doi:10.3772/j.issn.1002-0470.2020.02.002

一种运算和数据协同优化的深度学习编译框架①

吴林阳②*,**,*** 杜伟健*,**,*** 陈小兵*,**,*** 庄毅敏*,**,***

(*中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190)

(**中国科学院大学 北京 100049)

(***上海寒武纪信息科技有限公司 上海 201308)

摘 要 近年来,深度学习算法和深度学习处理器已被广泛应用于工业界,如何从软件层面充分挖掘深度学习处理器的性能成为目前编译器领域研究的热点和难点。现有的深度学习编译框架更侧重于对程序的运算部分进行优化,对数据的优化非常有限,这并不能发挥深度学习处理器的峰值性能。本文分析了深度学习算法和硬件平台的特点,提出一种运算和数据协同优化的深度学习编译框架 CDUCA,它包含计算图引擎、代码生成器、数据优化器3个不同层次的组件,在多个层次对运算和数据进行协同优化,最终生成高效的可部署模型。本文在现场可编程门阵列(FPGA)平台上评估了 CDUCA,实验结果表明,对于典型的深度学习应用,CDUCA 生成的模型性能能达到手工优化模型性能的86.5%。

关键词 深度学习;深度学习处理器;编译器;编译优化

0 引 言

随着大数据时代到来,深度学习算法已被广泛应用于安防、教育、医疗、贸易、交通、家居等各个领域。然而,深度学习应用对硬件计算能力要求极高,通用处理器已经无法满足应用的实时性需求。在此背景下,深度学习处理器蓬勃发展,逐渐成为处理深度学习应用的主要载体。

深度学习算法和硬件的快速发展给编译器设计带来了巨大的挑战,主要体现在2个方面:

(1) 通用性。如何设计统一的编译框架,支持不同的硬件平台后端(GPU、TPU、CPU、加速器)。

(2) 性能。针对特定的硬件平台后端,如何把性能(延时、吞吐率)优化到最好。

目前已有一些相关工作针对以上2个问题提出了解决方案。比如,Chen 等人[1]提出了一种自动化、端到端的编译优化框架 TVM。为了优化程序的性能,TVM 设计了面向算法的高级优化和面向硬件的低级优化两层优化架构,并采用了操作融合、访存延迟隐藏等关键技术。TVM 通过中间表示来支持不同的硬件后端,同时在语言层面支持用户自定义硬件优化原语,从而解决通用性问题。又比如,Vasilache 等人[2]提出了一种领域专用语言和编译框架 TC(tensor comprehensions)。为了优化程序性能,TC 在总体设计上采用了多面体编译框架(优化循环结构),并在其中集成了如存储层次提升、映射到寄存器、向量化、并行化等面向硬件结构的优化技术。相关工作还有 DLVM[3]、Latte[4]等。

然而,上述相关工作在考虑性能优化时,更侧重

① 国家重点研发计划(2017YFA0700900,2017YFA0700902,2017YFA0700901,2017YFB1003101),国家自然科学基金(61472396,61432016,61473275,61522211,61532016,61521092,61502446,61672491,61602441,61602446,61732002,61702478,61732020),北京市自然科学基金(JQ18013),973 计划(2015CB358800),"核心电子器件、高端通用芯片及基础软件产品"科技重大专项(2018ZX01031102),中国科学院科技成果转移转化重点专项(KFJ-HGZX-013),中国科学院战略性先导科技专项(B类)(XDB32050200)以及军队医学(AWS17J011)资助项目。

② 男,1991 年生,博士生;研究方向:智能处理器编程软件,计算机系统结构;联系人,E-mail: wulinyang@ict.ac.cn

(收稿日期:2019-03-07)

于对程序的运算部分进行优化,对数据的优化非常有限,比如 TVM 只在高级优化层面对数据布局进行转换。本文基于以下 2 点观察:

(1) 深度学习处理器往往需要支持可变大小的向量乘矩阵、矩阵乘矩阵操作,为了配合指令集和硬件结构进行访存、运算优化,数据往往需要根据不同的操作以及不同的运算规模进行不同的变换。

(2) 深度学习算法在推理过程中存在大量的常量数据(权值、偏置等),可以采用部分求值的思想在编译阶段对运算、数据进行提前优化。

提出一种运算和数据协同优化的深度学习编译框架 CDUCA(computation and data unified compile architecture),以解决深度学习性能优化问题。本文研究的内容不涉及如何解决通用性问题。

本文剩余章节内容组织如下。第 1 节介绍 CDUCA 的总体设计。第 2 节介绍 CDUCA 各个模块以及优化技术。第 3 节对 CDUCA 生成模型的性能进行了多个维度的实验评估。第 4 节对本文的工作进行总结和展望。

1 总体设计

深度学习算法的主体是神经网络。神经网络由一系列的基本操作按一定的拓扑结构连接组成,每个操作包含一组或多组输入、输出神经元,神经元可以在操作之间共享。进一步,神经网络可以用计算图来表示,图节点表示操作,边表示神经元。训练好的模型数据也是神经网络的重要组成部分,比如卷积操作的权值。

CDUCA 编译框架接收原始的神经网络计算图和模型数据,输出可部署到深度学习处理器上执行的模型文件,其总体框架如图 1 所示。

CDUCA 包含计算图引擎、代码生成器、数据优化器 3 个不同层次的组件/模块。

(1) 计算图引擎 采用线性变换、常量折叠等优化技术,对原始的计算图和模型数据进行面向算法的高级优化,生成优化后的计算图和模型数据。

(2) 代码生成器 采用基于 cost model 的启发式搜索策略,对计算图引擎优化后的计算图和模型数据进行运算和数据协同编译优化,生成高效的目标平台代码。

(3) 数据优化器 采用数据对齐、维度变换、精度调优等优化技术,对计算图引擎优化后的模型数据进行面向硬件平台的二次优化,生成可部署模型。

图 1 CDUCA 总体架构图

此外,得益于层次化的设计结构,CDUCA 编译框架具有很强的扩展性,它能够方便地集成各种编译优化技术。比如,CDUCA 可以在计算图引擎模块集成操作聚合技术,可以在代码生成器模块集成多面体循环优化技术。如何扩展这些优化技术不作为本文讨论的重点。

2 CDUCA 模块

本小节介绍 CDUCA 各个模块以及优化技术。

2.1 计算图引擎

计算图引擎采用线性变换和常量折叠技术来对运算和数据进行协同优化。在介绍计算图引擎的工作原理之前,本小节先介绍这 2 种优化技术。

神经网络算法中常见的操作包含卷积、全连接、激活、池化、批规范化、缩放,本文将这些操作归为 2 类:线性变换操作和非线性变换操作。所有的线性变换操作都能够表示成向量或矩阵乘加的形式,如下:

$$\boldsymbol{Y} = \boldsymbol{XW} + \boldsymbol{B} \tag{1}$$

所有的非线性变换操作都无法表示成式(1)的形式。其中,$\boldsymbol{X}$、$\boldsymbol{Y}$ 是变量,$\boldsymbol{W}$、$\boldsymbol{B}$ 是常量(提前训练好的模型数据)。按照上面的分类,卷积、全连接、批规范化、缩放操作是线性变换操作,最大池化、激活是非线性变换操作。以全连接操作为例,式(1)中的 $\boldsymbol{X}$、$\boldsymbol{Y}$ 分别表示全连接的输入、输出神经元矩阵,$\boldsymbol{W}$ 表示权值矩阵,$\boldsymbol{B}$ 表示偏置矩阵。上面列举的其他线性变换操作也能转换成式(1),具体转换的方法本文不做详细描述。

假定有 2 个连续的线性变换操作,第 2 个操作使用第 1 个操作的输出作为输入:

$$\boldsymbol{Y}_1 = \boldsymbol{X}_1\boldsymbol{W}_1 + \boldsymbol{B}_1 \tag{2}$$

$$\boldsymbol{Y}_2 = \boldsymbol{X}_1\boldsymbol{W}_2 + \boldsymbol{B}_2 \tag{3}$$

由于线性变换满足分配律和结合律,又由于 $\boldsymbol{W}$ 矩阵和 $\boldsymbol{B}$ 矩阵是常量,因此式(2)、(3)可以做以下等价线性变换:

$$\boldsymbol{Y}_2 = (\boldsymbol{X}_1\boldsymbol{W}_1 + \boldsymbol{B}_1)\boldsymbol{W}_2 + \boldsymbol{B}_2 \tag{4}$$

$$\boldsymbol{Y}_2 = \boldsymbol{X}_1\boldsymbol{W}_1\boldsymbol{W}_2 + \boldsymbol{B}_1\boldsymbol{W}_2 + \boldsymbol{B}_2 \tag{5}$$

$$\boldsymbol{W}' = \boldsymbol{W}_1\boldsymbol{W}_2,\ \boldsymbol{B}' = \boldsymbol{B}_1\boldsymbol{W}_2 + \boldsymbol{B}_2 \tag{6}$$

$$\boldsymbol{Y}_2 = \boldsymbol{X}_1\boldsymbol{W}' + \boldsymbol{B}' \tag{7}$$

因此,原始的 2 步计算式(2)和式(3)通过线性变换为式(4)和式(5)和常量折叠式(6)优化,简化成了一步计算式(7)。这 2 种优化技术在减少运算量的同时也对常量模型数据进行了压缩,一方面减少了模型数据的存储开销,另一方面减少了运行时的访存量。

举一个具体的应用来说明这 2 种优化技术是实用的。ResNet[5] 是经典的图像分类网络,它包含大量的卷积 + 批规范化 + 缩放子结构,如图 2 所示。

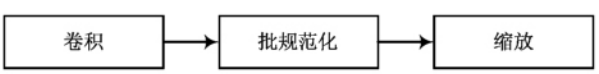

图 2 ResNet 子结构

上面 3 个操作都是线性变换操作,因此可以进行线性变换和常量折叠优化。具体的优化步骤参考式(2)~(7)。

计算图引擎的具体工作原理如下(图 3)。首先扫描原始的计算图,识别其中连续的线性变换操作;然后对连续的线性变换操作执行线性变换和常量折叠优化,把多个操作节点合并成一个操作节点,把多份模型数据合并成一份模型数据;最后输出简化后的计算图和模型数据。

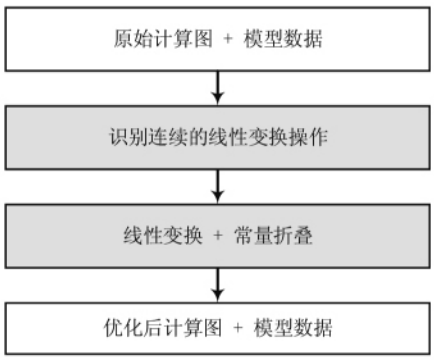

图 3 计算图引擎工作原理图

2.2 代码生成器

代码生成器在生成目标代码时充分考虑了对运算和数据进行协同优化。深度学习处理器为了优化访存性能,往往会在靠近计算单元的地方设计片上缓存(on-chip memory),访问片上缓存的速度远比从片外 DDR 快。为了简化结构设计,深度学习处理器并没有将片上缓存设计成 cache 这种软件透明的结构,因此软件编程人员需要显示地管理这些片上缓存[6,7]。合理利用片上缓存能够极大地提升程序的性能,单从优化运算的角度出发无法做到这一点。为了在支持可变大小的向量乘矩阵、矩阵乘矩阵操作应用场景下充分利用片上缓存,代码生成器必须同时考虑运算优化和数据优化。另一方面,由于深度学习处理器支持向量化操作以及按固定维度进行循环展开[8],从提高访存效率的角度出发,片外数据往往需要做特殊的对齐以及维度变换。

CDUCA 代码生成器在设计时充分考虑了上面几个问题,并提出了以下解决方案。

采用基于 cost model 的启发式搜索策略来协同优化运算和数据。cost model 给出程序的运行时间估计,它考虑的核心因素是访存时间、运算时间以及两者的覆盖率。增大覆盖率是为了隐藏计算、访存延时,在此基础下,尽量减小访存和运算时间就能提升程序的性能。在生成目标平台指令时存在多种选择,比如,分几次加载一块完整的数据,数据存放的

位置(片上缓存/片外 DDR),单次运算数据量的大小等。使用暴力搜索的开销过大,因此,本文提出了一种启发式的指令生成策略,一个重要的启发式策略就是尽量把片上缓存用满。

代码生成器的完整工作流程如下(图 4)。首先对计算图引擎优化后的计算图进行 cost model 建模,然后采用启发式搜索策略来生成高度优化的目标平台指令。

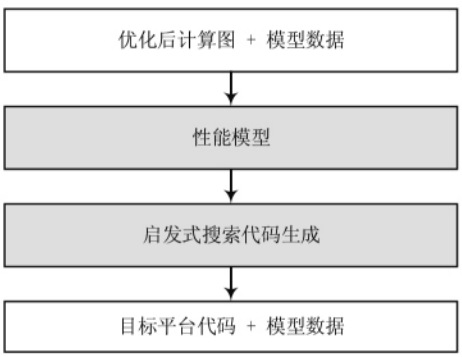

图 4 代码生成器工作原理图

2.3 数据优化器

数据优化器根据目标平台代码所需要的最优数据布局,对模型数据进行面向硬件平台的二次优化。主要的优化包括:

(1) 按照运算的要求对数据进行对齐,加快访存速度。深度学习处理器向量化运算对数据对齐有一定的要求,对齐访问能够加快访存速度。

(2) 按照运算的要求对模型数据进行维度变换,提高访存局部性。卷积等算法需要将数据解释成多维数组,多维数组的排布顺序会影响访存的跳转次数。为了尽量减少访存跳转,需要对数据进行维度变换。

(3) 精度调优,提高运算速度。深度学习处理器往往支持低精度运算[8,9],比如 16 比特量化、8 比特量化,不同的精度运算性能不同。因此,在编译时期,可以对模型数据进行精度选优,从而提高性能。

数据优化器的工作原理如下(图 5)。先按照目标平台代码对数据布局的要求,对模型数据进行上述 3 种优化;然后将目标代码和优化后的模型数据打包,生成可部署模型。

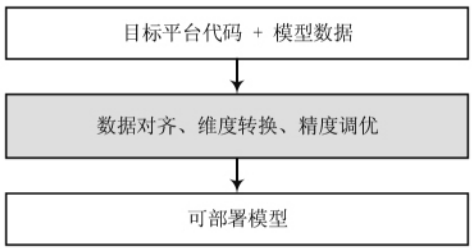

图 5 数据优化器工作原理

3 实验和评估

本文在 DLPlib[10] 的基础上进行了编程模型扩展,使其支持了网络级别的计算,并在内部实现了 CDUCA 编译优化框架。本文对 CDUCA 编译生成的模型进行了多方面的性能评估。接下来先介绍本文使用的硬件平台和测试集。

(1) 硬件平台

本文采用与 DaDianNao[11] 类似的设计,用 verilog 实现了大部分硬件逻辑,并在 SynopsysFPGA 上进行综合。在 100 MHz 的现场可编程门阵列(field programmable gate array, FPGA)主频下,单个处理器核心的峰值运算性能可以达到 0.2 GFOP/s。DaDianNao 包含多个处理核心,本文在测试时只使用了单个运算核心,但这并不表示 CDUCA 框架不支持多核系统。

(2) 测试集

本文选取了典型的图像分类网络如 AlexNet[12]、ResNet[5] 和典型的目标检测网络如 Yolov2[13]、Faster-RCNN[14] 作为测试集。

本文将 CDUCA 生成的模型与手工优化模型进行了性能对比。手工优化模型是指由专业人士使用汇编指令手工实现算法模型,这种方式能够把特定的算法模型性能优化到极致,其缺点是开发成本太高。

图 6 给出了 CDUCA 模型与手工优化模型性能的对比。其中 CDUCA-NOPT 系列表示的是关闭了计算图引擎的 CDUCA。对比 CDUCA-NOT 和 CDUCA 可以看出,计算图引擎优化对 ResNet50、Yolov2 性能提升很大(10% 以上),而对 AlexNet、Faster-RCNN 提升不大,因为 AlexNet 中不包含连续的线性变换操作,而 Faster-RCNN 主要开销并不是在线性操

作部分。表 1 给出了 CDUCA 模型和手工优化模型性能的比值,在平均值上,CDUCA 生成的模型性能能达到手工优化模型性能的 86.5%。

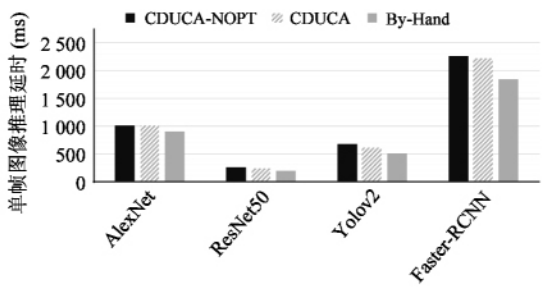

图 6 CDUCA 模型与手工优化模型性能对比

表 1 CDUCA 模型与手工优化模型性能对比

AlexNet	ResNet50	Yolov2	Faster-RCNN	Mean
89.1%	90.7%	83.0%	83.2%	86.5%

4 结 论

本文从深度学习算法和深度学习硬件平台的特点出发,提出了一种运算和数据协同优化的深度学习编译框架 CDUCA。在计算图引擎模块,采用线性变换和常量折叠技术对原始计算图和模型数据进行面向算法的协同优化;在代码生成器模块,采用基于 cost model 的启发式搜索技术,对运算和数据进行协同优化,生成目标代码;在数据优化器模块,采用数据对齐、维度变换、精度调优技术,对模型数据进行面向硬件平台的二次优化。CDUCA 在整个框架设计中贯穿运算和数据协同优化的思想。实验结果表明,CDUCA 生成的模型性能能达到手工优化模型性能的 86.5%。

此外 CDUCA 框架层次化的设计结构具有很强的扩展性,能够在各个模块集成各种编译优化技术。比如,CDUCA 可以在计算图引擎模块集成操作聚合技术,可以在代码生成器模块集成多面体编译优化技术,集成这些优化技术可以作为本文的后续研究工作。

总结来说,在深度学习编译优化领域,对运算和数据进行协同优化非常重要,这是后续通用深度学习编译框架设计中应该重点考虑的。

参考文献

[1] Chen T, Moreau T, Jiang Z, et al. TVM: an automated end-to-end optimizing compiler for deep learning[C]// Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, Berkeley, USA, 2018: 579-594

[2] Vasilache N, Zinenko O, Theodoridis T, et al. Tensor comprehensions: framework-agnostic high-performance machine learning abstractions[J]. *arXiv*: 1802.04730, 2018

[3] Wei E, Schwartz L, Adve V. DLVM: a modern compiler infrastructure for deep learning systems[J]. *arXiv*: 1711.03016, 2017

[4] Truong L, Barik R, Totoni E, et al. Latte: a language, compiler, and runtime for elegant and efficient deep neural networks[C]// Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation-PLDI 2016, Santa Barbara, USA, 2016: 209-223

[5] Krizhevsky A, Sutskever I, Hinton G. ImageNet classification with deep convolutional neural networks[J]. *Advances in Neural Information Processing Systems*, 2012, 25(2):84-90

[6] Liu S, Du Z, Tao J, et al. Cambricon: an instruction set architecture for neural networks[C]// 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, Korea, 2016: 393-405

[7] Chen T, Du Z, Sun N, et al. DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning[C]// Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, Salt Lake City, USA, 2014: 269-284

[8] Guo K, Zeng S, Yu J. A survey of FPGA-based neural network accelerator[J]. *arXiv*:1712.08934, 2017

[9] 周聖元,杜子东,刘道福,等. 低面积低功耗的机器学习运算单元设计[J]. 高技术通讯, 2019, 29(1):12-18

[10] Lan H, Wu L, Wang B, et al. DLPlib: a library for deep learning processor[J]. *Journal of Computer Science and Technology*, 2017, 32(2):286-296

[11] Chen Y, Sun N, Temam O, et al. DaDianNao: a machine-learning supercomputer[C]// Proceedings of the

47th Annual IEEE/ACM International Symposium on Microarchitecture, Cambridge, UK, 2014: 609-622

[12] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[J]. *arXiv*: 1512.03385, 2015

[13] Redmon J, Farhadi A. YOLO9000: better, faster, stronger[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, USA, 2017: 6517-6525

[14] Ren S, He K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks [J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015, 39(6):1137-1149

A computation and data unified compile architecture for deep learning

Wu Linyang[*,**,***], Du Weijian[*,**,***], Chen Xiaobing[*,**,***], Zhuang Yimin[*,**,***]

([*]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

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

([***]Cambricon Tech. Ltd, Shanghai 201308)

Abstract

In recent years, deep learning algorithms and deep learning processors have been widely used in the industry. How to fully exploit the potential of deep learning processors remains a big challenge for compilation framework. The existing deep learning compilation frameworks usually focus on optimizing the computational part of program, and the optimization for the data part is so limited, which can't exploit the peak performance of deep learning processors. Based on the characteristics of deep learning algorithms and hardware, this paper proposes a deep learning compilation framework CDUCA(computation and data unified compile architecture), to achieve computing and data collaborative optimization. CDUCA contains three different levels of components: computation graph engine, code generator and data optimizer. CDUCA performs hierarchical optimization for both computation and data, and then generates efficient deployable model. This paper evaluates CDUCA on the FPGA platform with several typical deep learning applications. The experiments show that CDUCA model can achieve a speedup of 86.5% on average compared to manual optimized model.

Key words: deep learning, deep learning processor, compiler, compile optimization