高技术通讯 2023 年 第33 卷 第12 期:1253-1264

doi:10.3772/j.issn.1002-0470.2023.12.003

MACO:基于访存视角的卷积网络自动代码优化①

张晓扬②*** 肖俊敏③* 姚家树*** 谭光明*

(*中国科学院计算技术研究所高性能计算机研究中心 北京 100190)

(**中国科学院大学 北京 100049)

摘 要 推理自动优化一直是人工智能(AI)与系统结构领域交叉的研究重点,但以访存为出发点的自动优化研究方案较少。本文从全局和局部两方面出发,针对数据布局和内核的自动优化问题,以访存的视角对卷积神经网络(CNN)自动代码优化中优化时间成本过高的问题进行研究。为有效分析访存,本文改进了经典的红蓝卵石访存模型的建模方法,提出了新的I/O下界估计方法,降低了多阶段复合算法的下界估计难度,并基于改进后的模型估计了卷积的I/O下界。根据卷积下界估计的结论,本文对数据流进行合理设计,有针对性地优化了自动模板生成技术下巨大的搜索空间,避免了大量无效搜索过程,使内核搜索效率较在未经优化的搜索空间中得到显著加速,并在一般性的卷积参数下较cuDNN有平均2.24倍的性能提升,保证了内核性能。同时本文借助神经网络实现了不同数据布局下的卷积性能预测,R2得分高于传统机器学习模型,且在ResNet-18、AlexNet和VGG-11模型中采用基于数据布局回溯算法和预测模型的混合布局策略较默认布局策略分别有1.28倍、1.32倍和1.29倍的性能提升。

关键词 内存优化;人工智能(AI);推理;数据布局;自动调优

0 引 言

推理优化一直是人工智能(artificial intelligence,AI)与系统结构领域交叉的研究重点。随着智能物联网产业的发展,越来越多的人工智能算法和智能硬件应运而生,如何将海量算法和智能硬件有机结合成为系统性能优化人员关注的问题。为实现应用快速部署迁移的需求,大量研究开始关注自动优化技术。同时,相比于硬件计算性能的高速发展,硬件的内存性能提升较为缓慢,这也直接导致了现有的硬件内存墙问题越来越严重,性能优化目标开始逐步从计算优化过渡到访存优化。

卷积神经网络(convolutional neural network,CNN)的自动优化技术通常包含2个方面:全局计算图优化和算子内核优化。全局计算图优化是针对确定的计算任务以及算子间的数据依赖影响,确定最合适的算子搭配;算子内核优化是针对单个算子,确定特定硬件平台上最合适的算子优化参数。从内存访问的角度看,在全局图优化中存在的一个重要问题就是数据布局的确定。由于具体的算子性能与数据布局有相关性,且不同的算子间存在布局依赖性,所以需要从全局角度确定性能最优的算子数据布局;而在算子内核优化方面,由于自动优化技术多基于机器学习驱动的迭代式搜索方式实现[1],在没有专家经验指导的搜索中会带来很多无效搜索,因此可以从降低访存提高数据重用的角度给予性能优化搜索的方向性指导。

① 国家自然科学基金(62172391,61972377,62032023,T2125013)和北京市科技计划(Z231100007423002)资助项目。

② 男,1993年生,博士生;研究方向:计算机系统结构;E-mail:zhangxiaoyang@ncic.ac.cn。

③ 通信作者,E-mail:xiaojunmin@ict.ac.cn。

(收稿日期:2022-03-18)

针对以上 2 个访存相关的自动优化问题，目前也开展了一些研究工作。Li 等人[2]最先针对图形处理器(graphics processing unit，GPU)上的不同数据布局的性能展开讨论，但其工作主要关注如何快速进行多维数据布局的转换，而不关注应当何时进行数据转换。Zhang 等人[3]提出了针对数字信号处理器(digital signal processor，DSP)进行数据布局转换时机决策的模型，但是基于分类预测模型在一定程度上会造成全局性能的预测偏差。许浩博等人[4]提出面向多任务的神经网络特征挖掘来重用计算以达到性能提升的架构设计，但这一设计存在一定局限性，如在单任务情况下不适用。而 Chen 等人[5]提出的张量虚拟机(tensor virtual machine，TVM)框架，完全利用搜索算法和机器学习模型进行全空间的内核优化搜索。Sabne[6]提出的加速线性代数(accelerated linear algebra，XLA)框架则充分利用了手工优化的经验实现内核级别的优化。吴林阳等人[7]从数据布局、精度等方面提出运算与数据协同的深度学习编译框架。这些框架的自动优化技术能够在一定程度上达到人工优化的效果，提高模型高性能部署的效率，但专家经验知识提取的优化模式限制了自动优化技术的发挥空间，而基于搜索空间的极大优化自由度又带来了难以接受的优化时间开销，如何兼顾两者的优势，成为内核自动优化过程中非常关键的问题。

1　相关背景

1.1　卷积数据布局

卷积在神经网络中是以张量形式存在的，张量是一个多维的数组结构。由于内存是个二维存储结构，所以对于多维数组来说，张量会以特定的顺序进行数据排列，维度的排列先后顺序会决定数据在内存中的组织形式，即对应不同的数据布局。卷积网络中的张量通常包含 4 个维度，即待处理特征图数量 N、特征图通道数 C、特征图宽度 W 和高度 H。这些维度的排列组合带来了不同的数据布局，如 NCHW、NHWC、CHWN 等。如图 1 所示，逻辑视图展示了一个 $N=2$(上下 2 个独立块)、$C=2$(每个独立块叠放的 2 层)、$W=4$、$H=3$ 的四维张量逻辑结构，NCHW 和 CHWN 分别展示了张量逻辑结构在 2 种数据布局下的映射关系(纹理一一对应)。

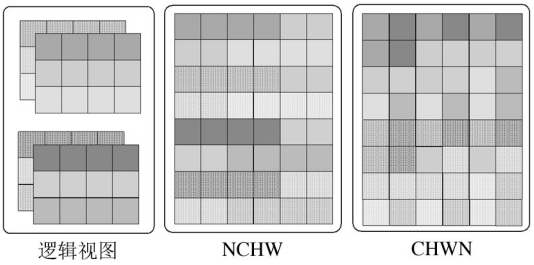


图 1　张量逻辑布局与物理布局关系示意图

1.2　内核自动优化技术

内核自动代码优化技术是指通过对专家经验的归纳总结或者通过合理设计搜索空间，将优化手段通过模式匹配或者搜索方法等方式自动应用到代码优化中，以提高目标代码的性能，实现代码与硬件的深度耦合。传统的代码优化是件劳动量较大的工作，而自动优化技术的出现能够有效降低人力成本，使代码的大规模部署成为可能。自动优化技术是学术界和工业界都一直关注的问题。对于卷积算法而言，常见的解决方案是对特定卷积算法进行模板设计，预留一些可优化的参数轴，通过调优的方式在预留的参数组合空间内来确定合适的参数组合，如 TVM[5]，但模板设计需要一定的专业知识，这提高了开发者的优化门槛，同时也将优化的空间限制到了开发者的经验认知内。所以为了进一步释放自动优化的空间，一些自动模板生成技术应运而生，如 Ansor[8]是通过规则匹配的方式，根据算法的简单描述自动生成多种模板进行内核搜索，如采取不同的循环顺序、选择不同的并行深度等等。图 2 所示的是两类基于搜索的自动优化技术路线的差异。

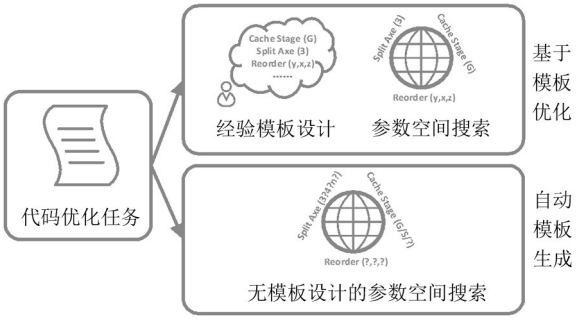


图 2　基于模板的自动优化与自动模板生成技术

1.3 红蓝卵石访存通信模型

红蓝卵石游戏模型是由 Hong 和 Kung[9] 于 1981 年提出的一个两级内存访问模型。这一模型可以对算法访存的下界进行估计,而访存下界可以用于指导算法的数据流设计,进而为算法的内核实现提供优化方向。

红蓝卵石游戏模型所设计的游戏是基于一个描述算法操作过程的有向无环图进行的。令 $G(V, E)$ 为有向无环图的符号表示,其中 V 表示算法操作的顶点集合,E 表示 2 个操作的依赖关系的边集。如果 $G(V, E)$ 满足以下 4 个属性,那么满足这些属性的划分则被称为 S-划分。

属性 1:V 被划分为 h 个子集 $(V_1, V_2, \cdots\cdots, V_h)$,满足 V_i 之间互不相交,但它们的并集是 V。

属性 2:对于每个 V_i,都有一个最多包含 S 个顶点的支配集 D_i。D_i 是 V 的一个节点子集,使得从 G 的输入到 V_i 的节点的任何路径都包含 D_i 中的一些节点。

属性 3:V_i 的最小集被定义为 V_i 中没有任何属于 V_i 的顶点的顶点集。

属性 4:$V_1, V_2, \cdots\cdots, V_h$ 之间没有循环依赖。

下面的定理描述了基于 S-划分模型的通信下界 Q:

$$Q \geqslant S \cdot (P(2S) - 1) \tag{1}$$

2 问题与挑战

2.1 卷积网络数据布局的全局规划问题

数据布局反映的是数据在内存中的排布方式,其差异会影响到数据的局部性特征,进而带来不同的访存读写效率,从而影响卷积的计算效率。而这种数据局部性特征,由于卷积张量的参数维度不同,在不同卷积层上的表现是不同的。图 3 所示的是 AlexNet[10] 对应 5 层卷积在 CUDA(compute unified device architecture)平台上 2 种数据布局下的性能表现,可以看到其中的性能差别[4]。

这些性能差距不仅来源于卷积核本身,也来源于卷积输入数据和输出数据的张量布局。图 4 展示了卷积神经网络中张量计算的关系。

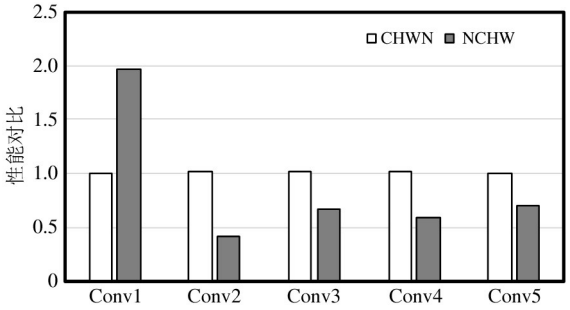


图 3 AlexNet 不同卷积层在不同数据布局下的性能表现

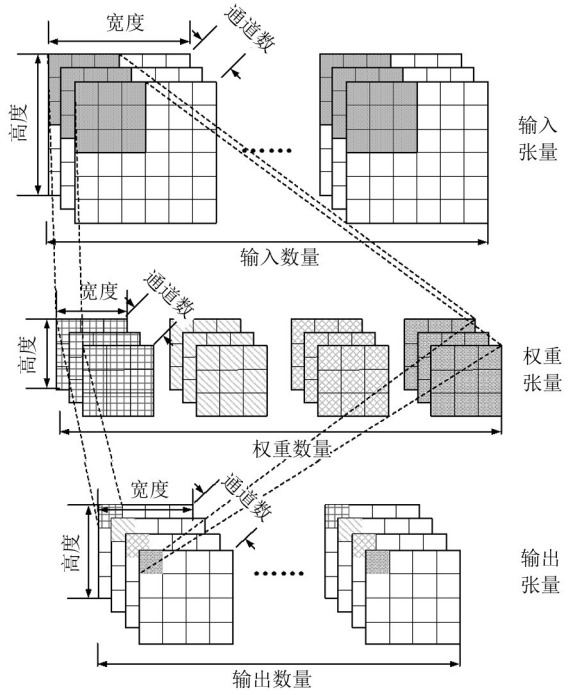


图 4 卷积层的计算过程

在卷积神经网络中,张量以四维结构呈现,虽然计算过程包含了 3 个张量共计 12 个维度的计算关系,但从图中可以发现,并非所有维度都是独立的变量。输出通道取决于卷积核的数量,而权重张量的通道数一般等于输入张量的通道数,输出张量与输入张量有如下对应关系:

输出高度 / 宽度

$$= \left\lceil \frac{\text{输入高度 / 宽度} - \text{权重高度 / 宽度} + 1}{\text{跨步}} \right\rceil$$

其中跨步表示 2 次卷积计算的跨度大小。由于张量是多维数据结构,而内存是二维数据结构,所以张量需要从高维转化为低维将数据存放到内存中,而不同的存放顺序决定了数据计算时的连续性。

由于不同层的数据局部性表现不同,如果相邻

层的最优数据布局不一致,则需要损失某些层的性能,或进行布局转化,但布局转化又会带来额外的时间开销。对于整个网络而言,为了避免过多的数据布局转换开销,需要合理安排各个层的数据布局,以充分挖掘网络的整体性能。目前已有不少研究工作针对 GPU 的数据布局进行研究,如 Majeti 等人[11]探讨了中央处理器(central processing unit,CPU) + GPU 混合架构的布局自动调整问题。Weber 和 Goesele[12]尝试对 CUDA 应用程序的数据布局和内核优化配置进行自适应优化。Zheng 等人[13]应用数据布局优化来提高 GPU 上长短期记忆网络-循环神经网络(long-short-term memory recurrent neural network,LSTM RNN)的性能。但如何从全局角度决策网络层的数据布局,仍然是个有挑战性的问题。因为转换开销的存在,全局的数据布局搭配不再是简单的线性叠加问题。而为了能够合理决策网络层的数据布局,并且将这一数据布局对全局的影响加以刻画,则需要对相应层的性能进行预测,但对于连续域的时间预测问题,精准预测又较困难。

2.2 算子内核的自动优化问题

对于卷积神经网络而言,算子的内核优化主要针对卷积。卷积是卷积神经网络中计算量最大的部分,也是访存行为最复杂的部分。卷积算子优化是根据计算特点和硬件架构,对算子计算代码进行针对性改进以提升算子运算效率的工作。目前的卷积优化多基于人工优化,许多大型硬件厂商为了提高相应算法在自家平台上的性能表现,也提供了平台专用的算子内核集合,如目前主流的 CPU、GPU 硬件厂商如 Nvidia、Intel、ARM、AMD 等,都通过提供适合自身硬件的算子库来支持 AI 网络模型算子内核的加速,同时也有一些硬件平台会有第三方开发者进行支持,如尹宁[14]借助 OpenBLAS 为龙芯平台构建优化了深度学习算子库。但由于这些优化手段的针对性较强,对于新型网络或者不常用的卷积会存在一定的优化短板,表现出长尾效应。图 5 展示了 CUDA 的深度神经网络(CUDA deep neural network,cuDNN)与 TVM 调优结果的性能对比,可以发现 cuDNN[15]库在部分卷积中性能表现较好,而在另外一些卷积下却难以发挥出性能。这主要是因为很多优化手段是针对于常见卷积,而这些优化具有一定的滞后性,所以在算法和模型日新月异的发展趋势下,现有优化库的稳定性会导致软件和硬件之间的差距难以快速被填补。因此,基于平台和模型的内核自动优化技术成为解决这一问题的重要途径。

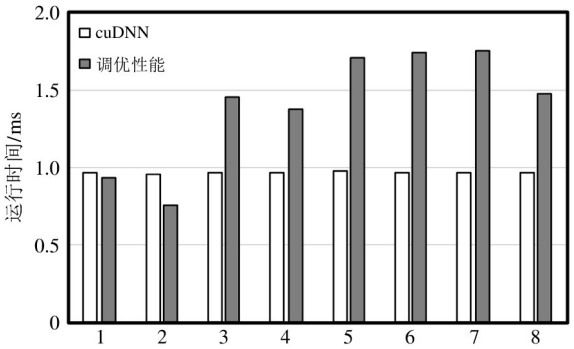


图 5 cuDNN 与调优性能的对比

自动优化技术虽然能够加速填补算法与硬件之间的性能沟壑,摆脱繁重的人力工作,但自动优化技术的应用仍旧存在很大的问题,其中之一就是巨大的参数搜索空间所带来的巨大优化开销。图 6 展示了 AlexNet 在 TVM 框架内的部分卷积参数搜索空间,可以看到搜索空间可达千万级别。而基于搜索的参数优化方法对运行时对数据反馈依赖很大,因此在如此海量的空间内获取一个性能较好的内核实现是非常耗时的。由于这一优化过程包含搜索、编译、测试等环节,在不剪枝搜索空间的情况下,很难通过简单的设备叠加来实现搜索效率的提高。而对于如 Ansor 这类自动模板生成的优化方法而言,TVM 的搜索空间仅仅是其中的一个子集,所以虽然这些方法给了更大的优化空间和优化机会,但同时

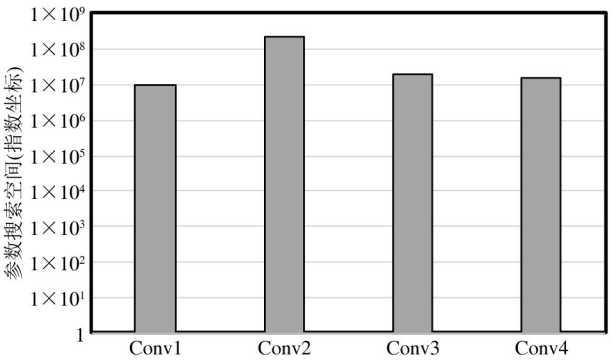


图 6 AlexNet 卷积在 TVM 中的参数搜索空间

也引入了更大的搜索开销,为自动优化技术的推广带来了障碍。

为了既能保留优化机会,同时又可以降低搜索开销,本文基于卷积核复杂的内存读写行为进行建模,并将建模结果应用到自动优化的过程中,以基于访存角度给予搜索空间合理的设计方向,能够有助于在缩减搜索空间的同时,提高搜索效率,并保证搜索的内核性能。

红蓝卵石游戏模型是进行算法访存下界估计的经典模型,利用这一模型能够确定更合适的搜索空间设计方向,有效降低不必要的搜索过程。然而,这一模型进行多阶段算法分析时难度很大。因为一般的红蓝卵石模型只适用于一个单一的计算过程,而对于卷积这类包含多阶段的问题,如果直接应用原始方法,会在不同阶段转换的过程中带来冗余的数据写回和读取操作。因此,在这种情况下,通过简单的数学证明来获取访存的通信下界是难以完成的,所以需要针对卷积算法分析发展更合适的推导方法。

2.3 主要解决的问题

根据上述分析,本文从全局图优化和算子优化角度出发,提出了布局规划和内核优化 2 个方面的研究,具体优化流程如图 7 所示。

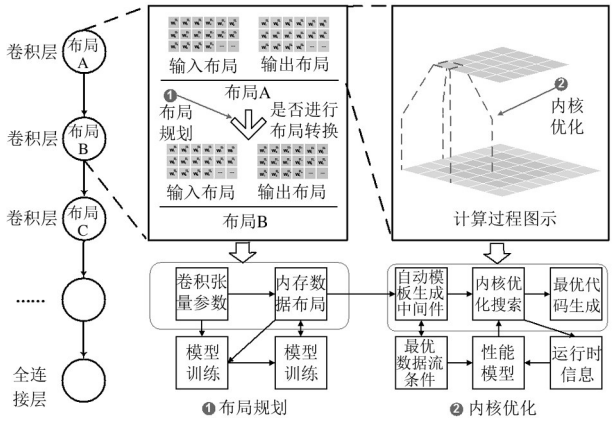


图 7 卷积网络的优化流程

在内核优化方面,本文力求从访存角度减少自动优化的搜索开销,高效地获得最优化的代码;而在布局规划方面,本文旨在通过算法设计和预测模型来获取合理的全局内存数据布局方案,以将自动优化后的内核进行有效的搭配,实现全局性能上的最优。本文的具体研究贡献如下。

(1)在算子内核优化层面,从访存行为特征出发,重新发展了传统的红蓝卵石模型,以适应多阶段算法的通信下界建模。借助提出的建模方法,对卷积的多阶段算法 I/O 下界进行分析,为卷积内核的优化方向提供参照标准。在这一结论的指导下,进行合理的数据流设计,进而从数据读写的角度提高卷积算法计算的数据局部性,提高数据重用率。并将这一数据流设计的条件,应用到自动模板生成技术中,以降低海量搜索空间所带来的巨大开销,为自动优化技术的应用扫清障碍。

(2)在全局图优化层面,针对全局数据布局决策需要性能预测为指导的问题,设计能够进行性能预测的数据布局分析模型,减少实际测试带来的优化开销,提出全局数据布局的规划算法,采用混合数据布局的方式来充分发挥卷积网络各层的性能。

3 卷积内核生成方法优化

对于卷积内核的自动优化,本文旨在进行合理的优化空间设计,并基于最先进的自动模板生成技术,从访存角度改进模板生成时的优化搜索空间。首先需要解决的问题是如何获得一个科学的访存最优下界;然后基于访存最优下界进行数据流的设计;最后根据数据流合理设计优化搜索空间。

红蓝卵石模型是建立最优访存下界的经典模型,但传统的红蓝卵石模型更适合于单阶段算法。对于卷积这类多阶段复合算法而言,直接推断 $P(S)$ 是十分困难的,因此需要对这一方法加以改进。本文采取对 $P(S)$ 的下界进行估计的方法,对于一个有向无环图 $G(V,E)$,这里使用符号$|\cdot|$来表示任意集合的顶点数目,例如$|V|$表示集合 V 中的顶点数量。令$\mathbb{P}_S$为包含有向无环图 $G(V, E)$ S-划分所有可能性的集合,并且每个$\mathbb{P}_S$中的元素表示一些 $G(V, E)$ 的 S-划分。令

$$H(S) = \min_{\{V_1,\cdots,V_h\}\in\mathbb{P}_S} \frac{|V|}{\max_{1\leqslant i\leqslant h}|V_i|} \qquad (2)$$

$H(S)$ 表示 $G(V,E)$ 任一 S-划分集合数量的下界。根据 $P(S)$ 的定义,结合式(1)和(2),Q 可表示为

$$Q \geqslant S \cdot (P(2S) - 1) \geqslant S \cdot (H(2S) - 1) \tag{3}$$

因此,只需估计 $H(2S)$ 而非 $P(2S)$。由于 $H(S)$ 依赖于 $\max\limits_{1\leqslant i\leqslant h} | V_i |$ 的值,所以对 V_i 的细粒度分析是关键。

如果能找出 V_i 与 $G(V,E)$ 所有子计算的关系,那么估计 V_i 的顶点数量成为可能。在推断 V_i 上界之前,首先记 $G_j(U_j, E_j)$ 为 $G(V,E)$ 一个多阶段划分中任一子图对应的子计算,那么该子计算的输入顶点必须是 $G_{j-1}(U_{j-1}, E_{j-1})$ 输出顶点,且 U_j 内部顶点集互不相交。

现在的目标是估计 $| V_i |$ 的最大值。为了方便表示,记 U_j 的输出集合为 $\overline{O}_j$,V_i 的支配集记作 D_i。此处定义最大顶点生成函数。对于给定的任意整数 k 和第 j 个子计算,定义的最大顶点生成函数如下:

$$\varphi_j(k) = \max_U | \Theta(D) \cap U \cap U_j | \tag{4}$$

$$\psi_j(k) = \max_U | \Theta(D) \cap U \cap \widetilde{O}_j | \tag{5}$$

其中,$| \Theta(D) \cap U \cap U_j |$ 和 $| \Theta(D) \cap U \cap \widetilde{O}_j |$ 分别表示由支配集 D 在 2 个顶点集合 $U \cap U_j$ 和 $U \cap \widetilde{O}_j$ 中生成的顶点数量。显然,φ_j 和 ψ_j 提供了 U_j 和 $\widetilde{O}_j$ 中的顶点数量上界估计,这些顶点是由满足 $| D \cap U_j | + | \Theta(D) \cap \widetilde{O}_{j-1} | \leqslant k$ 条件的 D 生成的。

此外,通过最大生成函数 φ_j 和 ψ_j 可获得强有力的分析方法来估计 $V_i \cap U_j$ 和 $V_i \cap \widetilde{O}_j$ 中 D_i 生成的顶点数。基于此,对任意 S-划分 $G(V,E)$ 的 $| V_i |$ 有一个上界:

$$\begin{aligned} T(S) = S + \max_{\sum_{j=1}^{n} k_j \leqslant S} & (\varphi_1(k_1) + \varphi_2(k_2 + \psi_1(k_1)) + \\ & \cdots\cdots + \varphi_n(k_n + \psi_{n-1}(k_{n-1} + \psi_{n-2}(k_{n-2}\cdots \\ & + \psi_1(k_1))))) \end{aligned} \tag{6}$$

假设有向无环图 $G(V, E)$ 描述了一个 n 阶段算法。所有子计算都对应于一个有向无环图的多阶段划分。给定大小为 S 的快速内存,为了实现整个算法的计算,在快速内存和慢速内存之间的 I/O 操作数 Q 满足:

$$Q \geqslant S \cdot \left(\frac{| V |}{T(2S)} - 1 \right) \tag{7}$$

虽然式(7)和(1)是相似的,但是对多阶段算法而言估计 T 要比得到 P 更容易。

基于以上的多阶段算法分析方法,下面以直接卷积为例建立 I/O 下界。

图 8 展示了直接卷积的有向无环图 $G(V,E)$。直接卷积包含 2 个阶段:通过输入图片和卷积核的输入生成大量的乘积项;将乘积项相加形成一个基于求和树的最终输出。求和树是一个树形结构的有向无环图子图,除了输入顶点外,该树的其他顶点的入度最多为 2,并且树的所有输入都会被累加到一起并且只有一个输出。在求和过程之后就完成了直接卷积。因此,$G(V,E)$ 的多阶段划分可以被记作 $G(V,E) = G_1(U_1,E_1) \cup G_1(U_1,E_1)$,对于直接卷积的有向无环图,其顶点总数为

$$\begin{aligned} | V | = & (2W_{\text{ker}}H_{\text{ker}}C_{\text{in}} - 1)W_{\text{out}}H_{\text{out}}C_{\text{out}} + W_{\text{in}}H_{\text{in}}C_{\text{in}} \\ & + W_{\text{ker}}H_{\text{ker}}C_{\text{in}}C_{\text{out}} \end{aligned}$$

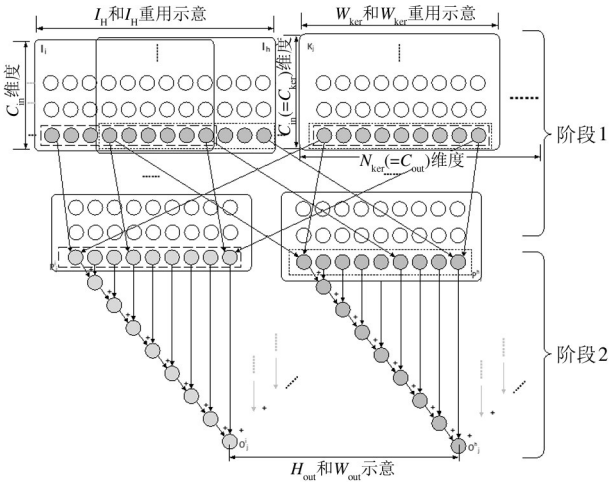


图 8 直接卷积的有向无环图

对于不同的滑动窗口,在一个输入图像中每个输入(元素)的最大重用数记作 R,它的值是 $R = \frac{W_{\text{ker}}H_{\text{ker}}}{\mu^2}$,$\mu$ 是跨步的大小。对于直接卷积这一两阶段的算法,可以证明 $\psi_1 = \varphi_1$,$\varphi_1(k_1) \leqslant 2S\sqrt{Rk_1}$,并且 $\varphi_2(k_2) \leqslant k_2 - 1$ 对于任何整数 k_1 和 k_2 都是可用的,这可以用于估计 $T(S)$。

对于直接卷积,$T(S) \leqslant 4S\sqrt{RS} + S - 1$,建立 I/O 直接卷积的下界为

$$Q_{\text{lower}}DC = \Omega\left(\frac{W_{\text{ker}}H_{\text{ker}}C_{\text{in}}W_{\text{out}}H_{\text{out}}C_{\text{out}}}{4\sqrt{2RS}} \right) \tag{8}$$

借助等式(8)中的通信下界可以建立近似 I/O 最优策略方法。在提出的一般 I/O 下界理论中,根

据式(6)可知,I/O 下界结果的最高阶项必定由某个 φ_j 决定。以直接卷积为例, φ_2 决定了 I/O 下界的最高阶项,也就是 φ_2 指向了涉及 I/O 操作最多的过程。这说明需要针对直接卷积的第 2 阶段进行优化。φ_2 指明应该用求和树上最少的输入得到最多的输出,即充分利用求和树的输入。而求和树上的输入来源于第 1 阶段的乘积顶点,也就是数据流设计应该将有效片上内存尽量分配给这些乘积顶点。由于直接卷积沿通道方向的数据只计算一次,所以不需要在通道方向上分配更多额外空间,只需要让输入图像的宽、高和卷积核数量维度上能够有更高的相互复用率,对应于最终输出就是其宽、高、通道维度。

图 9 展示了一个大小为 $x \times y \times z$ 的输出图像的子块(输出张量深色部分)。为了达到最小的片外内存访问,本文倾向于令 xyz 的结果逼近 S/N_p, 其中 N_p 是处理核心的数量。为了计算出子块 $x \times y \times z$ 的结果,需要对应于 $x' \times y'$ 位置的所有通道维度的输入以及与部分输出通道相关联的 z 个卷积核,如图 9 所示。由于片上内存受限而且需要用于存储尽可能多的输出结果,所以在加载图片输入和卷积核的时候,需要连续加载而不是一次性加载到快速内存中。在每个阶段,输入图片 $x' \times y' \times \alpha$ 的一部分和相应的 z 个卷积核权重 $H_{\text{ker}} \times W_{\text{ker}} \times \alpha$ 被加载到片上内存中。由于输入的第 i 通道仅能够被权重的第 i 层通道重用,所以为了保证输出子块在有限的片上内存中能够尽可能大,令 $\alpha = 1$, 也就是说数据流设计要求以固定的通道数加载 $x' \times y'$ 贴片并沿着通道维度的方向滑动。将 $x' \times y'$ 的输入贴片和相应的 z 卷积核权重加载到片上内存后,可以对输出子块执行部分和的计算。

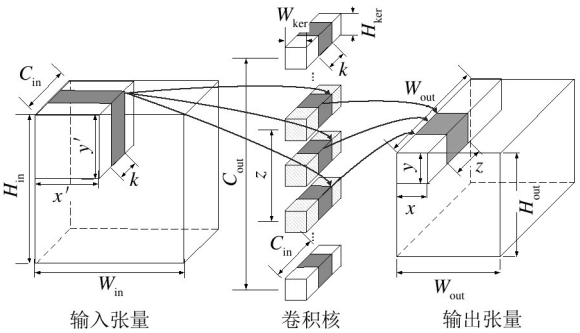


图 9　直接卷积数据流

为了更新整个输出子块,需沿通道方向连续滑动 $x' \times y'$ 输入块,并加载相应的输入和权重,执行部分更新。因此,更新每个输出子块只需要从片外内存加载所需的输入和权重到片上内存一次。同时,不同子块由 N_p 处理器并行地处理。在数据流设计中,共有 $(W_{\text{out}}H_{\text{out}}C_{\text{out}})/(xyz)$ 个输出子块。为更新每个子块,需从输入图像中输入 $x'y'C_{\text{in}}$ 个元素,从 z 个卷积核中输入 $W_{\text{ker}}H_{\text{ker}}C_{\text{in}}z$ 个元素。由于 $R = W_{\text{ker}}H_{\text{ker}}/\mu^2$, $x' \approx \mu x$, $y' \approx \mu y$, 数据读入 I/O 为

$$Q_{DC\ \text{reading}} \approx \frac{H_{\text{out}}W_{\text{out}}C_{\text{out}}}{xyz}\left(H_{\text{ker}}W_{\text{ker}}C_{\text{in}}\left(z + \frac{xy}{R}\right)\right)$$

$$\geqslant H_{\text{out}}W_{\text{out}}C_{\text{out}}H_{\text{ker}}W_{\text{ker}}C_{\text{in}}\left(2\sqrt{\frac{1}{Rxyz}}\right)$$

其中,当且仅当 $xy = Rz$ 时等式成立。基于 $R = W_{\text{ker}}H_{\text{ker}}/\mu^2$, $x' \approx \mu x$, $y' \approx \mu y$, $xy = Rz$ 条件的要求使得 $x'y' = zW_{\text{ker}}H_{\text{ker}}$。它决定了每个 $x' \times y'$ 贴片的最优尺寸。此外,存储输出 I/O 是 $H_{\text{out}}W_{\text{out}}C_{\text{out}}$。当 $xyz \approx S/N_p$ 且 $xy = Rz$ 时,I/O 的总量为

$$Q_{DC} \approx \frac{2H_{\text{out}}W_{\text{out}}C_{\text{out}}H_{\text{ker}}W_{\text{ker}}C_{\text{in}}}{\sqrt{RS/N_p}} + H_{\text{out}}W_{\text{out}}C_{\text{out}}$$

如果 $N_p = 1$ 且 $\dfrac{H_{\text{ker}}W_{\text{ker}}C_{\text{in}}}{\sqrt{SR}} >> 1$, 则数据流能够达到 I/O 下界。因为 S 在卷积网络的应用中一般不大于 kB 级别,所以很容易满足条件。

基于以上数据流设计,本文用科学的指导方法进行搜索空间合理剪枝。根据数据流给出的最优性条件,将其应用于自动模板生成技术中。通过将最优性条件应用到模板调度设计的过程中,模板的自动生成被限制到科学范围内,从而降低后续搜索带来的时间开销和硬件开销,而且也保证了在此空间内性能参数搭配的合理性。具体过程如图 10 所示。

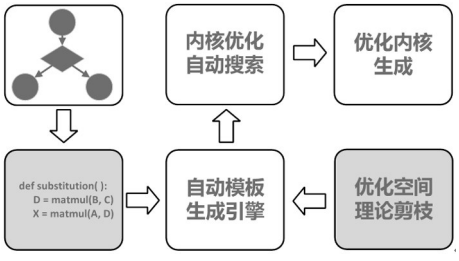


图 10　最优性条件的模板设计剪枝优化

4 全局卷积数据布局优化

为了挖掘网络的全局性能,发挥出自动调优后的卷积性能,需要确定每层卷积最合理的数据布局,因层制宜,所以本文采取混合张量布局策略,允许不同卷积层选择不同的张量布局。但由于不同的张量布局之间存在转换开销,因此需要从全局层面来决策整个张量布局的搭配。由于卷积网络通常较深,所以其对应的搭配空间较大,在自动进行布局规划的过程中,需要尽可能全面且迅速地判断规划方案的性能,进而做出最优决策。张量布局的规划算法过程如算法 1 所示。

算法 1 混合张量布局策略

输入:全局张量维度信息 T(第 i 层信息表示为 T_i),可选内存数据布局 L(第 i 种内存数据布局表示为 L_i);

输出:全局张量布局选择结果 *Ans*

```
Ans = [ ], Res = [ ]//存储最终输出结果和临时结果
Consume = INT_MAX, ConsumeNow = 0 // 总耗时和当前耗时初始化
LayoutChoice(Res, L) //调用回溯函数
return Ans//返回最终数据布局方法
def LayoutChoice(Res, L): //回溯函数
ConsumeNow += Time(T_i, L_i) //在 T_i 层的参数下选择 L_i 后的当前耗时
if (Res[-1].format != L_i): //相邻层布局不同
ConsumeNow += Trans(Res[-1] → L_i) //加入转换开销
if (ConsumeNow >= Consume): //超时退出
return
else if (ConsumeNow < Consume): //未超时
If (Res.size() == T.size()): //遍历完毕保存方案
Consume = ConsumeNow// 总耗时
Ans = Res// 方案结果
return
for L_i in L: //遍历方案
Res.add(L_i)
LayoutChoice(Res, L_i)
Res.remove(L_i)
ConsumeNow- = Time(T_i, L_i) //恢复回溯前的耗时
```

可以发现,算法 1 可以全面地考虑不同的情况,并及早对不合适的规划设计采取终止搜索行为。而效率方面,在于当确定相应规划策略时所获得的性能表现,也就是解决各层的性能预测问题。获得张量布局的性能最自然的方法是对设备相关的所有内存访问操作进行建模,并与不同张量布局的量化时间进行比较。但这种内存访问模型的建模难度大,同时由于计算行为还受到运行时其他因素的影响,所以准确性难以保证。因此,在实际情况中对所有操作进行建模是难以实现的。

为了更加准确地对不同层的合理数据进行标记,需要采取更加具备泛化能力的方法来获取张量布局的确定标准。同时,由于是全局优化问题,在很多设备上需要考虑张量布局的转化开销,因此无法简单地选择各层最好的张量布局,这就要求标准的获得以数值预测为好。

采用机器学习模型进行建模不失为一种泛化能力较强的方法。但机器学习的方法众多,且机器学习方法难以在数值预测上达到良好的性能。因此,本文选择采用神经网络架构搜索的方法进行模型的构建,以获得高精度的预测模型。

本文模型是基于自动优化后的内核数据训练而来的,具体的输入参数包括数据布局的编号、输入张量维度、权重张量维度、输出张量维度、计算跨步等等,以 R2 得分为评估目标,进行回归拟合。在使用时,当获取到相关的卷积层参数后可在模型的帮助下,获得不同数据布局编号下的性能预测结果。

本文的策略包含预处理和网络优化 2 个阶段。预处理阶段提取卷积网络中各层的配置信息,并基于算法 1 进行布局方案搜索,搜索过程中配合性能预测方法进行决策,将获得的结果返回给卷积网络进行合理的张量布局标记。部署卷积网络的设备根据标记将权重张量提前进行转换,并将流式数据按照相应层提前做好布局转换输出,如图 11 所示。

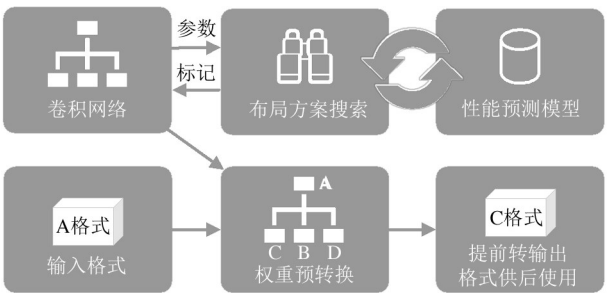


图 11 数据布局优化流程

5 实 验

5.1 数据布局模型测试

实验发现,相比于采用默认布局,采取模型决策的布局能够显著减少推理用时,ResNet-18、AlexNet、VGG-11 的性能在采取模型决策后分别提升了 1.28 倍、1.32 倍、1.29 倍,如图 12 所示。

通过对不同算法的测试发现,在相同的训练数据集下,支持向量机(support vector machine,SVM)回归模型[16]、线性回归模型[17]、梯度提升回归树(gradient boosted regression trees,GBRT)回归模型[18]对不同数据布局下卷积进行性能预测的 R2 得分分别为 0.162 584、0.312 771、0.897 487,而借助 autoML[19]技术构建的模型 R2 得分为 0.998 412,如图 13 所示。

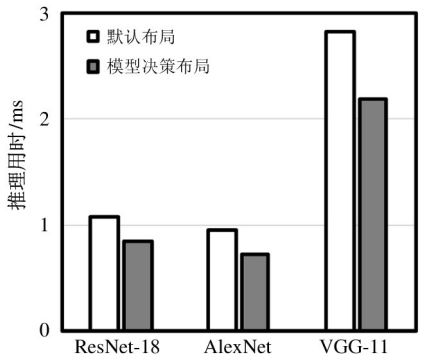


图 12 默认布局与模型决策布局的性能对比

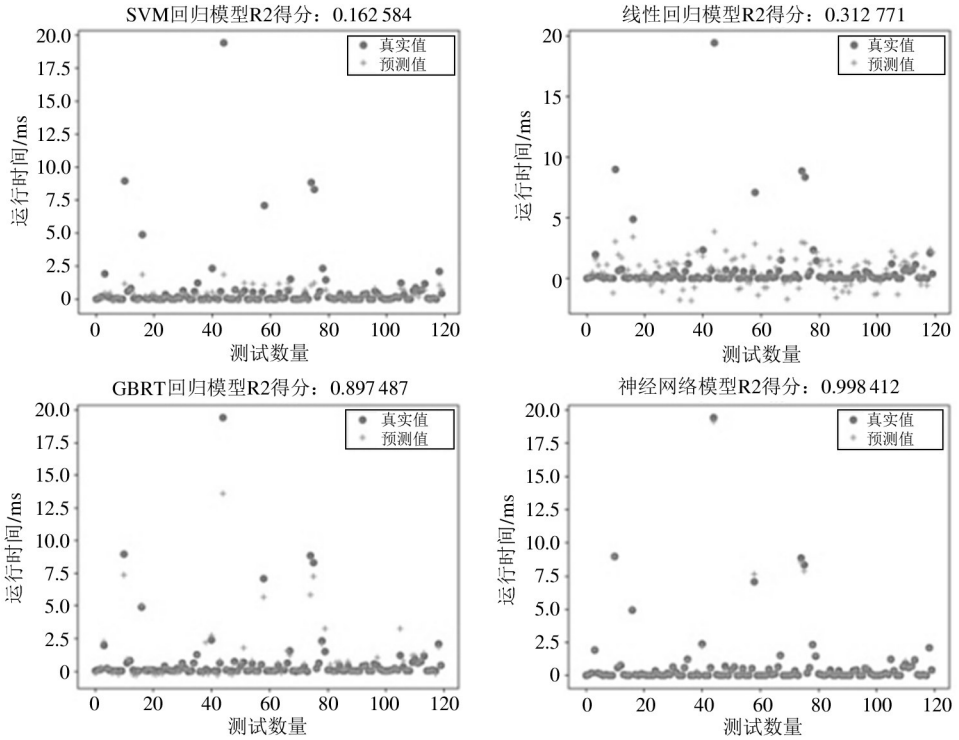


图 13 不同模型的 R2 得分情况

众所周知,R2 得分的取值范围为[-1,1],取值越接近 1 表示模型的预测表现越接近最优[20]。而在实际统计测试中也可以发现,通过神经网络模型预测的时间结果与实际的性能差距并不大。如图 14所示,在与表现最好的 GBRT 算法进行对比中发现,预测的阈值(即发生预测错误时,选定布局与最佳布局的性能差距)较大(即大于 0.01 ms)时,本文的方法分布比例仅占 2.2%,且最大性能预测差距不超过 0.05 ms,而 GBRT 在预测阈值较大时的分布比例占 5.6%,且约 2.8% 的性能预测差距超过 0.05 ms。这说明GBRT方法的波动性大。由于布局

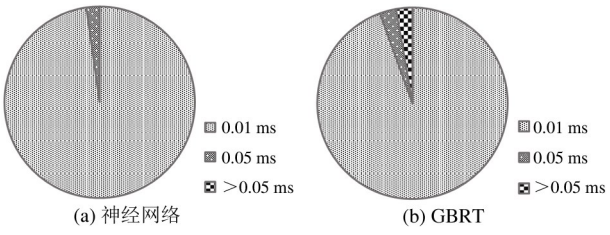


图 14 相应预测阈值下的分布比例

转换开销的存在,当性能预测的差距较小时对全局的性能影响不大,而在性能预测差距较大时产生的影响会显著提高。因为在全局数据布局预测时,模型预测的稳定性比较重要,所以应当尽量避免较大性能预测差距。

5.2　内核性能测试

图 15 展示了自动优化卷积相较于 cuDNN 的性能提升,可以看出,无论是 NCHW 还是 NHWC 布局的卷积,其性能较原始相比都有显著提升,整体加速比为 2.24 倍。其中,当卷积宽度和高度较小时,在单批次的情况下性能提升更加明显,随着卷积宽度和高度的增大,加速效果开始下降。同样地,当卷积的输出通道数增加时,性能的提升效果开始下降。这说明随着卷积的增大,设备更容易发挥出计算性能,原始的优化难度降低。

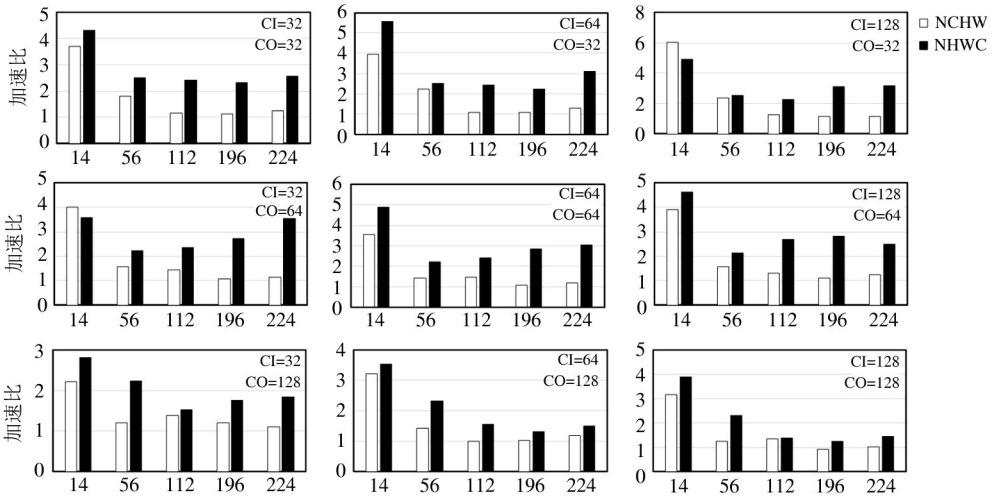


图 15　自动优化卷积与 cuDNN 卷积在 NCHW 和 NHWC 布局下相比的性能提升

同时,由于搜索空间在经过数据流重新设计后获得了剪枝,避免了大量无效的搜索过程,因此本文的方法相比于原始的搜索空间,搜索效率得到了提升。尤其在搜索的初始阶段,性能提升较快,说明本文的方法搜索起点更好。随着搜索过程的进行,本文的方法能够更快地找到一个性能更好的内核优化方案,如图 16 所示。

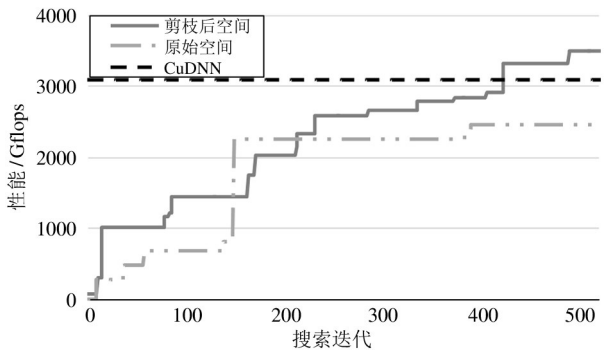
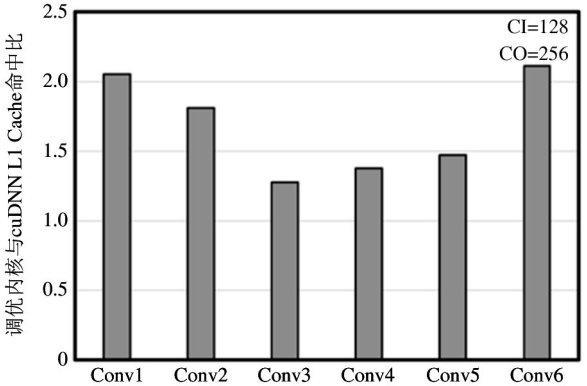


图 16　搜索空间剪枝前后的搜索效果对比

另外,图 17 展示了其中一些卷积的 L1 Cache 命中率测试结果。观察发现,经过调优后的内核 L1 Cache 命中率要显著高于 cuDNN 卷积内核,这从侧面印证了调优后的内核在数据重用方面的表现要优于 cuDNN。



	H	*W*	Stride
Conv1	14	14	1
Conv2	56	56	1
Conv3	112	112	1
Conv4	14	14	2
Conv5	56	56	2
Conv6	112	112	2

图 17　调优内核与 cuDNN 内核的 L1 Cache 命中比

5.3 端到端性能测试

图 18 展示了端到端的性能表现。当采取模型决策的混合数据布局后,获得的网络模型性能显著高于基准,ResNet-18、AlexNet、VGG-11 的性能分别提升了 1.07 倍、1.62 倍、1.34 倍,说明这些自动优化方法在端到端的性能提升上发挥了作用。

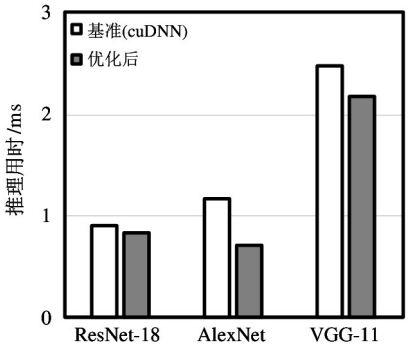


图 18 端到端性能测试

6 结 论

本文改进了经典的红蓝卵石访存模型的建模方法,通过提出新的访存下界估计方法,降低了多阶段复合算法的建模难度,使卷积 I/O 下界的推导成为可能。获得卷积 I/O 下界后,通过对卷积数据流的合理设计,搜索空间得到了优化剪枝,并将其应用于最新的自动模板生成方法的搜索空间设计中,通过避免大量无效搜索来降低搜索时间的成本开销。而对于全局的数据布局确定问题,通过神经网络架构搜索技术设计的新型神经网络,配合数据布局回溯算法,能够实现对卷积数据布局的预测和搭配。实验证明,本文所使用的神经网络能够有效预测出不同卷积的性能差异,并提供一个较为合理的数据布局搭配方法,较默认的数据布局在多个网络上性能均有 1.3 倍左右的提升。同时,经过对红蓝卵石模型的改进,本文的方法成功获得了多阶段算法的访存下界,并且经过访存分析后重新设计的数据流,能够有效降低自动模板生成技术下的内核搜索空间,降低搜索成本。在此基础上,本文获得的内核性能得到了保证,其较 cuDNN 有 2.24 倍的加速,同时搜索效率获得有效提升。

未来的工作主要包括两方面:一方面是全局方面引入更多的优化维度,除卷积布局外,还可在图结构、算法等方面进行性能预测,以带来更大的全局优化空间;另一方面是局部优化上引入对算子融合等情况下的空间设计以及更加简单的约束域推导方式。

参考文献

[1] 池昊宇, 陈长波. 基于机器学习的编译器自动调优综述[J]. 计算机科学, 2022,49(1):241-251.

[2] LI C, YANG Y, FENG M, et al. Optimizing memory efficiency for deep convolutional neural networks on GPUs [C]// Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Salt Lake City: IEEE, 2016:633-644.

[3] ZHANG X, XIAO J, ZHANG X, et al. Tensor layout optimization of convolution for inference on digital signal processor[C]//2019 IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom). Xiamen: IEEE, 2019: 184-193.

[4] 许浩博, 王颖, 王郁杰,等. 面向多任务处理的神经网络加速器设计[J]. 高技术通讯, 2021,31(5):457-463.

[5] CHEN T, MOREAU T, JIANG Z, et al. TVM: an automated end-to-end optimizing compiler for deep learning [C]// The 13th USENIX Symposium on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2018:578-594.

[6] SABNE A. XLA: compiling machine learning for peak performance [EB/OL]. [2022-03-18]. https://research.google/pubs/pub50530/.

[7] 吴林阳, 杜伟健, 陈小兵, 等. 一种运算和数据协同优化的深度学习编译框架[J]. 高技术通讯, 2020,30(2):120-125.

[8] ZHENG L, JIA C, SUN M, et al. Ansor: generating high-performance tensor programs for deep learning[C]// The 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). Virtual Event: USENIX Association, 2020:863-879.

[9] HONG J W, KUNG H T. I/O complexity: the red-blue pebble game[C]// Proceedings of the 13th Annual ACM Symposium on Theory of Computing. Milwaukee: Association for Computing Machinery, 1981:326-333.

[10] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Ima-

geNet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017,60(6): 84-90.

[11] MAJETI D, MEEL K S, BARIK R, et al. Automatic data layout generation and kernel mapping for CPU + GPU architectures[C]// Proceedings of the 25th International Conference on Compiler Construction. Barcelona: Association for Computing Machinery, 2016:240-250.

[12] WEBER N, GOESELE M. Adaptive GPU array layout auto-tuning[C]// Proceedings of the ACM Workshop on Software Engineering Methods for Parallel and High Performance Applications. New York: Association for Computing Machinery, 2016:21-28.

[13] ZHENG B, NAIR A, WU Q, et al. EcoRNN: fused LSTM RNN implementation with data layout optimization [EB/OL]. [2022-03-18]. https://arxiv.org/pdf/1805.08899.pdf.

[14] 尹宁. 基于龙芯平台的深度学习算子库的构建与优化[D]. 合肥:安徽大学, 2021:1-51.

[15] CHETLUR S, WOOLLEY C, VANDERMERSCH P, et al. cuDNN: efficient primitives for deep learning[J]. [2022-03-18]. https://arxiv.org/pdf/1410.0759.pdf.

[16] 孟田华, 卢玉和, 丁少军,等. 基于 SVM 的温度预测回归模型[J]. 现代计算机, 2020(20):3-6,13.

[17] KRÄMER W, SONNBERGER H. The linear regression model under test[M]. New York: Physica-Verlag HD, 2012:1-190.

[18] PRETTENHOFER P, LOUPPE G. Gradient boosted regression trees in scikit-learn[EB/OL]. [2022-03-16]. https://hdl.handle.net/2268/163521.

[19] ZIMMER L, LINDAUER M, HUTTER F. Auto-Pytorch: multi-fidelity meta learning for efficient and robust autoDL [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021,43(9):3079-3090.

[20] LEE K W, WON Y J, SONG Y B, et al. Development of machine learning ensemble model using artificial intelligence[J]. Journal of the Korean Society for Heat Treatment, 2021,34(5):211-217.

MACO: memory-based automatic code optimization of CNNs

ZHANG Xiaoyang[*,**], XIAO Junmin[*], YAO Jiashu[*,**], TAN Guangming[*]

([*]High Performance Computer Research Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**]University of Chinese Academy of Sciences, Beijing 100049)

Abstract

Inference automatic optimization has been the focus of research at the intersection of artificial intelligence (AI) and system architecture fields. However, there are fewer optimization research schemes based on memory. In this paper, the high time cost of automatic optimization of convolutional neural networks (CNN) data layout and kernel is studied and discussed from the perspective of memory from both global and local aspects. To perform the access analysis efficiently, the classical red-blue pebble game is re-explored and a new method is proposed to estimation I/O lower bound which reduces the difficulty of lower bound estimation for multi-stage composite algorithms. This work analyses the convolutional I/O lower bound based on the improved model and re-designs the data flow with the estimated results. This work purposefully optimizes the huge search space under the auto-template generation technique to avoid a large number of invalid search processes, so that the kernel search efficiency is significantly accelerated compared with the unoptimized search space, and the performance is improved by an average of 2.24 × compared with cuDNN with general convolutional parameters, which ensures the kernel performance. This work also implements the convolutional performance prediction under different data layouts with the help of neural networks, and the R2 score is higher than that of traditional machine learning models. The performance of the hybrid layout strategy based on data layout backtracking algorithm and prediction model has 1.28 ×, 1.32 ×, and 1.29 × improvement over the default layout strategy in ResNet-18, AlexNet, and VGG-11 models, respectively.

Key words: memory optimization, artificial intelligence (AI), inference, data layout, auto-tuning