面向高效部署的视觉大模型推理加速方法研究

左霖泽
中车工业研究院有限公司，中国·北京 100071

摘要：尽管剪枝与量化领域已有大量研究，但二者的平衡及联合优化以提升推理性能仍是学术界和工业界关注的核心问题。本研究针对视觉大模型在计算复杂度和部署成本方面的挑战，旨在探索剪枝与量化技术的有效结合以实现推理加速，提出一种融合结构化剪枝与混合精度量化的轻量化方法。该方法先通过对（Batch Normalization，BN）[1]层通道缩放因子及卷积层、全连接层权重分布的分析，结构化裁剪冗余通道 / 核以减少计算量和模型体积，再在剪枝后结合量化策略实现性能与精度的权衡。以 YOLOv8 为实验对象，先在 90%、83%、50% 三种剪枝率下开展剪枝实验，其中 50% 剪枝率时，模型参数量从原始 11.14M 减少至 5.68M，计算量从 14.28G 降至 7.45G，经微调后 mAP 保持在 0.6861，仅略低于原始模型；随后选用 50% 剪枝率剪枝后的模型，采用 FP16 与 INT8 混合精度量化策略进一步优化，使推理时间压缩至 1.4ms，准确率保持在 0.6578，为相对最优策略。实验结果表明，该方法成功压缩了模型体积、降低了计算负担，在保持精度基本稳定的前提下显著提升了推理速度，在提高模型部署效率方面展现了良好的应用前景，特别适用于边缘计算和移动端场景。

关键词：视觉大模型；模型轻量化；量化；剪枝；模型部署

Research on Inference Acceleration Methods for Visual Large Models with Efficient Deployment

Zuo Linze
CRRC ACADEMY, China Beijing 100071

Abstract: Although there has been extensive research on pruning and quantization, the balance and joint optimization of these techniques to improve inference performance remain a key concern in both academia and industry. This study addresses the challenges of computational complexity and deployment cost in visual large models and aims to explore the effective combination of pruning and quantization to accelerate inference. We propose a lightweight method that integrates structured pruning with mixed-precision quantization. The approach first analyzes the channel scaling factors of (Batch Normalization, BN)[1] layers and the weight distributions of convolutional and fully connected layers to structurally prune redundant channels/kernels, reducing both computation and model size. After pruning, a quantization strategy is applied to balance performance and accuracy. Experiments are conducted on YOLOv8 with pruning rates of 90%, 83%, and 50%. At a 50% pruning rate, the model parameters decrease from 11.14M to 5.68M and the computation reduces from 14.28G to 7.45G, while the mAP after fine-tuning remains 0.6861, slightly lower than the original model. Subsequently, the 50% pruned model is further optimized using a mixed FP16 and INT8 quantization strategy, compressing inference time to 1.4ms while maintaining an accuracy of 0.6578, representing the relatively optimal strategy. The results demonstrate that this method effectively reduces model size and computational burden, significantly improves inference speed while maintaining stable accuracy, and shows strong potential for enhancing deployment efficiency, particularly in edge computing and mobile scenarios.

Keywords: Visual large models; Model lightweighting; Quantization; Pruning; Model deployment

1 视觉大模型推理及加速概述

随着深度学习技术的飞速发展，视觉大模型（如 Vision Transformer[2]、ResNet[3] 系列等）在图像分类、目标检测、语义分割等计算机视觉任务中取得了突破性成果，其性能精度不断逼近甚至超越人类水平。然而，视觉大模型的参数规模与计算复杂度呈指数级增长，动辄数十亿甚至上百亿的参数的模型，不仅需要巨大的存储资源，更在推理过程中消耗大量计算算力，导致其难以部署在嵌入式设备、移动终端等资源受限平台，严重制约了人工智能技术在边缘计算、实时场景中的普及与应用。

推理加速技术作为解决视觉大模型部署难题的核心手段，能够在保证模型精度损失可控的前提下，降低模型的

计算量、存储量与功耗，实现模型的高效部署。其中，剪枝与量化技术因其操作简便、通用性强、无需重新设计模型架构、资源消耗低等优势，成为目前视觉大模型轻量化与推理加速的主流技术路径。深入研究剪枝与量化的联合优化方法，解决二者融合过程中的精度与效率平衡问题，对于推动视觉大模型在实际场景中的落地应用、提升人工智能技术的普惠性具有重要的理论价值与工程意义。

1.1 视觉大模型剪枝概述

模型剪枝[4] 技术旨在去除神经网络中的冗余参数，从而减小模型的体积，降低计算资源消耗，并可能加速推理过程。在深度学习的应用中，尤其是视觉大模型中，随着网络层数和参数数量的增多，模型的计算复杂度和内存消耗也随之增加，导致推理速度变慢，并难以在边缘设备或移动设备上高效运行。剪枝技术的应用可以有效应对这些问题，使得视觉大模型能够在资源受限的硬件环境中实现更高效的推理和部署。

网络剪枝的目标是通过删除一些不重要或冗余的网络参数，减少计算量和存储需求，从而使得模型更轻量化和高效化。网络剪枝的核心思想是，通过识别和去除网络中的“无用”部分，来提升网络的运行速度并减少其计算资源的消耗，同时尽可能地保持其准确性。

1.2 视觉大模型量化技术概述

量化[5] 是将值从连续的浮点空间映射到离散的、位数更低的整数空间的过程，其中全精度值被映射到称为“量化级别”的有限集合中的某个值。这种映射通常通过一个常数分段函数来实现，例如将输入范围[r1, r5] 划分为几个区间，每个区间对应一个量化级别 q1, q2, q3, q4 等。例如，范围[r1, r2] 内的所有值被映射到 q1，范围[r2, r3] 映射到 q2，以此类推。两个连续量化级别之间的距离称为步长。准确的量化依赖于计算最优的量化级别和步长（见图 1）。在下文中，我们将从不同的角度讨论神经网络中各种类型的量化方法。

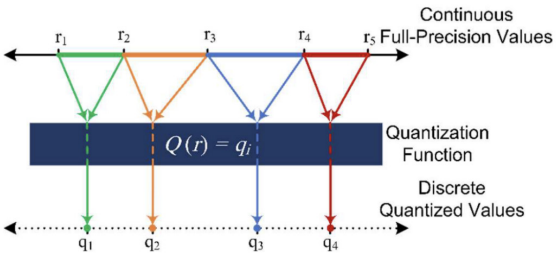


图1 使用量化函数将连续全精度值映射到离散量化值

针对神经网络，尤其是视觉大模型，量化技术的应用旨在减少模型的存储需求、降低计算复杂度并加速推理过程。随着模型规模的增长，其巨大的参数量和中间激活值占用了大量的内存和计算资源，限制了模型在资源受限设备（如移动端和边缘设备）上的部署。量化通过用更少的比特表示原始值，有效缓解了这些问题，使得视觉大模型能够在更高效的硬件环境中运行。

2 视觉大模型剪枝量化联合方法实验

剪枝和量化作为两种重要的模型优化技术，因其操作简单、无需特定结构设计且资源消耗低，在通用性和易实现性上相较于知识蒸馏、紧凑架构设计和动态网络技术更具优势，适合更广泛的轻量化需求。

模型剪枝通过移除冗余的神经网络参数，减少计算量和存储需求。剪枝不仅可以有效地减小模型的体积，还能提升推理速度，同时保持模型的精度。剪枝方法包括权重剪枝、结构剪枝和动态剪枝等，依据不同的剪枝策略，能够在不同场景下实现最佳性能。

模型量化作为另一种常见的优化手段。它通过将网络中的浮点数权重和激活值转换为低位数表示（如整型），有效地降低了计算量和存储消耗。量化后的模型不仅加速了推理过程，还能够减少功耗，适用于硬件平台的限制，尤其是在嵌入式系统或移动设备上，具有重要的应用价值。量化方法包括权重量化、激活量化、以及联合量化等，它们的设计目标是尽量保持原模型的准确性同时降低其资源需求。

模型剪枝和量化的结合，能够在保证模型精度的前提下显著提高推理效率，尤其适用于大规模视觉模型在资源受限环境中的部署。因此，深入研究模型剪枝和量化的技术，对于提升视觉大模型的推理加速性能、推动人工智能技术在边缘计算和移动端的应用具有重要意义。

目前，尽管在这两个领域已有大量的研究成果，但如何在剪枝与量化之间找到一个有效的平衡点，如何在剪枝与量化的联合优化中获得更高的推理性能，依然是学术界和工业界关注的核心问题。因此，本研究旨在探索剪枝与量化技术的前沿研究进展，并探索其有效结合，以实现对现有技术进行系统性的分析与优化及视觉大模型的推理加速。

本文提出了一种融合结构化剪枝与混合精度量化的视觉大模型轻量化方法，以提升其在资源受限平台上的运行效率。

2.1 视觉大模型剪枝量化联合方法

本研究提出的轻量化方案主要通过结构化剪枝及低位量化两种手段实现模型加速：首先基于 BN 层的通道缩放

因子（γ 值）及卷积层、全连接层的权重分布，识别网络中冗余或贡献较小的通道 / 核，并以结构化方式进行裁剪，减少计算量和模型体积；在剪枝后保留的参数基础上，采用基于层级动态特性差异的低位量化策略，实现性能与精度的权衡。

2.2 基于结构化方法的视觉大模型剪枝

结构化剪枝[6] 通过物理去除整个模块（如卷积核、神经元、通道或层）来压缩神经网络的规模，相比非结构化剪枝，结构化剪枝对网络结构产生了更为深远的影响。这种方法在减少计算量和内存占用方面具有显著优势，特别适用于在硬件平台（如 GPU、TPU 等）上加速推理。结构化剪枝的一个主要优点是它能够显著提高硬件执行效率，并且不依赖于特定的硬件加速器或软件库，这使得它在实际应用中具有更广泛的适用性。因此，结构化剪枝逐渐成为主流的剪枝技术，本研究主要关注于在参数依赖关系的约束下进行神经网络的结构化剪枝。

（1）稀疏训练：

在进行剪枝操作之前，首要步骤是对模型进行稀疏性训练[7]。稀疏训练的目的是引导网络在保持性能的同时，逐步抑制部分冗余参数的权重幅度，使其趋近于零，从而为后续的结构化剪枝提供可靠依据。只有在网络充分适应稀疏结构后，剪枝才能在尽量减少信息丢失的前提下有效执行。

当前主流的视觉大模型网络结构中都引入了批归一化 BN 层，以加快训练收敛速度。BN 层的核心作用在于保持各层激活输入的分布一致，从而缓解“内部协变量偏移”问题，使激活值分布始终位于非饱和区间，避免梯度消失，提升反向传播的稳定性。在具体实现中，BN 层通过学习缩放因子 γ 和偏移系数 β，对输入特征进行归一化处理，使得每一层能够适应输出特征的分布变化。BN 层以 z_{in} 和 z_{out} 分别作为输入与输出，利用 γ 和 β 对标准化后的数据进行线性变换，完成如下操作：

$$z_{out}=\gamma\cdot\frac{z_{in}-\mu}{\sqrt{\sigma^2+\varepsilon}}+\beta \tag{1}$$

其中 μ 和 σ^2 分别表示当前最小批的均值和方差，ε 为防止除零的极小常数。通过该过程，BN 层能够稳定网络训练过程，提高模型学习能力，并为基于通道缩放因子的剪枝提供有效的结构基础。

稀疏性训练的核心目标是对 BN 层中的缩放因子 γ 和偏移系数 β 引入稀疏性约束。在损失函数中增加对 γ 和 β 的正则化项，通过惩罚其绝对值使其部分值趋近于零，从而为后续通道级剪枝提供依据。引入的稀疏约束强度由超参数 λ 控制，λ 值越大，稀疏性约束越强，网络中冗余参数被压缩得越彻底。该策略能够在不显著影响模型收敛性的前提下，有效提升剪枝效率与模型压缩效果。

稀疏性训练通过对 BN 层中的缩放因子 γ 和偏移系数 β 引入正则项，实现对网络结构的压缩。在原始损失函数基础上，引入稀疏性约束后的整体损失函数形式如下：

$$L=\sum_{(x,y)}\ell(f(x),y)+\lambda_1\sum_{\gamma}g(\gamma)+\lambda_2\sum_{\beta}g(\beta) \tag{2}$$

其中，$\ell(f(x),y)$ 表示模型的预测损失，λ_1 与 λ_2 分别控制对 γ 和 β 的稀疏性惩罚权重。对于 L1 范数的稀疏约束，其表达形式为：

$$g(\gamma)=|\gamma|,g(\beta)=|\beta| \tag{3}$$

与原始梯度项相比，增加的梯度部分如下：

$$\Delta_{\gamma}=\frac{\partial(\lambda_1\cdot g(\gamma))}{\partial_{\gamma}}=\lambda_1\cdot sign(\gamma) \tag{4}$$

$$\Delta_{\beta}=\frac{\partial(\lambda_2\cdot g(\beta))}{\partial\beta}=\lambda_2\cdot sign(\beta) \tag{5}$$

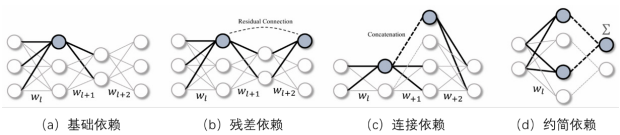
在训练过程中，为了防止稀疏约束对网络收敛造成过强干扰，采用动态调整策略对 λ_1 进行逐步衰减，其变化公式如下：

$$\lambda_1=0.01\cdot(1-0.9\cdot e^{-n/e}) \tag{6}$$

其中 n 表示当前训练步数，e 为总训练轮次。通过这种方式训练得到的稀疏模型，其通道权重分布将呈现更明显的差异，便于后续通道重要性评估与结构化剪枝的实施。此时可以基于 γ 值对通道重要性进行排序，并应用剪枝算法去除权重贡献较低的部分，从而实现有效的模型压缩。

（2）神经网络中的依赖：

本文的方法从全连接（FC）层[8] 出发展开，首先考虑一个由三个连续层组成的线性神经网络，如图 2（a）所示，分别由二维权重矩阵 ω_l、ω_{l+1} 和 ω_{l+2} 参数化。通过结构化剪枝，可以通过去除神经元来使该简单神经网络变得更加精简。在这种情况下，很容易发现一些参数之间存在依赖关系，记作 $\omega_l \Leftrightarrow \omega_{l+1}$，这就要求 ω_l 和 ω_{l+1} 的第 k 个神经元，必须同时去除 $\omega_l[k,:]$ 和 $\omega_{l+1}[k,:]$。



（a）基础依赖　（b）残差依赖　（c）连接依赖　（d）约简依赖

图 2 不同结构中相互依赖关系示意图

研究人员通常通过手动设计和模型特定的方案来处理层之间的依赖关系，并在深度神经网络上实现结构化剪枝。然而，正如图 2（b-d）所示，存在多种类型的依赖关系。逐个分析每种依赖关系是不可行的，尤其是简单的依赖关系可能会嵌套或组合成更复杂的模式。

为了解决结构化剪枝中的依赖关系问题，本研究中引入了依赖图，该方法提供了一种通用且完全自动化的依赖建模机制。

（3）依赖图方法：

为了实现结构化剪枝，首先需要根据层之间的相互依赖关系对层进行分组。目标是找到一个分组矩阵$G \in R^{L\times L}$，其中 L 代表待剪枝网络中的层数，且 G_{ij}=1 表示第 i 层和第 j 层之间存在依赖关系。为了方便起见，我们令$Diag(G)=1_{1\times L}$，以便启用自依赖关系。通过这个分组矩阵，可以很容易地找到与第 i 层具有相互依赖关系的所有耦合层，记作$g(i)$：

$$g(i)=\{j \mid G_{ij}=1\} \tag{7}$$

然而，由于现代深度网络可能包含数千层并且具有复杂的连接结构，从神经网络中估计分组模式并非易事，导致分组矩阵 G 既大又复杂。在这个矩阵中，G_{ij} 不仅由第 i 层和第 j 层决定，还受到它们之间的中间层的影响。因此，这种非局部和隐式的关系在大多数情况下无法通过简单规则来处理。为了克服这个挑战，我们并不直接估计分组矩阵，而是提出了一种等效的、易于估计的依赖关系建模方法——即依赖图，通过它可以高效地推导出 G。

首先考虑一个分组 $g=\{\omega_1,\omega_2,\omega_3\}$，其中存在依赖关系 $\omega_1 \Leftrightarrow \omega_2$、$\omega_2 \Leftrightarrow \omega_3$ 和 $\omega_1 \Leftrightarrow \omega_3$ 导出。通过仔细检查这种依赖关系建模，我们可以观察到其中存在一些冗余。例如，依赖关系 $\omega_1 \Leftrightarrow \omega_3$ 可以通过递归过程从 $\omega_1 \Leftrightarrow \omega_2$ 和 $\omega_2 \Leftrightarrow \omega_3$ 导出。具体而言，我们首先以 ω_1 为起点，检查它与其他层的依赖关系，例如：$\omega_1 \Leftrightarrow \omega_2$ 之后，ω_2 提供了一个新的起点，用于递归地扩展依赖关系，从而触发 $\omega_2 \Leftrightarrow \omega_3$。这个递归过程最终会得到一个传递关系 $\omega_1 \Leftrightarrow \omega_2 \Leftrightarrow \omega_3$。在这种情况下，我们只需要两个依赖关系就能描述分组 g 中的关系。因此，前面讨论的分组矩阵 G 对于依赖关系建模也是冗余的，因此可以压缩成一个更紧凑的形式，减少边的数量，同时保持相同的信息。

本研究证明了一个新的图 D，它衡量相邻层之间的局部依赖关系，称为依赖图（Dependency Graph），可以有效地替代分组矩阵 G。依赖图与 G 的不同之处在于，它只记录具有直接连接的相邻层之间的依赖关系。图 D 可以视为 G 的传递性约简，它包含与 G 相同的顶点，但边的数量尽可能少。形式上，D 的构建方式是，对于所有 G_{ij}=1，在 D 中存在一条从顶点 i 到 j 的路径。因此，可以通过检查 D 中顶点 i 和 j 之间是否存在路径来推导出 G_{ij}。

然而，研究发现，在实践中直接在层级上构建依赖图可能存在问题，因为一些基本层（如全连接层）可能具有两种不同的剪枝方案，例如所讨论的 $\omega_l[k,:]$ 和 $\omega_{l+1}[k,:]$，它们分别压缩输入和输出的维度。此外，网络还包含一些非参数化的操作，如跳跃连接，它们也会影响层之间的依赖关系。为了解决这些问题，我们提出了一种新的表示方法，将网络 $F(x,w)$ 分解为更细粒度的基本组件，表示为 $F=\{f_1,f_2,...,f_L\}$，其中每个组件 f_i 既可以是一个有参数的层（如卷积层），也可以是一个非参数化的操作（如残差加法）。我们不再在层级上建模关系，而是集中于层之间输入和输出之间的依赖关系。具体来说，我们将组件 f_i 的输入和输出分别表示为 f_i^- 和 f_i^+。对于任何网络，最终的分解可以形式化为 $F=\{f_1^-,f_1^+,...,f_L^-,f_L^+\}$。这种表示方法简化了依赖关系建模，并允许为同一层采用不同的剪枝方案。

基于上文提到的方法，可以将神经网络表示为方程 8，其中可以区分出两种主要类型的依赖关系，即层间依赖关系和层内依赖关系，如下所示：

$$(f_1^-,f_1^+)\leftrightarrow(f_2^-,f_2^+)\leftrightarrow...(f_L^-,f_L^+) \tag{8}$$

符号↔表示两个相邻层之间的连接性。对这两种依赖关系的分析可以得出简单但通用的依赖建模规则：

层间依赖关系（Inter-layer Dependency）：当两个层 f_i^- 和 f_j^+ 连接时，依赖关系 $f_i^-\leftrightarrow f_j^+$ 会持续出现，即 f_i^- 和 f_j^+ 之间有连接时，会产生这种依赖关系。

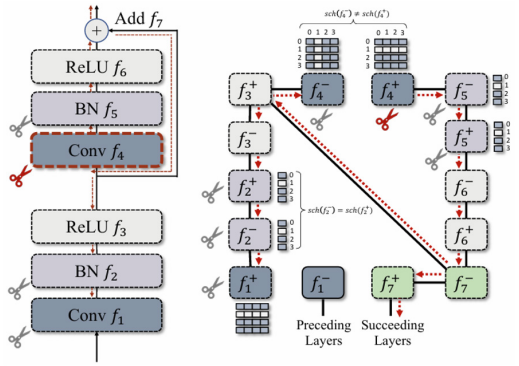
层内依赖关系（Intra-layer Dependency）：当且仅当 f_i^- 和 f_i^+ 共享相同的剪枝方案时，才会存在依赖关系 $f_i^-\leftrightarrow f_i^+$，即$\mathrm{sch}(f_i^-)=\mathrm{sch}(f_i^+)$，其中 sch 表示剪枝方案。

如果已知网络的拓扑结构，则层间依赖关系可以轻松估算。对于连接的层 $f_i^-\leftrightarrow f_j^+$，依赖关系始终存在，因为 f_i^- 和 f_j^+ 对应着网络中的相同中间特征。接下来的步骤是阐明层内依赖关系。层内依赖关系要求单层的输入和输出应同时进行剪枝。许多网络层满足这一条件，例如批量归一化层，它的输入和输出共享相同的剪枝方案，表示为$\mathrm{sch}(f_i^-)=\mathrm{sch}(f_i^+)$，因此它们将被同时剪枝，如图 3 所示。相比之下，像卷积层这样的层，其输入和输出有不同的剪枝方案，即$\omega[:,k,:,:]=\omega[k,:,:,:]$，如图 3 所示，这导致$\mathrm{sch}(f_i^-)\neq\mathrm{sch}(f_i^+)$。在这种情况下，卷积层的输入

和输出之间没有依赖关系。图 3 层的分组通过在依赖图（DepGraph）上进行递归传播实现，从 f_4^+ 开始。在这个例子中，由于上述所示的剪枝方案不同，卷积层的输入 f_4^- 和输出 f_4^+ 之间没有层内依赖关系。

基于上述规则，可以正式地建立依赖关系建模，如下所示：

$$D(f_i^-, f_j^+) = \mathbb{1}[f_i^- \leftrightarrow f_j^+] \vee \mathbb{1}[i = j \wedge \text{sch}(f_i^-) = \text{sch}(f_j^+)] \quad (9)$$



(a) CNNs (b) 在依赖图上的传播

图 3

其中，符号 ∨ 和 ∧ 分别表示逻辑运算中的“或”（OR）和“与”（AND）运算，而 1 是一个指示函数，当条件成立时返回“True”。第一个项检查由网络连接引起的层间依赖关系，而第二个项检查由共享剪枝方案在层输入和输出之间引入的层内依赖关系。值得注意的是，依赖图（DepGraph）是一个非对称矩阵，且 $D(f_i^-, f_j^+) = D(f_j^+, f_i^-)$。因此，我们可以检查所有输入 – 输出对来估计依赖图。在图 3 中，可视化了带有残差连接的卷积神经网络（CNN）模块的依赖图。

（4）分组级别剪枝：

在前面的章节中，已经提出了一种通用的方法，用于分析神经网络中的依赖关系，这自然地导致了一个分组级别的剪枝问题。评估分组参数的重要性对剪枝来说是一个重大挑战，因为它涉及多个耦合的层。在本节中，本文利用一个简单的基于范数的标准来建立一种实际的分组级别剪枝方法。

给定一个参数组 $g = \{w1, w2, ..., w_{|g|}\}$，现有的标准如 L2 范数重要性 $I(w) = \|w\|_2$ 可以为每个 $w \in g$ 生成独立的评分。估计分组重要性的自然方式是计算一个聚合评分 $I(g) = \Sigma_{w \in g} I(w)$。然而在不同层上独立估计的重要性评分往往是不可加的，因此由于分布和幅度的差异，这些评分可能是无意义的。为了让这种简单的聚合方式在重要性估计中发挥作用，本研究提出了一种分组级别的稀疏训练方法，如图 4(c) 所示，以便使得那些被置零的参数组能够被安全地从网络中移除。

具体而言，对于每一个具有 K 个可剪枝维度（由 $w[k]$ 表示）的参数 w，本研究引入了一个简单的正则化项用于稀疏训练，其定义如下：

$$R(g,k) = \sum_{K=1}^{K} \gamma k \bullet I_{g,k} = \sum_{K=1}^{K} \sum_{w \in g} \gamma_k \| w[k] \|_2^2 \quad (10)$$

其中，$I_{g,k} = \sum_{K=1}^{K} \sum_{w \in g} \gamma_k \| w[k] \|_2^2$ 表示第 k 个可剪枝维度的重要性，而 γ_k 表示对这些参数施加的收缩强度（即正则化系数）。本研究采用一种可控的指数策略来确定每个 γ_k，其公式如下：

$$\gamma_k = 2^{\alpha (I_g^{\max} - I_{g,k}) / (I_g^{\max} - I_g^{\min})} \quad (11)$$

在该方法中，本研究使用一个归一化得分来控制每个维度的收缩强度 α_k，其取值范围在 2^0 到 2^α 之间。在稀疏训练完成后，进一步使用一个简单的相对评分来识别并移除不重要的参数：

$$\hat{I}_{g,k} = N \bullet \frac{I_{g,k}}{TopN(I_g)} \quad (12)$$

其中，$TopN(I_g)$ 表示参数组 g 中前 N 个最重要维度的总得分。

2.3 基于低位量化的视觉大模型量化

视觉大模型在训练和推理过程中所使用的参数通常为 32 位浮点数，但浮点运算的计算成本远高于整数运算，对其进行量化可以显著压缩剩余权重参数的体积，并降低计算开销。

在神经网络中实现量化的过程，要求将卷积、矩阵乘法、激活函数、池化、拼接等操作全部转换为等效的整数运算（通常选择 8 位）。具体而言，在每次运算前后需分别添加量化与反量化操作：即在操作之前将输入从浮点数转换为 8 位整数（量化），在操作完成后将输出从 8 位整数转换回浮点数（反量化）。通过这一过程，模型在保持功能不变的前提下，能显著提升运算效率并减少内存消耗。

int8 量化[9] 的具体方法是将输入的 float32 类型数据转换为低位的数据类型，并传输至卷积层进行计算。同时，卷积层中的权重参数也被转换为低位的数据类型，从而实现整体的低位宽运算。

在量化操作中，需要将一组浮点数输入映射为低位的数据（0~255）范围的整数值。其基本流程如下：

首先，统计该组输入数据的最小值和最大值，然后使用公式 13 和公式 14 对每个输入数据进行量化与反量化处理。

$$q = round(\frac{x - x_{min}}{x_{max} - x_{min}} \times 255) \quad (13)$$

$$x \approx \frac{q}{255} \times (x_{max} - x_{min}) + x_{min} \quad (14)$$

其中，x 表示原始浮点数输入，q 为量化后的整数值，x_{min} 和 x_{man} 分别为该输入集合的最小值和最大值。通过这一映射，float32 数据得以压缩为低精度整数表示，显著提升了后续计算过程中的执行效率与内存利用率。

将 float32 数据类型转换为定点数类型（如 fp16、int8）的具体量化过程如下所示。在量化过程中，首先通过计算量化间隔 s 来确定每一个整数值所代表的浮点数范围，该公式为：

$$s = \frac{r_{fmax} - r_{fmin}}{2^n - 1} \quad (15)$$

其中，r_{fmax} 和 r_{fmin} 分别表示待量化浮点数据的最大值和最小值，n 是量化所使用的比特位数，s 即为量化步长。

随后，计算浮点数最小值对应的零点偏移量（即 zero-point），用于将浮点数 0 映射到整数量化空间中的适当位置。该计算公式为：

$$\varphi_x = [\frac{x_{fmin}}{s}] \quad (16)$$

完成以上准备工作后，使用以下主公式对浮点数输入 x_f 进行量化：

将 float32 数据类型量化为定点数据类型的公式如下所示：

$$x_q = [\frac{x_f}{s}] - \varphi_x \quad (17)$$

其中，[·] 表示带符号的四舍五入函数，定义如下：

$$[x] = sgn(x) \cdot [|x| + 0.5] \quad (18)$$

该函数的含义是：先对数值取绝对值加上 0.5 后再向下取整，并恢复原符号，实现了数学意义上的四舍五入。

通过上述步骤，可以将浮点数准确映射到定点数表示，从而在不显著损失模型精度的前提下，极大地降低模型的存储空间和推理计算量，为模型在边缘设备和移动终端中的部署提供了可行性基础。

卷积计算是专为卷积神经网络设计的核心操作。在对卷积神经网络进行量化之后，其卷积运算的计算原理将由浮点数运算转化为整数量化运算，其过程如下所示。

在未量化前，卷积操作可表示为：

$$a_f = \sum (x_f * \omega_f) \quad (19)$$

其中，x_f 表示输入特征图的浮点值，ω_f 表示卷积核的浮点权重，a_f 为浮点输出。

在量化之后，输入 x_f、权重 ω_f 被分别量化为整数形式 x_q 和 ω_q，并引入量化参数（缩放因子）S_x、S_ω 以及零点偏移 φ_x 、φ_ω。此时卷积计算公式转化为：

$$a_f = s_x s_\omega \sum (x_q + \varphi_x)(\omega_q + \varphi_\omega) \quad (20)$$

在输出端，量化输出 a_q 与其对应的浮点值 a_f 之间满足如下关系：

$$a_f = s_a (a_q + \varphi_a) \quad (21)$$

将上述公式联立，可推导出量化输出的表达式为：

$$a_q = [\frac{s_x s_\omega}{s_a} \sum (x_q + \phi_x)(\omega_q + \phi_\omega)] - \phi_a \quad (22)$$

通过这种方式，卷积操作可以在整数量化空间中高效执行，仅在输入和输出时进行浮点与整数之间的转换。这一过程显著降低了计算资源消耗，同时保持模型的表达能力。

2.4 视觉大模型剪枝量化联合方法实验

本研究所采用的轻量化流程包括以下六个步骤：首先，选取数据集作为任务的标准数据源，构建并初始化网络模型结构，为后续压缩实验提供统一的基准环境。在原始模型的训练阶段，进一步引入稀疏性正则化机制，对卷积层通道施加稀疏约束，引导网络在训练过程中自然抑制冗余特征通道，从而为后续剪枝操作提供可靠依据。

完成稀疏训练并收敛后，依据 BN 层中缩放因子 γ 值的绝对大小以及卷积层和全连接层的权重分布，对网络中权重贡献较小的通道和神经元进行结构化剪枝，形成稀疏结构，显著降低参数数量与计算开销。剪枝虽然可以有效减小模型体积，但不可避免地对模型表达能力造成一定损失。因此，在完成参数压缩后，本文对轻量化模型进行了少量 epoch 的微调训练，以适应新的网络结构与数据分布，修复因剪枝与量化所引起的精度波动。在剪枝完成后，为进一步压缩模型规模并提升推理效率，对保留参数进行量化处理。

最终，将优化后的模型部署在目标平台上，并通过验证集对其检测精度、推理速度、模型大小等关键性能指标进行综合评估，从而验证所提方法在实际应用中的可行性与有效性。该方法的整体流程如图 4 与图 5 所示，完整展现了从稀疏训练、结构剪枝到参数量化的全链路轻量化过程。

2.5 视觉大模型剪枝量化联合方法实验结果与分析

本研究的实验采用图形加速计算平台，GPU 型号为

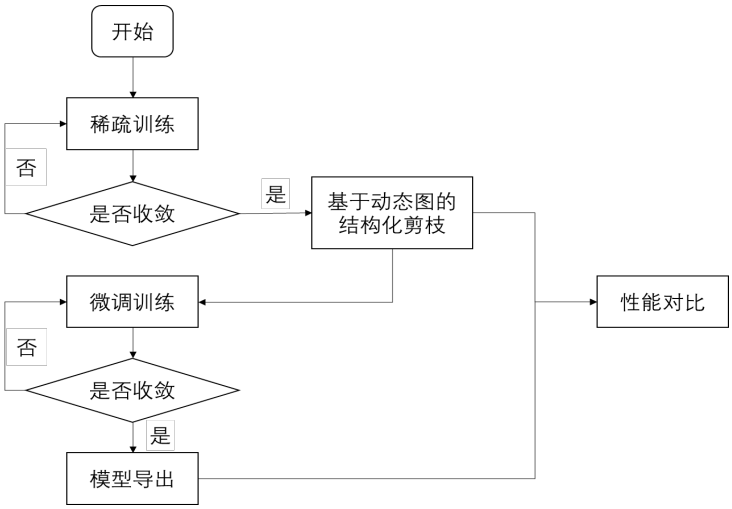


图 4 视觉大模型结构化剪枝流程

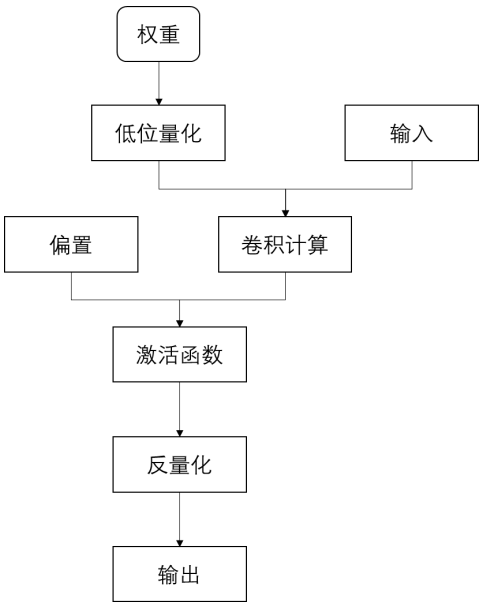


图 5 视觉大模型低位量化流程

表 1 视觉大模型yolov8基于本文提出的剪枝方法在剪枝率为90%的剪枝实验结果

	计算量（G）	参数量（M）	准确率（mAP）
剪枝前	14.27743	11.13792	0.69987
第一轮剪枝后	11.701916	8.935245	0.04456
第一轮微调后	11.701916	8.935245	0.699082
第二轮剪枝后	8.821654	6.285332	0.00
第二轮微调后	8.821654	6.285332	0.698123
第三轮剪枝后	6.327300	4.338866	0.00
第三轮微调后	6.327300	4.338866	0.692150
第四轮剪枝后	3.972012	2.952178	0.00
第四轮微调后	3.972012	2.952178	0.693130
第五轮剪枝后	1.428131	1.032790	0.00
第五轮微调后	1.428131	1.032790	0.662101

表 2 视觉大模型yolov8基于本文提出的剪枝方法在剪枝率为83%的剪枝实验结果

	计算量（G）	参数量（M）	准确率（mAP）
剪枝前	14.27743	11.13792	0.69987
第一轮剪枝后	8.701916	6.39559	0.03360
第一轮微调后	8.701916	6.39559	0.6921514
第二轮剪枝后	5.641652	3.83549	0.00
第二轮微调后	5.641652	3.83549	0.698326
第三轮剪枝后	3.925361	2.430846	0.00
第三轮微调后	3.925361	2.430846	0.6921514
第四轮剪枝后	2.979854	1.656523	0.00
第四轮微调后	2.979854	1.656523	0.713639
第五轮剪枝后	2.434190	1.21149	0.00
第五轮微调后	2.434190	1.21149	0.7169010

NVIDIA GeForce RTX4090，所使用的数据集为 NEU 带钢表面缺陷目标检测标准数据集。

为评估剪枝对模型计算复杂度、参数规模及检测性能的影响，本文以视觉检测模型 YOLOv8[10] 为基础，分别在

表 3 视觉大模型yolov8基于本文提出的剪枝方法在剪枝率50%情况下的剪枝实验结果

	计算量（G）	参数量（M）	准确率（mAP）
剪枝前	14.27743	11.13792	0.69987
第一轮剪枝后	12.90744	9.354339	0.056690
第一轮微调后	12.90744	9.354339	0.719054
第二轮剪枝后	10.37608	7.945423	0.00
第二轮微调后	10.37608	7.945423	0.710987
第三轮剪枝后	9.025612	7.120674	0.00
第三轮微调后	9.025612	7.120674	0.701802
第四轮剪枝后	8.219365	6.323421	0.00
第四轮微调后	8.219365	6.323421	0.696721
第五轮剪枝后	7.451352	5.680920	0.00
第五轮微调后	7.451352	5.680920	0.686113

表 4 视觉大模型yolov8基于本文提出的剪枝方法在剪枝率50%情况下的量化实验结果

量化策略	阶段	准确率（mAP）	推理时间（ms）
Fp16量化	量化前	0.7169010	5.6
	量化后	0.7017843	2.1
int8量化	量化前	0.7169010	5.6
	量化后	0.6073468	1.1
Fp16、int8混合量化	量化前	0.7169010	5.6
	量化后	0.6577890	1.4

剪枝率为 90%、83% 和 50% 的情况下进行了多轮剪枝与微调实验，详细结果见表 1 至表 3。

在剪枝率为 90% 的实验中，YOLOv8 模型的参数量由原始的 11.14M 逐步减少至 1.03M，计算量由 14.28G 减少至 1.43G，模型压缩显著。虽然每轮剪枝后模型准确率（mAP）都会显著下降，但经过微调后逐步恢复，最终准确率为 0.6621，仅略低于剪枝前的 0.69987，显示该方法具备在极端压缩场景下的可用性。然而，由于压缩比例过高，模型性能仍出现一定幅度下降，说明当剪枝率过高时，模型结构会遭到较大破坏。

在剪枝率为 83% 的实验中，模型参数压缩至 1.21M，计算量降至 2.43G，第五轮微调后 mAP 达 0.7169。整体来看，在保持高压缩率的同时实现了较高的检测精度，说明该剪枝率下模型结构更加紧凑，冗余部分被有效去除，具备“以剪促精”的潜力。尤其在第四轮和第五轮微调后，准确率反而优于剪枝前，展现了该策略在移动端或资源受限场景中的高实用性。

剪枝率为 50% 的实验中，在参数量减少一半至 5.68M、计算量降至 7.45G 的情况下，模型在第五轮微调后仍保持 0.6861 的 mAP，仅略低于原始模型，剪枝后首次微调甚至达到 0.7191 的最优表现，显示中等剪枝率可在压缩模型的同时提升或保持其性能。

从三个剪枝实验可以看出，“剪枝 – 微调交替”策略优于单次大幅剪枝，每轮剪枝后都通过微调恢复性能，有效避免了性能骤降的风险。同时，各轮剪枝的影响存在阶段性规律：前几轮剪枝对性能影响较小，而后三轮尤其是高剪枝率条件下，对准确率打击明显，说明模型存在一定的冗余容忍阈值。合理的多轮渐进式剪枝能够更稳定地压缩模型，并保持良好性能。

总体而言，本文剪枝策略在不同压缩目标下均取得了良好结果，特别是在剪枝率为 83% 时展现出出色的精度保持能力和压缩效果，为 YOLOv8 等大模型在资源受限环境中的高效部署提供了可行路径，也为后续轻量化研究提供了重要参考价值。

为进一步提升模型的部署效率，本文选择剪枝表现最好的模型 (83% 剪枝率的模型) 作为基准模型，在其基础上进行了后训练量化实验，分别测试了三种主流量化方法：FP16 量化、INT8 量化及 FP16 与 INT8 混合量化，其具体实验结果如表 4 所示。

因为量化操作并未改变模型的结构，因此计算量与参数量保持不变，分别为 2.43G 和 1.21M。在不改变计算量和参数量的前提下，FP16 量化将模型推理时间由 5.6ms 降

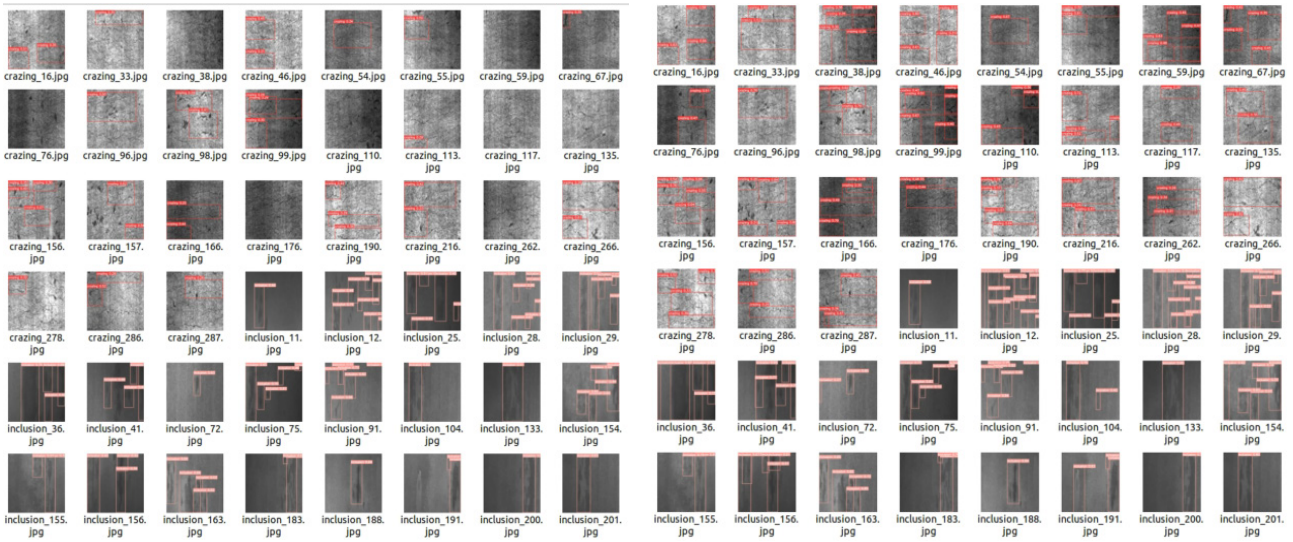


图 6 最优的轻量化实验前（右）后（左）推理检测结果对比

低至 2.1ms，准确率由 0.7169 下降至 0.7018，精度损失较小，表现出较好的速度 - 精度平衡能力。INT8 量化在加速方面更为显著，推理时间进一步压缩至 1.1ms，但准确率降至 0.6073，出现较大幅度的精度损失，说明该策略对模型表示能力影响较大，适合对速度极端敏感、对精度容忍度较高的应用场景。

FP16 与 INT8 的混合量化策略在推理时间与精度之间取得了一定的折中效果，推理时间压缩至 1.4ms，同时准确率保持在 0.6578，从图 6 的推理结果对比来看，其损失是可接受的。综合考虑其在保持精度的同时大幅提高了推理速度，为相对最优的策略。

3 结语

综上所述，本文针对视觉大模型的高计算复杂度与部署成本问题，提出了一种结合结构化剪枝与混合精度量化的轻量化方法，并在 YOLO v8 网络上进行了实验验证。通过对 BN 层通道进行重要性分析实现结构化剪枝，配合低位宽参数量化策略，有效压缩了模型体积，降低了计算负担，并在保持精度基本稳定的前提下显著提升了推理速度。实验结果表明，该方法在提升模型部署效率方面具有良好的应用前景，尤其适用于边缘计算与移动端场景。

然而，本研究仍存在一定的探索空间与优化潜力。未来的研究工作可以从两个方向展开：一方面，应进一步尝试不同网络层之间的混合精度量化策略，根据各层对精度的敏感性选择更合理的位宽组合，以在精度与效率之间寻求更优权衡；另一方面，剪枝与量化的融合机制仍需进一步研究，尤其是在设计剪枝与量化协同优化策略、统一训练流程及提升模型鲁棒性方面。通过更深层次的联合压缩算法设计，有望实现更具泛化能力与部署效率的视觉模型，为轻量化深度学习在实际场景中的广泛应用提供有力支撑。

参考文献：

[1] BJORCK N, GOMES C P, SELMAN B, et al. Understanding batch normalization [J]. 2018, 31.

[2] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need [J]. 2017, 30.

[3] KOONCE B. ResNet 50 [M]. Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization. Springer. 2021: 63-72.

[4] 王琳，宋权润，耿世超等. 神经网络滤波器剪枝技术研究综述[J]. 2026, 62(2).

[5] 杨春，张睿尧，黄泷等. 深度神经网络模型量化方法综述[J]. 2023, 45(10): 1613-29.

[6] 卢海伟. 模式识别与人工智能袁 J. 基于层融合特征系数的动态网络结构化剪枝[J]. 2019, 32(11): 1051-9.

[7] HOEFLER T, ALISTARH D, BEN-NUN T, et al. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks [J]. 2021, 22(241): 1-124.

[8] BASHA S S, DUBEY S R, PULABAIGARI V, et al. Impact of fully connected layers on performance of convolutional neural networks for image classification [J]. 2020, 378: 112-9.

[9] 钱源. 基于 KL 散度的 int8 量化方法研究与框架实

现[D]. 中国科学院大学 (中国科学院人工智能学院)，2020.

[10] SOHAN M, SAI RAM T, RAMI REDDY C V. A review on yolov8 and its advancements; proceedings of the International conference on data intelligence and cognitive informatics, F, 2024 [C]. Springer.

作者简介：左霖泽（1998-），男，汉族，河北省保定市人，硕士研究生，助理工程师，研究方向：计算机视觉，模型轻量化与边缘部署，多智能体。