视觉 Transformer 量化剪枝联合优化方法研究

摘　　要

视觉 Transformer 架构相较于传统卷积神经网络能捕获到远距离特征，这带来模型性能提升的同时也导致其对计算资源和内存的要求也相应增加，因此在资源受限的边缘设备上部署困难。为此，本研究探索针对视觉 Transformer 的联合压缩技术的可行性，通过融合针对化注意力机制的量化策略和维度剪枝技术，提出了一种视觉 Transformer 的量化剪枝联合优化框架，以实现模型的压缩并便于部署到如 FPGA 等硬件平台。具体工作为：

本文开发了一种新颖的针对自注意力头激活的量化策略，该策略考虑到自注意力头中激活的异质分布特征，能够有效捕获量化参数以最小化多头自注意力层中的近似误差，同时通过运行时估计来降低量化参数存储成本，在保证模型性能的前提下将模型量化为 8 比特。在此基础上，结合针对输入维度的模型剪枝算法，本文还进一步提出了一种视觉 Transformer 的量化剪枝联合优化算法，通过评估线性层输入序列各维度的重要性来实现有效剪枝。为了减少剪枝对模型准确度的影响，使用 L1 正则稀疏化重要性评分，实现更高的剪枝比例，极大提升了网络的轻量化和计算效率。在 ImageNet-100 数据集上的实验表明，本文所提方法不仅能够灵活节约内存，而且能达到与基准模型相当甚至更好的性能。

关键词：视觉 Transformer；模型量化；模型剪枝；压缩

Research on Joint Quantization and Pruning Optimization Methods for Vision Transformer

ABSTRACT

Compared with traditional convolutional neural networks, the Visual Transformer architecture captures long-distance features, enhancing model performance but consequently demanding increasing computational resources and memory. This intensifies the difficulty of deployment on resource-limited edge devices. To address this, our study investigates the feasibility of joint compression techniques for Visual Transformer. By integrating a quantization strategy tailored for attention mechanisms and dimension pruning techniques, we propose a joint optimization framework for quantizing and pruning Visual Transformer, facilitating model compression and deployment on hardware platforms like FPGAs:

Specifically, this paper develops a novel quantization strategy for the activation of self-attention heads, taking into account the heterogeneous distribution characteristics of activations within the self-attention heads. This strategy effectively captures quantization parameters to minimize approximation errors in the multi-head self-attention layers while reducing the storage cost of quantization parameters through runtime estimation. By doing so, it quantizes the model to 8 bits while ensuring the preservation of model performance. Building upon this foundation, the paper further proposes a joint optimization algorithm for quantizing and pruning Visual Transformer. It implements effective pruning by evaluating the importance of each dimension in the input sequence to the linear layer. To minimize the impact of pruning on model accuracy, L1 regularization is employed to sparsify the importance scores, achieving a higher pruning ratio and significantly enhancing the network's lightweight and computational efficiency. Experiments conducted on the ImageNet-100 datasets demonstrate that the proposed method not only efficiently conserves memory but also achieves performance comparable to or even better than the baseline model.

Key words: Vision Transformer; Model Quantization; Model Pruning; Compression

目　　录

1 绪论

1.1 研究背景及意义

近年来，采用 Transformer 架构的模型在自然语言处理（natural language processing, NLP）[1,2]和计算机视觉（computer vision, CV）[3-5]任务中展现了出色的性能。Transformer 架构利用注意力机制来捕捉输入序列的上下文关联和长距离特征，从而提高了模型的准确度。然而，在对输入序列执行处理过程中，注意力机制的计算成本按照序列长度的平方递增，这使得全局信息的提取在增进模型精确度的同时，同样显著提升了模型的参数数量与计算的复杂度。同时受先前模型扩展研究的启发，发现增加模型与数据规模能进一步提高 Transformer 模型的性能，对于绝大多数已构建的 Transformer 及其衍生模型来说，这一事实加剧了它们对存储空间、运行内存及计算资源的依赖。以 ViT 模型为例，当其在 ImageNet-1K 数据集上实现约 78%的 Top-1 准确度时，需要的每秒浮点运算数（Floating-point Operations Per Second, FLOPS）高达 18B[5]。模型基于 Transformer 的结构所带来的显著计算消耗及存储需求，限制了其在资源受限设备上的应用，以及对响应时间有严格要求环境中的部署[6]。因此，如何在不牺牲准确率的前提下，优化模型和网络结构以实现更高效的运行，已经成为了一个紧迫的研究课题。

针对高效 Transformer，策略上可分为改善网络结构以增加信息传输效率和改进深层网络参数的初始化两大类。目前多数研究集中在第二种策略上，致力于发展与视觉 Transformer 模型匹配的深度模型压缩技术[5]。目前已有众多行之有效的技术用于压缩和加速深度神经网络，诸如低秩分解、量化方法、网络剪枝以及知识蒸馏等。然而，现有的神经网络加速平台主要侧重于对常用神经网络中的通用计算进行优化[6]，或是专门针对经典的卷积神经网络（CNN）和循环神经网络（RNN）结构设计优化方案[7–9]，这些架构和优化策略并未聚焦于 Transformer 模型中至关重要的注意力层。同时研究发现对于 Transformer 架构，将此类压缩技术应用其中很有潜力。例如，Lan[2]等人提出的 ALBERT 模型中，词嵌入层经由张量分解被分割为更小的矩阵，并且采用共享跨层参数来降低模型复杂性；Zafrir[10]等人利用对称线性量化方法，将权重和激活数据动态化至 8 位整数，实现 BERT 模型的四倍压缩；Voita[11]等人研究发现多头注意力中存在冗余，提出了对其相对重要性较低的注意力头部进行剪枝。而基于知识蒸馏，TinyBERT[12]中的学生网络从教师网络中学习 logits，在基本不损失性能的情况下将 BERT 缩小 40%。

然而，尽管在自然语言处理任务中，Transformer 的压缩与加速已经得到了广泛的研究，但对于计算机视觉任务中 Transformer 的加速研究却相对较少。同时，这些现有的方法大多只是在设计剪枝、量化、知识蒸馏、和轻量化网络设计等各自独立的方法，而对于如何联合应用剪枝、量化、和知识蒸馏等方法来进行视觉 Transformer 的压缩加速的研究却很少。

本论文针对视觉 Transformer 的联合压缩方法进行深入研究，通过采用有效的剪枝和量化方法对注意力机制进行优化，从而显著降低模型的计算复杂度和内存访问需求，以提高在资源受限场景下的模型推理速度。本研究提出的针对视觉 Transformer 模型的压缩与加速策略，为实际应用如物联网设备和边缘计算中的模型部署提供了一定的参考。

1.2 模型压缩研究现状

当前，针对模型压缩与加速算法，文献综述[13-15]主要集中在单独阐述剪枝、量化、知识蒸馏以及轻量化网络设计等策略。然而，这些综述对于结合剪枝、量化和知识蒸馏等多种技术的联合压缩与加速新方法的探讨尚显简略，且相关的参考文献也相对匮乏。其中模型剪枝和模型量化是当前最受关注的两类深度神经网络压缩方法，它们在优化角度和压缩形式上完全独立，可以同时应用和互相补充，从而实现更高的压缩比例和更好的模型性能。

1.2.1 模型量化研究现状

量化在计算机视觉任务中得到广泛的应用。Tim 等[16]实施了一种并行加速技术，将原本的 32 位梯度和激活值量化为特定的 8 位浮点数；另一方面，Jacob 等[17]则运用了伪量化技术，旨在模拟量化过程中可能出现的误差，进而将权重和激活值转化为 8 位定点数；此外，低比特量化技术是对二值化神经网络[18]的进一步优化，其中的代表性工作涵盖了二值量化[19-21]与三值量化[22-24]；过度追求高压缩比和运算速度而采用极低比特量化会显著降低模型的准确性。为了平衡压缩、加速与模型精度之间的关系，可以采用混合精度的策略。Asit 等[25]提出的宽低精度网络中，权重和激活分别被量化到 2bit 和 4bit，以此来实现这种平衡。另外，Wang 等[26]则引入了混合精度硬件感知自动量化方法，该方法结合了强化学习，以自动确定网络每一层的最优量化位宽。然而这些量化方法是为卷积神经网络设计的，没有考虑 Transformer 的特有结构。

在现有的针对 Transformer 模型的量化研究工作中，在现有的针对 Transformer 模型的量化研究工作中，Zafrir O 等[10]采用量化感知训练和对称线性量化方法，成功将 BERT 模型压缩至 8 位定点整数，不仅在下游任务中精度几乎无损失，而且存储空间占用减少了 75%。SHEN S 等[27]则利用权重矩阵的二阶海森信息，对 BERT 实施了混合精度和分组量化。另一方面，ZADEH A H 等[28]采纳了聚类理念，将 BERT 模型中 99.9%的权重量化至 3bit，其余权重保持原样存储，但这种方法需在专用硬件上执行推理。为了追求更高的模型压缩率，有文献[29]采用三值化权重分割实现权重的二值化，而文献[12]则通过引入二值注意力机制和知识蒸馏来达到相同目的。尽管二值化模型理论上可获得 32 倍的压缩比相较于原始模型，但上述两种方法均会导致模型精度显著下降，从而限制了其在实际应用中的效果。

但上述研究大部分着重于对权重进行量化，对于计算复杂度较高的注意力机制，当前多数模型在激活值矩阵的运算上依然依赖单精度浮点数，这导致了较高的计算成本。此外，先前的研究主要聚焦于自然语言处理任务，其设计初衷并非直接服务于计算机视觉任务。因此，在应用于计算机视觉领域时，这些模型往往需要进行额外的微调训练以适应不同的数据特性和任务要求。

1.2.2 模型剪枝研究现状

根据剪枝粒度的不同，剪枝可以分为结构化剪枝和非结构化剪枝。早期的方法中更多使用非结构化剪枝，剪枝程度通常使用设置的超参数阈值来控制。Magnitude 剪枝方法[30-32]被广泛认为是一种效果显著的非结构剪枝手段。其核心理念在于剔除那些绝对值较小的权重。GORDON M 在研究中发现[31]，当 BERT 模型在预训练阶段时，若采用 Magnitude 剪枝并在较低的稀疏度进行操作，模型在后续下游任务中的性能基本不受影响。然而，当稀疏度提高时，模型在下游任务上的表现会大幅下降。此外，研究者们也探索了基于梯度或其他敏感性标准的方法，以此来界定权重的重要性并设定相应的阈值。Lecun 等[37]提出最佳脑损伤(optimal brain damage, OBD)算法，它运用损失函数的 Hessian 矩阵来判断参数的重要性，并据此进行权重的裁剪。另外，有文献[33-34]介绍了一种新的剪枝策略，该策略根据权重在训练过程中的移动方向来评估其重要性，具体来说，若权重的更新趋势越接近 0，则该权重被视为较不重要。

结构化剪枝方法以网络中的完整算子为对象。现有的一系列 Transformer 剪枝策略集中于减少多头自注意力模块中的头。例如，Michel 等人[35]的研究表明，并非所有的注意

力头都是同等重要的，一些头部可以在不损失太多性能的情况下被移除。同样，Voita 等人[11]通过对多头自注意力机制的详细分析，发现特定的注意力头在处理某些特定任务时扮演着主要角色，而其余的注意力头对性能的贡献相对较低，因此可以被剪除。除了剪切多头自注意力的头部，层剪枝作为一种更为激进的结构化剪枝方法，通过直接删除模型中的特定层来显著减小模型规模和计算负担。Fan 等人[36]提出了一种结构化丢弃技术，可以根据需求动态地减少 Transformer 模型的深度。另外，Lagunas 等人[38]提出了一种块修剪策略，并将其与运动剪枝技术相融合，从而能够灵活地裁剪模型中的任意组件。然而，这些方法主要是针对自然语言处理领域的 Transformer 模型设计的，并未充分考虑到视觉 Transformer 的特殊结构和设计要素。

1.2.3 量化剪枝联合优化研究现状

对于剪枝与量化的结合，可以按照次序进行操作，也可以协同进行。其中按次序进行结合中剪枝操作与量化操作依旧是两阶段操作，很难实现剪枝与量化相互关联共同优化，得到最优的量化剪枝联合策略，因此协同进行的工作更值得关注。

Tung 等人[39]提出了一种名为并行剪枝量化的策略，该方法的特点是对剪枝和量化这两个过程进行同步优化。尽管在每个迭代步骤中，剪枝操作先于量化进行，但在每一轮的并行剪枝量化完成后，都会通过重新训练来恢复模型的精度。这种策略不仅优化了剪枝过程，也优化了量化过程，因此在性能上优于传统的先剪枝后量化方法。然而，这种策略的计算成本相对较高。Wang 等人[40]提出了一种创新性的轻量化网络设计方法，该方法融合了网络结构搜索、剪枝以及量化三个核心阶段。此方法依赖超网络架构和精度预测器来实现高效的网络设计。具体而言，超网络的设计允许网络的超参数（例如卷积核的大小）在结构化剪枝过程中进行灵活调整。同时，引入的精度预测器能够根据网络结构和量化策略的编码来预测模型的精度，从而进一步优化剪枝和量化的策略选择。

Gi[41]等人提出了一种名为均匀变分网络量化器的方法，旨在同时进行深度神经网络的剪枝和量化以达到压缩目的。此方法融合了基于随机失活技术的非结构化剪枝与动态精度量化策略。在非结构化剪枝方面，它依赖于为每个神经元分配的随机失活概率；而在量化方面，则根据剪枝过程中获得的概率，动态地调整每个权重的量化级别。通过这种方式，该方法成功地在剪枝与量化操作之间建立了有效联系，从而实现了两者的有机结合。但上述方法主要是为卷积神经网络设计的，没有考虑 Transformer 的特有结构，针对 Transformer 的压缩加速方法需要对现有的方法进行改进。

1.3 论文主要工作及结构安排

基于前述分析，本文深入探讨了针对视觉 Transformer 的联合压缩方法。本文通过剪枝技术对原始网络输入值的维度进行精简，同时运用量化手段将高精度的权重和激活值参数转换为低精度表示。本研究旨在探索模型量化与剪枝相结合的压缩方案之潜力。此外，研究以图像分类任务为应用场景，专注于研究如何有效压缩视觉 Transformer 模型，以便于其在边缘设备上的实际应用与部署。具体研究内容包括：

（1）针对视觉 Transformer 的量化算法。本文针对自注意头中异质分布的激活提出了一种针对每个头的激活量化策略，以最小化多头自注意力层中的近似误差。此外，本文使用运行估计学习量化参数，降低量化参数的存储成本。在保证模型性能的前提下将模型量化为 8 比特，很大程度上压缩和加速了原始的视觉 Transformer 模型。消融实验表明，MHA 模块相较于 FFN 模块对量化操作更敏感，可以进一步对 FFN 进行压缩。

（2）在本文提出的量化算法的基础上，结合针对激活值输入维度的剪枝算法，提出针对视觉 Transformer 的量化剪枝联合优化算法。在要修剪的组件之前加入用来设置掩码的重要性评分，并使用 L1 正则化来对其稀疏化，修剪重要性分数较小的维度获得轻量化网络。在保持模型高精度的同时，在原本的的量化方案基础上进一步降低模型大小和算成本。消融实验表明，剪枝率与模型参数减少的比例以及浮点运算次数的缩减比率高度一致，同时剪枝方法的有效性并未因稀疏率的减小而显著减弱，进一步证明了该策略的稳定性和有效性。

（3）本文在 ImageNet-100 上以 ViT 及其变体进行了广泛实验，结果表明，本文提出的量化剪枝联合优化方法能够实现灵活的内存节约，同时达到与默认训练策略相当甚至更好的性能，验证了本文算法的有效性。

本文的组织结构如下：

第一章为绪论。首先简单介绍了本课题的研究背景和研究意义；其次以模型量化和模型剪枝为切入点，分析了卷积神经网络和 Transformer 模型压缩的研究现状，提出了量化剪枝联合优化压缩方案的研究目标；最后对本论文的主要工作内容以及各章节的内容安排进行了概括叙述。

第二章为理论与相关知识，主要阐述了 Transformer 模型和模型压缩的相关内容。首先，本文详细描述了 Transformer 模型在文本和图像处理领域的不同变体结构，深入解析了文本 Transformer 与视觉 Transformer 的核心处理流程。其次，在神经网络模型压

缩的章节中，本文全面介绍了常见的神经网络模型压缩技术，并针对本文所采用的模型量化和模型剪枝的技术手段及其特点进行了详尽的阐述。

第三章详细介绍了本文所提出的视觉 Transformer 模型量化方法。首先，针对卷积神经网络和 Transformer 的训练感知量化方法进行了简要回顾。随后，重点阐述了本文所采用的专门针对视觉 Transformer 的模型量化方案，包括针对多头自注意力层的分头激活量化策略和使用运行估计来学习量化参数等；最后在 ImageNet 数据集上进行实验，结果表明本文所提量化方法显著地实现了对原始视觉 Transformer 模型的高效压缩和加速，消融实验的结果表明，本文所提出的分头量化策略能更好的保留模型性能，同时相较于注意力层，前馈神经网络层对量化操作更加敏感。

第四章对本文提出的视觉 Transformer 模型量化剪枝联合优化方法进行了介绍。首先阐述了量化剪枝联合压缩的可行性与必要性；然后介绍了本文使用的量化剪枝联合优化框架，再对具体的剪枝策略与量化策略进行详细介绍；最后在 ImageNet 数据集上进行实验，结果表明本算法在保持模型高精度的同时，可以在第三章提出的量化方案基础上进一步降低计算成本与模型参数。消融实验表明，通过精心设计剪枝率，可以在不损害模型准确度的前提下，实现模型的轻量化和计算效率的提升。

最后一部分是结论与致谢，该部分对全文的工作内容及所取得的成果进行了全面性的梳理与总结，并对未来可能的研究方向进行了展望与探讨。

2 理论基础与相关知识

2.1 Transformer 模型

2.1.1 文本 Transformer

Transformer 是一个基于自注意力机制的模型，其结构呈现编码器-解码器的架构[6]，如图 2.1 所展示。在进行文本处理任务之前，数据会先通过词嵌入层（Embedding Layer）进行转换，这一层负责将语言中的词汇映射为可供处理的向量形式，从而有效地表达原始文本所蕴含的语义内容。紧接着，位置编码层（Positional Encoding, PE）会为每个词汇添加位置信息，确保模型能够准确捕捉到文本中的顺序关系。在计算过程中，编码器部分仅执行一次运算，而解码器则需要循环进行运算，直至预测出结束信号为止[6]。下面将一一介绍 Transformer 模型架构的重要组件。

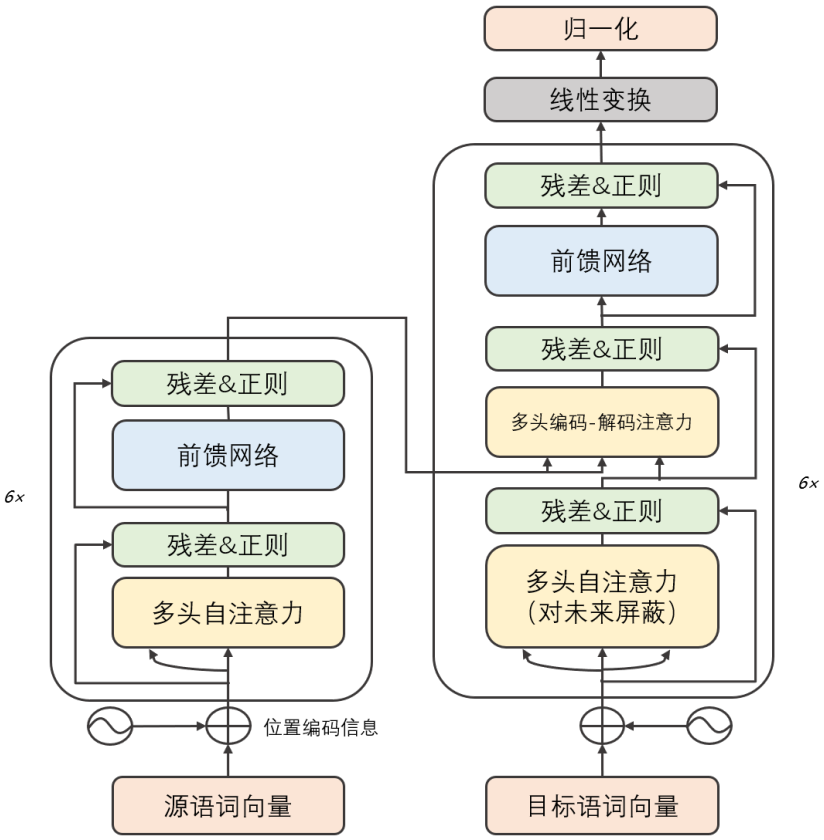


图 2.1　Transformer 结构图[42]

（1）位置编码

由于 Transformer 模型并非类似于 RNN 的循环结构来捕捉输入的序列顺序，而是通过并行处理的方式来同时处理所有输入，因此使用位置编码层为模型计算提供输入单词

序列的词序信息。Transformer 中利用正弦函数和余弦函数来计算固定的位置编码，如公式(2-1)和(2-2)所示[42]：

$$PE(pos,2i)=\sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{2.1}$$

$$PE(pos,2i+1)=\cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{2.2}$$

其中 i 代表词向量的维度索引，d_{model} 代表 Transformer 模型隐藏层特征嵌入维度，PE 为位置编码矩阵，pos 代表单词在句子中的位置索引。这种编码方式保证了不同位置在所有维度上都能获得不同的位置编码。

（2）编码器-解码器

在 Transformer 中，编码器由多个编码器子层堆叠而成。如图 2.2 所示，编码器的每个子层中包含多头自注意力层（Multi-Head Attention, MHA）和简单前馈神经网络（Feed Forward Network, FFN）两部分[6]。为了便于训练，每个子层的输出都采用了残差连接和层归一化来确保稳定的数据特征分布，降低信息损失，最后一层编码器的输出作为输入送到解码器进行交互多头注意力计算。

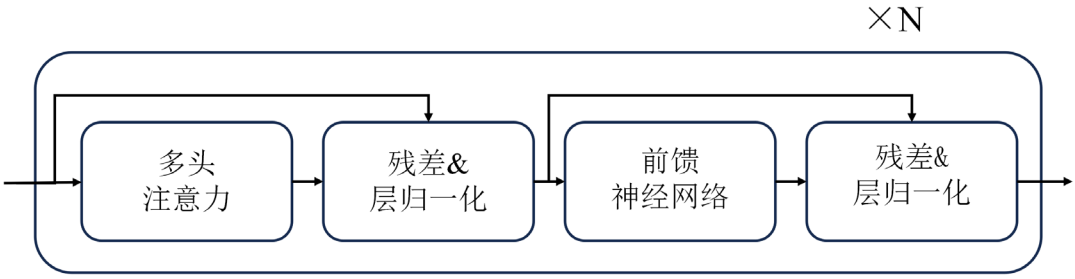


图 2.2　Transformer 编码器结构图

解码器每个子层包含遮掩多头自注意力层、多头自注意力层和简单前馈神经网络三个部分，如图 2.3 所示。遮掩多头自注意力层利用一个掩码矩阵，限制了每个位置对后续位置的注意力，从而保证预测仅依托于已生成的序列，拼屏蔽未来时间步的信息。在此基础上，解码器的交叉注意力层中，键（Keys）和值（Values）来自编码器的输出，查询（Query）来自解码器的上一子层，从而实现从编码器中得到所需的特征信息完成解码预测。解码器的最终输出会先通过线性层进行变换，随后经由 softmax 层进行归一化处理，最终将解码的 token 序列转化为对应的概率分布，进而指导序列生成任务的执行。

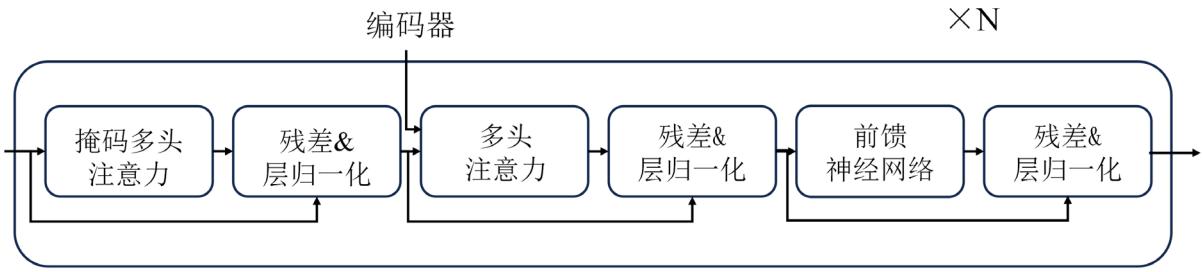


图 2.3　Transformer 解码器结构图

（3）注意力机制

注意力机制在模型中发挥着关键作用，它能够协助模型筛选出更为重要的信息，进而做出更为精确和高效的决策。在 Transformer 模型中，通过运用缩放点积注意力机制，可以计算出各个单词间的注意力分数，这使得每个单词都能获取到全文语义信息，进而有效地捕捉到整个序列中单词间的长距离依赖和全局上下文关系。计算公式如下[42]。

$$Attention(Q,K,V)=softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.3}$$

其中 Q、K、V 分别代表查询向量、键向量和值向量，d_k 表示键向量维度。

Transformer 在搭建编码器和解码器时所使用的是多头自注意力。多头自注意力由多个缩放点积注意力模块并行组合在一起，首先通过不同权重的线性层将 Q、K、V 投影至不相同的空间，在不同的投影空间内计算 Q 和 K 的内积得到注意力分数，并对其进行缩放和 softmax 处理，以突出重要的信息。然后用处理后的分数加权 V 得到单头注意力的输出，最后拼接所有头部的单头自注意力后通过线性层进行输出。图 2.4 为多头自注意力的结构图。

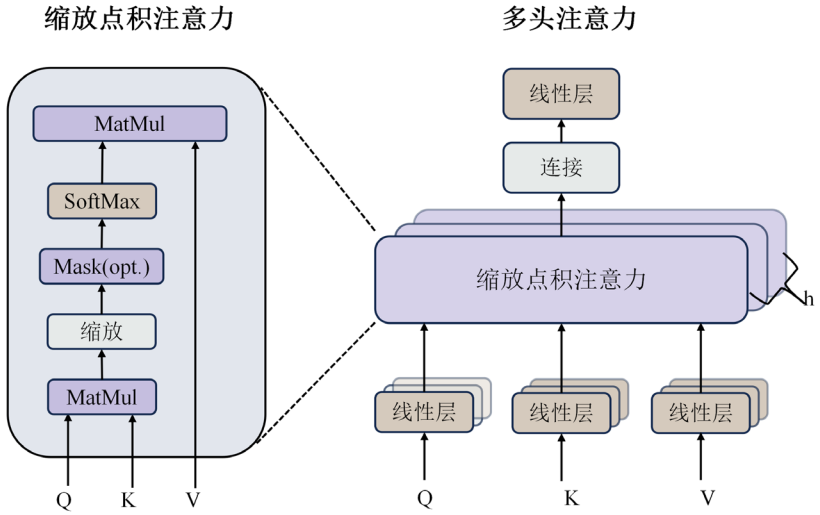


图 2.4　多头注意力结构图[42]

因为在不同的子投影空间内自注意力有不同的分布，因此多头自注意力可以从不同的角度寻找输入数据之间的关联信息，从而让每个注意力头关注每个词的不同特征部分，均衡单个注意力头关注的单一特征表达所带来的偏差。

2.1.2 视觉 Transformer

鉴于 Transformer 模型在自然语言处理任务中展现出的卓越性能，越来越多的研究者开始探索将其应用于计算机视觉领域。Dosovitskiy[43]等人将原始的 Transformer 模型直接用于图像分类，提出了一种完全基于自注意力机制的视觉 Transformer 模型 ViT（Vision Transformer），结构如图 2.5 所示。

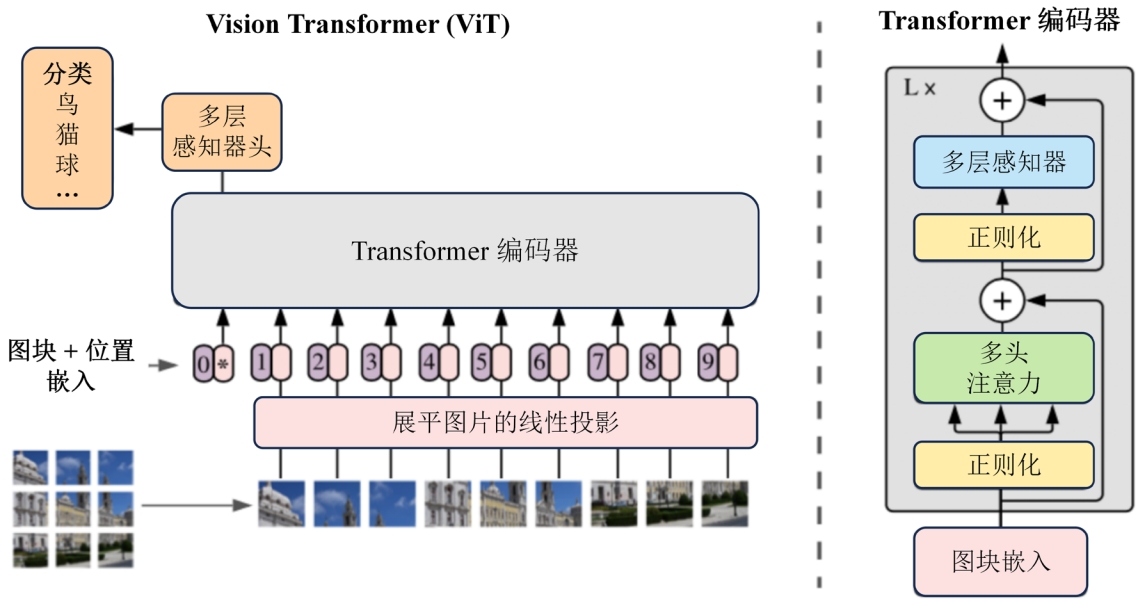


图 2.5　ViT 模型结构图[43]

ViT 的核心思想是将图像分割成一系列的小块，这些小块被视为序列中的元素，类似于 NLP 中的单词。然后，这些图像块被线性嵌入为一维的向量，并附加位置编码以保留它们在原始图像中的空间信息。这些向量随后被送入标准的 Transformer 结构中进行处理。ViT 的核心步骤主要为图像块嵌入、位置编码、Transformer 编码器和分类头四部分。

（1）图像块嵌入

在图像块嵌入步骤中，首先输入图像被分割成若干个小的正方形区域，每个区域称为一个 patch。这些 patches 按照图像的顺序被展平成一维的向量序列，并通过一个线性层转换成模型所需维度的嵌入。这个嵌入过程类似于自然语言处理中单词到词向量的转

换，将原始的二维图像数据转化为可以被 Transformer 处理的序列化数据，为模型提供了一种有效的方式来捕获图像中的局部特征。

（2）位置编码

由于 Transformer 架构本身并不具有处理序列元素顺序的能力，位置编码确保了模型能够利用图像块的空间排列信息，ViT 的位置编码直接使用了一维的位置嵌入变量。

（3）Transformer 编码器

ViT 包含多个 Transformer 编码器层，每层都由自注意力机制和前馈神经网络组成，与文本 Transformer 的编码器结构相同。

（4）分类头

分类头是位于编码器输出端的一个组件，它主要负责将编码器提取的高级特征转换为最终的分类预测，通常由一个或几个全连接层构成。为了捕获整个图像的信息，ViT 会使用一个特殊的 cls token 与图像 patches 一起被送入 Transformer 编码器。经过所有 Transformer 编码器层的处理后，最终的 cls token 被用于图像的分类任务，在最后一个编码器层的输出中，cls token 的表示被送入分类头，以预测图像的类别。

ViT 通过这些关键部分的结合，能够有效地处理视觉信息并在图像分类等任务上取得了卓越的性能。这种将 Transformer 架构应用于视觉领域的方法开辟了新的研究方向，包括对 ViT 结构的改进和优化，以及将其扩展到其他视觉任务上。

2.2 模型压缩方法

深度学习网络压缩的目标是，利用神经网络参数和结构存在的冗余性，对既有模型进行压缩处理，从而在尽量不损害模型性能的前提下，获得参数量更少、结构更为简洁的模型。当前主流的深度神经网络压缩方法根据其形式的不同大体上可以划分为四类：低秩分解、模型量化、模型剪枝和知识蒸馏。

模型剪枝和模型量化是当前最受关注的两类深度神经网络压缩方法，前者直接移除网络中的冗余参数，后者将高精度存储的权值参数以低位精度表示。它们在优化角度和压缩形式上完全独立，可以同时应用和互相补充，从而实现更高的压缩比例和更好的模型性能。故本文从量化剪枝联合优化方面对视觉 Transformer 压缩方法展开研究。

2.2.1 模型量化

为了保持较高的精度，神经网络中一般使用 64 位或 32 位浮点型数据进行计算。量

化是指用低精度的值来表示高精度的网络权重参数和激活值参数进行计算，由此降低内存消耗并加快训练进程。举例来说，通过将 32 位浮点型数据转换为 4 位整型数据来压缩模型，这种方式能够将模型的存储需求降低到原来的十六分之一，从而实现节省存储资源和简化计算操作的目的。

参数量化根据量化角度的不同有许多分类方式。根据量化后数值的间隔是否均匀，可以分为线性量化与非线性量化。线性量化的参数间隔的大小相等，实现简单且计算效率高；非线性量化采用非线性函数来映射浮点数到离散值，使用不均匀的划分区间，更有效地捕获信号信息，但实现复杂。

根据量化后范围是否对称，可以分为对称量化和非对称量化。在对称量化中，权重和激活值被量化到一个以零为中心的均匀分布的整数范围内，正数和负数被对称地量化到整数上；非对称量化允许使用不同的量化因子和零点，使得量化的过程可以更灵活地适应数据的实际分布，但需要额外存储量化因子和零点，增加了一些计算复杂度。

根据量化过程中是否包含训练环节，我们可以将量化方法分为两类：训练后量化与量化感知训练。在训练后量化方法中，对已训练完成的模型直接进行参数量化，此后不再进行微调操作。相对而言，量化感知训练方法在模型训练阶段就已经将量化的影响纳入考量，它能够在训练中根据实际情况动态地调整量化因子的大小，从而尽可能地减少精度的损失。因此，量化感知训练通常能够比训练后量化保持更高的精度。

此外，量化的精度也对模型性能有重要影响[44]。实际研究显示，分层量化虽计算快但误差大，逐通道量化误差小但内存成本高。因此，分组量化作为一种平衡方法，被广泛应用[10,27]。

2.2.2 模型剪枝

已有研究结果证实，对于特定的深层神经网络，部分权重和激活值存在冗余，即某些权重与激活值对神经网络的输出结果影响微乎其微，网络剪枝通过删除对性能不敏感的冗余和不起决定性作用的连接来减少模型的参数量和计算量。剪枝后的精度会有所损失，通常通过微调来保持性能。经典的剪枝流程如图 2.6 所示，主要分为三个步骤：

（1）对模型进行预训练，得到满足精度要求的模型作为基准模型。

（2）根据特定的剪枝策略对基准模型执行剪枝操作，减少网络结构中的层数、通道、权重等，得到轻量级的模型。

（3）使用原始训练数据对修建后的模型重新进行训练，微调以弥补因剪枝造成的精

度损失。

图 2.6　经典模型剪枝流程

根据剪枝对象重要性的评估不同，模型剪枝可以分为基于阈值、正则化惩罚项以及权重重要性的剪枝等方法。基于阈值的剪枝通过设定一个标准，根据权重的绝对值低于此标准的百分比来逐层或全局移除权重，这是一种直观且普遍有效的方法。然而，使用单一阈值可能会忽视个别连接的贡献度，有时会对模型性能产生负面影响。正则化惩罚项剪枝利用正则化技术在训练过程中倾向于较小的权重，训练完成后移除接近零的权值，但这种方法可能会造成所有参数以相同速率指数衰减，降低剪枝性能。而基于权重重要性的剪枝则通过评估每个权重对模型性能的贡献度来进行剪枝，优先移除那些对损失影响最小的权重，从而在剪枝过程中保持了模型性能。

根据剪枝粒度的不同，剪枝可以分为结构化剪枝和非结构化剪枝。非结构化剪枝主要通过权重剪枝和神经元剪枝来实现，是细粒度的剪枝策略，相较于规则化剪枝能取得很高的稀疏率，可以很好的平衡性能与压缩率，但在计算过程中需要设计适应于稀疏化的算子和硬件来进行加速,否则无法体现其价值。

不同于非结构化剪枝以单个元素为单位进行剪枝，结构化剪枝方法以网络中的完整算子为对象，规整的移除模型蕴含的冗余结构来减少网络的计算，是粗粒度的剪枝策略。按照剪枝的对象，结构化剪枝可以分为层剪枝、滤波器剪枝、通道剪枝、分组剪枝等。尽管结构化剪枝无需专门的硬件和软件库支持，但由于其剪枝的粒度相对较大，这可能导致模型在性能上遭受显著损失。

3 视觉 Transformer 量化优化方法

3.1 引言

一般情况下，神经网络模型的训练与推理阶段均采用单精度浮点数实现，但是浮点数计算会占用大量的计算资源与储存资源，极大地限制了 Transformer 等大模型在实际应用中的部署。参数量化可以在不改变网络结构的前提下将全精度值参数替换为低比特位宽进行网络计算，显著降低硬件平台的计算功耗。

以往针对卷积神经网络的策略简单将张量切片成固定大小的块，未考虑维度差异，不适用于 Transformer，因为它忽视了不同自注意力头通常有不同的注意力模式，即 MSA 层中每个自注意力头的激活具有独特的分布特征。因此对于量化操作，每个自注意力头的激活都应该有其独特的裁剪范围和偏移量。基于此，本文提出了一种基于每个自注意头的激活量化策略，同时使用运行估计来学习量化参数。

3.2 视觉 Transformer 量化方案

3.2.1 分头激活量化策略

针对自注意力头中异质分布的激活，本文提出了一种分头激活量化策略，如图 3.1 所示。其中不同的颜色代表不同的量化分组，B、N 和 D 分别代表批量大小、序列长度和隐藏通道的维数，N_h 和 D_h 分别指的是自注意力头的数量和每个头的维数。对于标准的输入序列，根据一定数量的隐藏通道来对激活值进行分组。对于查询、键、值以及注意力，本文基于自注意力头对激活值进行分组。

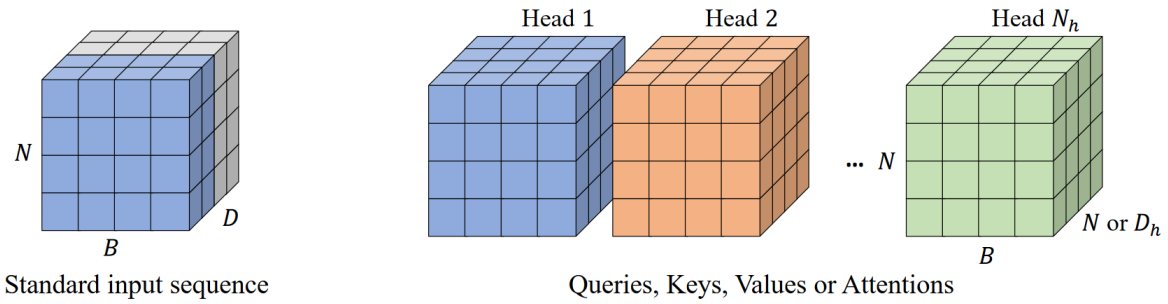


图 3.1　不同激活分组量化策略

具体实现为：对于 MSA 层中第 h 个头的输入激活中的元素 $x \in X^h$，进行 k 比特非对称线性量化，量化公式如下：

$$\overline{x} = \text{clip}(\text{round}((x - \beta^h) \cdot \frac{2^k - 1}{\alpha^h}), 0, 2^k - 1) \tag{3.1}$$

其中 α^h 和 β^h 是分别代表第 h 个头的量化裁剪范围和偏移量化参数。$\text{clip}(x, x_{low}, x_{up})$ 将任何数 x 裁剪到 $[x_{low}, x_{up}]$ 范围内。round(·) 表示舍入函数，本文采用随机舍入，理论上与最近舍入相比，它保证了更小的概率误差界限[48]。具体的定义如下：

$$\text{round}(x) = \begin{cases} \lceil x \rceil, p = x - \lfloor x \rfloor \\ \lfloor x \rfloor, p = \lceil x \rceil - x \end{cases} \tag{3.2}$$

对于激活值的反量化公式，定义如下：

$$\hat{x} = \overline{x} \cdot \frac{\alpha^h}{2^k - 1} + \beta^h \tag{3.3}$$

训练中采用量化感知训练实现伪量化，鉴于 round 函数在微分时的非连续性，本文选择利用直通估计器（straight-through estimator, STE）这一技术来确保梯度在反向传播时的有效计算。

3.2.2 量化参数学习

为了确定每一层的量化参数 α 和 β，一些先前的工作提出使用每个样本的统计数据[45,46]或基于梯度的方法[49,50]。具体来说，利用样本统计数据的方法是通过计算每个样本在每一层的当前最小值和最大值来确定量化参数的。然而，这种方法效率较低，并且需要额外的内存来存储这些量化参数。另一方面，基于梯度的方法，例如 LSQ+等非对称量化技术，可能会增加内存使用，因为它们需要存储压缩的激活和大量上下文信息，这与本研究减少内存占用的目标相悖。

因此，本研究提出了一种新的方法，即利用运行估计来动态更新训练过程中的量化参数。这种方法旨在更有效地计算量化参数，同时减少内存消耗，从而更好地满足优化内存使用的目标。其公式可以表示为：

$$\alpha^h = \lambda \alpha^h + (1 - \lambda)(\max(\mathrm{X}^h) - \min(\mathrm{X}^h)) \tag{3.4}$$

$$\beta^h = \lambda \beta^h + (1 - \lambda) \min(\mathrm{X}^h) \tag{3.5}$$

其中 λ 为超参数。由于量化参数在每个头中的不同样本之间是共享的，使用运行估计可以在训练时节省更多的内存。例如，设 N_h 为 MSA 层中自注意力头的数量，B 为批量大小，使用运行估计，只需要为 Softmax 层存储 $2N_h$ 额外的参数，与整体内存占用相

比这可以忽略不计。相比之下，使用每个样本的统计数据需要为一层保存 $2BN_h$ 额外的参数，内存需求较大。

3.2.3 其他模块量化

上文提及的分头激活量化策略应用在 Transformer 模型中的查询、键、值和注意力上，对于 FFN 层中的标准输入序列等其他类型的激活，本文采用通道分组量化方案。同时对于所有模块的权重参数使用对称线性量化进行压缩。

3.3 实验与分析

在本节中，本文针对图像分类任务的视觉 Transformer 模型 ViT 及其变体评估了所提出的分头激活量化策略的性能。结果表明，本文提出的量化策略在压缩和加速原始的视觉 Transformer 模型方面表现出色。此外，消融实验表明，本文提出的量化方法可以对模型灵活进行压缩以实现内存与精度的均衡。

3.3.1 实验设置

（1）数据集和评估指标

本文在 ILSVRC-2012ImageNet（在下文中将其称为 ImageNet）数据集上进行实验，该数据集包括 1.2M 个训练图像，来自 1K 个类别，以及 50K 个验证图像。ImageNet-100 是 ImageNet-1K 的子集，随机抽样了 100 个类及其相应的图像作为训练集和验证集。按照常用做法，本文通过 Top-1 准确率来衡量模型性能。此外，还记录了模型大小及计算量衡量量化效果。

（2）对比方法

为了验证分头激活量化策略的有效性，本文在 ViT 上评估了本文的量化方案。由于最近的模型通常在模型深度和宽度方面有多种变体，如表 3.1 所示。在实验中，采用视觉 Transformer 模型 ViT-S、ViT-B 作为基线模型。

表 3.1　ViT 模型变体细节表

模型	层数	隐藏层特征嵌入维度	多层感知器特征嵌入维度	头数
ViT-S	12	384	1536	6
ViT-B	12	768	3072	12
ViT-L	24	1024	4096	16

（3）实现细节

默认情况下，所有模型都是在单台机器 RTX 3080*2 上训练的，ImageNet 的总批量大小为 1,024。本文采用了 AdamW 优化器，配以余弦衰减学习率调度器。将初始学习率和权重衰减设置为1×10^{-4}和5×10^{-2}。此外，本文在与每个基线比较时采用了相同的训练策略。式(3.4)(3.5)中的λ设置为 0.9，α和β通过第一次训练迭代时的最小-最大激活值进行初始化，文中量化默认为 8 比特量化。

3.3.2 量化结果分析

在表 3.2 中，将基线模型与量化后的模型进行了比较。总体上，本文提出的量化可以在推理时减少大约一半的内存消耗，同时实现与强基线相当甚至更好的性能。

表 3.2　量化结果表

模型	量化方案	模型大小（MB）	推理时间（s）	Top-1 准确率(%)
ViT-S	Baseline	85.4	64.3	95.8
	8bit, weight only	42.7	57.5	96.2
	4bit, weight only	21.3	54.3	95.7
	8bit, weight and activation	42.7	19.2	95.4
ViT-B	Baseline	344.2	263.6	97.8
	8bit, weight only	272.3	234.6	97.7
	4bit, weight only	136.1	226.7	97.7
	8bit, weight and activation	272.3	81.7	98.0

根据观察结果，可以明确地看到，本文所介绍的分头激活量化方法相较于基础模型展现出了令人瞩目的卓越性能。此方法通过精细的量化策略，不仅成功地将模型的大小缩减了一半，更在推理速度上取得了显著的突破，而且几乎没有影响到模型的准确性。值得一提的是，在 ViT-B 上的实验中，采用了分头激活量化方法的模型在精度上反而超越了原始的基准模型。这一结果充分证明了本文所提出的量化方案不仅在减小模型大小方面有效，更能在某些情况下提升模型的性能。另一方面，尽管仅对模型权重进行量化也能大幅度减小模型的大小，但在减少推理时间上的效果并不显著。以上结果证实了本文提出的量化方案的有效性。

3.3.3 消融实验与分析

在本节中，基于 ImageNet-100 数据集，在视觉 Transformer 模型 ViT-S 上进行消融实验，探究不同量化粒度的影响和 transformer 不同模块对量化的敏感度，以便实现更加灵活的量化方案。

（1）不同量化粒度的影响

为了探索不同量化粒度的影响，本文将所提出的分头激活量化策略与分层量化进行比较。结果显示在表 3.3 中。

表 3.3　不同量化粒度结果表

模型	量化粒度	模型大小（MB）	推理时间（s）	Top-1 准确率（%）
ViT-S	Baseline	85.4	64.3	95.8
	Layer	42.7	18.8	95.1
	Head	42.7	19.2	96.3

总体而言，研究发现各种量化粒度的方案在内存使用上并没有显著的差异，这主要是因为每一网络层仅需存储为数不多的量化参数，而这些参数的数量并不会因为量化粒度的不同而发生大幅度的变化。因此，在内存消耗方面，各种量化方案的表现是相对一致的。

在实施分层量化策略时，本文对同一层的全部激活值采用了相同的裁剪范围和偏移量来进行量化处理。这种方法虽然在实施上相对简单，但在性能上却并未能超越基线水平。这可能是因为分层量化策略过于简单，没有充分考虑到不同部分之间的差异性，从而导致量化后的模型性能受到了一定的限制。相较之下，本文所提出的分头量化方法则显得更为精细和有效。这种方法充分考虑了不同头和通道组的差异化统计特征，对每个部分进行了更为精准的量化处理。通过这种方式，不仅有效地减少了量化误差，还提高了模型的性能。与分层策略和基准方法相比，分头量化方法在推理速度与性能之间达成了更为理想的平衡，展现出了更优的表现。

（2）不同模块对量化的敏感度

MSA 层和 FFN 层是 Transformer 模型的主要组件。同时，它们在模型推理期间消耗了大量计算资源。为了研究压缩不同模块的效果，本文在 ImageNet-100 上训练 ViT-B 模型，结果显示在表 3.4 中。

表 3.4　量化 Transformer 不同模块结果表

模型	量化模块	模型大小（MB）	Top-1 准确率（%）
ViT-S	Baseline	85.4	95.8
	MHA	66.4	95.7
	FFN	66.4	91.5
	CLS	80.1	95.8
	ALL	42.7	92.4

首先通过深入的观察和分析，可以发现，在变换器中前馈神经网络部分对于各种调整和变化表现得最为敏感。尽管对前馈神经网络进行量化处理，如同对多头注意力层进行量化一样，都可以有效地缩减模型的整体大小，从而达到节省存储空间和提高处理速度的目的。然而，对前馈神经网络的量化处理却会显著地降低模型的性能。这可能是因为量化过程中引入的误差对前馈神经网络的影响更为显著，导致其无法准确地处理和传递信息。

相反，根据本文所创新提出的多头注意力量化方法，研究发现多头注意力部分对于量化处理表现出了较高的稳健性。这意味着，即使在经过量化处理后，多头注意力部分几乎不会对模型的整体性能产生影响，适合进行量化操作。此外，同时压缩多头注意力和前馈神经网络层实现了最大的内存节省，但也导致了较低的精准度。不过，人们可以根据实际需求和应用场景，针对模型中的不同模块进行有选择性的压缩和优化。通过这种方式，可以在保证模型性能可接受的前提下，实现内存占用和精准率之间的有效权衡。

4 视觉 Transformer 量化剪枝联合优化方法

4.1 引言

在模型压缩领域，网络剪枝和参数量化均占据着核心地位，作为两种主流的压缩技术。其中，剪枝技术专注于通过优化模型结构来实现压缩，而量化技术则聚焦于调整参数的存储效率以达到压缩目的。这两种技术的压缩机制互为补充，可协同作用，以达成更高的压缩效果。剪枝与量化的结合策略多样，包括先执行剪枝后实施量化，以及剪枝与量化并行的方式。具体到剪枝操作，根据其剪枝的粒度不同，可以分为结构化剪枝和非结构化剪枝两种形式。而在量化方面，根据是否涉及训练过程，可以分为训练感知量化和训练后量化。训练感知量化是在训练过程中引入伪量化操作，模拟量化效果，通过浮点硬件来近似实现定点操作；而训练后量化则直接对预先训练好的浮点模型进行参数量化，无需额外的训练步骤。

本节着重探讨了非结构剪枝与训练感知量化在视觉 Transformer 模型上的协同压缩优化方法。该方法旨在保持任务准确率的同时，实现更佳的模型压缩与运算加速效果。通过这种协同优化技术，可以得到更为简洁、采用低位宽表示的稀疏模型。当前，针对此类模型，已有众多研究致力于设计更适合低位宽计算和非结构化稀疏计算的专用运算结构，这些专用结构能够带来显著的能效提升。

4.2 视觉 Transformer 量化剪枝联合优化

4.2.1 量化剪枝联合优化框架

为了减少视觉 Transformer 在训练时的计算消耗，本文设计了一个通用的、节省内存的量化剪枝联合优化框架。在实现剪枝量化协同的方式上，本文的研究更多是先剪枝后量化方法的延申，将先剪枝后量化的过程多次迭代。

具体针对多头注意力机制而言，其优化后的计算流程如图 4.1 所示。步骤为：①对于输入的激活值矩阵 Q、K、V，首先进行剪枝，再量化 Q、K 到 4 位或 8 位定点整数，输入线性层，计算缩放点击注意力；②计算分数矩阵并反量化并送入 softmax 函数得到 P；③将 P 量化到 4 位或 8 位定点整数并将其与量化后的 V 相乘，得到单头注意力机制输出结果；④拼接单头注意力机制输出，先剪枝再量化，通过线性层得到多头注意力输出。第②步与第③步进行反量化与量化操作的原因是直接将量化的数据送入 softmax 会

显著影响模型在数据集上的精度。

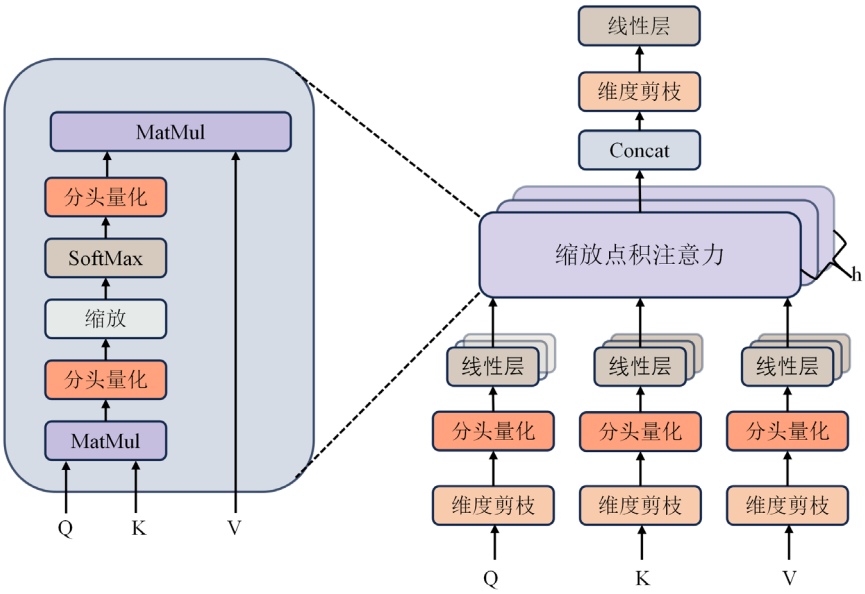


图 4.1　注意力层优化框架

4.2.2 剪枝策略

典型的视觉 Transformer 架构包含多个关键组件，如多头注意力、多层感知器、层归一化、激活函数以及 shortcut 连接。在这种结构中，主要的计算负担集中在多头自注意力和多层感知器模块上。为了有效降低 Transformer 的计算复杂度，本文所提出的剪枝策略特别针对多头自注意力层和多层感知器进行优化。

在多头自注意力层和多层感知器层中，全连接层是计算和存储负担的主要来源。为了降低计算量，本剪枝算法通过修剪全连接层的输入参数来实现优化。具体而言，算法通过学习线性投影的相关性重要分数，进而修剪线性投影的维度。对于特征 $X \in \mathbb{R}^{n \times d}$，其中 n 表示通道数量，d 表示每个通道的维度，本剪枝算法就是根据每个维度的重要性分数将 $X \in \mathbb{R}^{n \times d}$ 修剪为 $X \in \mathbb{R}^{n \times \hat{d}}$，其中 $\hat{d} < d$，从而从线性投影的相关组件中剔除无用特征。

为了方便学习重要性分数，本文将重要性分数假设为实数，这样重要性分数可以随着梯度的下降进行优化。为了增强重要性分数的稀疏性，本研究采用了 L1 正则化方法，并将其纳入损失函数中，其中稀疏率 λ 为超参数。在完成稀疏化正则训练后，对正则化后的重要性分数绝对值进行排序，根据预设的剪枝率来确定阈值，将待剪枝的特征维度中重要性分数绝对值低于阈值设置为 0，其余设置为 1 得到掩码，从而对特征维度进行

修剪。多头自注意力层和多层感知机层都包含两个全连接层，在这些全连接层之前进行维度修剪，以有效降低全连接层的计算复杂度，整体的剪枝框架如图 4.2 所示。

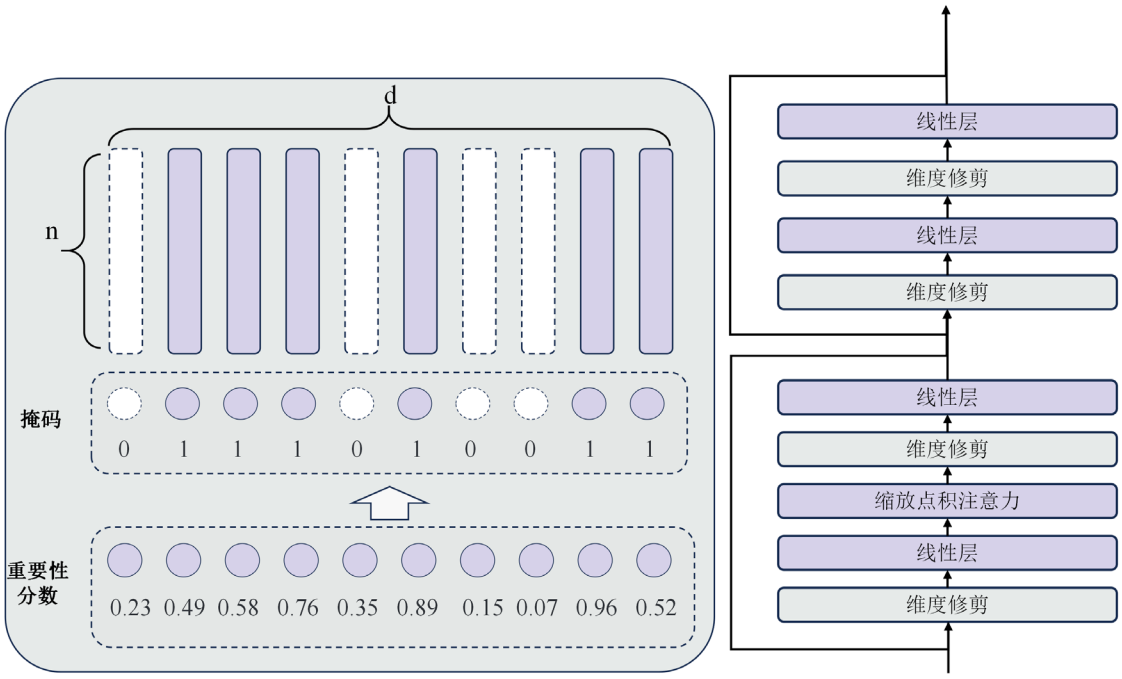


图 4.2　视觉 Transformer 剪枝框架图

4.2.3 量化策略

本节采用的量化策略与 3.2 节一致，对 Transformer 模型中的查询、键、值和注意力使用分头激活量化策略。对于 FFN 层中的标准输入序列等其他类型的激活，采用通道分组量化方案，同时对于所有模块的权重参数本文使用对称线性量化进行压缩。

4.3 实验与分析

在本节中，本文针对图像分类任务的流行视觉 Transformer 模型 ViT 评估了所提出的量化剪枝联合优化策略。结果表明，本文所提的联合优化方案能很大程度上压缩和加速原始的视觉 Transformer 模型，且优于单一的量化方案。此外，消融实验研究确定了稀疏率、剪枝率与参数量和每秒浮点运算数之间的关系。

4.3.1 实验设置

（1）数据集和评估指标

本文在 ImageNet100 数据集上进行实验。通过 Top-1 准确率来衡量模型性能。此外，本文还记录了模型参数量及计算量衡量压缩和加速效果。

（2）对比方法

本文在 ViT 模型及其变体上评估了本文的压缩方案，为了全面评估这一压缩方案的有效性和实用性，采用选取了视觉 Transformer 模型中的两个重要代表——ViT-B 和 ViT-S 作为基线模型来进行实验。

（3）实现细节

默认情况下，实现细节与 3.3.1 节保持一致，稀疏率设置为 0.0001，稀疏正则化训练的学习率设置为6.25×10^{-6}。实验中设置的剪枝率包括 0.2 和 0.4。

4.3.2 量化剪枝联合优化结果分析

在表 4.2 中，将基线模型与量化剪枝后的模型进行了比较。总体上，本文提出的量化剪枝框架可以在量化的基础上进一步压缩和加速模型，同时实现与基线相当甚至更好的性能。

表 4.1　量化剪枝结果表

模型	参数量（M）	FLOPS（G）	Top-1 准确率（%）
ViT-S	22.1	4.6	95.8
量化剪枝 ViT-S（剪枝率 20%）	17.2	3.6	95.5
量化剪枝 ViT-S（剪枝率 40%）	12.5	2.6	94.8
ViT-B	86.4	17.6	97.8
量化剪枝 ViT-B（剪枝率 20%）	67.3	13.7	97.1
量化剪枝 ViT-B（剪枝率 40%）	48.4	10.1	96.7

相较于基准模型 ViT-S，我们在减少模型 20%的维度后观察到，FLOPS 降低了 22%，而模型的 Top-1 精确度仅轻微下滑 0.3%。在削减 40%的维度时，FLOPS 进一步降低了 43%，同时 Top-1 精确度也仅降低了 1%。

与 ViT-B 基准模型相比，我们的方法在几乎不损失精确度的情况下，同样有效地实现了模型的压缩。这些结果清晰地展示了 L1 正则化所带来的正则稀疏训练，对于在维持模型精确度的同时实现模型剪枝具有显著的效果。以上结果证实了本文提出的量化剪枝框架的有效性。

4.3.3 消融实验与分析

在 ImageNet-100 数据集的基础上，本研究对 ViT-B 模型实施了消融实验。通过运用各异的稀疏度和剪枝率，本文全面评估了所提出的针对视觉 Transformer 的量化剪枝方法的效能。实验目的在于明确稀疏率、剪枝率与参数保留率以及每秒浮点运算数保留率之间的内在联系。

表 4.2　不同稀疏率量化剪枝结果表

稀疏率	剪枝率	参数量（M）	FLOPS（G）	Top-1 准确率（%）
0.00001	0.2	67.3	13.7	97.1
	0.4	48.4	10.0	96.7
	0.5	37.5	8.1	95.6
	0.6	28.1	6.2	93.8
0.0001	0.2	66.2	13.7	97.1
	0.4	47.2	10.0	96.8
	0.5	37.5	8.1	95.5
	0.6	28.1	6.3	94.1

消融实验的结果详见表4.2，在对使用0.00001稀疏率进行训练的ViT-B模型进行尺寸缩减时，我们注意到，随着剪枝比例的提升，模型的准确率会略有下降。类似的情况在使用0.0001稀疏率训练的ViT-B模型中也得到了体现。然而，值得指出的是，在相同的剪枝比例下，由不同稀疏率训练出的模型，在精度上并无显著差异。

进一步观察发现，剪枝比例与模型的参数量和每秒浮点运算数的降低比例是一致的。这表明，正则稀疏率在修剪策略的有效性上并未产生显著影响，即通过精心设计剪枝率，可以在不损害模型准确度的前提下，实现模型的轻量化和计算效率的提升。

结　论

目前 Transformer 模型在自然语言处理和计算机视觉领域均表现出了出色的性能，但是由于其参数量大和计算复杂度高等特点，限制了其在资源有限的的设备上的部署。本研究针对视觉 Transformer 模型在图像分类任务上的应用场景，利用剪枝与量化联合压缩和加速视觉 Transformer 以便在边缘设备部署，主要工作和成果如下：

（1）探索了一种基于每个自注意力头的头部激活量化策略，该策略针对自注意头中异质分布的激活分头进行量化，可以更好地捕获量化裁剪范围和偏移，从而最小化多头自注意力层中的近似误差。此外，模型使用运行估计学习量化参数，以降低量化参数的存储成本。在保证模型性能的前提下将模型量化为 8 比特，很大程度上压缩和加速了原始的视觉 Transformer 模型。消融实验表明，FFN 模块相较于 MHA 模块对量化操作更敏感，可以针对模型中不同的模块进行不同压缩实现一个可接受的内存-精度权衡。

（2）在提出的量化算法的基础上，结合针对激活值的剪枝算法，提出了一种针对视觉 Transformer 的量化剪枝联合优化算法。在要修剪的组件之前添加可学习的重要性分数，并使用 L1 正则化来对其稀疏化，修剪重要性分数较小的维度获得轻量化网络。在原本的的量化方案基础上进一步降低计算成本与模型参数，同时保持模型的精度。消融实验表明，剪枝率与参数量以及浮点运算数减少的比率相匹配，同时剪枝方法的有效性不会受到稀疏率的显著影响。这意味着，通过精心设计剪枝率，可以在不损害模型准确度的前提下，实现模型的轻量化和计算效率的提升。

（3）为了充分验证上述两种压缩方法的实际性能表现，本文在 ImageNet-100 上进行了广泛的实验，结果表明，本文提出的量化剪枝联合优化方法能够实现灵活的内存节约，同时达到与默认训练策略相当甚至更好的性能，验证了本文算法的有效性。

致　　谢

感谢我的指导老师张琨，如果没有她非常有前瞻性的实施每周一次组会的话，我相信这篇论文是怎么也搞不出来了，以我的拖延症到时候大概率是换一个简单的方向随便应付了事。同时在论文的指导和修改方面张琨老师也给予了我很多的帮助，再次感谢一波。

感谢我的父母，在我过往的二十多年中一直支持我，我知道我的学费也是一笔不小的费用，他们一直在努力的给我一个更好的环境，所以我也在尽我所能去学习，得到一个更好的结果。最近几年回去明显发觉老爸的白头发变多了，不知道什么时候开始他也是满头白发了，老妈倒是没什么变化，但是一直说自己的更年期到了。希望老爸老妈可以健康百岁，爱你们。

感谢我在大学这四年认识的所有朋友与老师，感谢他们在我人生最青春的四年中的陪伴，在此点名感谢梁晓剑、李文清、冯秋实、苑梓康、李心航，感谢一直以来对我的包容，认识你们真的是一件幸运的事情，虽然没有在大学收获甜甜的爱情，但是收获了满满的友情啊。还有大学四年中遇到的和蔼可亲的老师们，感谢刘超老师在学业上的悉心教诲，感谢郝闻达老师在科创上的支持，有了他们才让我更加有信心面对学习和科创上的种种难题。

还要感谢吕慧婷学姐，很开心能够在大学中认识到如此优秀与美好的学姐，无论是在我刚刚步入大学感到茫然无措时，还是在我保研途中感到焦虑崩溃时，学姐都是很有耐心的安慰与鼓励我，让我可以有个地方歇一歇。虽然我平常喜欢一个人独自面对情绪的肆虐，但是有时感觉有个人可以依靠的感觉确实很安心。虽然学姐已经回到了广东工作了，但是希望这份来自秦皇岛的感谢能飘过大半中国的版图成为一种祝福到达学姐身边。秦皇岛的月色很美，惠州的晚风也很温柔，希望以后月色能与晚风相遇，如此美景总是值得期待的。祝福学姐万事如意，开心顺遂，永远有一个自由有趣的灵魂。

最后感谢我自己，至今已经是在东秦的第四个年头了，如无意外这篇论文将代表着我本科生涯的结束。感谢你至今为止坚持不懈的努力，拿到了美赛的 O 奖和挑战杯的一等奖，最后成功推免到了南开，我知道这一路上你很累，但是回报也算是丰收，给你一个大大的大拇指。哦对了，最后这半年减肥 35 斤，真的太强了，希望你以后能继续保持健康的生活，在学习之余也要有一个健康的身体。

参考文献

[1] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[2] Lan Z, Chen M, Goodman S, et al. Albert: A lite bert for self-supervised learning of language representations[J]. arXiv preprint arXiv:1909.11942, 2019.

[3] Wang W, Xie E, Li X, et al. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 568-578.

[4] Carion N, Massa F, Synnaeve G, et al. End-to-end object detection with transformers[C]//European conference on computer vision. Cham: Springer International Publishing, 2020: 213-229.

[5] 陈婉玉.面向视觉处理的 Transformer 模型压缩算法研究及实现[D].北京邮电大学,2023.DOI:10.26969/d.cnki.gbydu.2023.000130.

[6] 邓晗珂.Transformer 模型压缩算法研究及硬件加速器实现[D].华南理工大学,2022.DOI:10.27151/d.cnki.ghnlu.2022.001897.

[7] 徐睿, 马胜, 郭阳, 等. 基于 Winograd 稀疏算法的卷积神经网络加速器设计与研究[J]. Computer Engineering & Science/Jisuanji Gongcheng yu Kexue, 2019, 41(9).

[8] Gao C, Rios-Navarro A, Chen X, et al. EdgeDRNN: Recurrent neural network accelerator for edge inference[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2020, 10(4): 419-432.

[9] Pan Z, Chen P, He H, et al. Mesa: A memory-saving training framework for transformers[J]. arXiv preprint arXiv:2111.11124, 2021.

[10] Zafrir O, Boudoukh G, Izsak P, et al. Q8bert: Quantized 8bit bert[C]//2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS). IEEE, 2019: 36-39.

[11] Voita E, Talbot D, Moiseev F, et al. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned[J]. arXiv preprint arXiv:1905.09418, 2019.

[12] Jiao X, Yin Y, Shang L, et al. Tinybert: Distilling bert for natural language understanding[J]. arXiv preprint arXiv:1909.10351, 2019.

[13] 黄震华，杨顺志，林威，等. 知识蒸馏研究综述[J]. 计算机学报，2022，45(3): 624-653.

[14] 高晗，田育龙，许封元，等. 深度学习模型压缩与加速综述[J]. 软件学报，2020，32(1): 68-92.

[15] 邵仁荣，刘宇昂，张伟，等. 深度学习中知识蒸馏研究综述[J]. 计算机学报，2022，45(8): 1638-1673.

[16] Dettmers T. 8-bit approximations for parallelism in deep learning[J]. arXiv preprint arXiv:1511.04561, 2015.

[17] Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 2704-2713.

[18] Lin X, Zhao C, Pan W. Towards accurate binary convolutional neural network[J]. Advances in neural information processing systems, 2017, 30.

[19] Courbariaux M, Bengio Y, David J P. Binaryconnect: Training deep neural networks with binary weights during propagations[J]. Advances in neural information processing systems, 2015, 28.

[20] Hou L, Yao Q, Kwok J T. Loss-aware binarization of deep networks[J]. arXiv preprint arXiv:1611.01600, 2016.

[21] Juefei-Xu F, Naresh Boddeti V, Savvides M. Local binary convolutional neural networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 19-28.

[22] Zhu C, Han S, Mao H, et al. Trained ternary quantization[J]. arXiv preprint arXiv:1612.01064, 2016.

[23] Achterhold J, Koehler J M, Schmeink A, et al. Variational network quantization[C]//International conference on learning representations. 2018.

[24] Mellempudi N, Kundu A, Mudigere D, et al. Ternary neural networks with fine-grained quantization[J]. arXiv preprint arXiv:1705.01462, 2017.

[25] Mishra A, Cook J J, Nurvitadhi E, et al. Wrpn: Training and inference using wide reduced-precision networks[J]. arXiv preprint arXiv:1704.03079, 2017.

[26] Wang K, Liu Z, Lin Y, et al. Haq: Hardware-aware automated quantization with mixed precision[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 8612-8620.

[27] Shen S, Dong Z, Ye J, et al. Q-bert: Hessian based ultra low precision quantization of bert[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(05): 8815-8821.

[28] Zadeh A H, Edo I, Awad O M, et al. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference[C]//2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2020: 811-824.

[29] Bai H, Zhang W, Hou L, et al. Binarybert: Pushing the limit of bert quantization[J]. arXiv preprint arXiv:2012.15701, 2020.

[30] Chen T, Frankle J, Chang S, et al. The lottery ticket hypothesis for pre-trained bert networks[J]. Advances in neural information processing systems, 2020, 33: 15834-15846.

[31] Gordon M A, Duh K, Andrews N. Compressing bert: Studying the effects of weight pruning on transfer learning[J]. arXiv preprint arXiv:2002.08307, 2020.

[32] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. arXiv preprint arXiv:1510.00149, 2015.

[33] Sanh V, Wolf T, Rush A. Movement pruning: Adaptive sparsity by fine-tuning[J]. Advances in neural information processing systems, 2020, 33: 20378-20389.

[34] Li Y, Luo F, Tan C, et al. Parameter-efficient sparsity for large language models fine-tuning[J]. arXiv preprint arXiv:2205.11005, 2022.

[35] Michel P, Levy O, Neubig G. Are sixteen heads really better than one?[J]. Advances in neural information processing systems, 2019, 32.

[36] Fan A, Grave E, Joulin A. Reducing transformer depth on demand with structured dropout[J]. arXiv preprint arXiv:1909.11556, 2019.

[37] LeCun Y, Denker J, Solla S. Optimal brain damage[J]. Advances in neural information processing systems, 1989, 2.

[38] Lagunas F, Charlaix E, Sanh V, et al. Block pruning for faster transformers[J]. arXiv preprint arXiv:2109.04838, 2021.

[39] T Tung F, Mori G. Clip-q: Deep network compression learning by in-parallel pruning-quantization[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 7873-7882.

[40] Wang T, Wang K, Cai H, et al. Apq: Joint search for network architecture, pruning and

quantization policy[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 2078-2087.

[41] Gil Y, Park J H, Baek J, et al. Quantization-aware pruning criterion for industrial applications[J]. IEEE Transactions on Industrial Electronics, 2021, 69(3): 3203-3213.

[42] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[43] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. arXiv preprint arXiv:2010.11929, 2020.

[44] Jegou H, Douze M, Schmid C. Product quantization for nearest neighbor search[J]. IEEE transactions on pattern analysis and machine intelligence, 2010, 33(1): 117-128.

[45] Chen J, Zheng L, Yao Z, et al. Actnn: Reducing training memory footprint via 2-bit activation compressed training[C]//International Conference on Machine Learning. PMLR, 2021: 1803-1813.

[46] Fu F, Hu Y, He Y, et al. Don't waste your bits! squeeze activations and gradients for deep neural networks via tinyscript[C]//International Conference on Machine Learning. PMLR, 2020: 3304-3314.

[47] Cordonnier J B, Loukas A, Jaggi M. On the relationship between self-attention and convolutional layers[J]. arXiv preprint arXiv:1911.03584, 2019.

[48] Croci M, Fasi M, Higham N J, et al. Stochastic rounding: implementation, error analysis and applications[J]. Royal Society Open Science, 2022, 9(3): 211631.

[49] Choi J, Wang Z, Venkataramani S, et al. Pact: Parameterized clipping activation for quantized neural networks[J]. arXiv preprint arXiv:1805.06085, 2018.

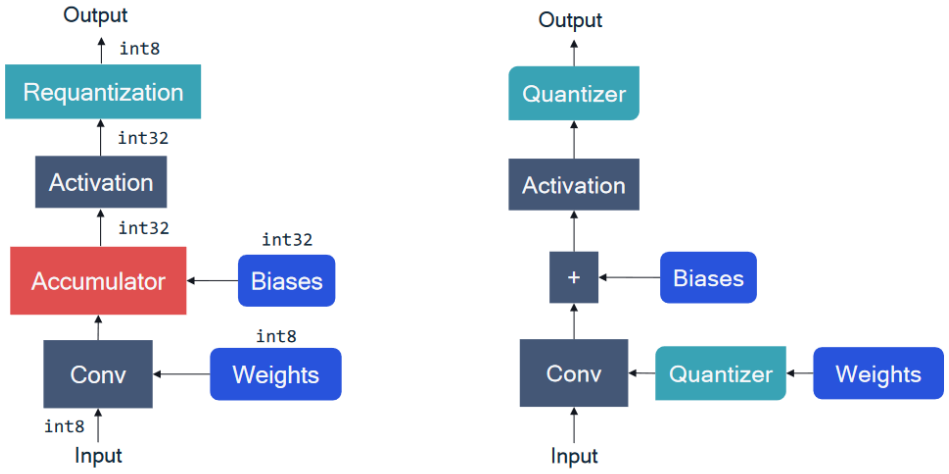
[50] Bhalgat Y, Lee J, Nagel M, et al. Lsq+: Improving low-bit quantization through learnable offsets and better initialization[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. 2020: 696-697.

附　　录

附录 A

A White Paper on Neural Network Quantization

To test how well a neural network would run on a quantized device, we often simulate the quantized behavior on the same general purpose hardware we use for training neural networks. This is called quantization simulation. We aim to approximate fixed-point operations using floating-point hardware. Such simulations are significantly easier to implement compared to running experiments on actual quantized hardware or using quantized kernels. They allow the user to efficiently test various quantization options and it enables GPU acceleration for quantization-aware training as described in section 4. In this section, we first explain the fundamentals of this simulation process and then discuss techniques that help to reduce the difference between the simulated and the actual on-device performance.



(a) Diagram for quantized on-device inference with fixed-point operations.

(b) Simulated quantization using floating-point operations.

Figure 4: Schematic overview of quantized forward pass for convolutional layer: a) Compute graph of actual on-device quantized inference. b) Simulation of quantized inference for general-purpose floating-point hardware.

Previously, we saw how matrix-vector multiplication is calculated in dedicated fixed-point hardware. In figure 4a, we generalize this process for a convolutional layer, but we also include an activation function to make it more realistic. During on-device inference, all the inputs (biases, weight and input activations) to the hardware are in a fixed-point format. However, when we simulate quantization using common deep learning frameworks and

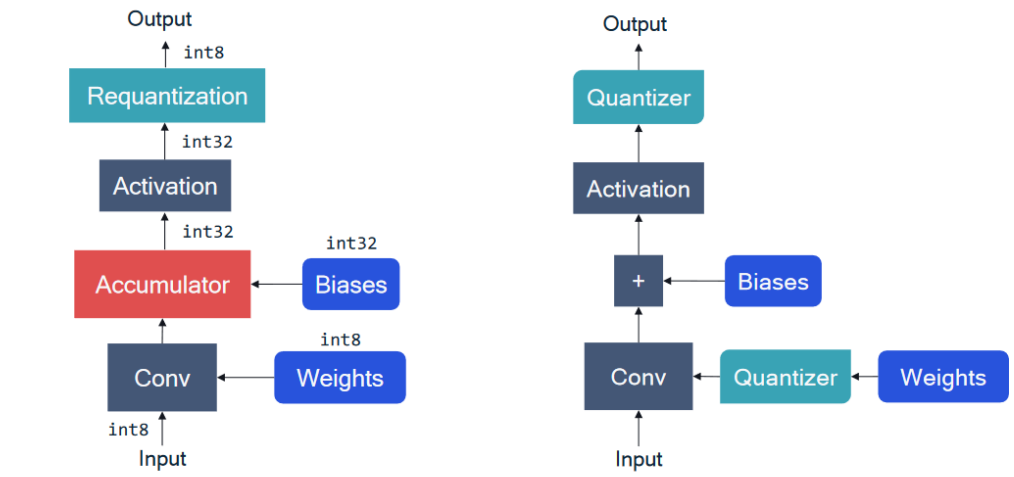
general-purpose hardware these quantities are in floating-point. This is why we introduce quantizer blocks in the compute graph to induce quantization effects.

Figure 4b shows how the same convolutional layer is modelled in a deep-learning framework. Quantizer blocks are added in between the weights and the convolution to simulate weight quantization, and after the activation function to simulate activation quantization. The bias is often not quantized because it is stored in higher-precision. In section 2.3.2, we discuss in more detail when it is appropriate to position the quantizer after the non-linearity. The quantizer block implements the quantization function of equation (7) and each quantizer is defined by a set of quantization parameters (scale factor, zero-point, bit-width). Both the input and output of the quantizer are in floating-point format but the output lies on the quantization grid.

中文译文 A

神经网络量化白皮书

为了测试神经网络在量化设备上的运行情况，我们通常在用于训练神经网络的通用硬件上模拟量化行为，这被称为量化模拟。我们的目标是采用浮点硬件来近似固定点运算。与在实际的量化硬件上运行实验或使用量化内核相比，这类模拟显著更易于实现。它们允许用户高效地测试各种量化选项，并且如第 4 节所述，它支持对量化感知训练进行 GPU 加速。在这一节中，我们首先解释这种模拟过程的基础知识，然后讨论帮助减少模拟与实际设备性能之间差异的技术。



(a) Diagram for quantized on-device inference with fixed-point operations.

(b) Simulated quantization using floating-point operations.

图 4：卷积层量化前向传播的示意概览：a) 实际设备上量化推理。b) 通用浮点硬件量化推理的模拟

在前文中，我们看到矩阵-向量乘法是如何在专用固定点硬件中计算的。在图 4a 中，我们将这个过程泛化到卷积层，但同时为了更加真实，我们还包括了一个激活函数。在设备上进行推理时，所有输入（偏置、权重和输入激活）都采用固定点格式呈现给硬件。然而，当我们使用常见的深度学习框架和通用硬件模拟量化时，这些数值是使用浮点格式的。这就是为什么我们在计算图中引入量化器模块以引发量化效果的原因。

图 4b 展示了同一个卷积层如何在深度学习框架中被建模。在权重和卷积之间加入了量化器模块以模拟权重量化，以及在激活函数之后以模拟激活量化。偏置通常不进行量化，因为它存储在更高精度中。在第 2.3.2 节中，我们将更详细地讨论在非线性之后置入量化器的适当时机。量化器模块实现了方程（7）中的量化函数，每个量化器都由一组量化参数（尺度因子、零点、位宽）定义。量化器的输入和输出都是浮点格式，但输出位于量化网格上。

附录 B

A Memory-saving Training Framework for Transformers

While both the training speed and the memory consumption are critical bottlenecks for Transformer training, this work focuses on the latter. As in common practice (e.g., checkpointing [13] and hashing [24]), there is a trade-off between the speed and memory consumption at training time for all ACT frameworks [16, 17]. In Mesa, the additional training overhead includes the quantization/de-quantization, the computation of min/max values, stochastic rounding and tensor reshaping for highly accurate CUDA dispatching. These operations can happen in almost all layers to achieve significant memory-saving while sacrificing the training speed. In an extreme case, Mesa could result in the throughput decreases by half on a single GPU when compressing a whole model.

On the other hand, considering that distributed training has been widely adopted in modern deep learning [1,3,26], the training cost can be dominated by data loading and communication overhead among GPUs, especially for small models. For example, we will show in Section 5.1 that training PVT-Ti [3] with Mesa only requires additional 15% GPU hours. Besides, to further mitigate the additional training overhead, we modularize Mesa such that it can flexibly target different components in a model during training. In practice, we suggest to compress the most memory-hungry modules (e.g., backbones) to achieve a target memory-saving with acceptable training speed. In Section 5.2, we show in Table 7 that

compressing different operations (e.g., Softmax and GELU) can achieve different speed and memory trade-offs. We will also provide the experiments on compressing different modules (e.g., MSA and FFN) in Table 8 for reference.

中文译文 B

神经网络量化白皮书

虽然训练速度和内存消耗对于 Transformer 训练都是关键的瓶颈，但这项工作主要关注于后者。像常见的实践（例如，检查点保存[13]和哈希[24]）那样，在训练时对于所有 ACT 框架[16, 17]来说，存在速度与内存消耗之间的权衡。在 Mesa 中，额外的训练开销包括量化/去量化、最小/最大值的计算、随机取整和为了高精度的 CUDA 调度而进行的张量重塑。这些操作几乎可以在所有层进行，以在牺牲训练速度的同时实现显著的内存节省。在极端情况下，当压缩整个模型时，Mesa 可能导致单个 GPU 上的吞吐量减少一半。

另一方面，考虑到分布式训练在现代深度学习[1,3,26]中已经广泛采用，训练成本可能会受到数据加载和 GPU 间通信开销的主导，特别是对于小型模型。例如，我们将在第 5.1 节展示，使用 Mesa 训练 PVT-Ti[3]只需要额外增加 15%的 GPU 小时数。此外，为了进一步减轻额外的训练开销，我们将 Mesa 模块化，使其能在训练期间灵活地针对模型中的不同组件。实践中，我们建议压缩最耗内存的模块（例如，主干网络），以在可接受的训练速度下实现目标内存节省。在第 5.2 节，我们将在表 7 中显示压缩不同操作（例如，Softmax 和 GELU）可以实现不同的速度和内存权衡。我们还将提供在表 8 中压缩不同模块（例如，MSA 和 FFN）的实验供参考。