

以子图融合为最小单位的混合精度推理

崔丽群, 胡磊

(辽宁工程技术大学软件学院, 辽宁葫芦岛 125105)

摘要: 近几年卷积神经网络作为深度学习最重要的技术,在图像分类、物体检测、语音识别等领域均有所建树。在此期间,由多层卷积神经网络组成的深度神经网络横空出世,在各种任务准确性方面具有显著提升。然而,神经网络的权重往往被限定在单精度类型,使网络体积相较于特定硬件平台上的内存空间更大,且floating point 16、INT 8等单精度类型已无法满足现在一些模型推理的现实需求。为此,提出一种以子图为最小单位,通过判断相邻结点之间的融合关系,添加了丰富比特位的混合精度推理算法。首先,在原有单精度量化设计的搜索空间中增加floating point 16半精度的比特配置,使最终搜索空间变大,为寻找最优解提供更多机会。其次,使用子图融合的思想,通过整数线性规划将融合后的不同子图精度配置,根据模型大小、推理延迟和位宽操作数3个约束对计算图进行划分,使最后累积的扰动误差减少。最终,在ResNet系列网络上验证发现,所提模型精度相较于HAWQ V3的损失没超过1%的同时,相较于其他混合精度量化方法在推理速度方面得到了提升,在ResNet18网络中推理速度分别提升18.15%、19.21%,在ResNet50网络中推理速度分别提升13.15%、13.70%。

关键词: 子图融合;混合精度推理;约束问题最优化求解;GPU加速

DOI: 10.11907/hjtk.231538

开放科学(资源服务)标识码(OSID):



中图分类号: TN911.73

文献标识码: A

文章编号: 1672-7800(2024)006-0044-09

Mixed-Precision Inference with Subgraph Fusion as the Minimum Unit

CUI Liqun, HU Lei

(College of Software, Liaoning Technical University, Huludao 125105, China)

Abstract: In recent years, convolutional neural networks, as the most important technology in deep learning, have made achievements in fields such as image classification, object detection, and speech recognition. During this period, deep neural networks composed of multi-layer convolutional neural networks emerged, showing significant improvements in accuracy in various tasks. However, the weights of neural networks are often limited to single precision types, resulting in a larger memory space compared to specific hardware platforms, and single precision types such as floating point 16 and INT 8 can no longer meet the practical needs of some model inference today. To this end, a mixed precision inference algorithm is proposed, which uses subgraphs as the minimum unit and adds rich bits by judging the fusion relationship between adjacent nodes. Firstly, adding a floating point 16 semi precision bit configuration to the search space of the original single precision quantization design increases the final search space, providing more opportunities for finding the optimal solution. Secondly, using the idea of subgraph fusion, the accuracy of different fused subgraphs is configured through integer linear programming. The computational graph is divided based on three constraints: model size, inference delay, and bitwidth operands, reducing the accumulated disturbance error in the end. In the end, it was verified on the ResNet series network that the proposed model had an accuracy loss of no more than 1% compared to HAWQ V3, while also improving inference speed compared to other mixed precision quantization methods. In the ResNet18 network, the inference speed was improved by 18.15% and 19.21%, respectively, and in the ResNet50 network, the inference speed was improved by 13.15% and 13.70%, respectively.

Key Words: subgraph fusion; mixed precision inference; constrained optimization problem-solving; GPU acceleration

收稿日期: 2023-05-23

基金项目: 辽宁省高等学校基本科研项目(LJKMZ20220699)

作者简介: 崔丽群(1969-),女,硕士,辽宁工程技术大学软件学院副教授,研究方向为智能图像处理,图像与视觉信息计算;胡磊(1997-),男,CCF会员,辽宁工程技术大学软件学院硕士研究生,研究方向为模型压缩。

0 引言

近年来,深度学习已成为热门研究领域,其适用性和重要性得到了广泛验证并在多个领域得到应用。深度学习通过训练多层网络结构对未知数据进行分类或回归,卷积神经网络作为最重要的技术在图像分类、物体检测、语音识别、自然语言处理等领域发挥重要作用。

随着多层卷积神经网络组成的深度神经网络流行,其在各种任务的准确性方面具有显著改进。然而,深度神经网络也面临一个难题,即庞大的参数量、计算量和运算复杂度使其难以部署在特定的硬件平台上^[1]。神经网络的权重通常被限制在单精度类型,从而对模型的占用空间带来了挑战,解决该问题的常用方法是量化神经网络,即使用低精度量化整数值来表示模型参数和特征图,通过对模型的权重和激活进行量化,以有效减少神经网络模型大小,加速模型推理,降低计算资源的消耗。

目前, floating point 16、INT 8等单精度类型已无法满足某些模型推理的实际需求,现有硬件上出现了多种新型运算单元,对不同的数据类型提供了不同的支持。然而在模型推理过程中,这些被设计处理支持不同数据类型的运算单元通常处于闲置状态。为此,提出了一种HAWQ V3方法,针对模型的权重和特征图使用混合精度纯整数量化算法,采用纯整数乘法、加法及静态位移进行量化,整个推理过程中不执行浮点和整数除法计算,在模型推理过程中最大程度利用空闲的运算单元。同时,HAWQ V3提出一种新的硬件感知混合精度量化公式,将不同层的混合精度配置问题形式化为整数线性规划(Integer Linear Programming, ILP)问题,该方法通过定义最小模型扰动、设定模型大小、推理延迟和位宽操作数的约束优化混合精度配置。然而,HAWQ V3忽略了不同层之间的融合关系,仅将优化对象定义为单一层次的网络,搜索空间仅限于4比特和8比特,因此搜索空间较为局限。此外,在设定约束时,以层为单位会导致层与层之间的扰动累积,从而导致误差累积,影响最终优化结果。

为缓解上述问题,本文提出一种以融合且具有更加丰富比特位的混合精度推理算法。在权重和特征图表示方面增加了 floating point 16半精度的比特配置,将搜索空间扩大至 3^L ,提供了更多的机会寻找最优解。然后,采用融合方式对计算图进行划分会使总的累加次数随之减少,模型所累积的扰动误差减少。融合算法根据不同算子的特性决定融合方式减小求解的误差,使得被优化的网络在特定硬件的计算单元上提升推理速度。本文的重要贡献为:

(1)指出了HAWQ V3的局限性。HAWQ V3在为每层网络选择最优比特位配置时,仅提供4比特和8比特两种选择,对于具有L层的模型而言其搜索空间较为局限。在设定约束时仅以每层为单位进行处理,将导致模型层与层

之间的扰动误差累积,最终产生较大误差影响优化结果。

(2)针对网络每层的权重和特征图表示,增加 floating point 16半精度的比特配置,将搜索空间扩大至 3^L 。相较于原先搜索空间而言,为寻找最优解提供了更多机会。

(3)设计了针对子图融合的ILP,并应用子图融合方式对原有的计算图进行划分。该方法相较于以层为单位进行划分可减少总计算次数,降低最终求解过程中模型所累积的扰动误差。

1 相关研究

近年来,人们在改善神经网络模型准确性和效率的平衡间作出了许多贡献,大致可划分为:①设计新的神经网络结构^[2],提出平衡网络深度/宽度/分辨率3个维度的缩放方法;②共同设计神经网络体系结构和硬件^[3];③剪枝操作^[4];④知识蒸馏^[5];⑤量化^[6]。由于本文采用的是量化方法,接下来将进一步概述量化的相关工作。

量化是一种压缩神经网络模型的方法,低精度量化用于权重和激活,在不改变原网络架构的情况下减小模型大小,且在量化过程中允许使用低位矩阵乘法或卷积,虽然能提升速度和效率,但可能会导致精度大幅下降。为此,Zhang等^[7]提出对量化和位宽操作,兼容深度神经网络及其相关量化器的联合训练方法,在CIFAR-10和ImageNet数据集的实验表明该方法能很好地适用于各种网络结构,虽然一直使用浮点数进行量化,但量化程度受限。Elthakeb等^[8]提出周期性正则化辅助量化,利用正弦函数的周期性、可微性和局部凸性轮廓,自动推进网络权重值在共同确定的水平上量化,然而使用统一精度量化会导致精度下降。尽管这些工作都取得了一些进步,但始终是在统一精度量化方向,均会造成精度下降。

目前,新的方向是采用混合精度量化。在混合精度量化中某些层保持较高精度,而其他层保持较低精度,卷积神经网络在相同的压缩比下可获得更高的准确率。Tianshu等^[9]提出一种简单、有效的混合精度量化神经网络,逐渐减小比特宽度以改善精度和压缩之间的权衡。结果表明,由于更好的保留了原始图像信息,较高精度的底层可显著提高1位网络的性能,而较低精度的后验层有助于k-bit网络的正则化,但该方法在较大的空间中进行搜索时需要大量数据,极其耗费时间。

后续,又在混合精度方向提出了许多新方法。Huanrui等^[10]提出位级稀疏量化来解决混合精度量化问题,相较于以前的方法,BSQ在CIFAR-10和ImageNet数据集的各种模型结构上精度和位缩减更高,但局限于只能对权重进行位宽的选择,激活值只能手动设置。Vincent等^[11]提出给每一层引入表示bit-length的变量来自动学习,采用插值方式进行非整型bit量化的表示,证明了该方法可有效保持准确性,但需要耗费大量的计算资源。Wang等^[12]

通过强化学习算法搜索空间,结合硬件仿真器反馈的硬件指标(Latency 和 Energy)约束搜索,相较于固定位宽(8位)量化而言,有效地将延迟降低了34%~61%,将能耗降低了65%,但需要通过强化学习进行搜索,搜索过程需要花费大量时间。Elthakeb 等^[13]使用强化学习为模型的每一层的权重和激活值搜索位宽,但采用的是经典的PPO算法,在实验设计方面更精细,仍需消耗大量计算资源和时间。Wu 等^[14]利用可微分NAS的搜索方法搜索位宽,实验表明该方法在CIFAR-10和ImageNet数据集上超越了ResNet,但其局限在于只跟Tenary Neural Network(三值神经网络)进行了比较。

此外,Dong 等^[15]提出通过Hessian光谱获得的灵敏度寻找到合适的混合精度,以有效减少计算时间及计算资源,但该方法仅提供相对灵敏度,要求手动确定位宽精度,无形中也消耗了大量时间。Dong 等^[16]通过基于Pareto前沿的方法避免手动确定位宽精度,无需任何手动选择即可自动确定不同层的位精度,理论分析表明使用平均Hessian轨迹的灵敏度度量更好,但无法同时满足不同的硬件需求。为此,Yao 等^[17]通过HAWQ V3的ILP解决方案生成具有各种约束(模型大小、BOPS和延迟)的混合精度设置,在硬件上消耗几秒钟即可解决。Hubara 等^[18]通过改进传统的ILP,使其能够部署在硬件上。

然而,HAWQ V3也存在着一些问题。其中,HAWQ V3在使用ILP量化每一层时,只有4比特和8比特两种选择,对于具有L层的模型而言,ILP的搜索空间较小,仅为 2^L ;在利用ILP解决问题时以层为单位进行计算,将增加层与层之间的扰动误差。为解决上述问题,本文增加floating point 16半精度的比特配置以扩大搜索空间,为寻找最优解提供了更多机会,然后提出子图融合的方法,利用子图切分有效减少了扰动误差。

2 背景知识

2.1 计算图

计算图 $G=(V, E)$ 是有向无环图(Directed Acyclic Graph),可定义神经网络模型, V 为所有算子集合, E 描述各算子之间的依赖关系^[19]。集合 V 中的每个元素用来表示不同类型的算子,例如卷积层、全连接层、批量归一化处理层。集合 E 的每个元素是由二元组 (u, v) 组成,算子 u 表示起点张量,算子 v 表示终点张量。计算图 $G=(V, E)$ 的详细描述和定义如图1所示。

2.2 计算子图

计算子图 $G'=(V', E')$ 是有向无环图的一个子图, V' 为所有算子集合 V 中的一个子集, E' 用来描述 V' 子集中各算子间的依赖关系,且 V' 的所有结点都具有很强的连通性。集合 V' 中各元素都像集合 V 一样表示不同类型的算子。集合 E' 的每个元素也像集合 E 中的一样由二元组 (u', v')

组成。其中,算子 u' 表示起点张量, v' 表示终点张量。计算子图 $G'=(V', E')$ 的信息如图2可见。

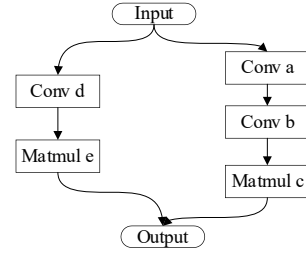


Fig. 1 Computation graph
图1 计算图

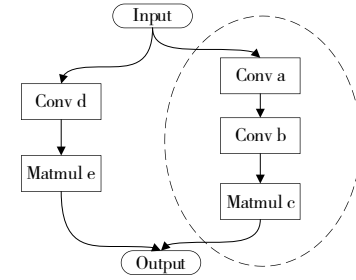


Fig. 2 Computation subgraph
图2 计算子图

2.3 混合精度量化

量化是深度学习模型在各大硬件平台落地的重要基础技术。研究表明,将深度学习模型从全精度量化到8bit以下的研究与实践非常广泛。早期研究主要集中于训练后对权重进行量化,而现在研究关注于训练过程中完成权重、激活、误差和梯度的量化。在量化过程中,通常采用均匀量化将所有层的精度统一降为低比特宽度的INT 8、INT 4、INT 2等,量化将神经网络权重/激活限制为有限值集,如式(1)所示。

$$Q(r) = \text{Int}(r/s) - Z \quad (1)$$

式中: Q 为量化运算符; r 为真实值数(激活或权重); S 为实值缩放因子; Z 为零点,选择零点使得0值精确地映射到量化值;Int通过舍入操作将浮点值映射为整数值,但相较于原始FP32浮点数模型,可能会导致精度明显下降。

为了优化模型的推理速度,量化研究开始向混合精度量化方向发展。HAWQ V3将敏感层保持在较高的精度,而将不敏感层保持在较低的精度,但并非所有层都能从量化过程中获得线性比例的推理速度提升。事实上,一些特定层在量化为低精度时,无法获得任何推理速度提升,而有些层则在该量化过程中获得超越线性比例的加速收益。因此,对这些特定层进行量化可能会损害整个模型的最终精度。

2.4 子图切分调度

该过程首先合并相邻节点,然后将合并后的节点整体融合,从整个计算图中切分出来。最终,定义在计算图 G 上的切分调度 S ,如式(2)所示。

$$S = \{(V_1, F_1), (V_2, F_2), \dots, (V_k, F_k)\} \quad (2)$$

式中: V_i 表示第 i 个子图中的所有算子; F_i 表示两个相邻节点之间的融合关系。

通过计算图 G 在切分调度 S 下分解成连续的子图, 即这些连续的子图之间相互独立, 并没有重合的节点。然后给定 GPU 上的计算图 G 和切分调度 S , 目标是求出一个最

优的切分调度 S^* , 使给定计算图 G 延迟最小, 如式 (3) 所示。

$$S^* = \operatorname{argmin} \operatorname{Cost}(G, S) \quad (3)$$

式中: Cost 为在计算图 G 和切分调度 S 上定义的函数; 函数 $\operatorname{Cost}(G, S)$ 被定义为在切分调度 S 之后运行计算图 G 的延迟。相邻结点之间的融合关系如图 3 所示。

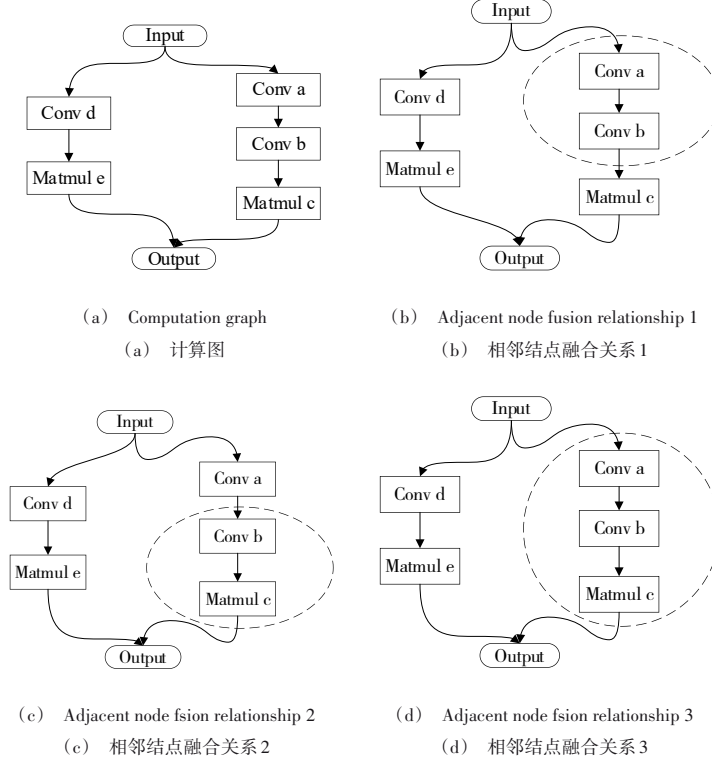


Fig. 3 Fusion relationship between adjacent nodes

图 3 相邻结点之间的融合关系

3 本文方法

3.1 硬件部署

传统的 CUDA Core 管道只支持 32 位浮点加法、32 位浮点乘法及 32 位到 8 位整数运算 (移位、加法等操作)^[20]。实验表明, 仅通过浮点运算次数和模型大小衡量神经网络的效率不够精确, 有些小型模型可能需要更多的浮点运算次数且延迟更高。由于本文使用了混合精度, 传统的 CUDA Core 已无法满足需求, 因此采用 Tensor Core 进行部署。Tensor Core 是一种新型的处理单元, 专为深度学习应用而设计, 是一种矩阵乘积累加计算单元, 可在一个时钟周期内实现两个 4×4 矩阵乘法和 4×4 矩阵加法 (即 MAC 运算)。Tensor Core 支持 INT 8 和 INT 4 精度, 已根据深度学习网络推理进行了优化, 但唯一可用的 API 是 WMMA 指令调用。WMMA 指令是在 Tensor Core 上以 INT 4 精度执行矩阵运算的微内核, 目前仍未有编译器可使用 WMMA 指令将量化为 INT 4 的神经网络映射到 Tensor Core。为了解决这一问题, 本文基于 HAWQ V3 论文中的方法, 扩展 TVM

以支持 INT 4 推理, 并采用类似 Cutlass 的调度策略, 为 TVM 中的 INT 8 和 INT 4 的 Tensor Core 实现了新的直接卷积调度^[21]。

3.2 基于子图融合的混合精度推理

目前, 在量化神经网络为低精度时需平衡精度和延迟, 通常运用 ILP 寻找最佳折中的位宽精度设置, 以形式化问题定义的方式解决。现有 3 种选择可用于每一层的量化精度 (INT4、INT8 和 floating point 16), 对于 L 层的模型而言 ILP 算法的搜索空间为 3^L 。ILP 问题的解决目标在于通过指定的约束条件, 在这些 3^L 可能性中找到最佳位宽配置, 从而在扰动 Ω 范围内实现最佳折中。

HAWQ V3 处理约束是以每一层为单位进行处理。若假设在没有融合的情况下, 每个层的误差都固定, 将导致模型层之间扰动的不断累积, 最终会导致过大的扰动误差。具体的划分方式如式 (4)~式 (7) 所示。

$$\text{Objective: } \min \sum_{i=1}^L \Omega_i^{(b_i)} \quad (4)$$

$$\text{subject to: } \sum_{i=1}^L M_i^{(b_i)} \leq \text{Model Size Limit} \quad (5)$$

$$\sum_{i=1}^L G_i^{(b_i)} \leq \text{BOPS Limit} \quad (6)$$

$$\sum_{i=1}^L Q_i^{(b_i)} \leq \text{Latecny Limit} \quad (7)$$

本文的创新点在于采用计算子图作为基本划分单元,而非以每一层为单位进行划分。相较于HAWQ V3,以计算子图为单位可减少累加次数、降低误差,从而提升推理速度。采用子图划分方式时可设置不同的分组次数,重新生成新的约束条件。由于在子图内部可能存在节点融合现象,可进一步减少计算次数和误差提升推理速度。

HAWQ V3假定整张计算图中的每个节点均互相独立扰动,而本文假设子图扰动相互独立,但子图内部节点之间均存在联系,提供均匀切分和不均匀切分两种切分方案。

3.2.1 均匀切分

将网络均分为3个均等子图,如式(9)一式(11)所示。

$$\text{Objective: } \min \sum_{i=1}^{\frac{L}{3}} \Omega_i^{(b_i)} + \sum_{i=\frac{L}{3}}^{\frac{2L}{3}} \Omega_i^{(b_i)} + \sum_{i=\frac{2L}{3}}^L \Omega_i^{(b_i)} \quad (8)$$

$$\text{Subject to: } \sum_{i=1}^{\frac{L}{3}} M_i^{(b_i)} + \sum_{i=\frac{L}{3}}^{\frac{2L}{3}} M_i^{(b_i)} + \sum_{i=\frac{2L}{3}}^L M_i^{(b_i)} \leq \text{Model Size Limit} \quad (9)$$

$$\sum_{i=1}^{\frac{L}{3}} G_i^{(b_i)} + \sum_{i=\frac{L}{3}}^{\frac{2L}{3}} G_i^{(b_i)} + \sum_{i=\frac{2L}{3}}^L G_i^{(b_i)} \leq \text{BOPS Limit} \quad (10)$$

$$\sum_{i=1}^{\frac{L}{3}} Q_i^{(b_i)} + \sum_{i=\frac{L}{3}}^{\frac{2L}{3}} Q_i^{(b_i)} + \sum_{i=\frac{2L}{3}}^L Q_i^{(b_i)} \leq \text{Latency Limit} \quad (11)$$

3.2.2 不均匀切分

假设计算图有18个节点,将其不均匀切分成3个子图,每个子图分别含有3、5、10个节点,具体切分过程如式(12)一式(15)所示。

$$\text{Objective: } \min \sum_{i=1}^3 \Omega_i^{(b_i)} + \sum_{j=1}^5 \Omega_j^{(b_j)} + \sum_{k=1}^{10} \Omega_k^{(b_k)} \quad (12)$$

$$\text{Subject to: } \sum_{i=1}^3 M_i^{(b_i)} + \sum_{j=1}^5 M_j^{(b_j)} + \sum_{k=1}^{10} M_k^{(b_k)} \leq \text{Model Size Limit} \quad (13)$$

$$\frac{1}{3} \sum_{i=1}^3 G_i^{(b_i)} + \frac{1}{5} \sum_{j=1}^5 G_j^{(b_j)} + \frac{1}{10} \sum_{k=1}^{10} G_k^{(b_k)} \leq \frac{1}{3} \text{BOPS Limit} \quad (14)$$

$$\sum_{i=1}^3 Q_i^{(b_i)} + \sum_{j=1}^5 Q_j^{(b_j)} + \sum_{k=1}^{10} Q_k^{(b_k)} \leq \text{Latency Limit} \quad (15)$$

由于每个子图的扰动相互独立,可预先计算每个子图的灵敏度。在计算灵敏度时,采用ILP问题求解方法,旨在寻找让灵敏度最小化的正确位宽精度。 $\Omega_i^{(b_i)}$ 表示第*i*个子图中每个算子在运行时具有 b_i 比特所产生的扰动, $\sum_{i=1}^3 \Omega_i^{(b_i)}$ 表示第一个子图含有3个算子,且每个算子在运行时的比特配置为 b_i ,在该假设下将原图切分为3个不

同大小的子图,每个子图所含的算子数目不相同,可将该问题转化为求解3个不同大小的子图所带来的扰动的最小值; $\sum_{j=1}^5 \Omega_j^{(b_j)}$ 表示第二个子图含有5个算子,每个算子在运行时的比特配置为 b_j ,在该假设下将原图切分为5个不同大小的子图,且每个子图所含算子数目不相同,将该问题被转化为求解5个不同大小的子图所带来的扰动的最小值; $\sum_{k=1}^{10} \Omega_k^{(b_k)}$ 表示第3个子图含有10个算子,每个算子在运行时的比特配置为 b_k ,在该假设下将原图切分为10个不同大小的子图,且每个子图所含算子数目也不相同,该问题被转化为求解10个不同大小的子图所带来的扰动的最小值; $M_i^{(b_i)}$ 表示在具有 b_i 比特配置时的第*i*个子图的大小; $\sum_{i=1}^3 M_i^{(b_i)}$ 表示第一个子图含有3个算子,每个算子在运行时的比特配置为 b_i ,在该假设下将原图切分为3个不同大小的子图,每个子图所含的算子数目也不相同,因此该约束被转化为求解3个不同大小的子图的大小的最小值; $\sum_{j=1}^5 M_j^{(b_j)}$ 表示第二个子图含有5个算子,且每个算子在运行时的比特配置为 b_j ,在该假设下将原图切分为5个不同大小的子图,每个子图所含算子数目也不相同,因此该约束被转化为求解5个不同大小的子图的大小的最小值; $\sum_{k=1}^{10} M_k^{(b_k)}$ 表示第3个子图含有10个算子,且每个算子在运行时的比特配置为 b_k 。在该假设下将原图切分为10个不同大小的子图,每个子图所含算子数目也不相同,因此该约束被转化为求解10个不同大小的子图的大小的最小值; $G_i^{(b_i)}$ 为计算第*i*个子图所需的对应BOPS,BOPS用来表示整个计算子图的权重与激活值的比特配置和算子的浮点运算次数乘积。

$$G_i^{(b_i)} = b_{w_i} b_{a_i} \text{MAC}_i \quad (16)$$

式中: MAC_i 用于计算第*i*个算子的总乘—累加运算; b_{w_i} 、 b_{a_i} 为权重和激活的位精度。

通常而言,使用哪种约束主要取决于最终的用户应用程序。当进行不均匀切分时,规定 $G_i^{(b_i)}$ 前的系数由该计算子图所对应的算子个数决定,如式(14)所示。 $\frac{1}{3} \sum_{i=1}^3 G_i^{(b_i)}$ 表示第一个子图含有3个算子且使用3个算子的BOPS之和的平均值表示该子图所对应的BOPS。 $\frac{1}{5} \sum_{j=1}^5 G_j^{(b_j)}$ 表示第二个子图含有5个算子且使用5个算子的BOPS之和的平均值表示该子图所对应的BOPS。 $\frac{1}{10} \sum_{k=1}^{10} G_k^{(b_k)}$ 表示第3个子图含有10个算子且使用10个算子的BOPS之和的平均值表示该子图所对应的BOPS; $Q_i^{(b_i)}$ 表示第*i*个子图在具有 b_i 比特配置时的运行延时,如式(15)所示; $\sum_{i=1}^3 Q_i^{(b_i)}$ 表示第一个子图含有3个算子,且每个算子在运行时的比特配置为 b_i ,在该假设下将原图切分为3个不同大小的子图,每个子图所含的算子数目也不相同,因此该约束被转化为求解3个不同大小的子图所产生的运行

延时之和的最小值; $\sum_{j=1}^5 Q_j^{(b_j)}$ 表示第二个子图中含有 5 个算子, 每个算子在运行时的比特配置为 b_j , 在该假设下将原图切分为 5 个不同大小的子图, 每个子图所含算子数目也不相同, 该约束被转化为求解 5 个不同大小的子图所产生的运行延时之和的最小值; $\sum_{k=1}^{10} Q_k^{(b_k)}$ 表示第 3 个子图中含有 10 个算子, 每个算子在运行时的比特配置为 b_k , 在该假设下将原图切分为 10 个不同大小的子图, 且每个子图所含的算子数目也不相同, 该约束被转化为求解 10 个不同大小的子图所产生的运行延时之和的最小值。

此外, 通过固定厂商提供的模板库方法来获取 latency Limit, 即使用 cuDNN 推理库在 GPU 上测量切分后子图的推理延时。

3.3 最小单位切分子图

通过上述对均匀切分和不均匀切分两种情况的充分讨论, 本文定义的最小单位切分子图的数量并不会严格限制在具体的子图数量上, 由调度器在量化的过程中对网络的实际情况进行动态切分。例如, 在对 ResNet18 网络进行子图切分的过程中 (见图 3b) 选择两个子图为最小切分单位 (见图 3d), 而本文选择 3 个子图作为最小切分单位, 由式 (6) 一式 (13) 通过整数线性规划求出具体的切分规则及最小单位数量。

4 实验结果与分析

本文实验验证了所提量化压缩方法, 在 ResNet18、ResNet50 网络混合精度量化的推理速度相较于 HAWQ V3 有所提高。

4.1 实验环境与数据集

实验硬件为 GPU 为 RTX 2080Ti (1 块), 11 GB 显存, CPU 为 i7-10 代 (6 核), RAM 为 64 GB, 主频为 1.10 GHz, 操作系统为 64 位 Linux。软件平台为 Xshell 7, 软件环境为 Python 3.8, 深度学习框架为 Pytorch 1.4。采用 ImageNet 数据集进行实验, 评价指标为 Performed Inference, Performed Inference 越小表示推理速度越快。

4.2 基线方法

4.2.1 基线实验配置

在基线实验中研究了 ILP 公式中不同约束大小对推理速度的影响, 基线实验设置了 3 个约束, 其中 BOPS 代表位宽操作, Latency 代表延迟, Model Size 代表模型大小。针对每个约束考虑 3 个不同的阈值, 分别为 0.75、0.5、0.25。在 BOPS 中, 0.75 表示 $4\text{bit_BOPS} * 0.25 + 8\text{bit_BOPS} * 0.75$, 0.5 表示 $4\text{bit_BOPS} * 0.5 + 8\text{bit_BOPS} * 0.5$, 0.25 表示 $4\text{bit_BOPS} * 0.75 + 8\text{bit_BOPS} * 0.25$; 同理在 Latency 和 Model Size 两个约束中的 3 个阈值也一致, 在设置阈值的同时研究 ILP 公式如何平衡折中获得最优量化模型。

4.2.2 基线实验结果

实验对单独设置的各个约束条件进行了测试, 实验结

果如表 1、表 2 所示。该表格分为 BOPS、Latency 和 Model Size 3 个部分, 每部分都具有相应的约束条件, 采用 ILP 求解器在最大化精度的前提下, 寻找最佳混合精度以满足约束条件。首先, 对 ResNet 18 网络的 BOPS、Latency 和 Model Size 进行实验, 然后以同样的约束条件在 ResNet50 网络上进行实验。

Table 1 Experimental results of ResNet18 inference baseline

表 1 ResNet18 推理基线实验结果

BOPS	Latency	Model Size	Performed Inference/ms	Std	Average Per Sample/ms
0.75	/	/	3.67	0.01	0.46
0.5	/	/	3.53	0.00	0.44
0.25	/	/	3.30	0.00	0.41
/	0.75	/	3.67	0.00	0.46
/	0.50	/	3.49	0.00	0.44
/	0.25	/	3.30	0.00	0.41
/	/	0.75	3.80	0.00	0.48
/	/	0.5	3.77	0.00	0.47
/	/	0.25	3.75	0.00	0.46

Table 2 Experimental results of ResNet50 inference baseline

表 2 ResNet 50 推理基线实验结果

BOPS	Latency	Model Size	Performed Inference/ms	Std	Average Per Sample/ms
0.75	/	/	8.71	0.00	1.09
0.50	/	/	8.24	0.00	1.03
0.25	/	/	7.91	0.00	0.99
/	0.75	/	8.69	0.00	1.09
/	0.50	/	8.37	0.01	1.05
/	0.25	/	7.95	0.00	0.99
/	/	0.75	9.05	0.01	1.13
/	/	0.50	8.79	0.05	1.10
/	/	0.25	8.63	0.00	1.08

由表 1、表 2 可知, 在 ResNet18 网络中当目标函数设置为 $Objective: \min \sum_{i=1}^L \Omega_i^{(b_i)}$ 时, 将位宽操作的上界设置为 0.75 后, 通过 8 个样本进行推理运算得到的 ResNet18 网络的推理速度为 3.67 ms; 将延迟的上界设置为 0.50 后, 最终所得到的推理速度为 3.49 ms; 将模型大小的上界设置为 0.25 时, ResNet18 网络的推理速度为 3.75 ms。在 ResNet50 网络中也得到类似结果。实验表明, 在 ResNet18 网络中 BOPS 设置为 0.25 时推理速度最快, 达到 3.30 ms; 在 Latency 和 Model Size 设置为 0.25 时速度最快, 分别达到 3.30 ms 和 3.75 ms; 在 ResNet50 网络中, BOPS 设置为 0.25 时推理速度最快, 达到 7.91 ms; 在 Latency 和 Model Size 设置为 0.25 时最快, 分别达到 7.95 ms 和 8.63 ms。

上述实验中, ResNet18、ResNet50 所有模型均采用 TVM 框架, 在硬件中以量化精度方法测试运行速度, 因此这些结果不仅是模拟结果, 也能在实际应用场景中得到验证。

4.3 本文实验

将本文提出的两种切分方法运用到基线模型上, 实验

结果如表3—表6所示。同时,将基于均匀切分或不均匀切分的基线方法与原基线方法进行比较,实验结果如表7—表10所示。

Table 3 Results of ResNet18 inference with equally split

表3 均匀切分 ResNet18 推理实验结果

BOPS	Latency	Model Size	Performed Inference/ms	Std	Average Per Sample/ms
0.75	/	/	3.56	0.01	0.45
0.50	/	/	3.36	0.00	0.42
0.25	/	/	3.26	0.01	0.41
/	0.75	/	3.64	0.00	0.46
/	0.50	/	3.31	0.00	0.41
/	0.25	/	3.11	0.00	0.39
/	/	0.75	3.73	0.01	0.47
/	/	0.50	3.73	0.00	0.47
/	/	0.25	3.71	0.00	0.46

Table 4 Results of ResNet50 inference with equally split

表4 均匀切分 ResNet50 推理实验结果

BOPS	Latency	Model Size	Performed Inference/ms	Std	Average Per Sample/ms
0.75	/	/	8.68	0.03	1.08
0.50	/	/	8.19	0.00	1.02
0.25	/	/	7.86	0.00	0.98
/	0.75	/	8.62	0.00	1.08
/	0.50	/	8.13	0.02	1.02
/	0.25	/	7.93	0.00	0.99
/	/	0.75	8.77	0.00	1.10
/	/	0.50	8.74	0.02	1.09
/	/	0.25	8.42	0.00	1.05

Table 5 Results of ResNet18 inference with unequally split

表5 不均匀切分 ResNet18 推理实验结果

BOPS	Latency	Model Size	Performed Inference/ms	Std	Average Per Sample/ms
0.75	/	/	3.47	0.00	0.43
0.50	/	/	3.34	0.00	0.42
0.25	/	/	3.16	0.00	0.39
/	0.75	/	3.55	0.00	0.44
/	0.50	/	3.31	0.00	0.41
/	0.25	/	3.07	0.01	0.38
/	/	0.75	3.65	0.01	0.46
/	/	0.50	3.63	0.01	0.45
/	/	0.25	3.57	0.01	0.45

Table 6 Results of ResNet50 inference with unequally split

表6 不均匀切分 ResNet50 推理实验结果

BOPS	Latency	Model Size	Performed Inference/ms	Std	Average Per Sample/ms
0.75	/	/	8.58	0.01	1.07
0.50	/	/	8.10	0.00	1.01
0.25	/	/	7.81	0.00	0.97
/	0.75	/	8.60	0.00	1.08
/	0.50	/	8.10	0.05	1.01
/	0.25	/	7.91	0.00	0.99
/	/	0.75	8.57	0.00	1.07
/	/	0.50	8.46	0.00	1.06
/	/	0.25	8.33	0.00	1.04

Table 7 Comparison of accuracy results between equally split

ResNet18 and HAWQ V3

表7 均匀切分 ResNet18 与 HAWQ V3 精度结果比较

BOPS	Latency	Model Size	HAWQ V3 Accuracy	Our Accuracy
0.75	/	/	70.40	70.26
0.50	/	/	70.22	69.98
0.25	/	/	68.72	68.65
/	0.75	/	70.40	70.22
/	0.50	/	70.34	70.15
/	0.25	/	68.56	68.52
/	/	0.75	71.20	70.10
/	/	0.50	70.50	70.10
/	/	0.25	70.01	69.65

Table 8 Comparison of accuracy results between equally split

ResNet50 and HAWQ V3

表8 均匀切分 ResNet50 与 HAWQ V3 精度结果比较

BOPS	Latency	Model Size	HAWQ V3 Accuracy	Our Accuracy
0.75	/	/	76.10	76.05
0.50	/	/	75.39	75.29
0.25	/	/	74.45	74.37
/	0.75	/	76.63	76.26
/	0.50	/	74.95	74.69
/	0.25	/	74.26	74.26
/	/	0.75	77.38	76.39
/	/	0.50	75.95	75.84
/	/	0.25	74.89	74.60

Table 9 Comparison of accuracy results between unequally split

ResNet18 and HAWQ V3

表9 不均匀切分 ResNet18 与 HAWQ V3 精度结果比较

BOPS	Latency	Model Size	HAWQ V3 Accuracy	Our Accuracy
0.75	/	/	70.40	70.15
0.50	/	/	70.22	69.95
0.25	/	/	68.72	68.52
/	0.75	/	70.40	70.16
/	0.50	/	70.34	70.08
/	0.25	/	68.56	68.46
/	/	0.75	71.20	69.50
/	/	0.50	70.50	69.55
/	/	0.25	70.01	69.04

Table 10 Comparison of accuracy results between unequally split

ResNet50 and HAWQ V3

表10 不均匀切分 ResNet50 与 HAWQ V3 精度结果比较

BOPS	Latency	Model Size	HAWQ V3 Accuracy	Our Accuracy
0.75	/	/	76.10	75.90
0.50	/	/	75.39	75.25
0.25	/	/	74.45	74.29
/	0.75	/	76.63	76.16
/	0.50	/	74.95	74.53
/	0.25	/	74.27	74.25
/	/	0.75	77.38	76.39
/	/	0.5	75.95	75.20
/	/	0.25	74.89	74.42

所有的实验都采用了相同的测速方法,并且保证测速的误差几乎可以忽略。由表 3、表 4 可知,在 ResNet18 和 ResNet50 网络中进行均匀切分时,推理速度相较于基线结果有所提升。在 ResNet18 网络中,当 BOPS 设置为 0.25 时推理速度最快,达到 3.26 ms,相较于基线结果提升 1.21%,在 Latency 和 Model Size 设置为 0.25 时推理速度最快,分别为 3.11 ms、3.71 ms,相较于基线结果分别提升 5.76% 和 1.07%。在 ResNet50 网络中,当 BOPS 设置为 0.25 时推理速度最快,达到 7.86 ms,相较于基线结果提升 0.63%,在 Latency 和 Model Size 设置为 0.25 时推理速度最快,分别为 7.93 ms、8.42 ms,相较于基线结果分别提升 0.25% 和 2.43%。

由表 5、表 6 可知,在 ResNet18 和 ResNet50 网络中进行不均匀切分时,推理速度不仅相较于基线结果有所提升,还相较于均匀切分推理速度也有所提升。在 ResNet18 网络中,当 BOPS 设置为 0.25 时推理速度最快,达到 3.16 ms,相较于基线结果提升 4.24%,相较于均匀切分结果提升 3.07%,在 Latency 和 Model Size 设置为 0.25 时推理速度最快,分别为 3.07 ms 和 3.57 ms,相较于基线结果分别提升 6.97%、4.80%,相较于均匀切分结果分别提升 1.29% 和 3.77%。在 ResNet50 网络中,当 BOPS 设置为 0.25 时推理速度最快,达到 7.81 ms,相较于基线结果提升 1.26%,相较于均匀切分结果提升 0.64%;在 Latency 和 Model Size 设置为 0.25 时推理速度最快,分别达到 7.91 ms 和 8.33 ms,相

较于基线结果分别提升 0.51% 和 3.48%,相较于均匀切分结果分别提升 0.25% 和 1.08%。在 ResNet18 网络进行均匀切分时且 Latency 设置为 0.25 时,相较于 HAWQ V3 的 Model Size 设置为 0.75 时,推理速度提升 18.15%。在 ResNet18 网络不均切分且 Latency 设置为 0.25 时,相较于 HAWQ V3 的 Model Size 设置为 0.75 时推理速度提升 19.21%。在 ResNet50 网络进行均匀切分且 BOPS 设置为 0.25 时,相较于 HAWQ V3 的 Model Size 设置为 0.75 时推理速度提升 13.15%。在 ResNet50 网络进行不均切分且 BOPS 设置为 0.25 时,相较于 HAWQ V3 的 Model Size 设置为 0.75 时,推理速度提升 13.70%。

由表 7、表 8 可知,在 ResNet18、ResNet50 网络中进行均匀切分时,本文方法相较于 HAWQ V3 的精度而言,在精度损失上没有超过 1%。由表 9、表 10 可知,在 ResNet18 和 ResNet50 网络中进行不均匀切分时,本文方法在精度损失上相较于 HAWQ V3 也没有超过 1%。

4.4 消融实验

将不均匀切分 ResNet18 进行消融实验,以验证模型各方面对结果的优化程度,实验结果如表 11 所示。

由此可知,在 ResNet18 和 ResNet50 网络中,当将 BOPS、Latency 和 Model Size 阈值分别设置为 0.9、0.6 和 0.3 时推理速度明显下降。在保持精度的前提下,将 3 个约束阈值分别设置为 0.75、0.5 和 0.25 时模型最优,可显著提升推理速度。

Table 11 Ablation experiment result of unequally split ResNet18

表 11 不均匀切分 ResNet18 消融实验结果

BOPS	Latency	Model Size	Performed Inference/ms	Std	Average Per Sample/ms
0.90	/	/	3.91	0.01	0.49
0.60	/	/	3.62	0.01	0.45
0.30	/	/	3.58	0.01	0.45
/	0.90	/	3.74	0.01	0.47
/	0.60	/	3.81	0.01	0.48
/	0.30	/	3.77	0.01	0.47
/	/	0.90	3.76	0.00	0.47
/	/	0.60	3.64	0.01	0.46
/	/	0.30	3.83	0.01	0.48

综上,本文所提混合精度推理方法通过子图融合进行调优,在均匀切分和不均匀切分的情况下,ResNet18 网络相较于 HAWQ V3 推理基线结果分别提升 18.15% 和 19.21%。在 ResNet50 网络中,推理速度分别提升 13.15% 和 13.70%,证实了本文方法可在 GPU 上提升推理速度。此外,本文对精度变化进行研究发现,在均匀切分和不均匀切分时,模型的推理速度相较于基线提升显著,且精度相较于 HAWQ V3 损失不超过 1%。

5 结语

目前,混合精量化存在搜索空间受限、扰动误差较

大等问题。为此,本文采用更丰富的比特位和子图融合方法,提出一种混合精度推理算法。该方法不仅扩大了网络每层权重和特征图表示的搜索空间,还能减少扰动误差,优化后的网络在特定硬件计算单元上的推理速度显著提升。在 ResNet18 网络中,均匀切分和不均匀切分使推理速度分别提升 18.15%、19.21%。在 ResNet50 网络中推理速度分别提升 12.38%、13.70%。同时,在这两个网络中无论采用何种切分方法,精度损失均不超过 1%。

未来,不仅局限于设计 GPU 上的 WMMA 指令,还会将重心放在特定指令,使其能在特定运算单元的加速器上运行。

参考文献:

- [1] LI J Y, ZHAO Y K, XUE Z E, et al. A review on deep neural network model compression[J]. Chinese Journal of Engineering Science, 2019, 41(10): 1229-1239.
李江昀, 赵义凯, 薛卓尔, 等. 深度神经网络模型压缩综述[J]. 工程科学学报, 2019, 41(10): 1229-1239.
- [2] TAN M, TAN M, LE Q. EfficientNet: rethinking model scaling for convolutional neural networks[C]// International Conference on Machine Learning, 2019: 6105-6114.
- [3] WU B, DAI X, ZHANG P, et al. FBNet: hardware-aware efficient ConvNet design via differentiable neural architecture search[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019: 10734-10742.
- [4] YANG T J, CHEN Y H, SZE V. Designing energy-efficient convolutional neural networks using energy-aware pruning [C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2017: 5687-5695.
- [5] HINTON G, VINYALS O, DEAN J. Distilling the knowledge in a neural network[DB/OL]. <https://arxiv.org/pdf/1503.02531.pdf>.
- [6] 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/CVF Conference on Computer Vision and Pattern Recognition, 2017: 2704-2713.
- [7] ZHANG D, YANG J, YE D, et al. LQ-Nets: learned quantization for highly accurate and compact deep neural networks[C]// Proceedings of the European Conference on Computer Vision, 2018: 365-382.
- [8] ELTHAKEB A, PILLIGUNDLA P, MIRESHGHALLAH F, et al. Gradient-based deep quantization of neural networks through sinusoidal adaptive regularization [DB/OL]. <https://arxiv.org/ftp/arxiv/papers/2020/2003.00146.pdf>.
- [9] TIANSHU C, QIN L, JIE Y, et al. Mixed-precision quantized neural networks with progressively decreasing bitwidth for image classification and object detection [DB/OL]. <https://arxiv.org/ftp/arxiv/papers/2019/1912.12656.pdf>.
- [10] HUANRUI Y, LIN D, YIRAN C, et al. BSQ: exploring bit-level sparsity for mixed-precision neural network quantization[DB/OL]. <https://arxiv.org/ftp/arxiv/papers/2021/2102.10462.pdf>.
- [11] VINCENT G, ANDREAS M, CIARAN B, et al. BitPruning: learning bitlengths for aggressive and accurate quantization[DB/OL]. <https://arxiv.org/ftp/arxiv/papers/2020/2002.03090.pdf>.
- [12] 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.
- [13] ELTHAKEB A, PILLIGUNDLA P, MIRESHGHALLAH F S, et al. Releq: an automatic reinforcement learning approach for deep quantization of neural networks[J]. IEEE Micro, 2020, 40(5): 37-45.
- [14] WU B, WANG Y, ZHANG P, et al. mixed precision quantization of ConvNets via differentiable neural architecture search [DB/OL]. <https://arxiv.org/ftp/arxiv/papers/2018/1812.00090.pdf>.
- [15] DONG Z, YAO Z, GHOLAMI A, et al. Hawq: hessian aware quantization of neural networks with mixed-precision [C]// Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019: 293-302.
- [16] DONG Z, YAO Z, ARFEEN D, et al. HAWQ-V2: hessian aware trace-weighted quantization of neural networks[DB/OL]. <https://arxiv.org/abs/1911.03852>.
- [17] YAO Z, DONG Z, ZHENG Z, et al. Hawq-v3: dyadic neural network quantization [C]// International Conference on Machine Learning, 2021: 11875-11886.
- [18] HUBARA I, NAHSHAN Y, HANANI Y, et al. Improving post training neural quantization: layer-wise calibration and integer programming[DB/OL]. <https://arxiv.org/ftp/arxiv/papers/2021/006.10518.pdf>.
- [19] WANG H B, HUANGFU W, LIU Y X, et al. Optimization algorithm and implementation of IoT service coverage in mobile communication networks based on computational graph[J]. Chinese Journal of IoT, 2019, 3(2): 100-107.
王浩彬, 皇甫伟, 刘娅汐, 等. 基于计算图的移动通信网络物联网业务覆盖优化算法及实现[J]. 物联网学报, 2019, 3(2): 100-107.
- [20] STRATTON J A, STONE S S, HWU W. MCUDA: an efficient implementation of CUDA kernels for multi-core CPUs [C]// 21st International Workshop on Languages and Compilers for Parallel Computing, 2008: 16-30.
- [21] CHEN T, MOREAU T, JIANG Z, et al. TVM: an automated end-to-end optimizing compiler for deep learning [C]// USENIX Symposium on Operating Systems Design and Implementation, 2018: 578-594.

(责任编辑:刘嘉文)