计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(1): 162−175, 2026

一种自注意力模块的低精度损失量化方法

林德铝　何　琨
（华中科技大学计算机科学与技术学院　武汉　430074）
（derrylin@hust.edu.cn）

A Low-Precision Loss Quantification Method for Self-Attention Module

Lin Delü and He Kun
（*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan* 430074）

Abstract　With the rapid progress of deep learning technology and the continuous exploration of massive datasets, the self-attention module has been widely applied in various fields such as natural language processing, computer vision, and large language models. Although the self-attention module significantly improves the detection accuracy of deep learning models, its huge computational demand makes it particularly difficult to deploy on computing devices with limited computing power. Integer quantization, as one of the key technologies for deploying models on low-power computing chips, faces the problem of high precision loss caused by the structural characteristics of the self-attention module. To address this issue, a thorough analysis of the integer quantization error in the self-attention module is conducted, and two methods, pseudo-softmax vector quantization and block-wise pseudo-softmax vector quantization, are proposed. These two methods aim to significantly improve inference speed while effectively reducing the error caused by integer quantization by performing special integer quantization on the softmax vectors in the self-attention module. Experimental results show that compared with traditional direct quantization methods, the pseudo-softmax vector quantization method can reduce the quantization accuracy loss by 50%, while the block-wise pseudo-softmax vector quantization method can further reduce the accuracy loss by approximately 90%. These results fully demonstrate the effectiveness of the two quantization methods in reducing precision loss, providing strong support for the efficient deployment of the self-attention module on devices with limited computing power.

Key words　model quantization; self-attention module; low-precision loss; inference acceleration; divide and conquer

摘　要　随着深度学习技术的飞速进步和对海量数据集的持续发掘，自注意力模块在自然语言处理、计算机视觉以及大语言模型等多个领域得到了广泛应用。尽管自注意力模块显著提升了深度学习模型的检测精度，其巨大的计算需求却使得其在算力受限的计算设备上部署显得尤为困难。整数量化作为在低算力计算芯片中部署模型的关键技术之一，面临着由自注意力模块结构特点引起的较高精度损失问题。针对这个问题，对自注意力模块的整数量化误差进行了深入分析，提出了伪 softmax 向量量化方法和分块伪 softmax 向量量化方法。所提出方法通过对自注意力模块中的 softmax 向量进行特殊的整数量化，旨在显著提升推理速度的同时，有效降低整数量化带来的误差。实验结果表明，相比于传统的直接量化方法，伪 softmax 向量量化方法能够将量化精度损失降低 50%，而分块伪 softmax 向量量化方法更是能将精度损失减少约 90%。该结果充分证明了这 2 种量化方法在减少精度损失方面的有效性，为自注意力模块在算力受限设备上的高效部署提供了有力支持。

收稿日期：2024-10-19；**修回日期**：2025-05-19
基金项目：国家自然科学基金项目（62076105）
This work was supported by the National Natural Science Foundation of China (62076105).
通信作者：何琨（brooklet60@hust.edu.cn）

关键词 模型量化；自注意力模块；低精度损失；推理加速；分治

中图法分类号 TP181

DOI: 10.7544/issn1000-1239.202440804 **CSTR:** 32373.14.issn1000-1239.202440804

随着深度学习技术的迅猛发展，基于神经网络设计的模型在学术界和工业界均得到了广泛的关注和应用。为了解决传统深度学习模型在处理序列数据时的不足，基于自注意力(self-attention)设计的Transformer模型[1]应运而生，并迅速在自然语言处理任务如机器翻译[2]、文本摘要[3]、人机对话[4]等领域中崭露头角。随后，Dosovitskiy等人[5]将Transformer模型的创新理念引入计算机视觉领域，提出了具有划时代意义的ViT(vision Transformer)模型。ViT模型在ImageNet[6]竞赛中一举超越了传统的卷积神经网络，预示着它将成为未来几年计算机视觉领域的研究焦点。同时，大语言模型的发展也离不开Transformer结构的支撑，主流的大语言模型如GPT[7-8]，LLaMA[9]均是以Transformer为核心进行设计的。

然而，随着Transformer结构的广泛应用，如何在工业界实现其轻量化部署成为了一个亟待解决的问题。在工业界的模型部署实践中，模型整数量化已成为主流的轻量化部署手段。这种方法通过线性映射函数将模型的浮点数运算转换为低比特整数运算，从而在边缘计算设备上进行高效的低比特整数线性运算，显著提升了模型的部署效率。在卷积神经网络和全连接线性层中，通过网络融合的方式，将卷积神经网络或全连接层与批归一化[10]和ReLU激活函数合并为一个整体，再对整体进行低比特整数量化运算，从而大幅提高基于卷积神经网络和全连接层设计的模型推理效率。

对于Transformer模型而言，由于其包含了大量的自注意力模块，这些模块的量化过程往往面临较大的挑战。自注意力模块通过查询(query)矩阵$\boldsymbol{Q}$和键(key)矩阵$\boldsymbol{K}$的相乘，并经过softmax函数进行归一化，再与值(value)矩阵$\boldsymbol{V}$相乘得到加权输出。由于softmax函数的非线性特性，自注意力模块的量化往往会导致较大的精度损失。在工业界的Transformer模型部署场景中，自注意力模块部分通常仍采用浮点数进行计算，这在一定程度上限制了整体模型的量化率和部署计算效率。因此，开展对自注意力模块的低精度损失整数量化方法的研究显得尤为重要。

针对上述问题，本文深入剖析了自注意力模块量化误差的成因，并提出了一种针对性的高效量化方法。这种方法能够显著提升自注意力模块的量化精度，从而优化模型的部署效率。此外，本文还将这种自注意力模块的量化方法在英伟达的图形处理器GPU中进行了实现，并通过一系列严谨的算子实验验证了所提出方法的有效性，为Transformer模型的轻量化部署提供了有力的理论和技术支持。

本文的主要贡献包括2个方面：

1)深入分析并指出了自注意力模块的直接量化存在的误差问题，提出了伪softmax向量量化和分块伪softmax向量量化方法；

2)在英伟达的图形处理器中实现了这些量化方法的算子，在平均绝对误差和耗时2个指标上验证了所提量化方法的有效性。

1 相关工作

模型的整数量化是一种普遍应用的轻量化部署技术，在模型推理过程中扮演着至关重要的角色。通过采用较低的数值精度，如8 b整数(integer 8，INT8)进行推理，能够显著降低显存占用和延迟，进而提升数据吞吐率。这一技术主要是通过将原始的深度神经网络高精度浮点数据类型转换为低比特定点数据类型实现，常见的低比特定点数据类型有8 b整数和4 b整数等。低比特整数量化推理有2个主要优点：一是可以更快地通过内存层次结构移动数据，以最大限度地利用计算资源；二是在较低的数值精度下硬件可以实现更高的每秒运算次数，因为这些乘法器需要更少的硅面积和功率。

8 b整数量化推理作为定点数量化推理的代表性技术，在深度学习模型推理中广泛应用。8 b整数量化推理是指使用8 b整数格式进行深度学习模型推理，以实现更高的计算效率和能效。随着深度学习算法和模型应用范围的日益扩大，对高效、节能的推理计算需求也急剧增长，这使得8 b整数推理技术得到了广泛的研究和应用。

量化通常针对线性算子设计，Jacob等人[11]提出了8 b整数量化方法，使深度学习神经网络的线性算子仅依赖整型数据进行计算，不仅将存储空间缩减至原来的1/4，还显著提升了推理效率。Sun等人[12]

则尝试使用 8 b 浮点数(floating point 8, FP8)进行量化,以减轻量化带来的推理精度损失,但这通常需要硬件架构的支持。另外,Lin 等人[13]提出的 AWQ 量化方法,通过对线性层权重矩阵进行分组量化,实现了以 4 b 整数进行存储和计算。Dettmers 等人[14]则提出了一种全新的 4 b 数据格式 NormalFloat,它更好地适应了高斯分布数据,有效降低了模型量化带来的精度损失。

尽管线性算子的量化取得了显著成果,但直接将其应用于包含自注意机制的模型往往会导致精度显著下降。因此,针对这些模块,需要研究如何在提升速度的同时保证精度。例如,Lin 等人[15]提出的 FQ-ViT 模型使用了 Log-Int-Softmax 模块,以替代自注意力模块中的 softmax 模块,使其能够进行 8 b 整数推理。然而,这种方法改变了模型结构,需要重新进行微调训练,可能导致模型整体精度下降。Yuan 等人[16]提出了一种基于孪生均匀量化(twin uniform quantization)方法的模型 PTQ4ViT,将视觉自注意力模块中的激活值分为 2 个范围,并对每个范围进行单独的均匀量化。这种量化方式相对于线性量化而言,计算复杂度更高,可能降低 8 b 整数量化的计算效率。

从计算硬件设备的角度,Wang 等人[17]利用数学变换,将 softmax 整数量化的指数和对数计算转换为有限输入范围的计算以及一系列简单的计算,比如常数乘法运算、移位运算、加法运算等。但是该方法需要硬件结构支持才能发挥最大作用,在当下主流的边缘计算推理芯片中不具备通用性。Vasyltsov 等人[18]基于分段常数近似和查找表(look-up table, LUT)提出了一种高效的 softmax 计算方法,利用了整数量化的精度,可以与大部分传统的硬件加速器兼容。但是该方法的资源消耗以存储为主,需要利用高速而昂贵有限的片上存储。

在算法层面上,张强等人[19]提出了一种基于余弦相似性的 softmax 损失函数 CS-Softmax,该函数在分类任务中表现出了优异的性能。但是其依旧是采用浮点数进行计算,无法提升模型推理速度。

针对上述挑战,本文提出了一种针对自注意力模块的量化方法。它可以在不改变 softmax 算法和线性量化方式的前提下,在通用的图形处理器推理设备上支持 8 b 量化。同时,本文以图 1 所示的常见的 softmax 向量分布为例,对其进行量化误差分析,论证所提出的量化方法可以有效控制 softmax 向量的量化精度损失。

2 自注意力模块量化误差分析

通常自注意力模块算子的输入为查询矩阵 $\boldsymbol{Q}$、键矩阵 $\boldsymbol{K}$ 和值矩阵 $\boldsymbol{V}$,算子计算主要包含 2 个批量矩阵乘以及 1 个 softmax 函数,其计算方式如式(1)所示。

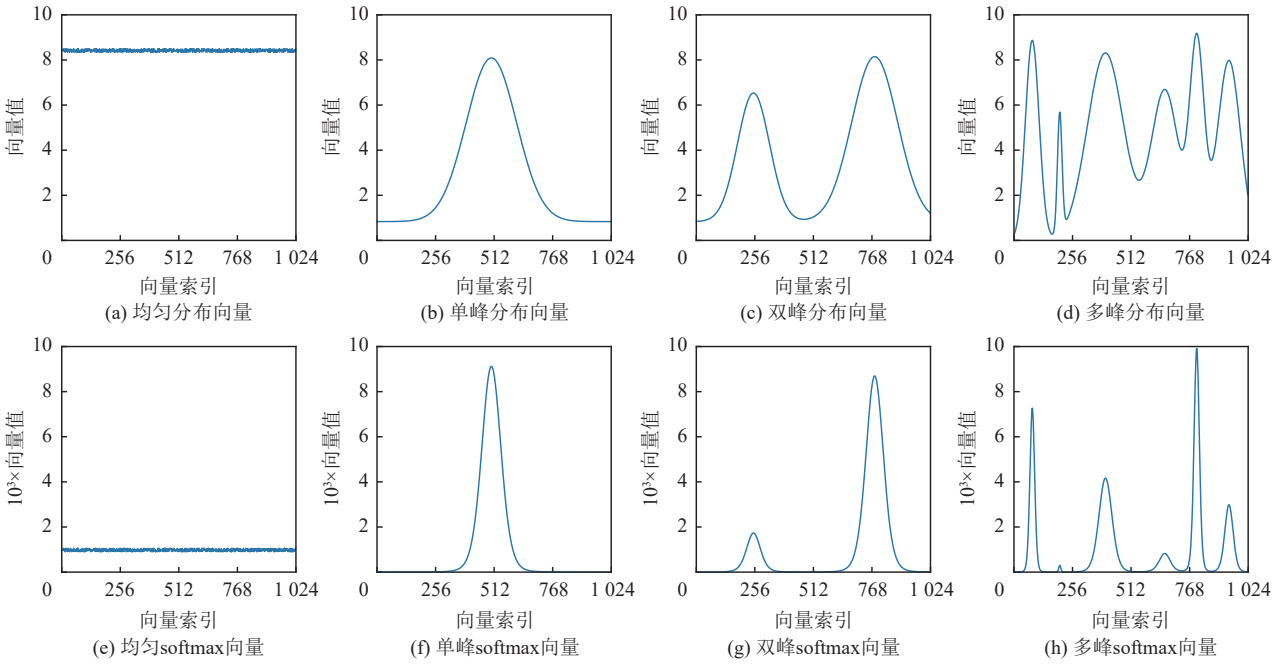


Fig. 1 Original vector and softmax vector diagram

图 1 原始向量及 softmax 向量图

$$\begin{aligned}
&\boldsymbol{Q}\in\mathbb{R}^{B\times L_Q\times d},\\
&\boldsymbol{K},\boldsymbol{V}\in\mathbb{R}^{B\times L_{KV}\times d},\\
&softmax(\boldsymbol{x})=\frac{\exp(\boldsymbol{x})}{\sum\exp(\boldsymbol{x})},\\
&\boldsymbol{S}=softmax\left(mm\left(\frac{\boldsymbol{Q}}{\sqrt{d}},\boldsymbol{K}^{\mathrm{T}}\right)\right),\\
&attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=mm(\boldsymbol{S},\boldsymbol{V})\ 。
\end{aligned}\tag{1}$$

其中 $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$ 分别代表查询矩阵、键矩阵和值矩阵；B 为数据的批大小，L_Q 和 L_{KV} 分别为 $\boldsymbol{Q}$ 以及 $\boldsymbol{K}$ 或 $\boldsymbol{V}$ 的序列长度，即键矩阵和值矩阵的序列长度均为 L_{KV}；d 为每个序列的通道大小，$mm(\cdot)$为批量矩阵乘。

在自注意力模块的量化推理实践中，考虑到 softmax 的输出矩阵范围固定在 [0, 1] 之间，且理论上该范围内的任意值都是可能达到的。因为无符号 8 b 整数的范围为 [0, 255]，所以 softmax 输出向量的缩放系数通常被固定为 1/255，这种固定值的做法是造成自注意力模块量化误差的主要原因。为了解决这个问题，工业界常见的方案是放弃对自注意力模块中最后 1 个矩阵乘法的量化，即只量化第 1 个矩阵乘法运算。该做法虽然能够减少量化误差，但却降低了整体的量化率，无法充分发挥量化带来的性能提升和资源优势。通常，为了避免指数运算的溢出，softmax 函数在计算机内的计算方式如式(2)所示。

$$\begin{aligned}
&m(\boldsymbol{x})=\max_i x_i,\\
&f(\boldsymbol{x})=\left[\exp(x_1-m(\boldsymbol{x})),\cdots,\exp(x_n-m(\boldsymbol{x}))\right],\\
&l(\boldsymbol{x})=\sum f(\boldsymbol{x}),\\
&softmax(\boldsymbol{x})=\frac{f(\boldsymbol{x})}{l(\boldsymbol{x})},
\end{aligned}\tag{2}$$

即在计算指数运算前会先减掉向量的最大值，使得指数运算输出范围在 [0, 1]，然后再除以向量和，这与原始的 softmax 计算等价。

然而，由于 softmax 的输出特性，其指数运算无法只用定点数进行计算，只能使用浮点数，需要对其输出进行量化方可在之后的计算使用定点数运算。其输出范围为 [0, 1]，若直接将该范围用作计算缩放系数，则缩放系数为 1/255。现考虑对一个 1 024 维的向量直接进行计算并量化输出，显然，量化输出的向量取值范围为 [0, 255] 的整数，即将浮点数向量除以量化缩放系数再进行取整，如式(3)所示。

$$s_{\mathrm{s}}=\frac{1}{255},\ \boldsymbol{x}^{\mathrm{s}}_{\mathrm{INT8}}=\left\lfloor\frac{\boldsymbol{x}^{\mathrm{s}}_{\mathrm{f}}}{s_{\mathrm{s}}}\right\rceil,\tag{3}$$

其中 s_{s} 为 softmax 输出向量的 8 b 量化缩放系数，$\boldsymbol{x}^{\mathrm{s}}_{\mathrm{f}}$ 和 $\boldsymbol{x}^{\mathrm{s}}_{\mathrm{INT8}}$ 分别为 softmax 输出的浮点数向量和 8 b 整数量化向量，$\lfloor\cdot\rceil$为四舍五入取整。

现考虑对几种常见的 softmax 分布进行误差分析，以论证不同量化方式对 softmax 量化误差的影响。如图 1 所示，图 1(a)~(d)为原始 1 024 维向量，包括均匀、单峰、双峰及多峰分布形态的向量；而图 1(e)~(h)为对应向量经过 softmax 计算后的向量。其中，横坐标为对应向量的位置索引，纵坐标为向量值。

若直接对该 softmax 向量进行 8 b 整数量化，将其输出与量化前的 softmax 向量组合可得到量化误差图，如图 2 所示。其中，横坐标为 1 024 维向量的索引，纵坐标为向量对应位置的值，实线为 softmax 浮点数输出的分布，虚线为 softmax 量化输出的分布，填充区域代表这 2 种分布之间的误差，δ_{mean}，δ_{max}，δ_{min} 分别为平均绝对误差、最大绝对误差和最小绝对误差，其由式(4)计算得到。

$$\begin{aligned}
\delta_{\mathrm{mean}}&=\frac{\sum_{i=1}^{n}\left|x^{\mathrm{s}}_{\mathrm{f}(i)}-s_{\mathrm{s}}x^{\mathrm{s}}_{\mathrm{INT8}(i)}\right|}{n},\\
\delta_{\mathrm{max}}&=\max_{1\leqslant i\leqslant n}\left|x^{\mathrm{s}}_{\mathrm{f}(i)}-s_{\mathrm{s}}x^{\mathrm{s}}_{\mathrm{INT8}(i)}\right|,\\
\delta_{\mathrm{min}}&=\min_{1\leqslant i\leqslant n}\left|x^{\mathrm{s}}_{\mathrm{f}(i)}-s_{\mathrm{s}}x^{\mathrm{s}}_{\mathrm{INT8}(i)}\right|,
\end{aligned}\tag{4}$$

其中 $x^{\mathrm{s}}_{\mathrm{f}(i)}$ 和 $x^{\mathrm{s}}_{\mathrm{INT8}(i)}$ 分别为 softmax 浮点数向量 $\boldsymbol{x}^{\mathrm{s}}_{\mathrm{f}}$ 和量化整数向量 $\boldsymbol{x}^{\mathrm{s}}_{\mathrm{INT8}}$ 的第 i 个值，n 为向量长度。

由此可见，不论何种分布，量化后的向量会出现大量的 0，导致巨大的误差。这是因为 softmax 的输出和为 1，在较大的向量中会有大量的值远小于 1，然而缩放系数依旧使用最大绝对值为 1 来计算，从而导致误差增大。

3 softmax 量化方法设计

3.1 伪 softmax 向量量化方法设计

通过观察 softmax 的计算过程，可以发现一个有趣的特性，即在计算 softmax 之前，通常会先将输入向量中的每个元素减去该向量的最大值。这样做的目的是防止在指数运算过程中发生数值溢出，并确保结果向量中的所有值都小于等于 0，且至少有 1 个值为“0”(即原本的最大值)。接着，对这个经过调整的向量进行指数运算，得到的输出向量范围仍然在 [0, 1] 之间，并且至少有 1 个元素的值为“1”。这个向量与 softmax 的输出非常相似，只是没有进行最后的归一化步骤，因此我们可以称之为“伪 softmax 向量”。

伪 softmax 向量是一个非常适合做定点数量化的向量，因为其正好解决了原 softmax 向量有大量值远小于 1 的问题，伪 softmax 向量必存在且至少含有 1 个最大值“1”，为量化提供了良好的条件。同样地，对均

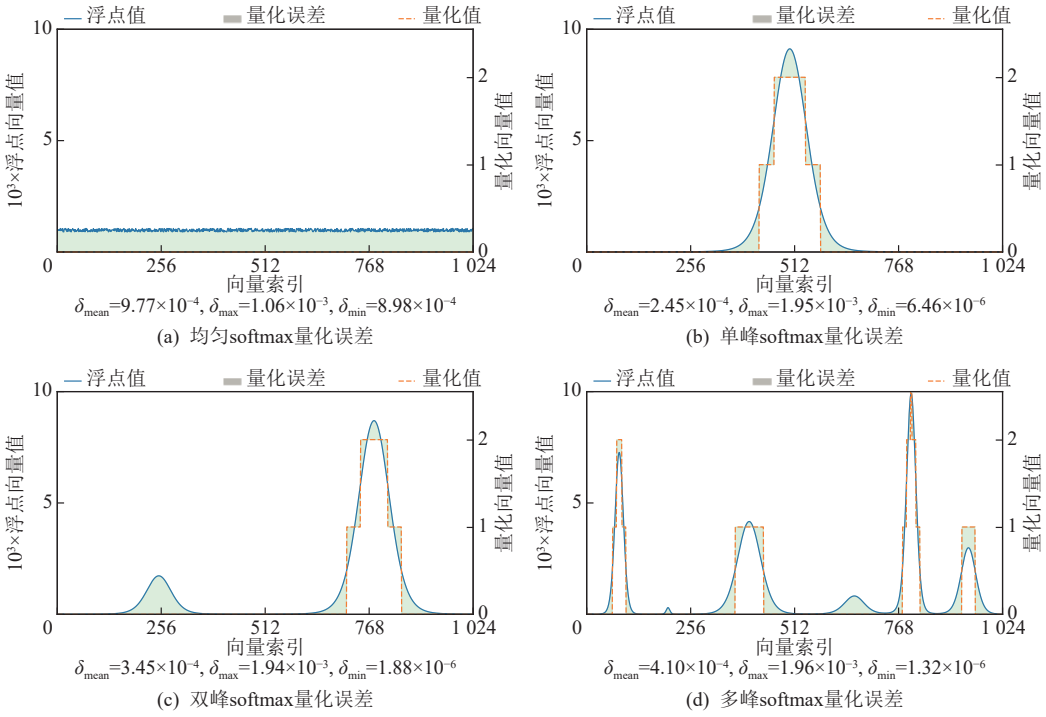


Fig. 2 softmax vector quantization error plot

图 2 softmax 向量量化误差图

匀、单峰、双峰以及多峰的输入进行伪 softmax 计算，并对其输出进行量化，分析定点数误差，如图 3 所示。

与原来的 softmax 向量输出相比，伪 softmax 向量未归一，误差会随之放大。因此，为了与原来的误差估计对齐，对于伪 softmax 向量的误差估计 δ_{mean}，$\delta_{\max}$，$\delta_{\min}$ 由式(5)计算得到。

$$
\begin{aligned}
&s_{\mathrm{f}}^{\mathrm{ws}}=\sum_{i} x_{\mathrm{f}(i)}^{\mathrm{ws}},\ s_{\mathrm{INT8}}^{\mathrm{ws}}=\sum_{i} x_{\mathrm{INT8}(i)}^{\mathrm{ws}},\\
&\delta_{\mathrm{mean}}=\frac{\sum_{i=1}^{n}\left|\frac{x_{\mathrm{f}(i)}^{\mathrm{ws}}}{s_{\mathrm{f}}^{\mathrm{ws}}}-\frac{x_{\mathrm{INT8}(i)}^{\mathrm{ws}}}{s_{\mathrm{INT8}}^{\mathrm{ws}}}\right|}{n},\\
&\delta_{\max}=\max_{1\leqslant i\leqslant n}\left|\frac{x_{\mathrm{f}(i)}^{\mathrm{ws}}}{s_{\mathrm{f}}^{\mathrm{ws}}}-\frac{x_{\mathrm{INT8}(i)}^{\mathrm{ws}}}{s_{\mathrm{INT8}}^{\mathrm{ws}}}\right|,\\
&\delta_{\min}=\min_{1\leqslant i\leqslant n}\left|\frac{x_{\mathrm{f}(i)}^{\mathrm{ws}}}{s_{\mathrm{f}}^{\mathrm{ws}}}-\frac{x_{\mathrm{INT8}(i)}^{\mathrm{ws}}}{s_{\mathrm{INT8}}^{\mathrm{ws}}}\right|,
\end{aligned}
\tag{5}
$$

其中 $x_{\mathrm{f}(i)}^{\mathrm{ws}}$ 和 $x_{\mathrm{INT8}(i)}^{\mathrm{ws}}$ 分别为伪 softmax 浮点数向量 $\boldsymbol{x}_{\mathrm{f}}^{\mathrm{ws}}$ 和量化整数向量 $\boldsymbol{x}_{\mathrm{INT8}}^{\mathrm{ws}}$ 的第 i 个值，n 为向量长度。由图 3 可见，相比于原始的 softmax 向量量化，伪 softmax 向量量化的精度误差要明显小于它，最大绝对误差降低为原来的数百分之一。

然而，伪 softmax 与原 softmax 不等价，需要一些额外的操作。通常，softmax 输出的注意力权重矩阵会与值矩阵进行相乘得到自注意力模块的输出，softmax 向量量化是为能够使其与值矩阵计算转换为定点数矩阵运算。伪 softmax 向量与原 softmax 向量仅差归一化，即除以向量和，为线性运算。因此，可在伪 softmax 向量与值矩阵完成定点数运算后再进行归一，即除以伪 softmax 向量和，其表示如式(6)所示。

$$
\begin{aligned}
\boldsymbol{x}&=\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d}},\\
attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})&=softmax(\boldsymbol{x})\cdot\boldsymbol{V}=\\
&\frac{f(\boldsymbol{x})}{l(\boldsymbol{x})}\cdot\boldsymbol{V}=\\
&\frac{f(\boldsymbol{x})\cdot\boldsymbol{V}}{l(\boldsymbol{x})},
\end{aligned}
\tag{6}
$$

其中 $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$ 分别代表自注意力模块中的查询矩阵、键矩阵和值矩阵，d 代表每个注意力头的向量维度，$f(\cdot)$ 代表伪 softmax，$l(\cdot)$ 则为伪 softmax 向量和。

对式 (6) 进行 8b 整数量化后可由式(7)表示。

$$
\begin{aligned}
&\boldsymbol{Q}_{\mathrm{INT8}}=\left\lfloor\frac{\boldsymbol{Q}}{s_{\boldsymbol{Q}}}\right\rceil,\ \boldsymbol{K}_{\mathrm{INT8}}=\left\lfloor\frac{\boldsymbol{K}}{s_{\boldsymbol{K}}}\right\rceil,\ \boldsymbol{V}_{\mathrm{INT8}}=\left\lfloor\frac{\boldsymbol{V}}{s_{\boldsymbol{V}}}\right\rceil,\\
&s_{\mathrm{s}}=\frac{1}{255},\\
&\boldsymbol{x}=\frac{s_{\boldsymbol{Q}}s_{\boldsymbol{K}}}{\sqrt{d}}\cdot\boldsymbol{Q}_{\mathrm{INT8}}\boldsymbol{K}_{\mathrm{INT8}}^{\mathrm{T}},\\
&\boldsymbol{S}_{\mathrm{INT8}}^{\mathrm{w}}=\left\lfloor\frac{f(\boldsymbol{x})}{s_{\mathrm{s}}}\right\rceil,\\
&attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=\frac{f(\boldsymbol{x})\cdot\boldsymbol{V}}{l(\boldsymbol{x})}=\frac{s_{\boldsymbol{V}}s_{\mathrm{s}}\boldsymbol{S}_{\mathrm{INT8}}^{\mathrm{w}}\boldsymbol{V}_{\mathrm{INT8}}}{l(\boldsymbol{x})},
\end{aligned}
\tag{7}
$$

其中 $s_{\boldsymbol{Q}}$，$s_{\boldsymbol{K}}$，$s_{\boldsymbol{V}}$，s_{s} 分别代表 $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$ 伪 softmax 输出向量的缩放系数，$\boldsymbol{S}_{\mathrm{INT8}}^{\mathrm{w}}$ 为 8 b 整数量化的伪 softmax

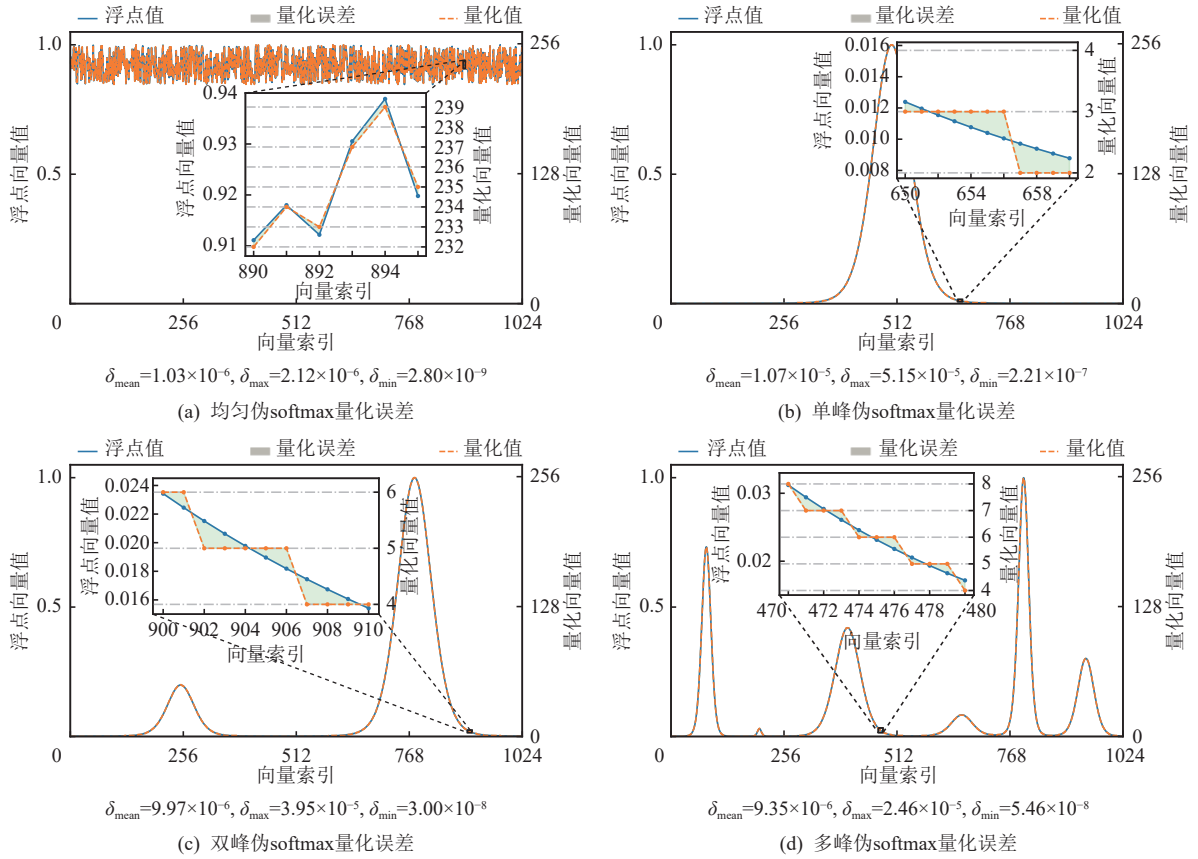


Fig. 3 Pseudo softmax vector quantization error plot

图 3 伪 softmax 向量量化误差图

输出向量。通过式(7)的变换，可以将自注意力模块中的 2 个矩阵乘计算都转换为 8 b 整型计算，进一步增加了自注意力模块的量化率，提升其计算效率。

3.2 分块伪 softmax 向量量化方法设计

在图像分类、人脸识别等将 softmax 作为输出层激活函数的场景当中，其输出向量往往只关注概率最高的某一个或某几个类别，对于较小的几个峰值通常只会认为其是一种数据波动，几乎不会影响模型整体的输出准确率。对于自注意力模块而言，其通常被应用在需要对图像细节进行特征提取的场景中，如目标检测、自动驾驶感知等领域。对于自注意力模块中的 softmax 向量，除了需要关注某些峰值区域，同样地对于隐蔽目标的关注也需要被保留。并且，一个 Transformer 结构往往需要若干层自注意力模块进行串联，这就会导致 softmax 量化误差的指数级别的放大，从而可能对隐蔽物体完全失去注意力，降低整体模型的召回率。

例如，对于上述的多峰分布 softmax 向量，其中就包含较高的波峰和较小的波峰。当 softmax 向量的波峰信息之间的大小相差较大时，较小的波峰在量化时会更容易丢失信息，从而可能导致量化模型识别准确率降低。在下文中，称 softmax 输出向量的波峰为“信息峰”。

在图 4 中，右上图为最大信息峰的放大图，可见 softmax 的浮点向量和量化向量十分吻合，而左下图为最小信息峰的放大图，相比之下误差会更大。为了充分评估不同大小信息峰的量化误差，在图 4 中，δ_{mean}, δ_{max}, δ_{min} 的计算由式(8)得到：

$$\begin{aligned}
\delta_{\text{mean}} &= \frac{\sum_{i=p}^{q}\left|\frac{x^{\text{ws}}_{\text{f}(i)}}{s^{\text{ws}}_{\text{f}}}-\frac{x^{\text{ws}}_{\text{INT8}(i)}}{s^{\text{ws}}_{\text{INT8}}}\right|}{(p-q+1)\max\limits_{p\leqslant i\leqslant q}x^{\text{ws}}_{\text{f}(i)}},\\
\delta_{\text{max}} &= \frac{\max\limits_{p\leqslant i\leqslant q}\left|\frac{x^{\text{ws}}_{\text{f}(i)}}{s^{\text{ws}}_{\text{f}}}-\frac{x^{\text{ws}}_{\text{INT8}(i)}}{s^{\text{ws}}_{\text{INT8}}}\right|}{\max\limits_{p\leqslant i\leqslant q}x^{\text{ws}}_{\text{f}(i)}},\\
\delta_{\text{min}} &= \frac{\min\limits_{p\leqslant i\leqslant q}\left|\frac{x^{\text{ws}}_{\text{f}(i)}}{s^{\text{ws}}_{\text{f}}}-\frac{x^{\text{ws}}_{\text{INT8}(i)}}{s^{\text{ws}}_{\text{INT8}}}\right|}{\max\limits_{p\leqslant i\leqslant q}x^{\text{ws}}_{\text{f}(i)}},
\end{aligned} \tag{8}$$

其中 $[p, q]$ 为 softmax 输出向量某个信息峰的索引范围。与式(5)不同点在于，式(8)需要评估量化误差对单独 1 个信息峰的影响，因此需要除以信息峰的峰值以统一评估数量级。由图 4 所示，在最常见的

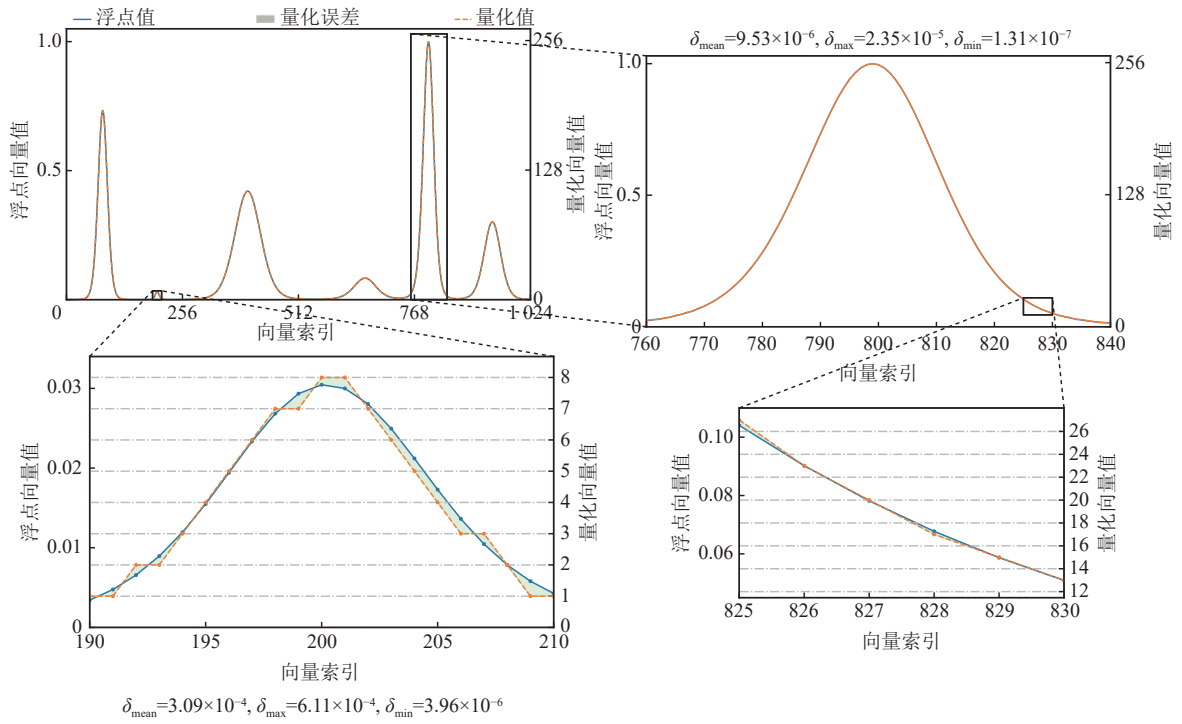


Fig. 4 Quantization error plot of pseudo softmax vector information peak

图 4 伪 softmax 向量信息峰量化误差图

多峰分布的 softmax 向量中，最大信息峰和最小信息峰的量化误差可以相差数十倍，即该量化方式只关注大信息峰（即峰值接近于 1 的信息峰）的信息精度而丢弃小信息峰（即峰值接近于 0 的信息峰）的信息精度。这是由于在方差较大的多峰向量中，可能会出现部分信息峰的峰值远小于 1 而依旧使用针对最大波峰设计的缩放系数计算，从而导致较大的误差。

通过上述分析，希望尽可能使 softmax 向量中的每个信息峰变大，最好是这些信息峰的峰值都能够达到或接近 1，这样就能以最大绝对值 1 计算缩放系数来量化每个信息峰，并使其能够充分利用缩放系数的量化范围，从而降低这种缩放系数所带来的量化误差。另外，如果 1 个 softmax 向量长度足够小，其包含的信息也会更少，即信息峰会更少，那么其在量化时就可以减少对信息峰精度的影响。因此，希望将 softmax 向量当中的每个信息峰都单独取出来进行量化，如图 5 所示，即对每个信息峰单独进行伪 softmax 计算，这样可以使得该信息峰的峰值能够达到 1。并且，希望这样的向量长度短一些，以降低量化误差对信息峰的影响。然而，信息峰的分布是黑盒的，无法判断每个 softmax 向量中有多少个信息峰以及这些信息峰是如何分布的。既然如此，可以考虑直接对伪 softmax 向量进行分块，虽然不能完全使不同的信息峰独立开来，但是能尽可能保证不同信

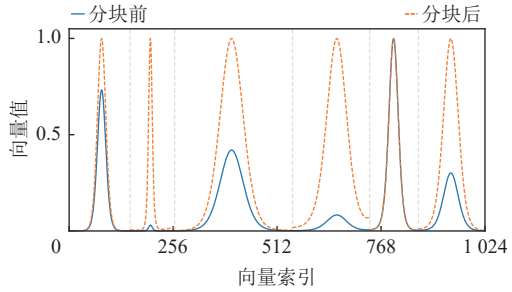


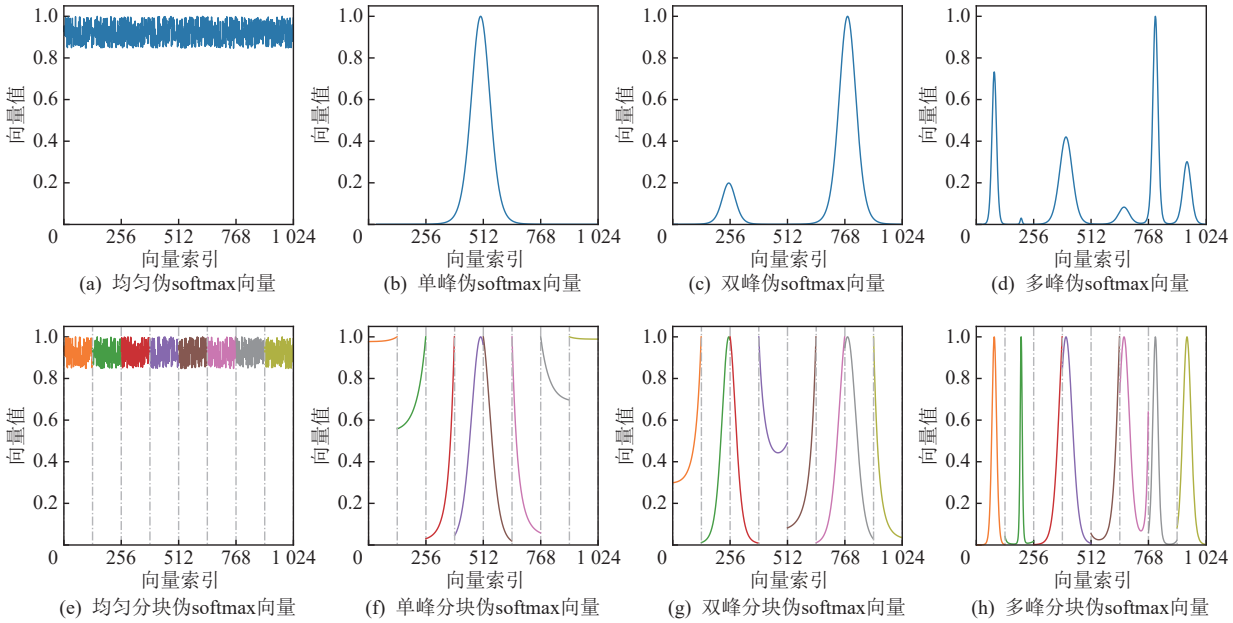
Fig. 5 Blocking diagram of multimodal pseudo softmax vector information peak

图 5 多模态伪 softmax 向量信息峰分块图

息峰的峰值分布在不同子向量中，从而使每个子向量的最大信息峰峰值能够达到 1。

为了得到分块伪 softmax 向量，需要将 softmax 分块，将其等量划分为 k 块，对每块进行伪 softmax 计算。通过分块计算，每块的伪 softmax 向量的最大值都能达到 1，从而可以保证整个伪 softmax 的输出向量至少含有 k 个值为 1 且可大幅增加接近 1 的值，从而拉高整体的向量值以减少量化误差。如图 6 所示，将上述分布的向量以 128 维为 1 块，每块独立计算伪 softmax，可见每一块向量块都可以达到最大值 1，以使每个向量块都尽可能分布均匀。

此外，可以借鉴 FlashAttention[20] 利用多线程技术对自注意力模块并行计算的方式，该方法同样可以



(a) 均匀伪softmax向量 (b) 单峰伪softmax向量 (c) 双峰伪softmax向量 (d) 多峰伪softmax向量

(e) 均匀分块伪softmax向量 (f) 单峰分块伪softmax向量 (g) 双峰分块伪softmax向量 (h) 多峰分块伪softmax向量

Fig. 6 Comparative diagram between pseudo softmax vector and block pseudo softmax vector

图 6 伪 softmax 向量与分块伪 softmax 向量对比图

利用多线程技术对伪 softmax 向量进行分块量化并计算，再进行合并计算。通过此方法，伪 softmax 向量的量化及计算可以与前后 2 个矩阵乘进行合并，可以大幅降低与全局内存的交互，使得所有的计算都在寄存器上进行，减少总体的计算时间。Flash-Attention 采用分治算法对 softmax 进行分块，将对 1 个 softmax 向量的计算划分为若干子向量的计算，如式(9)所示。

$$m(\boldsymbol{x}) = m\left(\left(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^k\right)\right) = \max\left(m\left(\boldsymbol{x}^1\right), m\left(\boldsymbol{x}^2\right), \cdots, m\left(\boldsymbol{x}^k\right)\right), \tag{9}$$

其中 $\boldsymbol{x}$ 为 softmax 的输入向量，$\boldsymbol{x}^i(i=1,2,\cdots,k)$为向量 $\boldsymbol{x}$ 的子向量块。$m(\cdot)$由式(2)定义。则 $f(\boldsymbol{x})$和 $l(\boldsymbol{x})$可以由式(10)计算得到：

$$f(\boldsymbol{x}) = \left[\exp(m(\boldsymbol{x}^i - m(\boldsymbol{x}))f\left(\boldsymbol{x}^i\right)\right]_{i=1,2,\cdots,k},$$
$$l(\boldsymbol{x}) = \sum_{i=1}^{k}\exp(m(\boldsymbol{x}^i - m(\boldsymbol{x}))l\left(\boldsymbol{x}^i\right), \tag{10}$$

其中式(10)的 $f(\cdot)$和 $l(\cdot)$与式(2)等价。通过该分块变换，可以由式(1)得到 softmax 向量：

$$softmax(\boldsymbol{x}) = \frac{f(\boldsymbol{x})}{l(\boldsymbol{x})} = \left[\frac{\exp(m(\boldsymbol{x}^j) - m(\boldsymbol{x}))f(\boldsymbol{x}^j)}{\sum_{i=1}^{k}\exp(m(\boldsymbol{x}^i) - m(\boldsymbol{x}))f(\boldsymbol{x}^i)}\right]_{j=1,2,\cdots,k}。 \tag{11}$$

对于自注意力模块的 8 b 整数量化计算，首先需要对输入的查询矩阵、键矩阵以及值矩阵进行分块并量化，如式(12)所示。

$$\boldsymbol{Q} = \left(\boldsymbol{Q}^1, \boldsymbol{Q}^2, \cdots, \boldsymbol{Q}^i, \cdots, \boldsymbol{Q}^{N_Q}\right)^{\mathrm{T}},$$
$$\boldsymbol{K} = \left(\boldsymbol{K}^1, \boldsymbol{K}^2, \cdots, \boldsymbol{K}^j, \cdots, \boldsymbol{K}^{N_{KV}}\right)^{\mathrm{T}},$$
$$\boldsymbol{V} = \left(\boldsymbol{V}^1, \boldsymbol{V}^2, \cdots, \boldsymbol{V}^j, \cdots, \boldsymbol{V}^{N_{KV}}\right)^{\mathrm{T}},$$
$$\boldsymbol{Q}^i_{\mathrm{INT8}} = \left\lfloor\frac{\boldsymbol{Q}^i}{s_{\boldsymbol{Q}}}\right\rceil, \boldsymbol{K}^j_{\mathrm{INT8}} = \left\lfloor\frac{\boldsymbol{K}^j}{s_{\boldsymbol{K}}}\right\rceil, \boldsymbol{V}^j_{\mathrm{INT8}} = \left\lfloor\frac{\boldsymbol{V}^j}{s_{\boldsymbol{V}}}\right\rceil。 \tag{12}$$

其中 $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$ 分别代表查询矩阵、键矩阵和值矩阵；$s_{\boldsymbol{Q}}$，$s_{\boldsymbol{K}}$，$s_{\boldsymbol{V}}$分别为 $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$ 的缩放系数。之后便可以由式(13)分块计算伪 softmax。

$$\boldsymbol{x}^{ij} = \frac{\boldsymbol{Q}^i\boldsymbol{K}^{j\mathrm{T}}}{\sqrt{d}} = \frac{s_{\boldsymbol{Q}}s_{\boldsymbol{K}}}{\sqrt{d}}\boldsymbol{Q}^i_{\mathrm{INT8}}\boldsymbol{K}^{j\ \mathrm{T}}_{\mathrm{INT8}},$$
$$\hat{\boldsymbol{S}}^{ij} = \exp(\boldsymbol{x}^{ij} - \max_i(\boldsymbol{x}^{ij})), \tag{13}$$

其中 $\hat{\boldsymbol{S}}^{ij}$为分块伪 softmax 输出矩阵块，$\max_i(\boldsymbol{x}^{ij})$为在第 i 块上对 j 维度上的向量求最大值。之后便可以对分块伪 softmax 进行量化，并与值矩阵块进行定点数计算，如式(14)所示。

$$s_{\mathrm{s}} = \frac{1}{255},\ \hat{\boldsymbol{S}}^{ij}_{\mathrm{INT8}} = \left\lfloor\frac{\hat{\boldsymbol{S}}^{ij}}{s_{\mathrm{s}}}\right\rceil,$$
$$\hat{\boldsymbol{S}}^{ij}\boldsymbol{V}^j = s_{\mathrm{s}}s_{\boldsymbol{V}}\cdot\hat{\boldsymbol{S}}^{ij}_{\mathrm{INT8}}\boldsymbol{V}^j_{\mathrm{INT8}}。 \tag{14}$$

之后便可以根据式(9)和式(10)进行合并，得到：

$$\boldsymbol{x}^{ij}_{\mathrm{m}} = \max_i\left(\boldsymbol{x}^{ij}\right),\ m_i = \max_{j=1,2,\cdots,N_{KV}}\left(\boldsymbol{x}^{ij}_{\mathrm{m}}\right),$$
$$\hat{\boldsymbol{S}}^i\boldsymbol{V} = \sum_j\exp(\boldsymbol{x}^{ij}_{\mathrm{m}} - m_i)\hat{\boldsymbol{S}}^{ij}\boldsymbol{V}^j =$$
$$\sum_j\exp\left(\max_i\left(\frac{\boldsymbol{Q}^i\boldsymbol{K}^{j\mathrm{T}}}{\sqrt{d}}\right) - m_i\right)\hat{\boldsymbol{S}}^{ij}\boldsymbol{V}^j。 \tag{15}$$

因为伪 softmax 未进行归一化操作,需要对得到的输出除以伪 softmax 的和以与原来的自注意力模块计算等价,如式(16)所示。

$$\boldsymbol{S}^i=\frac{\hat{\boldsymbol{S}}^i}{l(\boldsymbol{x}^i)},\ l(\boldsymbol{x}^{ij})=\sum \boldsymbol{x}^{ij},$$

$$l(\boldsymbol{x}^i)=\sum_j\left(\exp\left(\max_i\left(\frac{\boldsymbol{Q}^i\boldsymbol{K}^{j\mathrm{T}}}{\sqrt{d}}\right)-m_i\right)l(\boldsymbol{x}^{ij})\right),$$

$$attention(\boldsymbol{Q}^i,\boldsymbol{K},\boldsymbol{V})=\boldsymbol{S}^i\boldsymbol{V}=\frac{\hat{\boldsymbol{S}}^i\boldsymbol{V}}{l(\boldsymbol{x}^i)}=$$

$$\frac{\sum_j\left(\exp\left(\max_i\left(\frac{\boldsymbol{Q}^i\boldsymbol{K}^{j\mathrm{T}}}{\sqrt{d}}\right)-m_i\right)\hat{\boldsymbol{S}}^{ij}\boldsymbol{V}^j\right)}{\sum_j\left(\exp\left(\max_i\left(\frac{\boldsymbol{Q}^i\boldsymbol{K}^{j\mathrm{T}}}{\sqrt{d}}\right)-m_i\right)l(\boldsymbol{x}^{ij})\right)},\quad(16)$$

其中 m_i 为 j 维度向量的全局最大值。考虑到查询矩阵块$\boldsymbol{Q}^i$之间的计算是独立的,因此第 i 个查询矩阵块经过自注意力模块计算便可以当作输出的第 i 块。

同样地,采用均匀、单峰、双峰以及多峰分布的输入进行误差分析。与图 3 相比,图 7 在同个区域的整型量化明显有更多的划分层,并与浮点向量更贴合。同样地,为了与之前的误差评估方式一致,需要对每一个向量块转换回去后再计算误差,δ_{mean},δ_{max},δ_{min}的计算由式(17)得到。

$$\delta_{\mathrm{mean}}=\frac{\sum_j\sum_{i=p_j}^{q_j}\left|\frac{x_{\mathrm{f}(i)}^{\mathrm{kws}(j)}}{s_{\mathrm{f}}^{\mathrm{ws}}}-\frac{x_{\mathrm{INT8}(i)}^{\mathrm{kws}(j)}}{s_{\mathrm{INT8}}^{\mathrm{ws}}}\right|\max_i x_{\mathrm{f}(i)}^{\mathrm{ws}(j)}}{n},$$

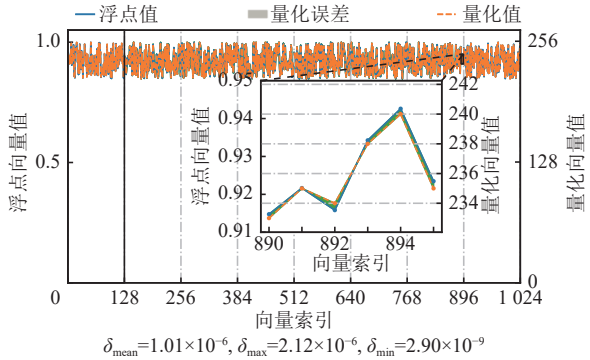
$$\delta_{\mathrm{max}}=\max_j\max_{p_j\leqslant i\leqslant q_j}\left|\frac{x_{\mathrm{f}(i)}^{\mathrm{kws}(j)}}{s_{\mathrm{f}}^{\mathrm{ws}}}-\frac{x_{\mathrm{INT8}(i)}^{\mathrm{kws}(j)}}{s_{\mathrm{INT8}}^{\mathrm{ws}}}\right|\max_i x_{\mathrm{f}(i)}^{\mathrm{ws}(j)},\quad(17)$$

$$\delta_{\mathrm{min}}=\min_j\min_{p_j\leqslant i\leqslant q_j}\left|\frac{x_{\mathrm{f}(i)}^{\mathrm{kws}(j)}}{s_{\mathrm{f}}^{\mathrm{ws}}}-\frac{x_{\mathrm{INT8}(i)}^{\mathrm{kws}(j)}}{s_{\mathrm{INT8}}^{\mathrm{ws}}}\right|\max_i x_{\mathrm{f}(i)}^{\mathrm{ws}(j)},$$

其中 $x_{\mathrm{f}(i)}^{\mathrm{kws}(j)}$ 和 $x_{\mathrm{INT8}(i)}^{\mathrm{kws}(j)}$ 分别为第 j 块伪 softmax 浮点向量块和量化向量块的第 i 个值,而 $x_{\mathrm{f}(i)}^{\mathrm{ws}(j)}$ 为原始伪 softmax 浮点向量中与第 j 块伪 softmax 浮点向量块对应的子向量的第 i 个值,$[p_j, q_j]$ 表示第 j 块伪 softmax 向量块的索引范围。根据分块伪 softmax 的定义,伪 softmax 浮点向量块$\boldsymbol{x}_{\mathrm{f}}^{\mathrm{kws}(j)}$和原始伪 softmax 浮点向量的子向量$\boldsymbol{x}_{\mathrm{f}}^{\mathrm{ws}(j)}$具有如式(18)所示的关系:

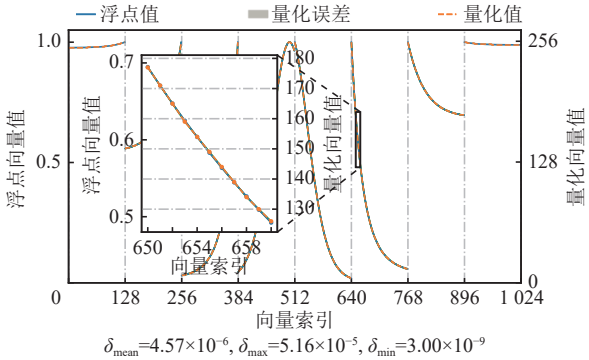
$$\boldsymbol{x}_{\mathrm{f}}^{\mathrm{kws}(j)}=\frac{\boldsymbol{x}_{\mathrm{f}}^{\mathrm{ws}(j)}}{\max_i \boldsymbol{x}_{\mathrm{f}(i)}^{\mathrm{ws}(j)}}。\quad(18)$$

由图 7 可见,通过分块伪 softmax 量化方式,伪 softmax 向量块的量化误差有一定的下降。虽然整体向量的量化误差没有数量级级别的下降,但是分块



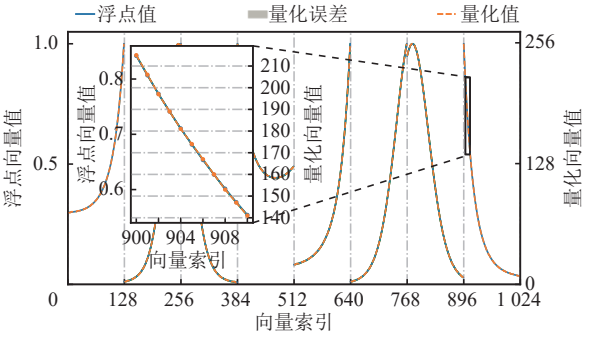
$\delta_{\mathrm{mean}}=1.01\times10^{-6}$, $\delta_{\mathrm{max}}=2.12\times10^{-6}$, $\delta_{\mathrm{min}}=2.90\times10^{-9}$

(a) 均匀分块伪softmax量化误差



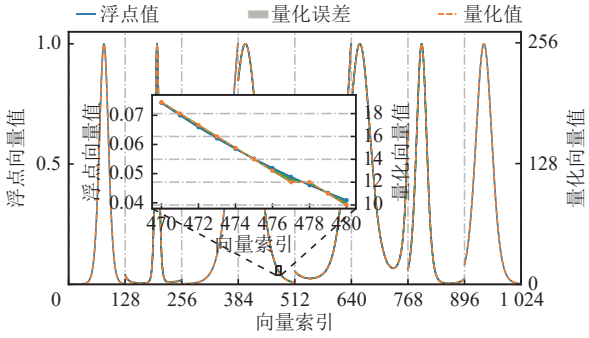
$\delta_{\mathrm{mean}}=4.57\times10^{-6}$, $\delta_{\mathrm{max}}=5.16\times10^{-5}$, $\delta_{\mathrm{min}}=3.00\times10^{-9}$

(b) 单峰分块伪softmax量化误差



$\delta_{\mathrm{mean}}=3.82\times10^{-6}$, $\delta_{\mathrm{max}}=3.95\times10^{-5}$, $\delta_{\mathrm{min}}=1.03\times10^{-10}$

(c) 双峰分块伪softmax量化误差



$\delta_{\mathrm{mean}}=3.51\times10^{-6}$, $\delta_{\mathrm{max}}=2.35\times10^{-5}$, $\delta_{\mathrm{min}}=3.63\times10^{-9}$

(d) 多峰分块伪softmax量化误差

Fig. 7 Error plot of block pseudo softmax vector quantization

图 7 分块伪 softmax 向量量化误差图

的方式可以使每个小信息峰转变为大信息峰，从而降低小信息峰的量化误差。对于自注意力模块而言，这些小信息峰的信息也十分重要，在后面对算子的实验中论证了这一点。

为了更显著地表示分块伪 softmax 方法对多峰向量中不同大小信息峰量化误差的影响，图 8 展示了与图 4 中相同信息峰在分块伪 softmax 方法下的量化误差图。可见，通过分块伪 softmax 量化的方法，小信息峰的精度可以达到甚至超过大信息峰的精度。相比于全局伪 softmax 量化方法，分块伪 softmax 量化方法可以使多峰向量中的小信息峰的量化误差下降为原来的数百分之一，充分保留了 softmax 输出向量的全部精度信息，不会像全局伪 softmax 量化方法那样侧重于大信息峰的量化精度。

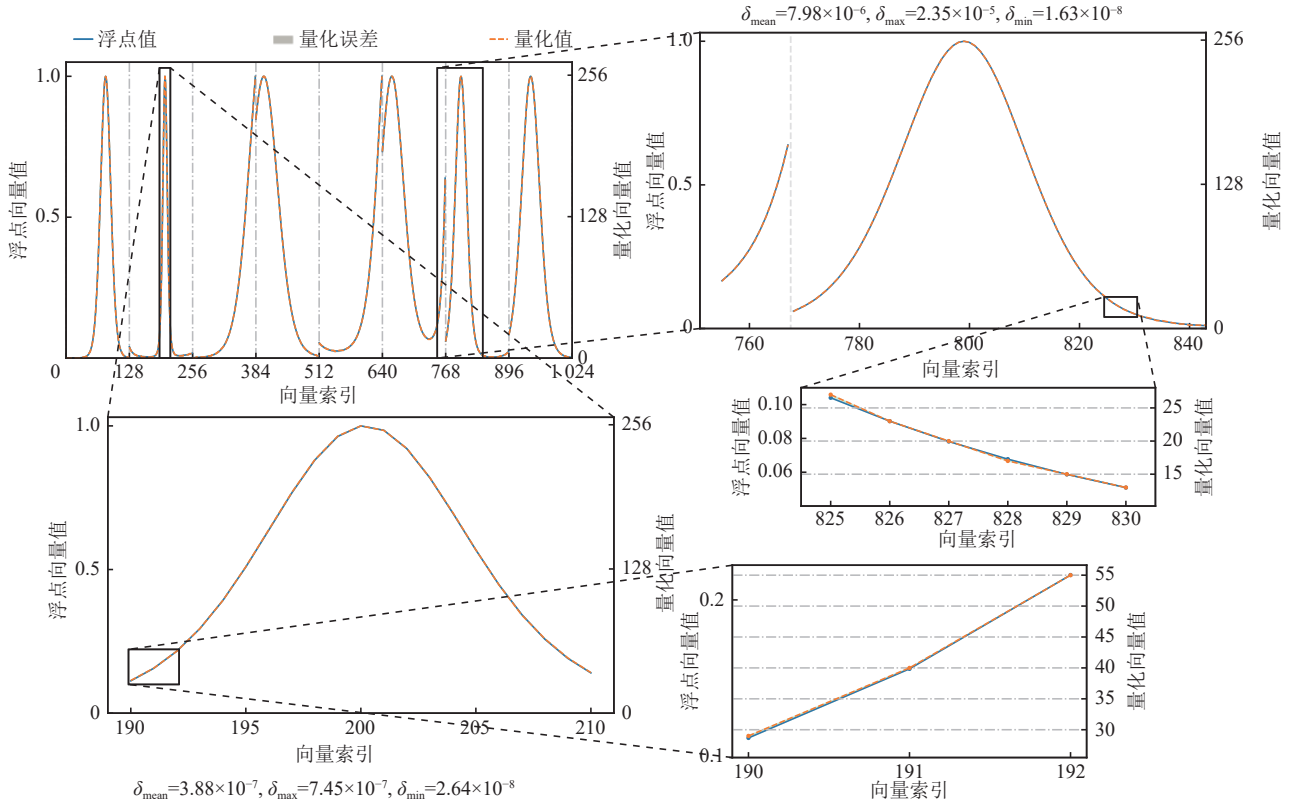


Fig. 8 Error plot of block pseudo softmax vector information peak quantization

图 8 分块伪 softmax 向量信息峰量化误差图

4 实验与分析

4.1 分块伪 softmax 向量量化方法设计

自注意力模块通过对查询矩阵和键矩阵的乘积进行 softmax 计算得到权重系数矩阵，再与值矩阵进行加权计算得到自注意力输出值。其输入通常为查询矩阵、键矩阵以及值矩阵，其中查询矩阵的大小为 (L_Q, B, H×d)，而键矩阵与值矩阵的大小均为 (L_{KV}, B, H×d)，其中 L_Q 和 L_{KV} 分别表示查询矩阵以及键或值矩阵的序列长度，B 表示批数据大小，H 表示注意力头的个数，d 表示每个注意力头的通道数。

如 3.2 节所述，为了尽可能降低 8 b 整数自注意力模块的精度损失，需要对 softmax 分块量化和计算，并可以通过 FlashAttention 的方式与前后 2 个矩阵乘法进行合并，以降低显存的读写量。

8 b 整数的自注意力模块算子实现流程为：

1）对查询矩阵进行分块，每块 64×d，并在每个线程中对键矩阵和值矩阵分块遍历，每块同样为 64×d，且每个线程块的线程数为 128。

2）每个线程束（32 线程）负责 1 个查询矩阵块的子块 16×d 的计算，及 16×d 的查询矩阵块子块与键矩阵块 64×d 的矩阵乘，利用 MMA（matrix multiply-accumulate）指令集在 Tensor Core 上计算。在图灵架构的显卡下，1 个 Tensor Core 最多可以支持 8×16 和 16×8 的矩阵乘；而在安培架构的显卡下，最多可以支持 16×32 和 32×8 的矩阵乘。查询矩阵块与键矩阵块的矩阵乘计算如式（19）所示。

$$\boldsymbol{Q}^{i}_{\mathrm{INT8}}\boldsymbol{K}^{j}_{\mathrm{INT8}}{}^{\mathrm{T}}=\begin{pmatrix}\boldsymbol{q}_{1,1} & \cdots & \boldsymbol{q}_{1,\frac{d}{k}}\\ \vdots & & \vdots\\ \boldsymbol{q}_{\frac{16}{m},1} & \cdots & \boldsymbol{q}_{\frac{16}{m},\frac{d}{k}}\end{pmatrix}\begin{pmatrix}\boldsymbol{k}_{1,1} & \cdots & \boldsymbol{k}_{1,\frac{d}{k}}\\ \vdots & & \vdots\\ \boldsymbol{k}_{\frac{64}{n},1} & \cdots & \boldsymbol{k}_{\frac{64}{n},\frac{d}{k}}\end{pmatrix}^{\mathrm{T}},\tag{19}$$

其中$\boldsymbol{Q}^{i}_{\mathrm{INT8}}$和$\boldsymbol{K}^{j}_{\mathrm{INT8}}$分别为 8 b 整数的查询矩阵块和键矩阵块, d 为每个自注意力头的通道维度, $\boldsymbol{q}_{i,j}$和$\boldsymbol{k}_{i,j}$分别为查询矩阵块和键矩阵块的子矩阵, m, k, n 分别为 1 个 Tensor Core 计算单元能计算的矩阵维度, 在安培架构中 m=16, k=32, n=8。

3)将根据式(12)得到的矩阵块反量化为浮点数后, 在最后 1 个维度上计算伪 softmax 块。该伪 softmax 块的值范围为 [0,1], 且每行都至少存在 1 个最大值“1”, 分布充分均匀。

4)将得到的浮点数伪 softmax 块根据式(13)进行量化, 再与值矩阵块相乘。矩阵块的相乘同样使用 MMA 指令集在 Tensor Core 上计算, 如式(20)所示。

$$\hat{\boldsymbol{S}}^{ij}_{\mathrm{INT8}}\boldsymbol{V}^{j}_{\mathrm{INT8}}=\begin{pmatrix}\hat{\boldsymbol{s}}_{1,1} & \cdots & \hat{\boldsymbol{s}}_{1,\frac{64}{k}}\\ \vdots & & \vdots\\ \hat{\boldsymbol{s}}_{\frac{16}{m},1} & \cdots & \hat{\boldsymbol{s}}_{\frac{16}{m},\frac{64}{k}}\end{pmatrix}\begin{pmatrix}\boldsymbol{v}_{1,1} & \cdots & \boldsymbol{v}_{1,\frac{d}{k}}\\ \vdots & & \vdots\\ \boldsymbol{v}_{\frac{64}{n},1} & \cdots & \boldsymbol{v}_{\frac{64}{n},\frac{d}{k}}\end{pmatrix},\tag{20}$$

其中$\hat{\boldsymbol{S}}^{ij}_{\mathrm{INT8}}$和$\boldsymbol{V}^{j}_{\mathrm{INT8}}$分别为 8 b 整数的伪 softmax 输出块与值矩阵块, $\hat{\boldsymbol{s}}_{i,j}$, $\boldsymbol{v}_{i,j}$分别为伪 softmax 输出块和键矩阵块的子矩阵块。

5)根据式(15)和式(16)将各个值矩阵块根据伪 softmax 块加权相乘后合并, 合并完后再除以伪 softmax 的每行和, 以对其进行归一化。

由于分块伪 softmax 量化需要与前后 2 个矩阵乘法进行合并, 即每个线程都会承担矩阵块乘法即伪 softmax 块的量化与反量化。通常, 伏特架构的英伟达显卡开始支持了更加高效的 Tensor Core 编程, 其在硬件层面上对小矩阵乘法做了深度优化, 使用 1 个线程簇(32 线程)计算矩阵块乘法, 显著提升了矩阵的计算速度。

调用 Tensor Core 的方式主要有 2 种: WMMA (warp level matrix multiply-accumulate)指令集与 MMA 指令集。WMMA 是一种高级指令集, 可以非常方便调用 Tensor Core 模块, 但是需要共享内存作为矩阵存储媒介。而 MMA 是一种低级指令集, 与硬件版本相关, 需要了解 1 个矩阵块在 32 个线程中的输入输出存储分布, 可直接在寄存器上进行操作, 无需共享内存介入。

考虑到 8 b 定点数算子的实现需要进行反复的量化与反量化, 使用更加底层的 MMA 指令集进行矩阵乘法, 便可以在寄存器上操作数据的量化与反量化, 避免频繁的量化与反量化操作给 8 b 定点数算子带来额外的数据读写延时。

4.2 自注意力模块量化精度及耗时实验

在第 3 节中, 讨论并分析了关于 softmax 的多种量化方式以及精度损失。下面针对自注意力模块算子对这几种量化方式进行消融实验。本节实验对比了自注意力模块的英伟达官方闭源版本(含单精度与半精度对比)、softmax 直接量化版本、伪 softmax 量化版本以及分块伪 softmax 量化版本(含 WMMA 指令集与 MMA 指令集对比)。

表 1 展示了自注意力模块的浮点计算与 8 b 整数量化的实验结果。可以看出, 实验复现的自注意力模块版本在性能上接近官方闭源版本, 这为后续的量化实验提供了可靠的基准。

Table 1 Experimental Results of Self-Attention Module Quantification

表 1 自注意力模块量化实验结果

方法	数据类型	耗时/ms	平均绝对误差
官方闭源版本	单精度	8.83	0
	半精度	4.55	4.30×10^{-5}
softmax 直接量化	8 b 整数	6.09	2.05×10^{-2}
伪 softmax 向量量化	8 b 整数	6.10	1.12×10^{-2}
分块伪 softmax 向量量化(WMMA 指令)	8 b 整数	1.90	2.60×10^{-3}
分块伪 softmax 向量量化(MMA 指令)	8 b 整数	1.16	2.60×10^{-3}

在量化实验方面, 尝试了直接量化、伪向量量化以及分块伪向量量化 3 种不同的方法。从算子平均绝对误差的角度来看, 这 3 种方法的误差是逐级减少的, 这说明随着量化策略的优化, 所提方法能够逐渐减少量化带来的精度损失。

此外, 实验还利用了 MMA 底层指令集的支持, 将 softmax 计算前后的量化与反量化数据都在寄存器中执行, 避免了共享内存的介入。这种优化策略不仅提高了数据处理的效率, 还大幅降低了在 GPU 上的计算耗时。这对于实际应用中提高模型的推理速度和降低资源依赖具有重要意义。

4.3 算子耗时及数据读写量分析

通过表 1 还可以发现, 全局 softmax 量化的耗时会比半精度浮点数计算要高。对此, 本节设计一些实验, 对这部分的异常实验结果进行分析。

对自注意力模块中 3 个部分即 2 个矩阵乘以及 1 个 softmax 算子分别测速分析，实验结果如表 2 所示。

Table 2 Speed Measurement Result of Each Self-Attention Module

表 2 各自注意力模块测速结果

方法	数据类型	模块	耗时/ms
官方版本	半精度	矩阵乘（前）	1.15
		softmax	2.56
		矩阵乘（后）	1.08
伪 softmax 向量量化	8 b 整数	矩阵乘（前）	2.33
		softmax	2.45
		矩阵乘（后）	1.15
		量化处理	0.17

由表 2 可以得到，8 b 整数量化的 2 个矩阵乘法的速度不降反升，这是不符合量化计算特性的。更进一步观察可以发现，对于矩阵乘（前），其输入矩阵大小分别为 (256, 1 024, 32) 和 (256, 32, 1 024)，而输出矩阵大小为 (256, 1 024, 1 024)。可见，输入和输出矩阵大小相差巨大，这会导致写入数据会远大于读出输出。从而可以计算出半精度和 8 b 整数量化下的矩阵乘（前）的读写量。

表 3 展示了矩阵乘（前）模块的数据读写量。由于 8 b 量化计算的输出是 32 b 整数（4 B），相比于半精度计算输出仍为半精度（2 B），其数据写入量要高 1 倍。而查询矩阵与键矩阵的矩阵乘计算输出矩阵要远大于输入矩阵，因此在该矩阵乘中，其数据写入量会决定其耗时。同样地，对于矩阵乘（后）的计算，其也对 GPU 的全局内存进行了大量的写入，因此无法发挥 8 b 整数量化的计算效率。

Table 3 The Amount of Data Read and Write of Matrix Multiplication (Front)

表 3 矩阵乘（前）的数据读写量

计算类型	输入类型	输出类型	读写量/MB
半精度	半精度 (2 B)	半精度 (2 B)	读：33.55 写：536.87
8 b 整数	8 b 整数 (1 B)	32 b 整数 (4 B)	读：16.78 写：1 073.74

而对于分块量化方法而言，因为其对 softmax 进行了分块计算，从而可以与前后 2 个矩阵乘法进行分块合并计算，无需对 GPU 的全局内存进行额外的读写，可以大幅提升其计算速度。因此，采用分块伪 softmax 向量量化方法，不仅可以在一定程度上降低量化损失，还能大幅提升计算速度。

4.4 量化模型精度实验

本节实验以模型单精度浮点数推理作为精度对照组，分别采用伪 softmax 向量量化方法以及分块伪 softmax 向量量化方法对自注意力机制的 softmax 向量进行 8 b 整数量化，并部署到主流的多目 3D 目标检测模型 BEVFormer[21] 中计算检测精度。

实验对 BEVFormer 的 3 种规格大小（tiny, small, base）的模型均进行实验，评测指标为自动驾驶开源 3D 检测数据集 NuScenes[22] 在多目 3D 目标检测任务上的评测指标 NDS（NuScenes detection score），实验结果如表 4 所示。

Table 4 Impact of the softmax Quantization Method on Model Accuracy

表 4 softmax 量化方法对模型精度的影响

模型	数据类型	量化方法	NDS
BEVFormer tiny	浮点数		35.3
	8 b 整数	伪 softmax 量化法	32.1
	8 b 整数	分块伪 softmax 量化法	35.4
BEVFormer small	浮点数		47.7
	8 b 整数	伪 softmax 量化法	45.4
	8 b 整数	分块伪 softmax 量化法	47.7
BEVFormer base	浮点数		51.5
	8 b 整数	伪 softmax 量化法	48.8
	8 b 整数	分块伪 softmax 量化法	51.4

对自注意力机制中的 softmax 向量执行伪 softmax 向量量化后，3 种规格大小的 BEVFormer 模型的 NDS 均出现了 2~3 个百分点的下降，而采用分块伪 softmax 向量量化方法的 3 种规格大小的 BEVFormer 模型相比于单精度浮点数推理精度几乎没有精度损失。

该实验充分论证了分块伪 softmax 向量量化方法对模型精度保留的有效性，其可以在不影响模型整体精度的情况下，提升整体模型 8 b 量化计算的占比，从而提升模型整体的计算速度并降低对资源的依赖程度。

5 结果与讨论

从实验结果来看，相比于原始的 softmax 向量直接量化方法，所提出的伪 softmax 向量量化和分块伪 softmax 向量量化方法的算子平均绝对误差均有显著的下降。其次，考虑到自注意力模块在 8 b 量化计算模式下对全局内存的读写量相比于半精度计算存在较大的不平衡现象，直接导致了伪 softmax 向量量化

的计算效率下降。对于分块伪 softmax 向量量化方法,本文使用了英伟达图形处理器中底层的矩阵计算指令(WMMA 和 MMA)对该算子的矩阵计算进行了合并,避免了中间计算结果对全局内存的频繁读写操作,从而大幅提升了分块伪 softmax 向量量化算子的计算效率。

需要注意的是,对于分块伪 softmax 向量量化计算,其并不完全适用于所有自注意力模块场景。考虑到其是采用底层指令进行计算,底层指令对矩阵大小通常会有一定要求。在本文方法实现中,要求自注意力模块的输入向量序列长度需要为 128 的整数倍,以便于底层计算指令的高效计算。对于大部分深度学习模型而言,其内部参数的长度通常满足要求。因此,所提出的分块伪 softmax 向量量化可以适用于目前大部分的深度学习模型。

6 总 结

本文针对自注意力模块直接量化方法的量化误差进行了深入的分析,并指出了误差较大的原因。随后,提出了伪 softmax 向量量化和分块伪 softmax 量化方法,分析并解决了 softmax 量化精度的问题。通过对浮点计算的算子和各个整数量化方法计算的算子进行了充分的实验,评估其输出的平均绝对误差和算子耗时,论证了所提出方法的有效性。

作者贡献声明:林德铝提出算法思路,设计和执行实验,撰写论文;何琨提出指导意见并修改论文。

参 考 文 献

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Proc of the 31st Conf on Neural Information Processing Systems. Red Hook, NY: Curran Associates Inc, 2017: 5998−6008

[2] Liao Xingbin, Qin Xiaolin, Zhang Siqi, et al. Review of interactive machine translation[J]. Journal of Computer Applications, 2023, 43(2): 329−334 (in Chinese)
(廖兴滨, 秦小林, 张思齐, 等. 交互式机器翻译综述[J]. 计算机应用, 2023, 43(2): 329−334)

[3] Liu Di, Xi Xuefeng, Cui Zhiming, et al. Review of research on extractive-abstractive automatic text summarization technology[J]. Computer Technology and Development, 2023, 33(5): 1−8 (in Chinese)
(刘迪, 奚雪峰, 崔志明, 等. 抽取-生成式自动文本摘要技术研究综述[J]. 计算机技术与发展, 2023, 33(5): 1−8)

[4] Chen Jianpeng, Ma Jianhui, Wang Yijun. A survey of human-computer dialogue system based on multiple-round interaction[J]. Journal of Nanjing University of Information Science & Technology, 2019, 11(3): 256−268 (in Chinese)
(陈健鹏, 马建辉, 王怡君. 基于多轮交互的人机对话系统综述[J]. 南京信息工程大学学报, 2019, 11(3): 256−268)

[5] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[C/OL]// Proc of the 9th Int Conf on Learning Representations. Washington: ICLR, 2021[2025-03-29]. https://openreview.net/forum?id=YicbFd-NTTy

[6] Deng J, Dong W, Socher R, et al. ImageNet: A large-scale hierarchical image database[C]//Proc of the 22nd IEEE Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2009: 248−255

[7] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding with unsupervised learning[R]. San Francisco: OpenAI, 2018

[8] Floridi L, Chiriatti M. GPT-3: Its nature, scope, limits, and consequences[J]. Minds and Machines, 2020, 30(4): 681−694

[9] Touvron H, Lavril T, Izacard G, et al. LLaMA: Open and efficient foundation language models[J]. arXiv preprint, arXiv: 2302.13971, 2023

[10] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]//Proc of the 32nd Int Conf on Machine Learning. Stroudsburg, PA: ACL, 2015: 448−456

[11] Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C]//Proc of the 31st IEEE Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2018: 2704−2713

[12] Sun Xiao, Choi J, Chen C Y, et al. Hybrid 8-bit floating point training and inference for deep neural networks[C]//Proc of the 33rd Conf on Neural Information Processing Systems. Red Hook, NY: Curran Associates Inc, 2019: 4901−4910

[13] Lin Ji, Tang Jiaming, Tang Haotian, et al. AWQ: Activation-aware weight quantization for LLM compression and acceleration[C/OL]// Proc of the 7th Conf on Machine Learning and Systems. 2024[2025-03-29]. https://proceedings.mlsys.org/paper_files/paper/2024/file/42a-452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf

[14] Dettmers T, Pagnoni A, Holtzman A, et al. Qlora: Efficient finetuning of quantized LLMs[C]// Proc of the 37th Conf on Neural Information Processing Systems. Red Hook, NY: Curran Associates Inc, 2023: 10088−10115

[15] Lin Yang, Zhang Tianyu, Sun Peiqin, et al. FQ-ViT: Post-training quantization for fully quantized vision transformer[C]//Proc of the 31st Int Joint Conf on Artificial Intelligence. Burlington: Morgan Kaufmann, 2022: 1173−1179

[16] Yuan Zhihang, Xue Chenhao, Chen Yiqi, et al. PTQ4ViT: Post-training quantization for vision transformers with twin uniform quantization[C]//Proc of the 19th European Conf on Computer Vision. Berlin: Springer, 2022: 191−207

[17] Wang Meiqi, Lu Siyuan, Zhu Danyang, et al. A high-speed and low-complexity architecture for softmax function in deep learning [C]//Proc of the 14th IEEE Asia Pacific Conf on Circuits and Systems. Piscataway, NJ: IEEE, 2018: 223−226

[18] Vasyltsov I, Chang W. Efficient softmax approximation for deep

neural networks with attention mechanism[J]. arXiv preprint, arXiv: 2111.10770, 2021

[19] Zhang Qiang, Yang Jibin, Zhang Xiongwei, et al. CS-Softmax: A cosine similarity-based Softmax loss function[J]. Journal of Computer Research and Development, 2022, 59(4): 936−949 (in Chinese)
(张强, 杨吉斌, 张雄伟, 等. CS-Softmax: 一种基于余弦相似性的Softmax 损失函数[J]. 计算机研究与发展, 2022, 59(4): 936−949)

[20] Dao T, Fu D Y, Ermon S, et al. FlashAttention: Fast and memory-efficient exact attention with IO-awareness[C]//Proc of the 36th Conf on Neural Information Processing Systems. Red Hook, NY: Curran Associates Inc, 2022: 16344−16359

[21] Li Zhiqi, Wang Wenhai, Li Hongyang, et al. BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers[C]//Proc of the 19th European Conf on Computer Vision. Berlin: Springer, 2022: 1−18

[22] Caesar H, Bankiti V, Lang A H, et al. NuScenes: A multimodal dataset for autonomous driving[C]//Proc of the 33rd IEEE Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2020: 11618−11628



Lin Delü, born in 1998. Master. His main research interests include high performance computing, model quantization, inference acceleration for large language models and autonomous driving models.
林德铝, 1998 年生。硕士。主要研究方向为高性能计算、模型量化、大语言模型以及自动驾驶模型的推理加速。



He Kun, born in 1972. PhD, professor. Distinguished member of CCF. Her main research interests include adversarial learning, deep representation learning, graph machine learning, and combinatorial optimization.
何　琨, 1972 年生。博士, 教授。CCF 杰出会员。主要研究方向为对抗学习、深度表示学习、图机器学习、组合优化。