计算机科学
COMPUTER SCIENCE

LegoViT:边缘端视觉推理中ViT模型块粒度缩放技术

周豪捷, 吴晓宁, 高志强, 韩锐, 张青龙, 刘驰, 陈铮, 赵玉, 王硕

引用本文

周豪捷, 吴晓宁, 高志强, 韩锐, 张青龙, 刘驰, 陈铮, 赵玉, 王硕. LegoViT:边缘端视觉推理中ViT模型块粒度缩放技术[J]. 计算机科学, 2026, 53(4): 269-276.

ZHOU Haojie, WU Xiaoning, GAO Zhiqiang, HAN Rui, ZHANG Qinglong, LIU Chi, CHEN Zheng, ZHAO Yu, WANG Shuo. LegoViT:Block-grained Scaling Techniques for ViT Models in Edge-side Visual Inference [J]. Computer Science, 2026, 53(4): 269-276.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.250900024

LegoViT:边缘端视觉推理中 ViT 模型块粒度缩放技术

周豪捷[1] 吴晓宁[2] 高志强[3] 韩 锐[1] 张青龙[1] 刘 驰[1] 陈 铮[2] 赵 玉[2] 王 硕[2]
1 北京理工大学计算机学院 北京 100081
2 中国民航信息网络股份有限公司 北京 101318
3 中国人民武装警察部队工程大学 西安 710018
(3280165225@qq.com)

摘 要 近年来,ViT 模型凭借其强大的图像理解能力被广泛部署于边缘侧视觉应用。在资源受限边缘端推理中,ViT 模型需依据可用资源对其进行有效缩放来获取最优的推理精度-延迟平衡。然而,现有推理模型缩放技术往往仅能在整个模型粒度进行缩放,导致关键信息丢失,需消耗更多计算资源/推理延迟来获取同样的精度。对此,提出 LegoViT 方法,旨在从 ViT 模型前馈网络中识别出可缩放模型块,以支持运行时块粒度模型缩放。对比模型粒度缩放方法的测试结果表明,LegoViT 使 ViT 模型内存占用降低 22.37%,计算量减少 21.1%,推理延迟平均缩短 61.05%。

关键词:边缘侧;ViT;推理优化;块粒度缩放
中图分类号 TP391

LegoViT:Block-grained Scaling Techniques for ViT Models in Edge-side Visual Inference

ZHOU Haojie[1],WU Xiaoning[2],GAO Zhiqiang[3],HAN Rui[1],ZHANG Qinglong[1],LIU Chi[1],CHEN Zheng[2],ZHAO Yu[2] and WANG Shuo[2]
1 School of Computer Science and Technology,Beijing Institute of Technology,Beijing 100081,China
2 TravelSky Technology Limited,Beijing 101318,China
3 Engineering University of PAP,Xi'an 710018,China

Abstract In recent years,Visual Transformer (ViT) models have been widely deployed in edge-based visual applications because of their powerful image understanding capabilities. To achieve optimal inference accuracy-latency balance in resource-constrained edge-side inference,it is essential to scale ViT models effectively based on available resources. However,existing inference model scaling techniques can only perform scaling at the entire model granularity,leading to the loss of critical information and often requiring more computational resources or higher inference latency to achieve equivalent accuracy. This paper proposes LegoViT, a method that identifies scalable model blocks from the feedforward networks of ViT mo-dels,thus supporting runtime block-level model scaling. Comparative test results demonstrate that LegoViT achieves a 22.37% reduction in memory footprint of ViT models,a 21.1% decrease in computational overhead,and an average 61.05% reduction in inference latency.

Keywords Edge side,ViT,Inference optimization,Block-grained scaling

1 引言

近年来,在诸多计算机视觉领域应用中,以 Vision Transformer(ViT)[1]为代表的模型显著提升了图像分类等任务的性能,例如 Swin[2]在 ImageNet-1K 上达到 85.4%的 top-1 准确率。同时,随着边缘计算需求的增长,ViT 类大模型在边缘设备的应用成为研究热点。然而,边缘侧部署 ViT 大模型面临着硬件资源的内存限制与计算资源动态变化等挑战。因此,如何在资源受限条件下实现模型推理效率与精度的平衡,通过缩放模型结构以在资源动态变化时最大化模型精度,成为提升边缘 ViT 模型推理性能的关键问题[3-6]。

如图 1 所示,ViT 的网络架构以分块嵌入层和堆叠的编码器层为核心,每个编码器层通常由多头自注意力(MSA)与前馈神经网络(FFN)交替组成。其中,FFN 内部执行的两次

到稿日期:2025-09-02 返修日期:2025-12-29
基金项目:国家自然科学基金(62272046,62132019,62472033,61872337);工业和信息化部高质量发展专项(CEIEC-20240);北方自动控制技术研究所合作课题和北京理工大学培养课题(2023CX01017)
This work was supported by the National Natural Science Foundation of China(62272046,62132019,62472033,61872337),Special Program for High-Quality Development of the Ministry of Industry and Information Technology(CEIEC-20240) and Cooperative Project with the Northern Institute of Automatic Control Technology and Cultivation Project of Beijing Institute of Technology(2023CX01017).
通信作者:韩锐(hanrui@bit.edu.cn)

高维特征空间映射导致其计算量与内存占用随模型深度呈线性增长，使得 FFN 的参数量与计算量通常为 MSA 的 3～4 倍。在边缘端推理中，这种特性导致 FFN 成为 ViT 推理过程的关键性能瓶颈，也为边缘侧部署 ViT 大模型带来以下两个挑战。

1）ViT 模型的细粒度缩放。现有端侧推理中的模型缩放方法大多针对 CNN 模型架构设计[3-6]，尚未深入研究 ViT 不同模块对推理精度和延迟的影响，因此仅能对整个模型进行粗粒度缩放[4,7-8]，以此来权衡精度与延迟。

2）多作业运行下的总体调度优化。边缘设备常需同时处理多个作业，然而现有针对 ViT 模型推理的调度工作往往依赖模型粒度的资源缩放，无法更细粒度地对多个作业的推理精度-延迟进行权衡优化[9-11]，导致资源利用率较低。

针对以上挑战，本文提出了 LegoViT——一种面向 ViT 模型的块粒度推理缩放方法，其核心思想是通过创建训练环境来表征 ViT 块的输入与输出通道特征，使得块的派生版本能够独立于其他部分进行重新训练。在运行时，通过动态感知资源变化求解优化目标，仅通过切换 ViT 模型中部分派生块即可实现动态缩放，避免了模型尺寸变更后的重训练过程。

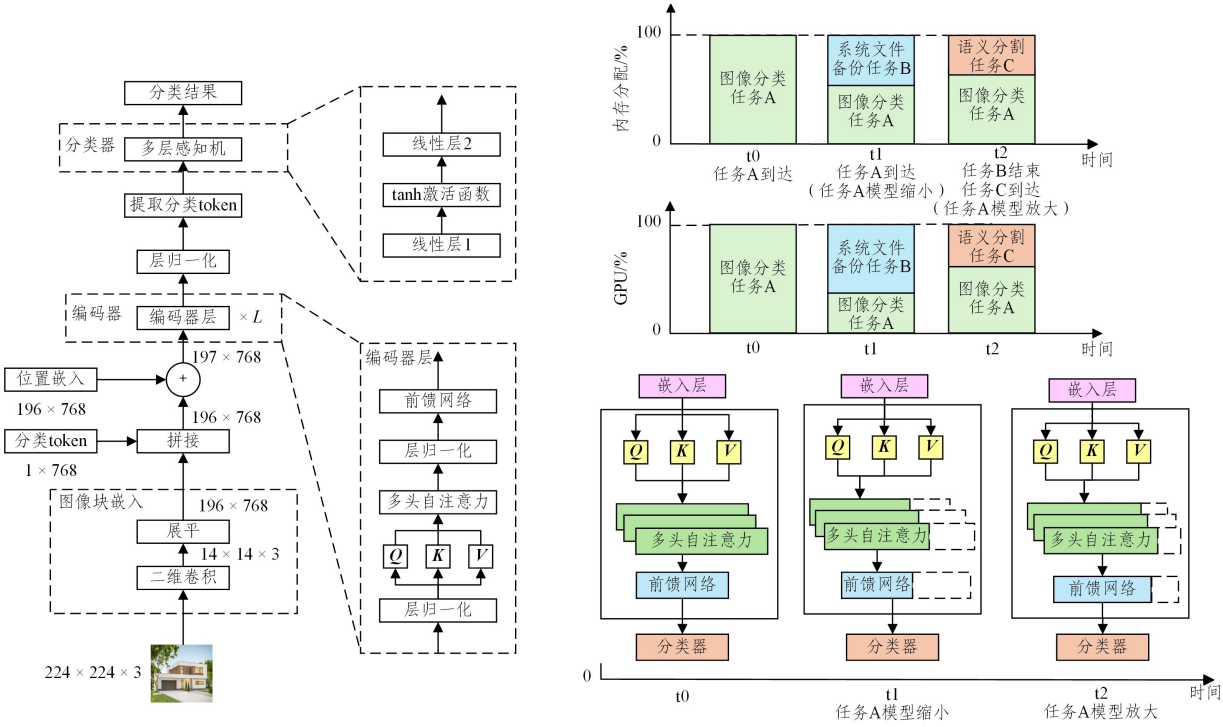

图 1 ViT 模型边缘端推理场景示意图

Fig. 1 Schematic diagram of ViT model inference in edge computing scenarios

本文基于 PyTorch 实现了 LegoViT 方法。实验使用 8 个不同规模的 ViT 模型进行 Cifar-10 图像分类任务，并在边缘计算场景中对比了 5 种现有 ViT 推理优化方法。结果显示，LegoViT 优化方法的模型内存占用平均减少 22.37%；计算量平均减少 21.1%；在模型精度不产生任何损失的前提下实现了平均 61.05%的推理延迟降低。

2 背景与相关工作

2.1 ViT 模型 FFN 模块分析

在 ViT 模型的边缘侧部署中，FFN 的计算效率问题尤为突出。以 ViT-Base 模型为例，当处理 224×224 的图像输入时，单个 FFN 层的浮点运算量达到 1.08×10^{12}，是 MSA 模块的 2.1 倍，其参数占比更是达到编码器层的 73.8%。

这种计算特性源于 FFN 的固有架构设计，其“先扩维再压缩”的设计虽然能提升模型的表达能力，但同时也会带来巨大的矩阵运算量。相比之下，MSA 虽然需要计算各个图像块之间的关系，但在常规图片尺寸下，这些关系的计算量反而比 FFN 更节省资源。因此，本文设计了针对 FFN 的块粒度缩放机制和动态优化策略，在保证高精度与低延迟的前提下，减少 FFN 部分在推理过程的计算资源消耗。

2.2 现有工作

近年来，很多研究者致力于优化移动系统中应用的推理性能，优化的内容包括利用中间结果实现推理任务早期退出[12-14]、推理结果的边缘缓存[15-17]和模型分割技术[18-20]等。

2.2.1 针对 CNN 的推理缩放技术

计算机视觉领域使用的传统 CNN 模型依赖堆叠层中的卷积核来提取局部特征，而 ViT 通过自注意力机制捕捉全局依赖。现有端侧推理中的模型缩放方法大多只针对 CNN 模型架构设计[3-6]，通过滤波器剪枝或移除卷积层通道来进行 CNN 模型的缩放，尚未对 ViT 不同模块（如 MSA 和 FFN）的缩放展开深入研究。

2.2.2 模型粒度的缩放

通过结构化剪枝技术在延迟与模型精度间取得权衡，已成为移动和边缘端视觉系统[21-24]的研究热点。这类方法在模型粒度上通过结构化剪枝对整个模型进行缩放，将其压缩为若干不同尺寸的派生模型（见图 2 上半部分）。但是现有缩

放技术[4,7-10]因模型重新训练耗时较长，仅能生成少量衍生模型。在实际运行时，它们根据可用资源动态选择其中一个模型。然而，这些模型间的较大性能差异可能导致显著的精度损失或推理延迟上升。

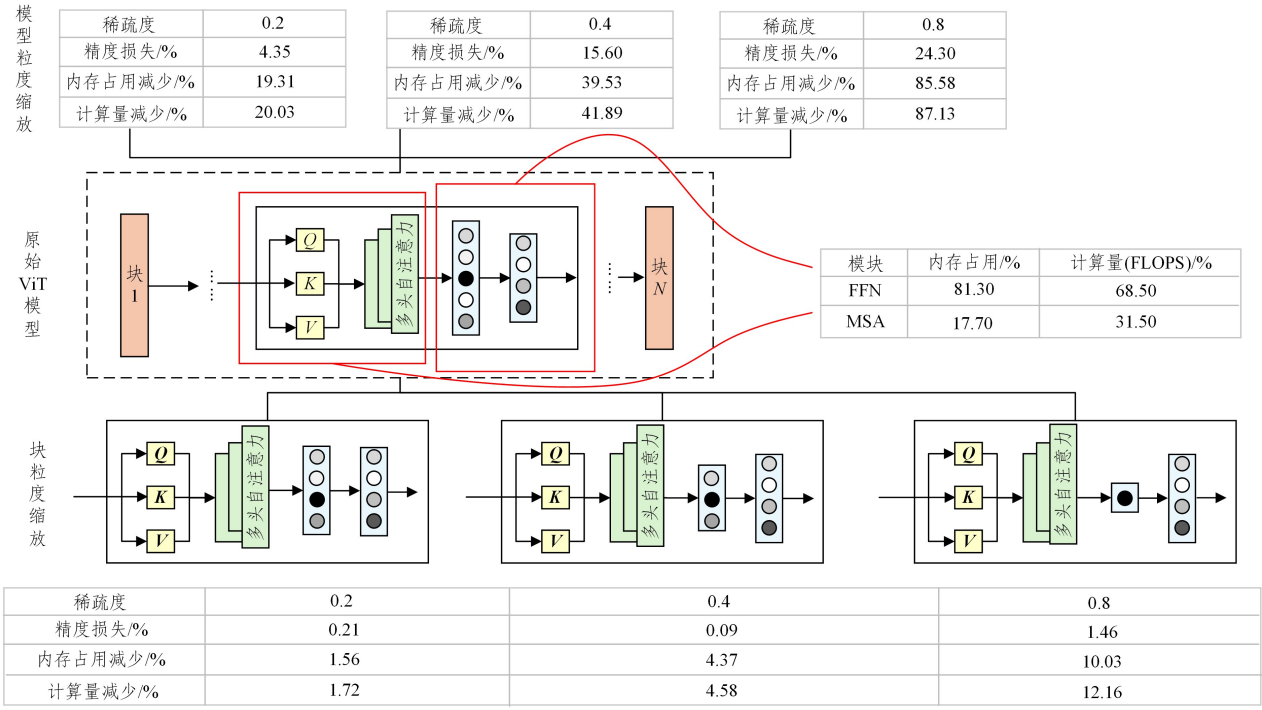

图 2 块粒度缩放与模型粒度缩放对比

Fig. 2 Comparation between block scaling and model scaling

2.2.3 模型剪枝推理优化

模型剪枝通过移除冗余参数降低计算复杂度，结合结构化设计可实现硬件友好的高效推理。近年来，针对 Transformer 架构的剪枝方法呈现出多样化的技术路线，分别从动态稀疏训练[25]、低秩近似表示[26]、基于正交变换的行列删除[27]和特征的波动性度量[28]等角度进行剪枝优化。然而，这些方法高度依赖于特定模型的校准数据，且需要额外引入大量的训练与优化开销，缺乏对广泛模型的通用适配。

3 LegoViT 设计

3.1 概述

LegoViT 是一种针对 ViT 模型的块粒度推理缩放方法，旨在实现边缘侧 ViT 模型的高效部署与推理，如图 3 所示。

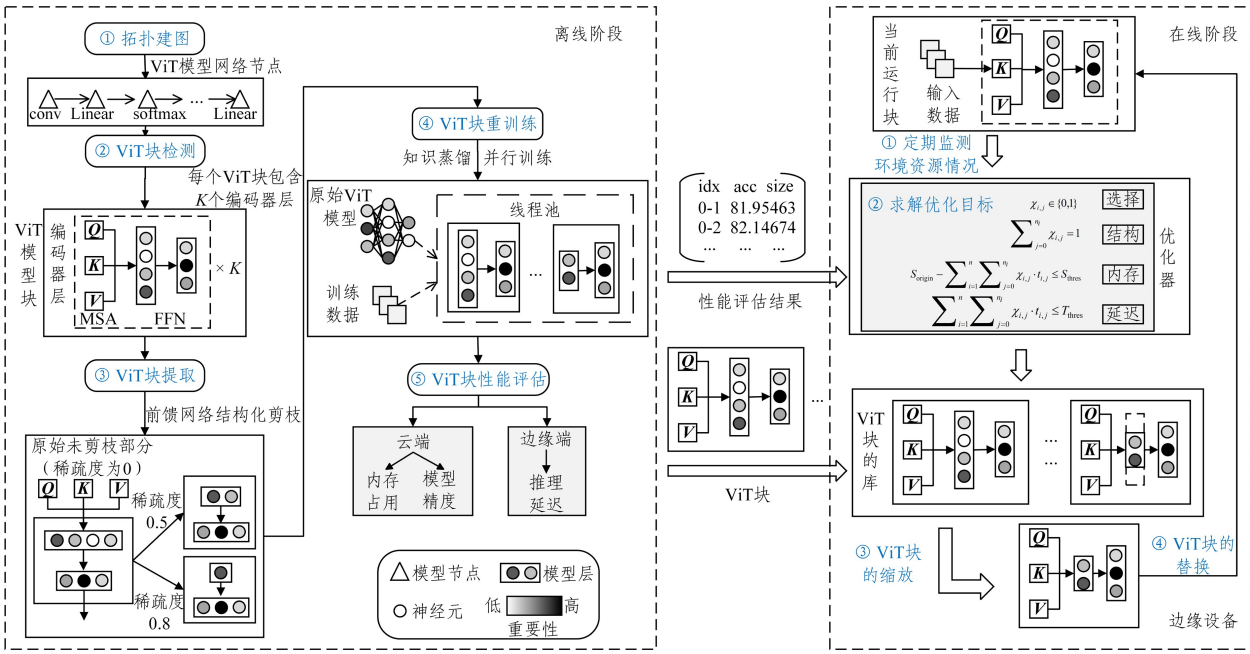

图 3 LegoViT 整体架构

Fig. 3 Overall architecture of LegoViT

LegoViT 分为离线与在线两个阶段，分别解决了两个关键的研究问题：1）如何快速地检测、提取和重训练多个 ViT 模型块；2）推理过程中如何根据实时的资源变化情况，寻找最优的 ViT 块组合。

3.2 离线阶段

离线阶段通过对 ViT 块的检测、提取与重训练，生成了一系列不同稀疏度的派生块，极大扩展了推理过程中 ViT 块组合的选择空间（见图 3 的左侧部分）。

3.2.1 拓扑建图

拓扑建图环节使用 PyTorch 中的 JIT 追踪模型的执行路径，将动态计算的模型转换为静态的计算图，然后通过广度优先搜索算法合并计算图中相邻的原子操作，捕捉各节点的输入/输出形状等信息。

3.2.2 块检测

块检测环节以上一环节的拓扑排序节点为输入，对 ViT 模型进行块粒度检测与划分。检测的具体计算式如下：

$$K=\max(2,floor(N_{\text{Encoder}}\times\rho)) \tag{1}$$

其中，N_{Encoder} 表示 ViT 模型中编码器层的总数量；ρ 的取值为 $[0,1]$，表示每个分割块中编码器层数量占总数的比例，仅反映 ViT 块的相对大小，模型最终精度-延迟表现对其并不敏感；$floor$ 表示向下取整；计算结果 K，即为每个 ViT 块中应该包含的编码器层数量。

3.2.3 块提取

块提取环节以划分出的多个 ViT 块为输入，通过结构化剪枝生成一系列不同稀疏度的派生块。此处采用 Torch-Pruning 作为 FFN 的结构化剪枝工具，以 0.2 为间隔在 $[0,1)$ 范围内设置了 5 个稀疏度。

3.2.4 块重训练

如图 4 所示，块重训练环节采用知识蒸馏方法，将 ViT 预训练模型作为教师模型，将同源的派生块作为学生模型，结合多线程技术并行化地执行重训练，极大地缩短了块重训练所需时间。

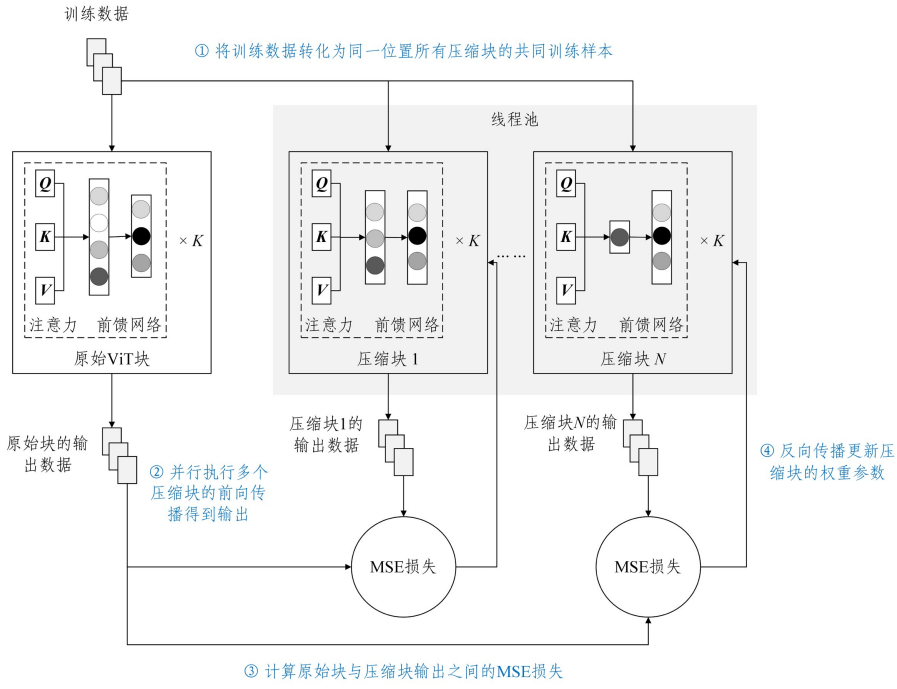

图 4 LegoViT 的块重训练环节

Fig. 4 Block re-training of LegoViT

3.2.5 块性能分析

块性能分析环节分析与记录所有压缩块的各方面性能指标，重点包括精度分析和延迟分析两个方面。

1）精度分析。对于原始块 b_i 的派生块 $b_{i,j}$，考虑 k 种不同的块组合，总共产生 k 个不同大小的新 ViT 模型，在 k 次测试中记录其相较于原始模型的下降值，最后取平均值 $A_{i,j}$ 作为派生块 $b_{i,j}$ 的精度下降记录值。

2）延迟分析。针对推理延迟的分析基于一个基本理论：模型块的推理延迟与包含的参数量成正比。对于原始块 b_i 和其派生块，假设参数量分别为 s_i 和 $b_{i,j}$，那么派生块 $b_{i,j}$ 的延迟下降百分比 $\phi_{i,j}=1-\frac{s_{i,j}}{s_i}$。只需要在实际运行时获取原始块 b_i 的其中一个派生块 $b_{i,k}$ 的推理延迟，即可推算出剩余派生块的延迟。但考虑到真实硬件中实际延迟还受到缓存、内存带宽等因素影响，该假设存在一定的局限性。

3.3 在线阶段

在实际运行中，选择将最小化整体精度损失作为优化目标，而非最大化内存节省或最小化推理延迟。这是因为模型精度是 ViT 模型在边缘视觉任务中的核心指标，若将内存或延迟作为优化目标，则会因低内存占用与低延迟的 ViT 块通常伴随着较大的精度损失而使模型整体的精度显著下降。

假设总共有 n 个未压缩的原始 ViT 模型块，原始块 b_i 共

有 n_i 个派生块。对于其中的派生块 $b_{i,j}$，它的内存占用下降值为 $s_{i,j}$，精度损失为 $A_{i,j}$，推理延迟为 $t_{i,j}$。另外，派生块 $b_{i,j}$ 拥有一个决策变量 $X_{i,j}$，表示该派生块是否被选中用以替换原始块。

每一次组合过程中，具体的优化目标如下：

$$\min \sum_{i=1}^{n}\sum_{j=0}^{n_i} A_{i,j} \cdot X_{i,j} \tag{2}$$

$$\text{s.t. } C1: X_{i,j} \in \{0,1\} \tag{3}$$

$$C2: \sum_{j=0}^{n_i} X_{i,j} = 1 \tag{4}$$

$$C3: S_{\text{origin}} - \sum_{i=1}^{n}\sum_{j=0}^{n_i} X_{i,j} \cdot s_{i,j} \leqslant S_{\text{thres}} \tag{5}$$

$$C4: \sum_{i=1}^{n}\sum_{j=0}^{n_i} X_{i,j} \cdot t_{i,j} \leqslant T_{\text{thres}} \tag{6}$$

约束条件 C1 为选择约束，即派生块 $b_{i,j}$ 的决策变量 $X_{i,j}$ 只能取 0 或 1。

约束条件 C2 为结构约束，即对于未压缩的原始块 b_i 的 n_i 个派生块中，有且仅有一个被选中。

约束条件 C3 为内存约束，其中 $\sum_{i=1}^{n}\sum_{j=0}^{n_i} X_{i,j} \cdot s_{i,j}$ 表示整体模型内存占用下降值的总和，S_{origin} 表示原始 ViT 模型的大小，S_{thres} 是设定的内存阈值。

约束条件 C4 为延迟约束，其中 $\sum_{i=1}^{n}\sum_{j=0}^{n_i} X_{i,j} \cdot t_{i,j}$ 表示所有派生块推理延迟的总和，T_{thres} 是设置的延迟阈值。

对于以上优化问题的求解，将其视为整数线性规划(ILP)问题。该问题的求解过程主要分为以下两个步骤。

1)去除整数约束条件 C1，即允许选择变量 $X_{i,j}$ 在[0,1]内连续取值，将 ILP 问题转换为 LP 问题 p^*，从而能够快速求出最优解 q^*。

2)使用分支定界法，首先从解 q^* 中选择一个非整数取值的选择变量 $X_{i,j}$，在左子节点中令 $X_{i,j}=0$，得到解 $q_{i,j}^{(0)}$；在右子节点令 $X_{i,j}=1$，得到解 $q_{i,j}^{(1)}$。对于每个子节点，设定上界(UB)为当前节点解求出的 LP 问题目标值，下界(LB)为当前可行整数解的 ILP 问题目标值。只有当解为整数解且 $UB<LB$ 时，才更新 LB，其他情况下则剪去分支。采用深度优先搜索策略，从根节点 q^* 出发，优先搜索 UB 最小的分支，最终得到 ILP 问题的最优解。

在上述分支定界法中，假设原始块总数为 n，每个原始块的派生块数量为 n_i，则总决策变量数为 $N=n\times n_i$。

该 ILP 问题的求解时间主要由 LP 松弛求解和分支定界法两步主导。LP 问题可在 $O(N^2)-O(N^3)$ 的多项式时间内求解，在 $N\leqslant 100$ 时，运行时间可控制在 10 ms 内；对于分支定界过程，考虑其平均情况下的时间复杂性。在最坏情况下，其时间复杂度为 $O(2^n)$，但通过内存和延迟约束限制可行解范围，可将其复杂度降至多项式量级(如 $O(N^2)$)。

在本文 LegoViT 方法中，模型被分为 4～6 个原始块，每个原始块拥有 5 个派生块(包括自身)，即 $N\leqslant 30$。同时，通过对其施加较为严格的内存与延迟约束，分支定界法可以在数十至数百毫秒内完成收敛，使其适合大多数边缘推理任务。在 N 值极大($N>100$)且约束松弛的情况下，才会导致秒级的 ILP 求解时间，影响整体系统的实时性。

4 实验评测

本文通过 PyTorch 实现优化方法，并深入分析了 LegoViT 推理卸载优化方法在模拟边缘计算环境中的实验设置、评测指标和实验结果。

4.1 实验设置

1)硬件设备。由于硬件条件限制，本实验未引入边缘设备，而是构建在一个由 3 台高性能服务器模拟的边缘环境之中。3 台设备采用相同的操作系统和 Linux 内核版本，设备之间基于 TCP 协议通信，内存与延迟等数据通过指定端口进行传输。具体配置如表 1 所列。

表 1 实验设备配置

Table 1 Experiment device configuration

处理器	操作系统	内存大小/GB	内核数
Intel(R) Xeon(R) Silver 4216 CPU	Ubuntu 18.04	251	64
Intel(R) Xeon(R) CPU E5-2680 v4	Ubuntu 18.04	251	56
Intel(R) Xeon(R) CPU E5-2660 v4	Ubuntu 18.04	503	56

2)实验超参数。离线阶段，设定块检测环节的 ρ 为 0.25(模型包含 12 个编码器层，即每 3 个划分出一个 ViT 块)；对每个 ViT 块依次使用 0/0.2/0.4/0.6/0.8 的剪枝比例；重训练环节设定训练轮数为 65，采用 Adam 优化器和 MSE 损失函数，学习率为 3×10^{-4}。在线阶段，设定 Tiny 规模模型的内存阈值为 15 MB，延迟阈值为 150 ms；Small 规模模型的内存阈值为 40 MB，延迟阈值为 450 ms。

3)实验模型及数据集。本文选择了 8 种不同规模的 ViT 模型作为实验对象，包括 ViT，DeiT，ViTMSN 等；采用 CIFAR-10 图像分类数据集进行初步验证(实验结果存在一定的局限性)。

4)对比方法。本文基于 PyTorch 在模拟边缘计算场景中分别对比了两类方法：模型压缩与卸载优化。其中模型压缩方法包括 SViTE，LoSparse，SliceGPT，FLAP；卸载优化方法包括 DDPG-based DEC-DNN-IO[30]。

4.2 实验测试与结果分析

在本实验中，首先使用 3 个 ViT 模型将 LegoViT 与模型压缩方法进行初步对比，图 5 展示了 LegoViT 与模型压缩方法的对比结果数据。在内存占用上，LegoViT 优于 SViTE 与 SliceGPT，其中相比 SViTE 减少了 9.3%的内存占用；在计算量上，LegoViT 与其他方法的表现相当；在推理延迟上，LegoViT 小幅度领先于 SViTE，SliceGPT 与 FLAP，分别提升了 1.53%，4.78%与 2.75%。总体上，LegoViT 与这些模型压缩基线方法在内存占用与计算量上差距较小，在推理延迟方面有一定的提升。

下面使用 8 个 ViT 模型将 LegoViT 与卸载优化方法进行 4 个指标的对比。

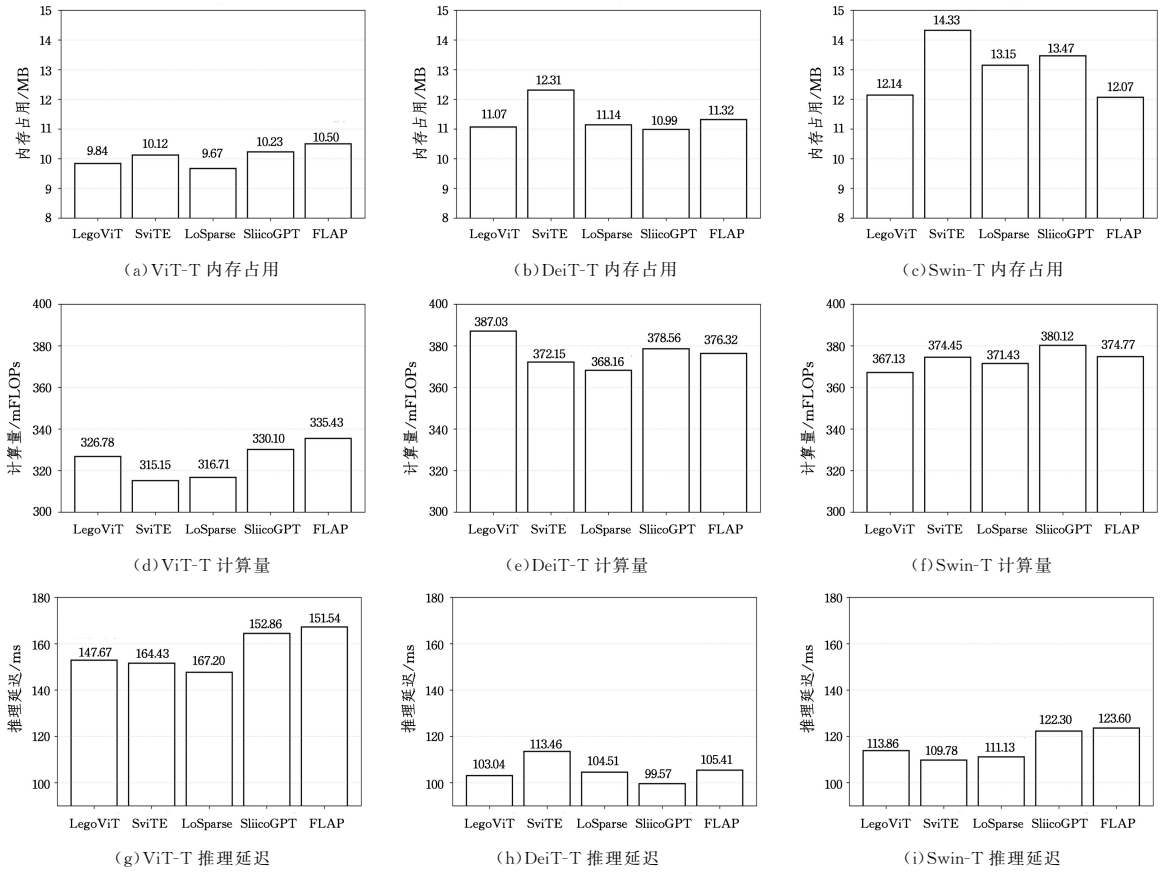

图 5 LegoViT 和模型压缩方法的对比结果

Fig. 5 Comparison results of LegoViT and model compression methods

4.2.1 资源占用分析

运行过程中，如图 6 所示，由于设备在不同时刻会有不同数量的作业到达，其分配给目标视觉任务的内存等资源会相应地减少或增加，LegoViT 可以在 ViT 模型推理的过程中监控实时的环境资源变化，智能选择最优的模型块进行缩放与切换，以剪枝后的轻量级模型块组合方式运行，从而减轻边缘端设备的负担，提高整体模型的推理效率。

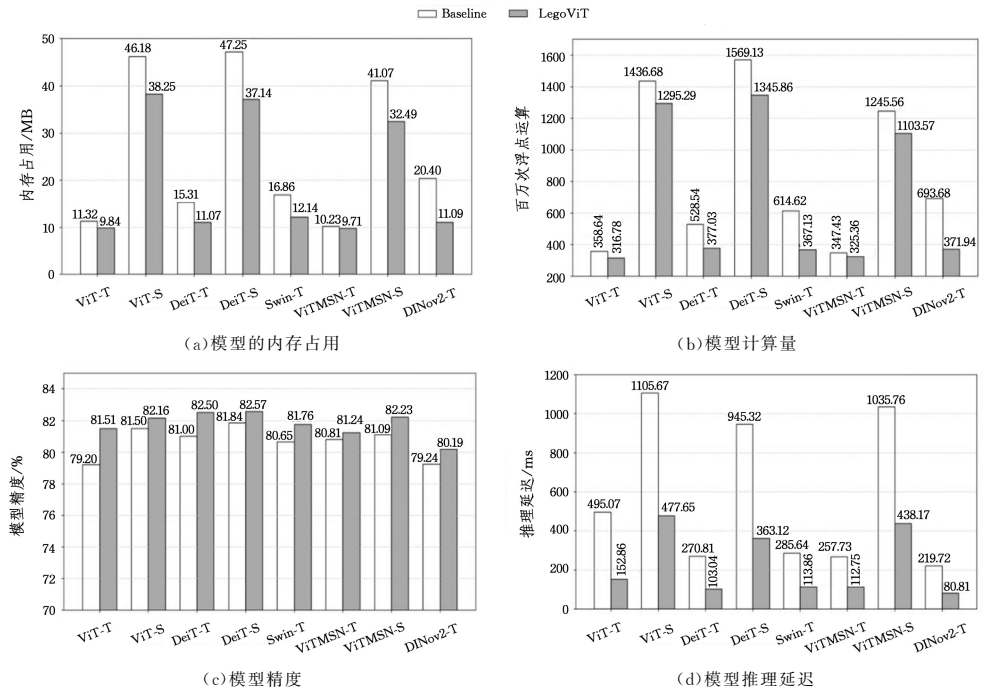

图 6 对比实验结果

Fig. 6 Results of comparative experiment

内存占用分析：模型的内存占用情况如图 6(a)所示，在 Tiny 规模的 ViT 模型上，内存占用平均降低了 23.9%，Small 规模的模型平均降低 19.82%，总体平均降低了 22.37%。

计算量分析：如图 6(b)所示，与层粒度的推理卸载方法相比，LegoViT 在所有模型上的计算量平均降低了 21.1%，Tiny 规模的模型平均降低 26.67%，Small 规模平均降低 11.82%。这是由于 LegoViT 在块粒度上对资源进行调度优化，在感知到资源变化时；根据优化目标的求解结果选择内存占用与计算量最小的 ViT 块进行替换，从而降低了边缘端设备的内存与计算开销。

4.2.2 精度与延迟分析

LegoViT 旨在实现同等模型精度表现下的推理性能提升，因此将精度无损提升(而非精度提升)作为优化目标。如图 6(c)与图 6(d)所示，在精度无损失的前提下，LegoViT 在所有 ViT 模型上实现了平均 61.05%的推理延迟降低，其中 Tiny 规模平均降低 62.46%，Small 规模平均降低 58.69%。

综上，LegoViT 在严格的延迟约束条件下将模型精度作为优化目标，使 ViT 模型内存占用降低 22.37%，计算量减少 21.1%，在精度无损条件下平均降低 61.05%推理延迟。

结束语 面对边缘端 ViT 模型推理过程的资源受限等问题和挑战，本文设计并实现了 LegoViT 推理优化方法，分别从 ViT 模型的块粒度的识别与多作业调度优化两个角度出发，设计了两阶段的推理优化框架。

未来的研究将重点关注以下方向。

1)ViT 模型自注意力与前馈网络混合剪枝。出于工程上的考量，本文仅对 ViT 模型中占据计算量主导地位的 FFN 结构进行剪枝，未来会加入关于 MSA 的通用性剪枝设计。

2)受硬件等条件限制，本研究目前仅在服务器上使用简单数据集进行初步验证，未来将引入边缘设备(如 Jetson)以及更多的真实数据集来验证 LegoViT 的有效性。

3)引入主流边缘设备与真实复杂数据集进行对比测试。受硬件等条件限制，本研究目前仅在高性能服务器上使用 CIFAR-10 数据集进行初步验证，未来将引入边缘设备(如 Jetson)以及更多的真实数据集来验证 LegoViT 的有效性。

参 考 文 献

[1] DOSOVITSKIY A,BEYER L,KOLESNIKOV A,et al. An image is worth 16×16 words:Transformers for image recognition at scale[J]. arXiv:2010.11929,2020.

[2] LIU Z,LIN Y,CAO Y,et al. Swin transformer:Hierarchical vision transformer using shifted windows[C]// Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021:10012-10022.

[3] FANG B,ZENG X,ZHANG M. Nestdnn:Resource-aware multi-tenant on-device deep learning for continuous mobile vision[C]// Proceedings of the 24th Annual International Conference on Mobile Computing and Networking. 2018:115-127.

[4] HAN R,ZHANG Q,LIU C H,et al. Legodnn:block-grained scaling of deep neural networks for mobile vision[C]// Proceedings of the 27th Annual International Conference on Mobile Computing and Networking. 2021:406-419.

[5] HAN S,MAO H,DALLY W J. Deep compression:Compressing deep neural networks with pruning,trained quantization and huffman coding[J]. arXiv:1510.00149,2015.

[6] LI H,HU C,JIANG J,et al. JALAD:Joint accuracy-and latency-aware deep structure decoupling for edge-cloud execution [C]// 2018 IEEE 24th International Conference on Parallel and Distributed Systems(ICPADS). IEEE,2018:671-678.

[7] KIM Y D,PARK E,YOO S,et al. Compression of deep convolutional neural networks for fast and low power mobile applications[J]. arXiv:1511.06530,2015.

[8] LI H,KADAV A,DURDANOVIC I,et al. Pruning filters for efficient convnets[J]. arXiv:1608.08710,2016.

[9] TANG Q,ZHANG B,LIU J,et al. Dynamic token pruning in plain vision transformers for semantic segmentation[C]// Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023:777-786.

[10] KONG Z,DONG P,MA X,et al. Spvit:Enabling faster vision transformers via latency-aware soft token pruning[C]// European Conference on Computer Vision. Cham:Springer Nature Switzerland,2022:620-640.

[11] SONG Z,XU Y,HE Z,et al. Cp-vit:Cascade vision transformer pruning via progressive sparsity prediction[J]. arXiv:2203.04570,2022.

[12] XU G,HAO J,SHEN L,et al. Lgvit:Dynamic early exiting for accelerating vision transformer[C]// Proceedings of the 31st ACM International Conference on Multimedia. 2023:9103-9114.

[13] LIU W,ZHOU P,ZHAO Z,et al. Fastbert:a self-distilling bert with adaptive inference time[J]. arXiv:2004.02178,2020.

[14] SCHUSTER T,FISCH A,GUPTA J,et al. Confident adaptive language modeling[J]. Advances in Neural Information Processing Systems,2022,35:17456-17472.

[15] MA X,ZHOU A,ZHANG S,et al. Cooperative service caching and workload scheduling in mobile edge computing[C]// IEEE INFOCOM 2020—IEEE Conference on Computer Communications. IEEE,2020:2076-2085.

[16] LIU Y,HE Q,ZHENG D,et al. Data caching optimization in the edge computing environment[J]. IEEE Transactions on Services Computing,2020,15(4):2074-2085.

[17] ZENG F,ZHANG K,WU L,et al. Efficient caching in vehicular edge computing based on edge-cloud collaboration[J]. IEEE Transactions on Vehicular Technology,2022,72(2):2468-2481.

[18] FAN W,GAO L,SU Y,et al. Joint DNN partition and resource allocation for task offloading in edge-cloud-assisted IoT environments[J]. IEEE Internet of Things Journal,2023,10(12):10146-10159.

[19] CHEN H,QIN W,WANG L. Task partitioning and offloading in IoT cloud-edge collaborative computing framework:a survey [J]. Journal of Cloud Computing,2022,11(1):86.

[20] LI X,QIN Y,ZHOU H,et al. An intelligent collaborative infe-

rence approach of service partitioning and task offloading for deep learning based service in mobile edge computing networks [J]. Transactions on Emerging Telecommunications Technologies, 2021, 32(9): e4263.

[21] HAN S, MAO H, DALLY W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. arXiv: 1510.00149, 2015.

[22] OH Y H, QUAN Q, KIM D, et al. A portable, automatic data qantizer for deep neural networks[C]// Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques. 2018: 1-14.

[23] REAGEN B, WHATMOUGH P, ADOLF R, et al. Minerva: Enabling low-power, highly-accurate deep neural network accelerators[J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 267-278.

[24] YANG T J, CHEN Y H, SZE V. Designing energy-efficient convolutional neural networks using energy-aware pruning[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 5687-5695.

[25] CHEN T, CHENG Y, GAN Z, et al. Chasing sparsity in vision transformers: An end-to-end exploration[J]. Advances in Neural Information Processing Systems, 2021, 34: 19974-19988.

[26] LI Y, YU Y, ZHANG Q, et al. Losparse: Structured compression of large language models based on low-rank and sparse approximation[C]// International Conference on Machine Learning. PMLR, 2023: 20336-20350.

[27] ASHKBOOS S, CROCI M L, NASCIMENTO M G, et al. Slicegpt: Compress large language models by deleting rows and columns[J]. arXiv: 2401.15024, 2024.

[28] AN Y, ZHAO X, YU T, et al. Fluctuation-based adaptive structured pruning for large language models[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2024: 10865-10873.

[29] XU X, YAN K, HAN S, et al. Learning-based edge-device collaborative dnn inference in iovt networks[J]. IEEE Internet of Things Journal, 2023, 11(5): 7989-8004.

ZHOU Haojie, born in 2002, postgraduate, is a member of CCF (No. A05537G). His main research interest is edge intelligence.

HAN Rui, born in 1985, assistant professor, Ph. D supervisor. His main research interests include cloud computing and edge intelligence.

(责任编辑:何杨)