tag{5.5}$$
$$\mathscr{Y}'' = \mathrm{TLN}\big(\mathscr{Y}' + \mathrm{MHA}^{cross}_{\mathscr{L}}(\mathscr{Y}', \mathscr{Z}_{enc})\big), \tag{5.6}$$
$$\mathscr{Y} = \mathrm{TLN}\big(\mathscr{Y}'' + \mathrm{TFFN}(\mathscr{Y}'')\big). \tag{5.7}$$

5.10 完整的 $\mathscr{L}$-Transformer 堆叠 N 个张量编码器层和解码器层，即可得到一个完整的张量 Transformer。张量形式为：

$$\mathscr{Z}^{(0)}_{\mathrm{enc}} = \mathscr{Z}_{\mathrm{src}} + \mathscr{P}_{\mathrm{enc}}, \tag{5.8}$$
$$\mathscr{Z}^{(\ell)}_{\mathrm{enc}} = \mathrm{TEncoderLayer}_{\ell}(\mathscr{Z}^{(\ell-1)}_{\mathrm{enc}}), \quad \ell = 1,\dots,N, \tag{5.9}$$
$$\mathscr{Y}^{(0)}_{\text{解码}} = \mathscr{Y}_{\mathrm{tgt}} + \mathscr{P}_{\mathrm{dec}}, \tag{5.10}$$
$$\mathscr{Y}^{(\ell)}_{\mathrm{dec}} = \mathrm{TDecoderLayer}_{\ell}(\mathscr{Y}^{(\ell-1)}_{\mathrm{dec}}, \mathscr{Z}^{(N)}_{\mathrm{enc}}), \quad \ell = 1,\dots,N. \tag{5.11}$$

最后，展开回 $\mathbb{R}^{T_t \times d}$ 并投影到词汇表大小V，可通过以下方式完成

$$Y = \mathrm{Mat}_p\big(\mathscr{Y}^{(N)}_{\mathrm{dec}}\big)\, W_{\mathrm{out}}, \qquad W_{\mathrm{out}} \in \mathbb{R}^{d \times V}.$$

（或者，也可以使用张量输出投影 $\mathscr{W}_{\mathrm{out}} \in \mathbb{R}^{d_s \times V \times p}$，然后再展开。）

5.11 与 p 个并行紧凑型 Transformer 的等价性 $\mathscr{L}$-积定义的一个关键推论是，在变换域中实现切片级解耦。设 $\widehat{\mathscr{X}} = \mathscr{L}(\mathscr{X})$ 和 $\widehat{X}_i = \widehat{\mathscr{X}}(:,:,i) \in \mathbb{R}^{T \times d_s}$。那么，一个张量 Transformer 层等价于对 $\widehat{X}1,\dots,\widehat{X}_p$ 分别应用 p 个独立的紧凑型（维度为d_s）标准 Transformer 层，使用变换后的参数，堆叠结果，然后应用 $\mathscr{L}^{-1}$。

形式化地说，如果 $T^{(i)}$ std : $\mathbb{R}^{T \times d_s} \to \mathbb{R}^{T \times d_s}$ 表示切片i 标准 Transformer 映射（使用切片特定的变换后权重），那么张量输出满足

$$\mathscr{Z} = \mathscr{L}^{-1}\Big(\mathrm{Stack}_3\big(T^{(1)}_{\mathrm{std}}(\widehat{X}_1),\dots,T^{(p)}_{\mathrm{std}}(\widehat{X}_p)\big)\Big). \tag{5.12}$$

切片间耦合。 尽管计算在变换域中按切片进行，但该模型并不等同于对嵌入维度进行静态划分。每个块之后，应用逆变换 $\mathscr{L}^{-1}$ 会混合p 个谱通道，因此下一层的输入切片是上一层切片输出的线性组合。由此，信息可以通过交替的 $\mathscr{L}$/ $\mathscr{L}^{-1}$ 映射在层与层之间跨切片传播，即使每个块都是在 $\mathscr{L}$域中按切片计算的，也能产生结构化的跨切片交互。

推论COROLLARY 5.11（并行计算）。在变换域中，p 个逐切片 *Transformer* 相互独立，可以并行执行。

5.12 训练及梯度传播 $\mathscr{L}$-乘积保持可微性，因为 $\mathscr{L}$和$\mathscr{L}^{-1}$ 是线性映射，且逐面乘法在变换域中等价于标准矩阵乘积。

命题PROPOSITION 5.12（通过 $\mathscr{L}$-乘积进行反向传播）。设 $\mathscr{Y} = \mathscr{X} *_{\mathscr{L}} \mathscr{W}$满足 $\mathscr{X} \in \mathbb{R}^{T\times d_s\times p}$ 且 $\mathscr{W} \in \mathbb{R}^{d_s\times d'\times p}$。给定 $G = \frac{\partial \mathscr{J}}{\partial \mathscr{Y}}$，梯度为

$$\frac{\partial \mathscr{J}}{\partial \mathscr{X}} = G *_{\mathscr{L}} \mathscr{W}^T, \qquad \frac{\partial \mathscr{J}}{\partial \mathscr{W}} = \mathscr{X}^T *_{\mathscr{L}} G,$$

其中 $(\cdot)^T$ 是 $\mathscr{L}$-转置。

5.13 计算复杂度我们比较标准 Transformer 在 d上运行时，与因子为p、切片宽度为$d_s = d/p$的张量 Transformer 的每层浮点运算次数。设$d_{ff} = 4d$且$d_{ff,s} = 4d_s$。

表TABLE 5.1每层浮点运算次数对比。投影和前馈网络的计算开销降低至原来的 $\approx p$，而注意力分数计算保持不变（其随 T^2 增长）。

运算	**标准**	**张量（$\mathscr{L}$-积）**
Q，K，V 投影	$3Td^2$	$3Tpd_s^2 = 3Td^2/p$
注意力分数（$QK^\top$）	T^2d	$pT^2d_s = T^2d$（不变）
注意力 $\times V$	T^2d	T^2d（不变）
输出投影	Td^2	Td^2/p
前馈网络（两层）	$8Td^2$	$8Td^2/p$
$\mathscr{L}$-变换（离散余弦变换/逆离散余弦变换）	—	Td 对数 p
总计	$12Td^2 + 2T^2d$	$12Td^2/p + 2T^2d + Td$ 对数 p

浮点运算次数与墙钟时间。 表 5.1 计算了算术代价，但实测运行时间还会受到以下因素影响：(i) 离散余弦变换/逆离散余弦变换步骤引入的额外内存受限遍历；以及 (ii) 切片式子编码器p 按顺序执行时的核启动/并行性影响。

采用算法 1 后所述的批处理并行实现，影响 (ii) 基本上得以消除，剩余开销主要由变换步骤和框架层面的内存流量构成。

标准 Transformer 的每层代价为 $\mathcal{O}(Td^2 + T^2d)$。张量 Transformer 将投影 和前馈网络 项降至 $\mathcal{O}(Td^2/p)$，而注意力项仍为 $\mathcal{O}(T^2d)$。变换开销$\mathcal{O}(Td\log p)$通常较小。

备注 9（加速比最大时）。当 $d \gg T$ 时，Td^2 项占主导地位，张量 *Transformer* 可使每层浮点运算次数有效减少 $\approx 1/p$ 倍。当 $T \gg d$ 时，注意力项 T^2d 占主导地位，加速比会降低；此时可将张量框架与高效注意力近似相结合 *[12, 34, 3, 4]*。

备注 10（内存）。参数存储减少了 $\approx 1/p$（占主导地位的 d^2 项），而注意力图存储（T^2 阶）保持不变。

6 数值实验 我们在文本分类任务上评估所提出的张量 $\mathscr{L}$-乘积 Transformer，并在受控的训练和预处理条件下将其与标准 Transformer 编码器（Std）进行比较。我们研究两个方面：(i) 在固定 p=4的情况下进行谱加权 （PE）；以及 (ii) 通过分解因子 p 进行张量分解 （在 IMDB 上对 $p \in \{2,4\}$ 进行消融）。结果以三个相互独立的随机种子（42、123、7）下的均值 $\pm$ 标准差表示。我们考虑两个基准数据集：IMDB（二分类情感分析，训练集/测试集各 25 K） [20] 和 AG News（4 类主题分类，120 K/7.6 K） [36]。我们报告

三种宽度 $d \in \{128, 256, 768\}$下的结果；最大设置（d=768）与 BERT-base 的嵌入维度相匹配 [5]，但从头开始[6]训练。

符号说明。标准表示原版 Transformer 编码器基线。**T*N-pe***表示因子为 p=N 且采用谱加权 *pe*的张量模型。当 p=4 时，我们考察：T4-标准（α_k=1）、T4-线性（α_k=k/p）、T4-调和（α_k=k）、T4-指数（α_k=$2^{(k-1)/(p-1)}$）以及 T4-可学习（学习得到的 α_k，额外增加 p 个可训练标量，并与所有其他参数采用相同的权重衰减）。

实证结论。 核心问题在于，将表示空间张量化能否在保持准确率的同时，使编码器参数缩减接近 $1/p$。在中等宽度下，我们观察到明显规律：IMDB 上的收益显著，而在 AG News 上，当 d=256 时，准确率略有损失。在更大宽度（d=768）下，差距缩小：张量模型与标准模型的表现相当，差异处于运行间变异性范围内（3 个随机种子）；同时，编码器参数减少 $4\times$，峰值内存降低 15%。这一扩展趋势是支持该方法适用于小模型以外场景的关键证据。

6.1 实验设置与评估流程

架构。 所有配置均使用n_{layers}=4 个编码器层，词汇表大小为 30000（字节对编-码），最大序列长度为 T=128；所有模型采用相同的截断和填充设置。表 6.1 总结了共享设置，表 6.2 列出了特定任务的宽度。对于因子为

表 6.1所有实验采用的共享架构设置。分词使用字节对编码（字节对编码） *[26]*。

组件	值
编码器深度	$n_{\text{layers}} = 4$
分词器 / 词汇表	字节对编码，30,000 个类型
最大序列长度	$T = 128$ （相同的截断/填充）

TABLE 6.2特定任务的架构设置。

设置	d	H	d_{ff}
IMDB	128	4	512
AG News（中等规模）	256	4	1024
AG News（BERT宽度）	768	8	3072

p，嵌入输出 $X \in \mathbb{R}^{T\times d}$ 被重塑为 $\mathscr{X} \in \mathbb{R}^{T\times d_s\times p}$，以 $d_s = d/p$为单位在离散余弦变换域 [1], 中处理切片，并通过逆 $\mathscr{L}$变换重新组合。

训练。 表 6.3 列出了所有实验共用的优化器和系统配置，表 6.4 则给出了数据集特定调度。

评估。 每种配置均使用全部 3 个随机种子运行。每个模型训练固定轮数（IMDB 为 20 轮，AG News 为 5 轮），不进行早停或检查点选择。我们报告**最终轮次**的测试准确率，该准确率是在留出测试集上计算的（IMDB 有 25 000 个样本；AG News 有 7 600 个样本）。所有模型在最终轮次之前均已充分趋于平稳（附录 B），且最后一轮的准确率在不同随机种子下保持稳定。峰值 GPU 内存通过 torch.cuda.max memory allocated 测量。推理延迟是在混合精度下使用 torch.no grad() 对完整测试集进行 3 次前向传播所得的均值。推理计时在训练结束后进行，不会影响任何训练或选择决策

6.2 公平比较协议 所有模型均采用完全相同的嵌入层和分类器层；只有编码器有所不同。这样可以将编码器单独作为架构和参数量差异的唯一来源。表6.5总结了哪些部分是固定的，哪些部分会变化。

[6]我们使用“BERT-base宽度”仅用于表示嵌入维度 $d = 768$。我们不使用预训练BERT权重进行初始化；此处的所有模型均采用相同的训练方案从随机初始化开始训练。

表6.3 所有实验采用的共享训练配置。

组件	值
优化器	AdamW（学习率 $= 3 \times 10^{-4}$，权重衰减 $= 0.01$）
学习率调度器	OneCycleLR（10% 线性预热，余弦退火至 10^{-5}） [28, 29]
梯度裁剪	1.0
精度	混合精度（PyTorch 2.1，torch.cuda.amp） [21]
硬件	NVIDIA Tesla T4（16 GB），CUDA 12.1
可复现性	torch.manual seed、numpy.random.seed、random.seed；cudnn.deterministic=True

表格 6.4数据集特定训练计划。

设置	轮次	批次大小
IMDB（$d = 128$）	20	128
AG News（$d = 256$）	5	128
AG News（$d = 768$）	5	64

表6.5受控比较：固定与变化的组件。

所有模型中均固定	可变（仅编码器）
嵌入层（词汇表 $\times d$）	编码器类型（标准 vs. $\mathscr{L}$-乘积）
分类器头（均值池化 + 线性）	分解因子 p
训练方案（优化器、学习率调度器、超参数）	谱加权策略（位置编码）
数据协议（序列长度 T、分词器、填充/截断、随机种子）	编码器参数量（及由此产生的压缩比）

对于因子为p的张量模型，编码器在宽度为$d_s = d/p$的p 个子编码器上运行，其编码器参数约为 $1/p$ Std 的编码器参数。在d较小时，嵌入层占总参数量的主要部分（例如 IMDB），因此编码器压缩无法完全体现为整体模型压缩。在d较大时，编码器在参数量中所占比例变高，因此编码器压缩可转化为显著的端到端节省（第6.5节）。

参数匹配基线。 为了区分架构效应与单纯的参数减少，我们纳入了一个参数匹配（PM）标准基线，其编码器参数量与 T4（p=4）大致相当。具体而言，**Std-1L** 是一个采用完整宽度d的单层标准编码器。在d=128时，其编码器参数量为 247,808（0.250×Std），与 T4-线性的 254,016（0.256×）非常接近。在d=256 时，其编码器参数量为 987,136（0.250×Std），与 T4-线性的 999,552（0.253×）相匹配。此比较揭示了架构层面的关键问题：在编码器预算相同的情况下，将资源投入单个全宽层，还是投入四个张量分解层更好？表6.6显示，T4-线性在两个数据集上均持续优于 Std-1L，这表明张量 $\mathscr{L}$-乘积结构带来的益处不止于简单地降低容量。

6.3 主要结果与位置编码策略比较 表 6.7 和表 6.8 报告了 IMDB 和 AG News 上的测试准确率、编码器参数和压缩比（d=256）。图 6.1 总结了编码器参数—准确率权衡。

在 IMDB 上，编码器压缩并未降低准确率；相反，所有p=4 张量变体的表现都优于 Std。最佳配置（T4-标准）达到 82.02% $\pm$0.03，高于 Std 的 80.77% $\pm$0.24 （+1.25 个百分点），同时使用了 0.256× 编码器参数。不同 PE策略之间的差距较小（从最好到最差相差 0.49 个百分点），这表明性能主要由张量分解本身决定，而谱加权只是次要的调优手段。表6.6中的参数匹配比较进一步

表 6.6参数匹配比较（*3* 个随机种子的均值 ±标准差）。编码器参数及相对于标准模型的比率。

数据集	模型	准确率（%）	编码器参数	比率
IMDB（天=128）	标准（4L，天=128）	80.77 ± 0.24	991 开尔文	1.000×
	Std-1L（1L，天=128）	77.50 ± 0.28	248 开尔文	0.250×
	T4-线性（4L，天$_s$=32）	**81.65 ± 0.06**	254 开尔文	0.256×
AG News（d=256）	标准（4L，d=256）	**91.40 ± 0.31**	3 949 开尔文	1.000×
	Std-1L（1L，d=256）	86.97 ± 0.15	987 开尔文	0.250×
	T4-线性 (4L, d_s=64)	90.76 ± 0.13	1 000 开尔文	0.253×

表6.7 *IMDB* 结果（d=128，*20* 个训练轮次，*3* 个随机种子）。嵌入（*3.84 M*）和分类器共享。最佳结果以粗体显示。

模型	p	准确率（%）	编码器	编码器比例
Std	—	80.77 ± 0.24	991 开尔文	1.000×
T4-标准	4	**82.02 ± 0.03**	254 开尔文	0.256×
T4-可学习	4	81.98 ± 0.20	270 开尔文	0.273×
T4-线性	4	81.65 ± 0.06	254 开尔文	0.256×
T4-指数	4	81.60 ± 0.14	254 开尔文	0.256×
T4-调和	4	81.53 ± 0.13	254 开尔文	0.256×
T2-线性	2	81.72 ± 0.17	500 开尔文	0.504×

印证了这一点：在编码器预算相同的情况下，T4-线性的表现显著优于 Std-1L，说明这种提升并非仅由参数减少所致。

表6.8*AG News* 结果（d=256，*5* 个训练轮次，*3* 个随机种子）。最佳结果以粗体显示。

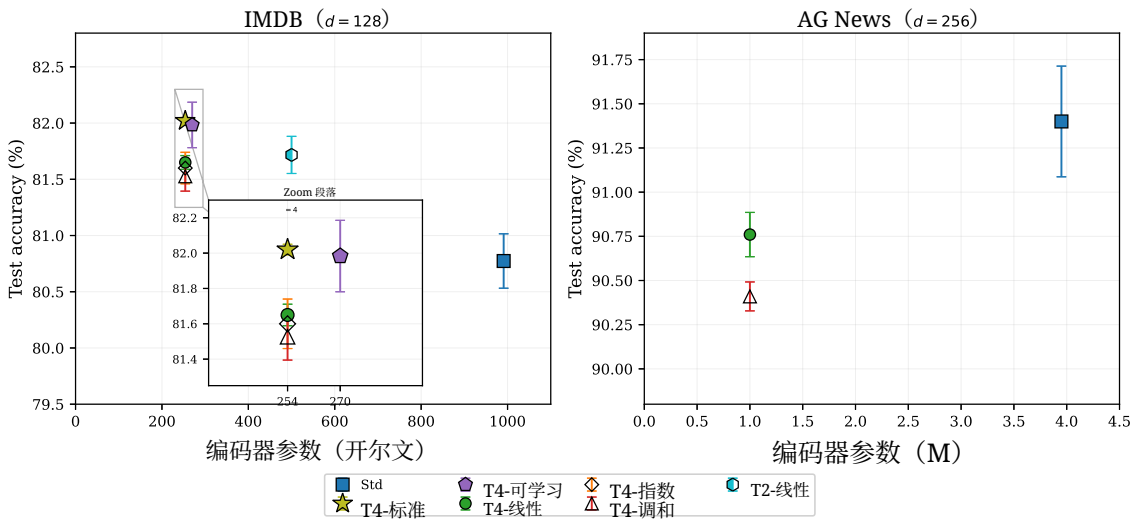
模型	p	准确率（%）	编码器	编码器比例
Std	—	**91.40 ± 0.31**	3.95 M	1.000×
T4-线性	4	90.76 ± 0.13	1.00 M	0.253×
T4-调和	4	90.41 ± 0.08	1.00 M	0.253×

在 AG News 的中等宽度设置下，张量化编码器实现了明显的压缩—准确率权衡。T4-线性将编码器压缩了 $4\times$ (3.95 M $\rightarrow$ 1.00 M)，代价是相较于 Std 准确率仅下降 0.64 个百分点。结合第6.5节的缩放实验来看，这一结果更易解读：在 d=768 时，差距会缩小。表6.6中的参数匹配基线进一步阐明了这一点：在编码器预算相同的情况下，T4-线性显著优于 Std-1L，这表明与单纯缩减深度相比，张量结构能更有效地利用参数。

图6.1展示了关键的实证趋势：张量因子分解使模型向左移动（编码器参数更少），同时准确率仅略有下降或没有下降；在 IMDB 上，准确率还有明显提升。在 AG News 的 d=256 设置下，张量化模型点的准确率略低于 Std，但编码器规模小得多，体现出显著的压缩优势。

6.4 谱加权（位置编码）消融图 6.2 比较了 IM 上的五种位置编码策略，p=4 DB. The 完整扫描（包括 AG News）的结果见附录 C。

可以得出两个结论。首先，谱加权并不脆弱：所有策略都保持着竞争力，且性能差异较小。其次，最佳加权方式取决于数据集：恒等加权（T4-标准）在 IMDB 上表现最佳，而线性加权在 AG News 上表现最佳，位于 d=256（附录 C）。这支持了这样一种解释： α_k 会引入频域偏置，而其最优形式取决于任务。



F图. 6.1. *IMDB*（左）和 *AG News*（右）上的编码器参数与测试准确率。误差线表示 ±1 个随机种子下的标准差 *s.* 张量 模型在使用约 4× 更少编码器参数的同时，达到了具有竞争力或更高的准确率，$p = 4$。

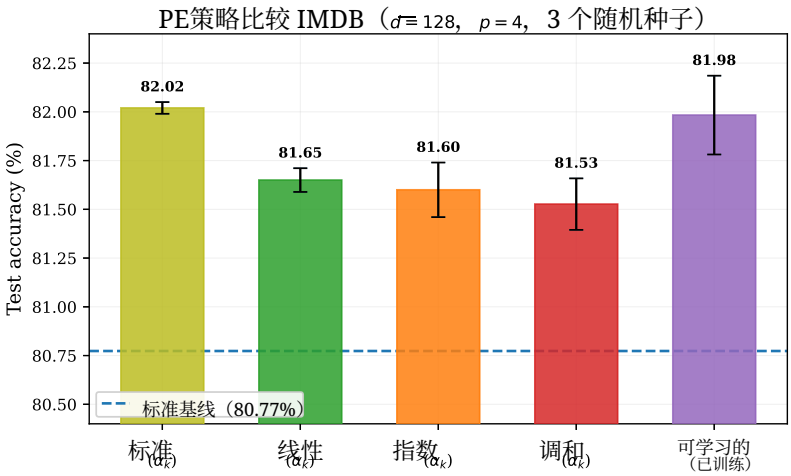


图 . 6.2. *IMDB* 上 $p = 4$ 时的位置编码策略比较（均值 ± 标准差，*3* 个随机种子）。虚线表示全宽标准基线。所有策略均优于标准基线，各策略间的总差异为 *0.49* 个百分点。

6.5 扩展至 BERT-base 维度 为测试该方法在更大宽度下的表现，我们在 AG News 上从头开始训练，使用d=768，其中 H=8、d_{ff}=3072 和 p=4（因此 d_s=192）。在这一宽度下，编码器占据了参数预算的主要部分，而 $1/p$ 编码器压缩可转化为显著的端到端节省。

表 6.9*AG News* 在 $d = 768$（*BERT-base*宽度）、$p = 4$时，从头开始训练，使用 *3* 个随机种子。张量模型将编码器压缩了 4×，并降低了峰值内存，同时保持准确率。

模型	准确率（%）	编码器	比率	Mem
Std	91.47 ± 0.09	28.4 M	1.00×	1.95 GB
T4-线性	**91.52 ± 0.21**	7.1 M	0.25×	1.66 GB

在 BERT-base宽度下，T4-线性与 Std 的结果在运行间变异性范围内一致（3 个随机种子），同时将编码器从 2,840 万个参数压缩至 710 万个参数（表 6.9）。不同于中等宽度，此时编码器压缩不再被嵌入层掩盖：模型总参数量从 5,140 万降至 3,020 万（表 6.10），峰值内存下降 15%。这正是结构化压缩

表6.10 在 $d = 768$ 时的参数细分。由于编码器占据了大部分参数，压缩编码器便会减少模型级别的参数量。

组件	Std	T4-线性
嵌入	23.0 M (45%)	23.0 M（76%）
编码器	28.4 M（55%）	7.1 M（24%）
总计	51.4 M	30.2 M (0.59×)

在实践中最为重要的场景。

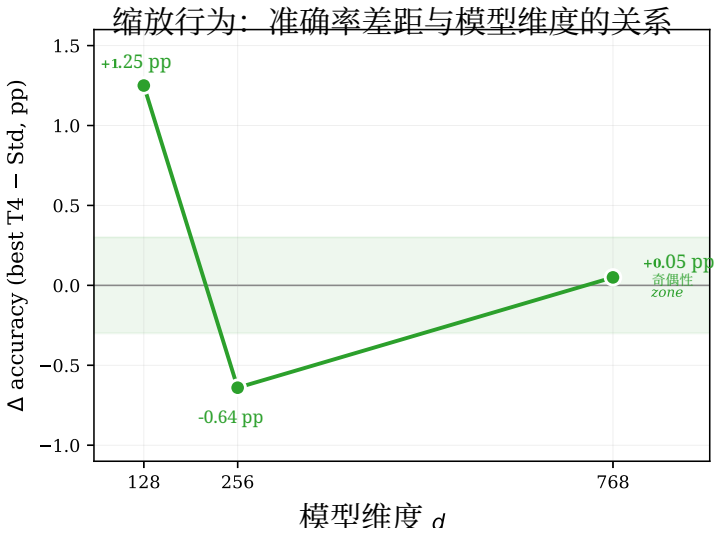


FIG. 6.3. 模型宽度 d 的函数关系下的准确率差距（最佳 *T4* 减去标准值）。阴影带表示 ±0.3 个百分点的一致性区域。当 $d = 768$ 时，张量模型达到一致性，同时将编码器压缩了 4×。

缩放趋势表明，单切片宽度 $d_s = d/p$ 是一个关键驱动因素。在 d=256 时，d_s=64 对 AG News 而言可能略显受限；在 d=768 时，d_s=192 似乎足以匹配标准值性能，同时保留 4× 的编码器压缩。这与理论相符：在变换域中，每个切片的行为都类似于一个紧凑型 Transformer，其容量取决于 d_s。

6.6 效率分析 表 6.11 报告了实际运行时间和内存测量结果，用于补充第 5.14 节中的 FLOPs 分析。所有测量均在单个 Tesla T4 GPU 上使用混合精度和序列长度 T=128 完成。在中等宽度（$d \leq 256$）下，尽管编码器削减了 4×，顺序切片执行仍会引入实际运行时间开销。在 d =768 下，计算量削减占据主导：每轮训练时间缩短，峰值内存也大幅减少。

6.7 研究结果总结 表 6.12 汇总了三种宽度下的主要结果。

在不同宽度下，张量 $\mathscr{L}$-乘积编码器实现了约 4× 编码器压缩，同时保持了有竞争力的准确率。在 IMDB 上，其性能提升了 1.25 个百分点，同时显著降低了方差。在 $d = 256$ 的 AG News 上，它以准确率降低 0.64 个百分点为代价，减少了 4× 编码器参数；参数匹配基线证实，这仍然是参数的一种高效用法。在 $d = 768$ 时，该方法达到奇偶持平，同时使模型总量减少 41%（表 6.10），峰值内存减少 15%（表 6.9）。综合来看，这些结果支持了主要论点：随着宽度增大，张量分解的效果愈发显著，而这恰恰是编码器压缩最有价值的应用场景。

7 结论 我们介绍了一种基于三阶张量 $\mathscr{L}$-乘积的 Transformer 多维表示学习框架。通过将词元嵌入从 $X \in \mathbb{R}^{T \times d}$ 重塑为 $\mathscr{X} \in \mathbb{R}^{T \times d_s \times p}$，其中 $d_s = d/p$，并在变换域中逐切片应用注意力和前馈运算，所得编码器在谱上等价于 p 个并行的紧凑型 Trans-

表 6.11相同深度下的效率比较（*4* 层，*3* 个随机种子的均值）。训练吞吐量按每轮次测量；推理使用完整测试集，并在混合精度下调用 *torch.no grad()*。

配置	模型	编码器参数	比率	训练令牌数/秒	训练轮次（秒）	峰值内存	批次
IMDB d=128	Std	991 开尔文	1.00×	123 开尔文	26 s	0.50 GB	128 / 256
	T4-线性	254 开尔文	0.26×	94 K	34 s	0.51 GB	128 / 256
AG News d=256	Std	3 949 开尔文	1.00×	177 开尔文	87 s	0.99 GB	128 / 256
	T4-线性	1 000 开尔文	0.25×	132 开尔文	116 秒	0.96 GB	128 / 256
AG News d=768	Std	28 352 开尔文	1.00×	60 K	257 秒	1.95 GB	64 / 128
	T4-线性	7 118 开尔文	0.25×	64 K	241 秒	**1.66 GB**	64 / 128

批次列：训练 / 推理批次大小。在$d \leq 256$下，尽管编码器削减了 4×，顺序切片执行仍会带来 ≈ 30%的实际运行时间开销。在$d = 768$下，计算量削减占主导：T4-线性每轮训练快 6%，且峰值内存少 15%。

TABLE 6.12所有配置的汇总（均值 ±标准，*3* 个随机种子）。Δ 表示最佳张量变体相较于标准方法的准确率差异

配置	Std	最佳 T4	Δ	编码节省
IMDB 天 $= 128$	$80.77_{\pm 0.24}$	$82.02_{\pm 0.03}$(标准)	+1.25	991 开尔文 → 254开尔文
AG News 天 $= 256$	$91.40_{\pm 0.31}$	$90.76_{\pm 0.13}$ (lin)	−0.64	3.95 M → 1.00 M
AG News $d = 768$	$91.47_{\pm 0.09}$	$91.52_{\pm 0.21}$(lin)	+0.05	28.4 M → 7.1 M

former，其作用于宽度为d_s的输入。这样可使编码器参数减少 $\approx 1/p$ （忽略偏置和归一化参数等低阶项），同时在逆变换后保留标准 Transformer 语义。

实证结果表明，在 IMDB 上，张量化编码器在采用 4× 编码器压缩时可取得更高的准确率，而参数量匹配的基线则证实，这些提升不能仅用参数量来解释。在中等宽度（$d = 256$）的 AG News 上，该方法呈现出明显的压缩与准确率权衡；而在 BERT-base 宽度（$d = 768$）下，张量模型在压缩编码器 4×并降低 GPU 内存峰值的同时，达到了与基线在统计意义上相当的表现。

该方法不会减少二次方级的注意力图项（计算和内存均随T^2增长），因此实际运行时间收益取决于具体场景和实现。特别是，在中等宽度下，顺序切片执行可能会引入开销；此外，目前该方法要求$p \mid d$，才能实现干净的张量化。

由此自然引出几个方向：(i) 将张量 $\mathscr{L}$-乘积编码器与高效注意力近似相结合，以解决 T^2 瓶颈；(ii) 实现全批量切片执行，以便更好地将理论上的 FLOP 降低转化为运行时间的改进；(iii) 将评估扩展到更广泛的基准和任务，包括预训练微调设置；(iv) 探索其他正交变换和可学习变换算子，以进一步使谱归纳偏置适应不同领域。

8 写作过程中对生成式人工智能和 AI辅助技术的声明 在准备这项工作期间，作者使用 ChatGPT 改进了LATEX 表格。使用该工具后，作者对内容进行了必要的审阅和编辑，并对出版物的内容承担全部责任。

9 利益冲突声明 我们特此声明，我们与其他个人或组织不存在任何可能不当影响（使我们的工作产生偏倚）的财务或个人关系。

附录 A. 补充理论与实现细节。

A.1 张量展开约定 我们遵循 [13] 中的标准张量记法和展开约定；所有推导仅依赖于n模乘积的标准单位恒等式：

$$(\mathscr{A} \times_n X)_{(n)} = X\,\mathscr{A}_{(n)}.$$

展开通过选择一个模作为行索引，并将其余所有模拼接为列，将张量转换为矩阵。这一操作很有用，因为许多多重线性恒等式（尤其是涉及n模乘积的恒等式）都可以通过标准矩阵乘法简洁地表示。

令 $\mathscr{A} \in \mathbb{R}^{n_1 \times \cdots \times n_N}$。模 n 展开 $\mathscr{A}_{(n)}$ 是如下矩阵

$$\mathscr{A}_{(n)} \in \mathbb{R}^{n_n \times \prod_{k \neq n} n_k},$$

它通过固定第n个指标，并按照一致的顺序枚举其余所有指标得到。

三阶情形。 对于 $\mathscr{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$，各展开满足

$$\mathscr{A}_{(1)} \in \mathbb{R}^{n_1 \times (n_2 n_3)}, \qquad \mathscr{A}_{(2)} \in \mathbb{R}^{n_2 \times (n_1 n_3)}, \qquad \mathscr{A}_{(3)} \in \mathbb{R}^{n_3 \times (n_1 n_2)}.$$

在本文中，模 3 展开尤为重要，因为 $\mathscr{L}$-变换沿管维度（模 3）作用。具体而言，将 $\mathscr{A}_{(3)}$ 写成列，即可将所有管 $\mathscr{A}(i,\ j,\ :) \in \mathbb{R}^{n_3}$ 按列堆叠。

当我们将词元嵌入张量化为 $\mathscr{X} \in \mathbb{R}^{T \times d_s \times p}$ 时，算子 $\mathscr{L}(\mathscr{X}) = \mathscr{X} \times_3 Z$ 可在展开形式下视为

$$\big(\mathscr{L}(\mathscr{X})\big)_{(3)} = Z\,\mathscr{X}_{(3)},$$

即，在管堆叠表示上进行标准矩阵乘法。这一视角阐明了该变换为何是线性的、可微的且易于高效实现。

A.2 详细说明 $\mathscr{L}$-多头注意力 </style}本附录将算法 1 展开为对 $\mathscr{L}$-多头注意力完全明确、面向实现的规范。其目标有两个方面。

(i) 可复现性与形状记录。 我们列出每个中间张量/矩阵及其维度，分别按切片 $(i = 1, \ldots, p)$ 和按头 $(j = 1, \ldots, h)$ 记录，包括各切片特定的投影查询/键/值、注意力分数矩阵，以及最终的逐切片输出投影。这消除了张量化注意力实现中常见的歧义来源（例如，头维度在哪里拆分、哪个轴被视为批次轴，以及如何执行拼接）。

（*ii*）逐切片等价性的操作含义。通过在 $\mathscr{L}$ 域中逐切片地表示计算过程，附录将等价性陈述具体化：对于每个固定切片 i，注意力更新恰好是对 $\widehat{X}^{(i)} \in \mathbb{R}^{T \times d_s}$ 应用标准缩放点积多头注意力，并使用切片专属的变换后参数。这阐明了该方法改变了什么（投影／前馈网络参数化，以及随$d_s = d/p$变化的计算缩放），以及没有改变什么（$T \times T$注意力图结构，以及关于T 的二次方评分计算）。

并行化说明。 尽管为清晰起见以显式切片循环呈现，详细表述也揭示了实践中采用的自然批处理策略：将切片索引（以及可选的头索引）视为批次维度，并通过批量矩阵乘法/融合注意力内核实现逐切片乘法。这是使变换域中理论上的逐切片独立性转化为高效 GPU 执行的关键步骤。

算法 2 $\mathscr{L}$-多头注意力（详细谱实现）

要求： 输入 $\mathscr{X} \in \mathbb{R}^{T\times d_s\times p}$；位置编码 $\mathscr{P} \in \mathbb{R}^{T\times d_s\times p}$；注意力头 h，$d_s = hd_h$；权重 $\{\mathscr{W}_{(j)Q}, \mathscr{W}_{(j)K}, \mathscr{W}_{(j)V}\}_{j=1}^{h}$ 其中 $\mathscr{W}_{(j)} \in \mathbb{R}^{d_s\times d_h\times p}$；输出 $\mathscr{W}_O \in \mathbb{R}^{d_s\times d_s\times p}$；可逆变换 $\mathbf{Z} \in \mathbb{R}^{p\times p}$。**确保：输出** $\mathscr{Y} \in \mathbb{R}^{T\times d_s\times p}$。**1：** $\mathscr{X}_{\text{pos}} \leftarrow \mathscr{X} + \mathscr{P}$ **2：** $\mathscr{X} \leftarrow \mathscr{X}_{\text{pos}} \times_3 \mathbf{Z}$ **3：对于 j** $= 1$ **到 h，执行 4：** $\widehat{\mathscr{W}}$**(j) Q** $\leftarrow \mathscr{W}$**(j) Q** $\times_3 \mathbf{Z}$ **5：** $\widehat{\mathscr{W}}$**(j) K** $\leftarrow \mathscr{W}$**(j) K** $\times_3 \mathbf{Z}$ **6：** $\widehat{\mathscr{W}}$**(j) V** $\leftarrow \mathscr{W}$**(j) V** $\times_3 \mathbf{Z}$ **7：结束循环 8：** $\mathscr{W}_O \leftarrow \mathscr{W}_O \times_3 \mathbf{Z}$ **9：对于 i** $= 1$ **到 p，执行 10：（遍历变换域切片；可并行化）11：** $\widehat{\mathbf{X}}^{(i)} \leftarrow \widehat{\mathscr{X}}$**(:,:,i)** $\in \mathbb{R}^{T\times d_s}$ **12：对于 j** $= 1$ **到 h，执行 13：** $\widehat{\mathbf{W}}$**(j) Q,i** $\leftarrow \widehat{\mathscr{W}}$**(j) Q (:, i)** $\in \mathbb{R}^{d_s\times d_h}$ **14：** $\widehat{\mathbf{W}}$**(j) K,i** $\leftarrow \widehat{\mathscr{W}}$**(j) K (:, i)** $\in \mathbb{R}^{d_s\times d_h}$ **15：** $\widehat{\mathbf{W}}$**(j) V,i** $\leftarrow \widehat{\mathscr{W}}$**(j) V (:, i)** $\in \mathbb{R}^{d_s\times d_h}$ **16：** $\widehat{\mathbf{Q}}^{(j)}$**i** $\leftarrow \widehat{\mathbf{X}}^{(i)}\,\widehat{\mathbf{W}}$**(j) Q,i** $\in \mathbb{R}^{T\times d_h}$ **17：** $\widehat{\mathbf{K}}^{(j)}$**i** $\leftarrow \widehat{\mathbf{X}}^{(i)}\,\widehat{\mathbf{W}}$**(j) K,i** $\in \mathbb{R}^{T\times d_h}$ **18：** $\widehat{\mathbf{V}}$**(j) i** $\leftarrow \widehat{\mathbf{X}}^{(i)}\,\widehat{\mathbf{W}}$**(j) V,i** $\in \mathbb{R}^{T\times d_h}$ **19：** $\widehat{\mathbf{S}}^{(j)}$**i** $\leftarrow \frac{1}{\sqrt{d_h}}\,\widehat{\mathbf{Q}}^{(j)}$**i (** $\widehat{\mathbf{K}}^{(j)}$**i)**$^{\top} \in \mathbb{R}^{T\times T}$ **20：** $\widehat{\mathbf{A}}^{(j)}$**i** $\leftarrow$ **Softmax(**$\widehat{\mathbf{S}}^{(j)}$**i)** $\in \mathbb{R}^{T\times T}$ **21：** $\widehat{\mathbf{head}}^{(j)}$**i** $\leftarrow \widehat{\mathbf{A}}^{(j)}$**i** $\widehat{\mathbf{V}}$**(j) i** $\in \mathbb{R}^{T\times d_h}$ **22：结束循环 23：** $\widehat{\mathbf{H}}_i \leftarrow$ **拼接(** $\widehat{\mathbf{head}}^{(1)}$**i** $\cdots$, $\widehat{\mathbf{head}}^{(h)}$**i)** $\in \mathbb{R}^{T\times d_s}$ **24：** $\widehat{\mathbf{W}}_{O,i} \leftarrow \widehat{\mathscr{W}_O}$**(:, :, i)** $\in \mathbb{R}^{d_s\times d_s}$ **ds 25：** $\widehat{\mathbf{Y}}_i \leftarrow \widehat{\mathbf{H}}_i\,\widehat{\mathbf{W}}_{O,i} \in \mathbb{R}^{T\times d_s}$ **26：结束循环 27：通过堆叠** $\{\widehat{\mathbf{Y}}_i\}_{\mathbf{i}=1}^{p}$ **作为正面切片，构造** $\widehat{\mathscr{Y}} \in \mathbb{R}^{T\times d_s\times p}$ **p 28：** $\mathscr{Y} \leftarrow \widehat{\mathscr{Y}} \times_3 \mathbf{Z}^{-1}$ **29：返回** $\mathscr{Y}$

A.3 逐切片等价性的流水线视图 图 A.1 从面向实现的角度展示了逐切片等价性结果，并明确呈现了数据流。一个张量编码器块可以理解为一个三步流水线：（i）沿管（模式 3）维度应用 $\mathscr{L}$ 变换，得到 p 个变换域正面切片 $\{\widehat{X}^{i}\}_{i=1}^{p}$；（ii）使用 p 个紧凑型标准 Transformer 块 $T^{(i)}$ std 独立处理这些切片，其宽度为 d^{s}（可并行执行，也可通过批处理内核执行）；（iii）堆叠各切片的输出，并应用 $\mathscr{L}^{-1}$ 以返回原始张量域。因此，该图阐明了 $\approx 1/p$ 编码器参数缩放的来源（每个子编码器处理的宽度为$d_s = d/p$），以及变换域中可实现的实际并行性。重要的是，该模型并非对嵌入维度进行静态划分：每个块之后，逆变换 $\mathscr{L}^{-1}$ 会以线性方式重新混合 p 个谱通道，因此输入下一层的切片是由上一层输出耦合而成的组合。由此，即使块内计算在 $\mathscr{L}$域中是逐切片进行的，信息仍可在深度方向跨切片流动。

A.4 关于 FFT/t-乘积与复值切片的说明本说明阐明：若使用傅里叶变换实例化 $\mathscr{L}$-乘积，即采用经典 t-乘积设定，会有哪些变化（以及哪些不会变化）。

复数变换域表示。当Z 取为离散傅里叶变换矩阵时，即使 $\mathscr{X}$为实数， $\mathscr{L}$-变换$\frown\mathscr{X} = \mathscr{X} \times_3 Z$ 通常也是复值的。因此，逐切片的“紧凑 Transformer”解释仍然成立，但所有线性代数运算都必须在 $\mathbb{C}$ 上理解，内积也必须采用厄米形式。

在 $\mathbb{C}$中的注意力是定义良好的（使用厄米转置）。在每个切片i内，注意力分数通过共轭转置计算：

$$\widehat{S}_i^{(j)} = \frac{1}{\sqrt{d_h}} \widehat{Q}_i^{(j)} (\widehat{K}_i^{(j)})^H,$$

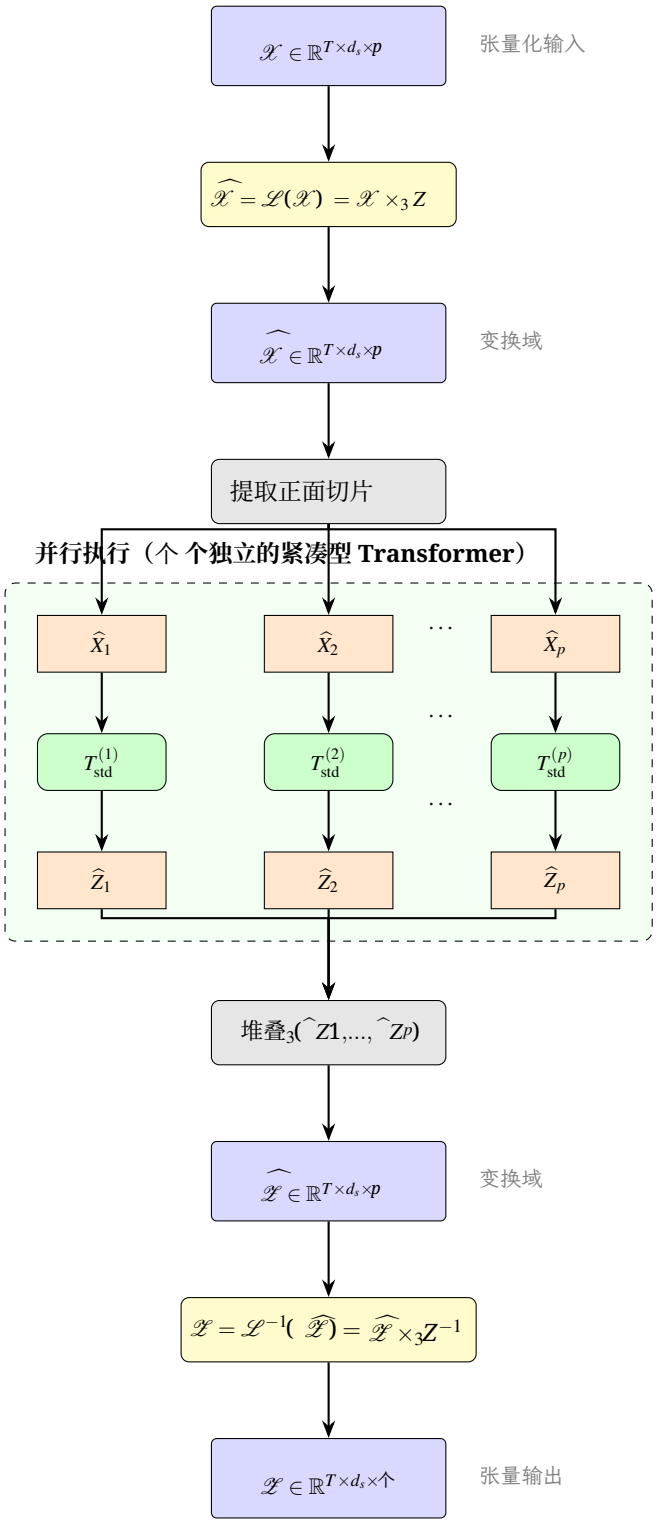
其余操作（Softmax、使用 $\widehat{V}_{(j)\,i}$进行加权、输出投影）则像实数情形一样逐切片进行。所得 $\widehat{\mathscr{Y}}$ 通常为复数；只有满足相应的共轭对称约束时，逆变换 $\mathscr{L}^{-1}(\widehat{\mathscr{Y}})$ 才能恢复实张量（如标准的基于 FFT 的实信号处理中那样）。

尽管快速傅里叶变换/张量积的选择在理论上十分自然且计算效率高，但在典型的深度学习技术栈中会带来一些实际问题：复值参数/激活值、复数分数上的 Softmax 可能存在歧义（通常通过取实部或幅值来处理），以及需要格外谨慎地确保输出为实值。因此，我们采用实正交变换（尤其是 DCT 型变换 [1]），使所有激活值和参数均保持为 $\mathbb{R}$，注意力计算与标准实值实现一致，并且所产生的频谱切片可直接解释为实值通道。此选择使该方法与标准训练流水线完全兼容，同时保留了在变换域中实现切片级解耦所需的关键代数性质。

附录 B. 训练曲线。 本附录通过展示基线与张量化编码器的优化动态，对第 6.1 节和第 6.3 节作补充。正文报告了跨随机种子汇总的最终性能（准确率/F1），而这些曲线有助于验证：(i) 所报告的改进并非源于训练不稳定；(ii) 谱加权主要影响收敛行为，而非造成运行过程反复波动。所有曲线均展示一个具有代表性的随机种子（随机种子 42）的结果；其余随机种子也呈现相同的定性趋势。

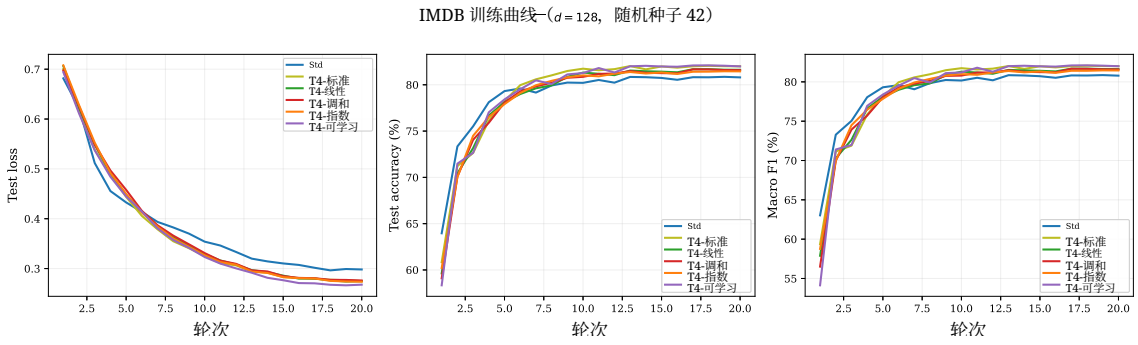
每幅图展示了主要评估指标（准确率和宏平均 F1）以及损失随训练轮次的变化。IMDB 实验使用 20 个训练轮次，而 AG News（$d = 256$）使用 5 个训练轮次，与表 6.4 所述的训练计划一致。

在各种 PE策略下，张量模型都能迅速进入稳定的高准确率状态，并将其维持下来，波动幅度有限，这与表 6.7 报告的较低种子间方差一致。此外，最终指标中观察到的 PE策略相对排序，在训练早期就已显现，这表明性能提升并非源于训练后期轮次的噪声。所有模型都在短短 5 轮的预算内达到峰值，因此与 IMDB 相比，收敛差异不太可能特别显著。张量变体保持稳定，而表 6.8 报告的准确率差距较小，这与曲线表现一致：在当前宽度下，张量模型以少量准确率为代价，大幅减少了编码器参数。



图A.1. 在 $\mathscr{L}$-积框架下张量 *Transformer* 的逐切片等价性。张量沿模式 *3* 进行变换，由 p 个紧凑型 *Transformer*（维度 d_s）逐切片独立处理，随后重新堆叠，并通过 $\mathscr{L}^{-1}$ 映射回原始域。

附录 C. PE 扫描完整结果。表 C.1 报告了在固定$p = 4$时，五种谱加权方案的完整对比。此处有两个目的：(i) 支持正文中的主张，即谱加权



图IG. B.1. *IMDB* 上的训练曲线（$d = 128$，随机种子 *42*）。

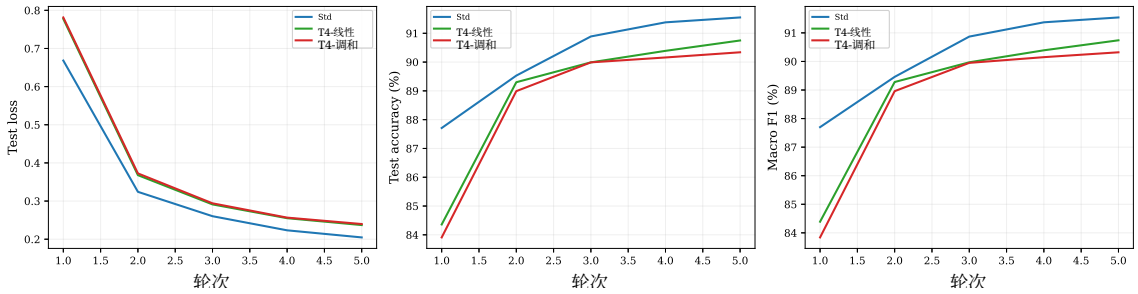


FIG. B.2. *AG News* 上的训练曲线（$d = 256$，随机种子 *42*）。

matters but is not brittle，以及 (ii) 记录各种策略的相对排名，而不局限于正文中展示的单张图。

每个元素表示 3 个随机种子下的平均测试准确率 ±标准差，“标准”表示全宽基线。粗体表示此处报告的各次运行中，每个数据集上表现最佳的张量策略。缺失元素应理解为“未运行/不可用”（如有可能，应补充这些数据以实现完整覆盖）。

TABLE C.1$p = 4$ 时的完整位置编码扫描（均值 ± 标准差，*3* 个随机种子）。各数据集的最佳结果以 粗体标示。

位置编码策略	**IMDB** $d = 128$	**AG News** $d = 256$
T4-标准 ($\alpha_k = 1$)	$\mathbf{82.02 \pm 0.03}$	—
T4-线性 ($\alpha_k = k/p$)	81.65 ± 0.06	$\mathbf{90.76 \pm 0.13}$
T4-指数 ($\alpha_k = 2^{(k-1)/(p-1)}$)	81.60 ± 0.14	—
T4-调和 ($\alpha_k = k$)	81.53 ± 0.13	90.41 ± 0.08
T4-可学习	81.98 ± 0.20	—
标准（基线）	80.77 ± 0.24	91.40 ± 0.31

在 IMDB 上，所有 PE策略的表现都优于标准方法，且各策略之间的差距不到一个百分点，这表明张量化是主要驱动因素，而加权则提供了次要的调优手段。在 AG News 上，$d = 256$时，线性加权在所报告的张量化实验中表现最佳。总体而言，结果支持正文中的结论：T4-线性是一个稳健的默认选择，而最佳策略仍可能因数据集而异。

附录D. 运行时间详情。 表D.1列出了墙钟时间和内存测量结果，以补充第6.6节中的 FLOPs 分析。其目的是阐明：(i) 张量分解减少了哪些内容（编码器参数和编码器浮点运算次数）；以及 (ii) 哪些方面目前受实现细节限制（顺序切片执行）。

“Train”表示每轮训练的平均墙钟时间，“Mem”表示运行期间 GPU 内存分配的峰值，“Infer”表示对整个测试集进行一次前向传播所需的墙钟时间。所有测量均在同一图形处理器（Tesla T4）上以混合精度进行。由于 AG News 在$d = 768$ 处使用的批次大小与较小配置不同，因此比较绝对推理时间时，应主要在相同配置内进行。

表格 D.1运行时间详情。训练为每轮的实际运行时间均值；内存为图形处理器内存分配峰值；推理为完整测试集上的前向传播。所有运行均使用 *NVIDIA Tesla T4* 和混合精度。

配置	模型	Enc.	训练（秒/轮）	内存（GB）	推理（秒）
IMDB ($d = 128$)	Std	991 开尔文	26	0.50	11.8
	T4-标准	254 开尔文	34	0.51	12.7
	T4-线性	254 开尔文	34	0.51	12.6
	T4-调和	254 开尔文	34	0.51	12.8
	T4-可学习	270 开尔文	34	0.51	12.8
	T2-线性	500 开尔文	29	0.49	11.9
AG News ($d = 256$)	Std	3.95 M	87	0.99	3.0
	T4-线性	1.00 M	116	0.96	3.3
	T4-调和	1.00 M	116	0.96	3.2
AG News ($d = 768$)	Std	28.4 M	260	1.95	5.0
	T4-线性	7.1 M	241	1.66	5.2

张量编码器实现了预期的参数量/浮点运算次数缩减，但当前实现会按顺序执行p 子编码器，这会导致较小宽度下的每轮训练时间增加（IMDB 和 AG$d = 256$）。相比之下，在较大宽度下峰值内存已经得到改善（$d = 768$），此时张量模型可将内存分配减少 15%。这表明了一个明确的工程优化方向：切片批处理（将切片索引视为批次维度）应能使浮点运算次数的减少转化为实际运行时间改进，尤其是在计算开销占主导地位的较大宽度下。

实际运行时间开销。 当前张量实现按顺序处理 p 个子编码器，因此无法在实际运行时间上充分发挥 $1/p$ 浮点运算次数的缩减效果。在 IMDB 上，这意味着每轮耗时 34 秒，而标准模型为 26 秒。在 AG News 上，当 $d = 256$ 时，每轮耗时 116 秒，而标准模型为 87 秒。

峰值内存。 在中等宽度下，峰值内存相近；在较大宽度下则有所改善。当 $d = 768$ 时，张量模型将峰值内存分配降低了 15%（1.66 GB，而标准模型为 1.95 GB），在 GPU 内存预算紧张时，这一改善具有实际意义。

切片批处理的潜力。 切片批处理实现（将切片维度视为批次维度）可以消除顺序执行的开销。鉴于编码器中的浮点运算次数减少了 $4\times$，预计在中等到较大的宽度下，批量执行的墙钟时间可与 Std 相当或优于 Std。

参考文献

[1] Nasir Ahmed、T. Natarajan 和 K. R. Rao。离散余弦变换。《IEEE计算机汇刊》，C-23(1):90–93，1974年。[2] Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell 等。语言模型是少样本学习者。《神经信息处理系统进展》，33:1877–1901，2020年。[3] Krzysztof Choromanski、Valerii Likhosherstov、David Dohan、Xingyou Song、Andreea Gane、Tamas Sarlos、Peter Hawkins、Jared Davis、Afroz Mohiuddin、Lukasz Kaiser 等。重新思考 Performer 中的注意力。arXiv预印本 arXiv:2009.14794，2020年。[4] Tri Dao、Dan Fu、Stefano Ermon、Atri Rudra 和 Christopher R´e。Flashattention：具备 IO感知能力的快速、内存高效精确注意力。《神经信息处理系统进展》，35:16344–16359，2022年。[5] Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。BERT：用于语言理解的深度双向变换器预训练。载于《计算语言学协会北美分会第2019年会议：人类语言技术论文集》，第1卷（长篇与短篇论文），第4171–4186页，2019年。[6] Frank L Hitchcock。张量或多元张量表示为乘积之和。《数学与物理学杂志》，6(1-4):164–189，1927年。[7] Edward J Hu、Yelong Shen、Phillip Wallis、Zeyuan Allen-Zhu、Yuanzhi Li、Shean Wang、Liang Wang、Weizhu Chen 等。LoRA：大语言模型的低秩适配。ICLR，1(2):3，2022年。[8] Xiaoqi Jiao、Yichun Yin、Lifeng Shang、Xin Jiang、Xiao Chen、Linlin Li、Fang Wang 和 Qun Liu。TinyBERT：为自然语言理解蒸馏 BERT。载于《计算语言学协会论文集：EMNLP 2020》，第4163–4174页，2020年。[9] Eric Kernfeld、Misha Kilmer 和 Shuchin Aeron。具有可逆线性变换的张量-张量乘积。《线性代数及其应用》，485:545–570，2015年。[10] Misha E Kilmer、Karen Braman、Ning Hao 和 Randy C Hoover。作为矩阵算子的三阶张量：用于成像应用的理论与计算框架。《SIAM矩阵分析与应用杂志》，34(1):148–172，2013年。[11] Misha E. Kilmer 和 Carla D. Martin。三阶张量的分解策略。《线性代数及其应用》，435(3):641–658，2011年。[12] Nikita Kitaev、Łukasz Kaiser 和 Anselm Levskaya。Reformer：高效的 Transformer。arXiv预印本 arXiv:2001 04451，2020年。。[13] Tamara G. Kolda 和 Brett W. Bader。张量分解及其应用。《SIAM评论》，51(3):455–500，2009年。[14] Zhenzhong Lan、Mingda Chen、Sebastian Goodman、Kevin Gimpel、Piyush Sharma 和 Radu Soricut。ALBERT：一种用于语言表示自监督学习的轻量级 BERT。arXiv预印本 arXiv:1909.11942，2019年。[15] James Lee-Thorp、Joshua Ainslie、Ilya Eckstein 和 Santiago Onta˜n´on。FNet：用傅里叶变换混合词元，2021年。[16] Yinhan Liu、Myle Ott、Naman Goyal、Jingfei Du、Mandar Joshi、Danqi Chen、Omer Levy、Mike Lewis、Luke Zettlemoyer 和 Veselin Stoyanov。RoBERTa：一种鲁棒优化的 BERT 预训练方法。arXiv预印本 arXiv:1907.11692，2019年。[17] Ilya Loshchilov 和 Frank Hutter。解耦权重衰减正则化。arXiv预印本 arXiv:1711.05101，2017年。[18] Y. Luo、H. Patel、Y. Fu、D. Ahn、J. Chen、Y. Dong 和 E. E. Papalexakis。TRAWL：面向大语言模型的张量约简与近似权重。arXiv预印本，2024年。[19] Yiran Luo、Het Patel、Yu Fu、Dawon Ahn、Jia Chen、Yue Dong 和 Evangelos E Papalexakis。TRAWL：面向大语言模型的张量约简与近似权重。载于《亚太知识发现与数据挖掘会议》，第402–413页。施普林格，2025年。[20] Andrew Maas、Raymond E Daly、Peter T Pham、Dan Huang、Andrew Y Ng 和 Christopher Potts。用于情感分析的词向量学习。载于《计算语言学协会第49届年会：人类语言技术论文集》，第142–150页，2011年。[21] Paulius Micikevicius、Sharan Narang、Jonah Alben、Gregory Diamos、Erich Elsen、David Garcia、Boris Ginsburg、Michael Houston、Oleksii Kuchaiev、Ganesh Venkatesh 等。混合精度训练。arXiv预印本 arXiv:1710.03740，2017年。[22] Alexander Novikov、Dmitry Podoprikhin、Anton Osokin 和 Dmitry P. Vetrov。神经网络张量化，2015年。[23] Ivan V. Oseledets。张量列车分解。《SIAM科学计算杂志》，33(5):2295–2317，2011年。[24] Colin Raffel、Noam Shazeer、Adam Roberts、Katherine Lee、Sharan Narang、Michael Matena、Yanqi Zhou、Wei Li 和 Peter J Liu。通过统一的文本到文本 Transformer 探索迁移学习的极限。《机器学习研究杂志》，21(140):1–67，2020年。[25] Victor Sanh、Lysandre Debut、Julien Chaumond 和 Thomas Wolf。DistilBERT：BERT 的蒸馏版本，更小、更快、更便宜、更轻量。arXiv预印本 arXiv:1910.01108，2019年。[26] Rico Sennrich、Barry Haddow 和 Alexandra Birch。使用子词单元进行低频词神经机器翻译。载于《计算语言学协会第54 1725届年会（第：长篇论文卷）论文集》，第 – 页，2016年 1 1715。[27] Pratyusha Sharma、Jordan T. Ash 和 Dipendra Misra。真相就在其中：通过层选择性秩约简提升语言模型的推理能力。载于《第十二届国际学习表征会议论文集》（ICLR），2024年。[28] Leslie N Smith。用于训练神经网络的周期性学习率。载于《2017年 IEEE冬季计算机视觉应用会议》（WACV），第464–472页。IEEE，2017年。[29] Leslie N Smith 和 Nicholay Topin。超收敛：使用大学习率快速训练神经网络。载于《多域作战应用中的人工智能与机器学习》，第11006卷，第369–386页。国际光学工程学会，2019年。[30] Marzieh Tahaei、Ella Charlaix、Vahid Nia、Ali Ghodsi 和 Mehdi Rezagholizadeh。KroneckerBERT：通过克罗内克分解和知识蒸馏显著压缩预训练语言模型。载于《计算语言学协会北美分会2022年会议：人类语言技术论文集》，

第2116–2127页，2022年。[31] Hugo Touvron、Thibaut Lavril、Gautier Izacard、Xavier Martinet、Marie-Anne Lachaux、Timoth´ee Lacroix、Baptiste Rozi`ere、Naman Goyal、Eric Hambro、Faisal Azhar、Aurelien Rodriguez、Armand Joulin、Edouard Grave 和 Guillaume Lample。Llama：开放且高效的基础语言模型。*arXiv* 预印本 *arXiv:2302.13971*，2023年。[32] Ledyard R Tucker。关于三模因子分析的一些数学注记。*Psychometrika*，31(3):279–311，1966年。[33] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、Lukasz Kaiser 和 Illia Polo- sukhin。Attention is all you need。载于神经信息处理系统进展，2017年。[34] Sinong Wang、Belinda Z Li、Madian Khabsa、Han Fang 和 Hao Ma。Linformer：具有线性复杂度的自注意力。*arXiv* 预印本 *arXiv:2006.04768*，2020年。[35] Liyuan Yu、Xiao Yao、Ye Liu、Xipeng Xiao 和 Zhiyuan Liu。十六个注意力头真的比一个好吗？神经信息处理系统进展，33:14014–14024，2020年。[36] Xiang Zhang、Junbo Zhao 和 Yann LeCun。用于文本分类的字符级卷积网络。神经信息处理系统进展，28，2015年。