doi:10.11887/j.cn.201803011 http://journal.nudt.edu.cn

二维矩阵卷积在向量处理器中的设计与实现*

张军阳,郭 阳
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:为了加快卷积神经网络模型的计算速度,便于大规模神经网络模型在嵌入式微处理器中的实现,以FT-matrix2000向量处理器体系结构为研究背景,通过对多核向量处理器体系结构的分析和对卷积神经网络算法的深入研究,提出将规模较小的卷积核数据置于标量存储体,尺寸较大的卷积矩阵置于向量存储体的数据布局方案。针对矩阵卷积中数据难以复用的问题,提出根据卷积核移动步长的不同动态可配置的混洗模式,通过对所取卷积矩阵元素进行不同的移位操作,进而大幅提高卷积矩阵数据的复用率。针对二维矩阵卷积由于存在数据相关性进而难以多核并行的问题,提出将卷积矩阵多核共享,卷积核矩阵多核独享的多核并行方案。设计了卷积核尺寸不变、卷积矩阵规模变化和卷积矩阵尺寸不变、卷积核规模变化的两种计算方式,并在主流CPU、GPU、TI6678、FT-matrix2000平台进行了性能对比与分析。实验结果表明:FT-matrix2000相比CPU最高可加速238倍,相比TI6678可加速21倍,相比GPU可加速663 805倍。

关键词:卷积神经网络;向量处理器;多核实现;矩阵卷积
中国分类号:TP391 **文献标志码**:A **文章编号**:1001-2486(2018)03-069-07

Design and implementation of two-dimensional matrix convolution based on vector processor

ZHANG Junyang, GUO Yang
(College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: In order to accelerate the computational speed of convolution neural network model and facilitate the implementation of large-scale neural network model in embedded microprocessor, the FT-matrix2000 vector processor architecture was taken as the research background. Through the analysis of the multi-core vector processor architecture and convolution neural network algorithm, a data layout scheme was proposed in which a smaller convolution kernel data was placed in a scalar memory bank and a larger convolution matrix was placed in a vector bank. Aimed at the problem that the data in the matrix convolution is hard to reuse, a dynamic shuffling pattern with different dynamic configurable parameters based on the moving steps of the convolution kernel was proposed, by carrying out different shift operations on the convolution matrix elements, the multiplexing rate of convolution matrix data was greatly improved. Aimed at the problem that two-dimensional matrix convolution is difficult to multi-core parallelism due to the existence of data correlation, a multi-core parallel scheme with convolution matrix sharing and convolution kernel matrix multi-core exclusive was proposed. Two computing methods of convolution kernel size unchanged, convolution matrix size changed and convolution matrix size unchanged and convolution kernel size changed were designed, a performance comparison and an analysis were carried out in mainstream CPU, GPU, TI6678 and FT-matrix2000. The final experimental results show that compared with the multi-core, the CPU can be accelerated up to 238 times, compared with TI6678, the speed can be accelerated 21 times, and compared with the high-performance GPU, the speed can accelerate 663 805 times.

Key words: convolution neural network; vector processor; multi-core implementation; matrix convolution

近年来,深度学习[1]受到了越来越多的关注,尤其是在图像处理[2]、语言处理[3]、机器翻译[4]等领域基于深度学习的目标识别技术都取得了重大突破,进而引发了机器学习和计算机视觉等领域的新一轮研究热潮。深度学习包括一系列神经网络模型,如卷积神经网络(Convolutional Neural Network, CNN)[5-6]、深度置信网络(Deep Belief Network, DBN)[7-9]、自动编码机[10]、循环神经网络(Recurrent Neural Network, RNN)[11]等常用模型。尤其是基于CNN的模型在图像识别领域取得了重大突破,当前几乎所有的图像识别类任务中识别率最好的模型都基于CNN。一般来说CNN模型由若干卷积层和池化层交替出现,最后由若干全连接层和分类层组成。其中,

* **收稿日期**:2017-04-09
基金项目:国家重点基础研究发展计划资助项目(2016YFB0200401);国家自然科学基金资助项目(61572025)
作者简介:张军阳(1987—),男,河南平顶山人,博士研究生,E-mail:zhangjunyang11@nudt.edu.cn;
郭阳(通信作者),男,研究员,博士,博士生导师,E-mail:guoyang@nudt.edu.cn

卷积层的计算量占整个模型的85%以上[12]。因此,当前的许多研究大多是针对CNN的加速器,如图形处理单元(Graphic Processing Unit, GPU)[13-14]、现场可编程逻辑门阵列(Field Programmable Gated Array, FPGA)[15-16]、专用集成电路(Application Specific Integrated Circuit, ASIC)[17-19]、向量数字信号处理器(Digital Signal Processor, DSP)等。

向量处理器体系结构就是其中的一种新颖体系结构[20]。一般包括标量处理单元和向量处理单元,标量计算单元负责标量任务的计算和流控;向量计算单元主要负责大规模的向量计算,它包括若干向量处理单元,每个处理单元上包含丰富的运算部件,其在拥有强大计算能力的同时也对软件的开发提出了新的挑战。针对该新型微处理器架构研究如何将各种不同的应用高效地实现向量化是当前面临的一大难题[21]。

本文针对CNN模型中矩阵卷积难以高效并行的难题,结合向量处理器的体系结构特点,设计了一种通过混洗操作来提高算法的并行性和数据复用率的方法。针对多输出卷积结果矩阵的计算,提出一种加速矩阵卷积计算的多核实现方案,并从体系结构、多级存储和计算模式等方面进行了算法优化和性能分析。

1 FT-matrix2000体系结构

FT-matrix2000是一款自主研发且面向高密度计算的高性能浮点多核向量处理器,单芯片集成12颗向量处理器内核,主频1 GHz,双精度峰值性能达到1152 GFLOPS。其单核结构如图1所示,每个单核是独立的超长指令字(Very Long Instruction Word, VLIW)体系结构,包括向量处理单元(Vector Processing Unit, VPU)和标量处理单元(Scalar Processing Unit, SPU),SPU负责标量计算和流控,VPU负责向量计算。VPU包括16个向量处理单元(Vector Processing Element, VPE),每个VPE包含1个局部寄存器文件和由3个浮点乘累加单元(Floating point Multiply ACcumulator, FMAC)、2个Load/Store和1个位处理单元(Bit Process, BP)组成的6个并行功能部件。SPU和VPU之间通过共享寄存器交换数据,支持广播指令将标量寄存器数据广播到向量寄存器。向量处理器核可同时发射11条指令,包括5条标量指令和6条向量指令,指令派发单元对执行包进行识别并派发到相应的功能单元去执行。FT-matrix2000提供96 KB的标量存储器(Scalar Memory, SM)和768 KB的阵列存储器(Array Memory, AM)用于向量访问。

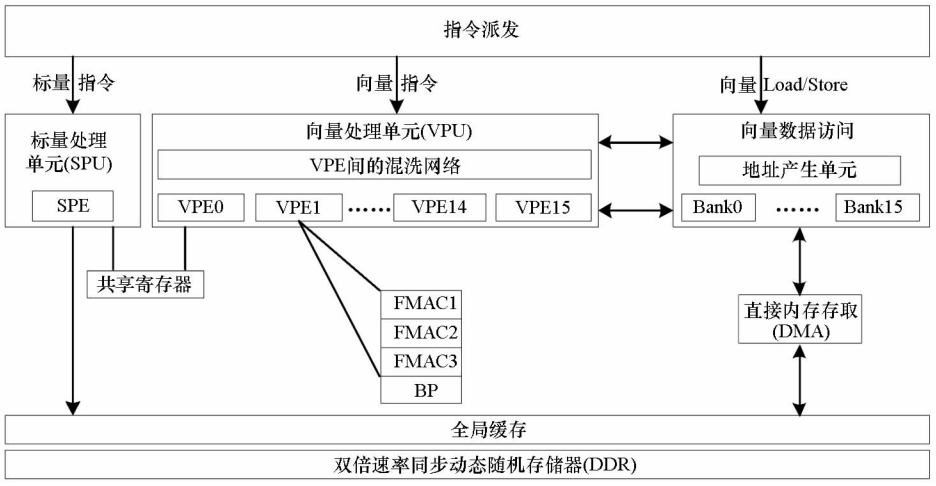


图1 FT-matrix2000处理器体系结构

Fig. 1 Architecture of FT-matrix2000

2 矩阵卷积算法概述

2.1 卷积神经网络的基本结构

一个具有代表性的卷积神经网络结构如图2所示,该模型主要用于早期的手写数字的识别任务中,而当前的神经网络模型与该模型大同小异,主要包括卷积层、下采样层、全连接层以及相应的正则化层,只是在模型的深度和广度以及连接方式上进行了改动。由于卷积神经网络模型中的卷积层的计算时间约占全部模型计算量的85%以上,本文主要研究加速CNN模型中卷积层的计算。

2.2 二维矩阵卷积

二维矩阵卷积常用于图像处理中,卷积神经

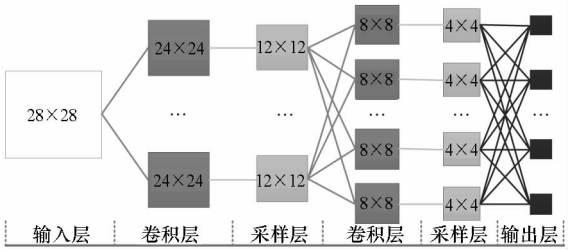


图 2　典型卷积神经网络结构

Fig. 2　Typical structure of CNN

网络中的卷积操作可以抽象成一组卷积核与一个输入特征图的卷积运算。每一个尺寸为 $k_x \times k_y$ 的卷积核在一个输入特征图上与一个 $k_x \times k_y$ 的卷积窗口进行对应点积操作并累加求和后得出输出特征图中的一个元素,卷积核在输入特征图上从上到下、从左到右进行如上操作,最终会得出一个二维的输出特征图。设水平移动步长为 S_x,垂直移动步长为 S_y,则输出特征图上位置为 (a,b) 元素的计算公式为:

$$O_{a,b}^{mo} = f\left(\sum_{mi \in A_{mo}} \left(\beta^{mi,mo} + \sum_{i=0}^{K_x-1}\sum_{j=0}^{K_y-1} \omega_{i,j}^{mi,mo} \times I_{aS_x+i,bS_y+j}^{mi}\right)\right) \tag{1}$$

式中:$\omega_{i,j}^{mi,mo}$ 是输入特征图 #mi 和输出特征图 #mo 之间的卷积核;$\beta^{mi,mo}$ 是输入特征图与输出特征图之间对应的偏置值;A_{mo} 表示输入特征图与对应输出特征图连接的集合;I^{mi} 表示第 mi 个输入特征图,$O_{a,b}^{mo}$ 表示第 mo 个输出特征图;$f(\cdot)$ 表示非线性激活函数,如 tanh、sigmoid 或 ReLU 等。

2.3　二维矩阵卷积的常用计算方法

图 3 为传统的矩阵卷积计算方式和矩阵相乘形式的卷积计算方式,输入为 3 个 3×3 的输入特征图,输入卷积核为 2 个 3 通道的 2×2 卷积核,通过卷积计算可以得出 2 个 2×2 的输出特征图。可以发现:如果采用传统矩阵卷积计算方式,由于都是二维的矩阵卷积计算,尤其是当卷积核规模较小的时候,算法的并行性难以得到保证。因此 Zhang 等[22]使用了一种新的计算方法,即将卷积与卷积核矩阵通过有规律的展开以普通矩阵相乘的方式进行计算。

其计算步骤为:将相应的卷积块以列向量的形式展开并组合成一个新的矩阵;将卷积核依行展开,所有的卷积核组成一个新的卷积核矩阵。将展开后的卷积矩阵和卷积核矩阵通过调用基础线性代数子程序库(Basic Linear Algebra Subprograms,BLAS)[23]函数库中的广义矩阵乘法(GEneral Matrix Multiplication,GEMM)完成卷积计算。

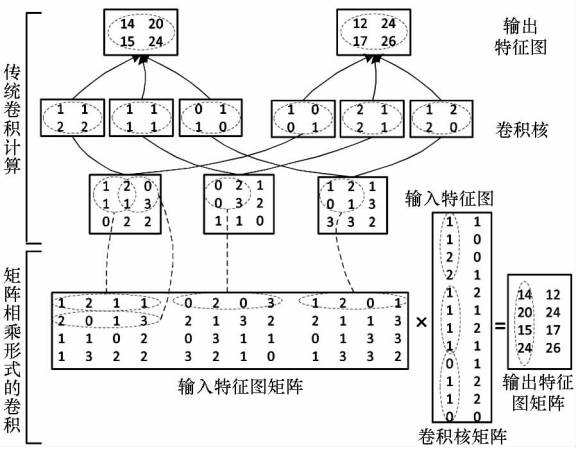


图 3　二维矩阵卷积计算的两种方式

Fig. 3　Two methods of two-dimensional matrix convolution for calculating

因为研究的矩阵卷积操作主要应用于卷积神经网络模型中,且卷积计算的结果通常要作为下一级运算的输入值;所以,神经网络模型中卷积层的计算结果有特定的二维结构。而图 3 中采用的矩阵相乘形式的卷积计算方式,虽然通过组合成大规模矩阵的形式增加了矩阵卷积计算的并行性,但是也破坏了卷积计算结果的二维结构,从而使得该计算结果不能直接作为下一级的输入值,可能需要对该卷积结果进行数据重排序才能作为下一级操作的输入值。同时,由于对卷积矩阵进行了重复展开,展开后的矩阵规模大大增加了,如 3×3 的卷积矩阵在 2×2 的卷积核规模下的展开为 4×4,数据量增加了 77.8%,而当卷积矩阵为 5×5 时,展开的卷积矩阵变成了 16×5,数据量增加了 220%。而片上存储对计算来说尤为重要,因此数据存储量的增加是该方法的一大缺陷。

3　二维矩阵卷积的向量化设计与实现

通过对第 2 节传统卷积计算方法的分析,提出一种新的二维矩阵卷积的向量化实现方法。该算法的实现流程如下:

步骤 1:根据多核向量处理器的核数 m,输入特征图的数量 n(尺寸为 $n_1 \times n_2$)、输入卷积核的数量 k(尺寸为 $k_1 \times k_2$)、向量处理器单核向量存储体的容量 v_1 和标量存储体的容量 v_2,合理划分多核处理器中每个核的计算负载;

步骤 2:将输入特征图元素按行连续存储于 AM 中,卷积核矩阵按行连续存储于 SM 中;

步骤 3:根据单核向量处理单元的个数加载 p 个输入特征图元素,标量加载卷积核第 1 个元素,

并广播至向量寄存器中；

步骤 4：将步骤 3 中通过广播得到的向量寄存器与向量加载的 p 的输入特征图元素进行乘加操作，并将计算结果累加至累加寄存器（ACCumulator register，ACC）中；

步骤 5：根据卷积核在输入特征图中的移动步长，配置相应的混洗模式，并将步骤 3 中所加载的 p 个输入特征图元素使用该混洗模式进行移位操作，同时标量顺序加载第 2 个卷积核元素，并广播至向量寄存器中，将该寄存器与移位后的寄存器中的元素进行乘法操作，并将乘法结果累加至步骤 4 中的累加寄存器 ACC 中；

步骤 6：根据卷积核的行数 k_1 和列数 k_2 重复上述过程，直到完成输出特征图第 1 行元素的计算；

步骤 7：根据输入特征图的行数 n_1 和列数 n_2，向量加载输入特征图的下一行元素，重复上述步骤，进而完成整个卷积结果矩阵的计算。

卷积神经网络模型中的卷积核规模一般都比较小，通常为 1×1、3×3、5×5、7×7、11×11 等，在此将卷积核存入标量存储体，输入特征图存入向量存储体。

3.1　单核程序向量化方法

1）以双精度浮点为例，单个数据为 8 B，为了叙述方便设输入特征图为 $N \times N$ 的方阵，则 $N \times N \times 8 = 768 \times 1024$ KB，所以 AM 中单次可以存放的最大输入特征图尺寸为 222×222。对于嵌入式处理器编程来说，当输入特征图的尺寸大于 222×222 时，需要从外存 DDR 中通过 DMA 来加载数据。

以一个例子来说明在不展开卷积矩阵的情况下如何同时计算输出卷积矩阵多个元素的向量化实现方法。考虑一个 1×6 的 *PEs*（PE_1，PE_2，PE_3，PE_4，PE_5，PE_6），输入特征图矩阵 $\boldsymbol{B}$ 为 6×6，卷积核矩阵为 2×2，步长为 1，为了叙述简洁，只描述输出结果矩阵第 1 行元素的计算（如图 4 所示，其他行计算过程与第 1 行类似）。

#1：6 个 *PEs* 加载输入矩阵 $\boldsymbol{B}$ 的第 1 行元素（$b_{1,1}$，$b_{1,2}$，$b_{1,3}$，$b_{1,4}$，$b_{1,5}$，$b_{1,6}$），标量加载卷积核矩阵的第 1 行第 1 个元素（$k_{1,1}$），$k_{1,1}$ 广播至向量寄存器中，即为 $k_{1,1}$、$k_{1,1}$、$k_{1,1}$、$k_{1,1}$、$k_{1,1}$、$k_{1,1}$，通过向量处理器的乘加指令（Vector Float MULtiply ADd，VFMULAD），每一个 *PE* 完成对应元素与 $k_{1,1}$ 的乘法，把结果累加到对应的 ACC 中；

#2：根据卷积核的移动步长配置相应的混洗模式（如图 5 所示），本实例中由于移动步长为 1，

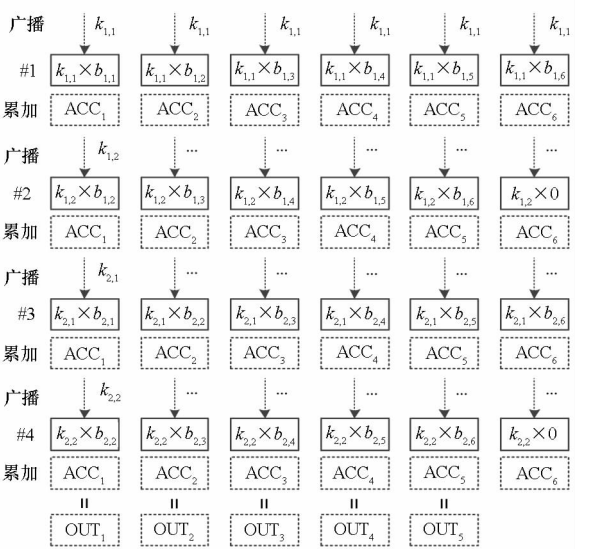


图 4　FT-matrix2000 矩阵卷积的算法映射

Fig. 4　Algorithm mapping of FT-matrix2000

配置的混洗模式为对应的向量寄存器中的元素向左移动 1 位，最左边的元素移出寄存器，最右边的元素补 0，移位后当前 *PEs* 所对应的元素为 $b_{1,2}$、$b_{1,3}$、$b_{1,4}$、$b_{1,5}$、$b_{1,6}$、0，同时加载卷积核的第 1 行第 2 个元素 $k_{1,2}$ 并广播至向量寄存器中，即为 $k_{1,2}$、$k_{1,2}$、$k_{1,2}$、$k_{1,2}$、$k_{1,2}$、$k_{1,2}$，使用向量乘加指令将乘法结果累加至#1 中的 ACC；

#3：6 个 *PEs* 加载输入矩阵 $\boldsymbol{B}$ 的第 2 行元素（$b_{2,1}$，$b_{2,2}$，$b_{2,3}$，$b_{2,4}$，$b_{2,5}$，$b_{2,6}$），广播卷积核矩阵第 2 行第 1 个元素（$k_{2,1}$），使用乘加操作完成对应元素与卷积核元素 $k_{2,1}$ 的乘法操作并将结果累加至#2 中；

#4：将#3 中的元素进行移位的同时广播卷积核矩阵第 2 行第 2 个元素至向量寄存器，使用乘加操作完成对应元素与 $k_{2,2}$ 的乘法，并将结果累加至#3 中。

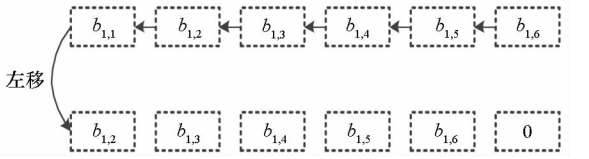


图 5　步长为 1 的混洗操作

Fig. 5　Shuffle mode of stride 1

通过分析，可以看出本方法有以下优势：不需要通过将输入特征图和卷积核重组成大规模的矩阵来提高并行性，减轻了矩阵展开所带来的存储压力，提高了所取数据的利用率。在向量化代码的优化过程中，可以通过软件流水的方式提高算法的并行性，在完全流水的情况下，每拍可以同时计算出输出特征图的一行元素，也可避免文

献[15]中所采用的 *PEs* 间数据的频繁移动。

2)当输入特征图尺寸超过向量存储体的容量时,向量存储体不能一次加载整个输入特征图。在高性能计算中,需平衡各存储层次间的负载搬移,提供内核高效计算所需数据,获得满意的计算效率。FT-matrix2000 拥有多级存储结构,如寄存器文件、L1D、Array Memory、Global Cache 和 DDR,且 L1D 可以配置成全 Cache 和静态随机存取存储器(Static Random Access Memory, SRAM),因此,当输入特征图数量超过向量存储体的容量时,使用全 SRAM 的方法,使用乒乓方式来平滑不同层级存储体之间的数据搬移,将内核计算与 DMA 搬移重叠起来,使内核处于满负荷运行(双缓冲机制如图 6 所示)。

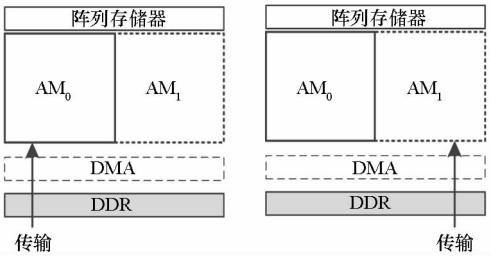


图 6　双缓冲机制示意

Fig. 6　Double buffer mechanism

值得注意的是,由于卷积计算有数据的重叠,在卷积矩阵分块传输时,根据卷积核的尺寸,第 2 次传输的卷积矩阵需要有部分数据与第一次传输的数据重叠。假设卷积核尺寸为 $m \times m$,若第 1 次传输 1 到 n 行,则第 2 次需重叠传输第 1 次传输的后 $m-1$ 行元素;由于卷积核一般较小,重叠传输数据的比重并不大。

3.2　矩阵卷积的多核程序设计分析

算法的并行不仅有单核内部的向量化实现,也包括多核的核间并行。FT-matrix2000 是一款多核处理器,因此,研究矩阵卷积的多核并行也是一个重要内容,尤其是针对大规模的矩阵卷积。由于核间并行需要考虑计算负载的多核划分、算法的相关性、核间的通信开销等,多核程序的设计更加复杂。

此外,在矩阵卷积的计算过程中,因卷积核的尺寸以及卷积核在输入特征图上的滑动步长不同,单个卷积的计算过程存在一定的数据相关性,因此,考虑到多核同步的通信开销,单输入特征图的卷积计算不易进行多核并行。因此主要针对单核实现单个卷积矩阵的向量化设计,当有多个独立的卷积计算时再由多核来并行进行加速,多输出矩阵卷积的多核实现方案如图 7 所示。

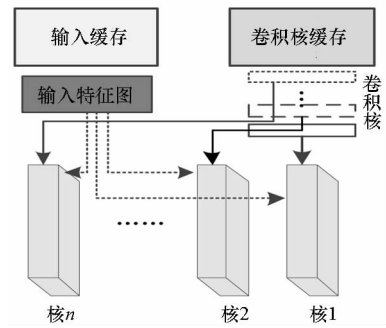


图 7　FT-matrix2000 矩阵卷积多核实现方案

Fig. 7　Multicore implementation of matrix convolution on FT-matrix2000

4　性能测试与分析

本实验的对比平台为 AMD A10-6700 APU,4 核 4 线程,DDR3 内存 16 G,主频 3.7 GHz。GPU 平台为 NVIDIA GeForce GTX1080TI,11 GB 显存,核心频率 1.5 GHz,使用 NVIDIA 专门为卷积计算高度优化的 cudnn 函数库,并基于 Torch 神经网络编程框架进行矩阵卷积的性能统计和优化。FT-matrix2000 平台单核核内向量存储空间 768 KB,标量存储空间 96 KB,核外 DDR 最大支持 128 GB,主频 1 GHz。使用 TI 的多核数字信号处理器 TMS320C6678,8 核,1.25 GHz 主频,每个核拥有 32 KB 的 L1P 和 32 KB 的 L1D,并基于 CCS5.5 软件编程平台完成所有程序代码的测试与性能统计。FT 平台基于 FT-matrix2000 软件开发环境完成 FT-matrix2000 代码编写与性能测试,核心代码程序采用基于手工汇编的方式进行软件流水和循环展开优化。

以固定卷积核尺寸和固定卷积矩阵尺寸两种方式来进行实验分析。(图 8~11 中类似 16×5 表示 16×16、5×5 的方阵,且为 64 位双精度浮点值)。

图 8 统计了随着卷积核尺寸变化,FT-matrix2000 对于 CPU 和 TI6678 的加速比,可以看出,FT-matrix2000 向量处理器相对于主流多核 CPU 取得了 24~44 倍的加速比,相对于 TI6678 取得了 14~17 倍的加速比。图 9 为卷积核规模不变,随着输入矩阵规模变化的 FT-matrix2000 基于 CPU 和 TI6678 的加速比,在此计算模式下,FT-matrix2000 相对主流 CPU 取得了 14~238 倍的加速比,相对 TI6678 取得了 13~21 倍的加速比。

图 8 和图 9 分别为两种卷积计算模式下,FT-matrix2000 向量处理器相对于主流 CPU 和 TI6678 取得的性能优势。由于本文算法主要是用于卷积

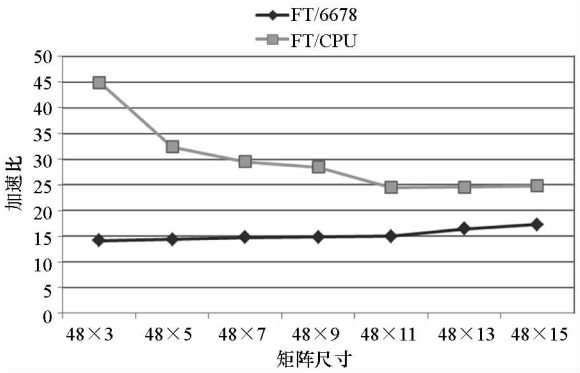


图 8 基于卷积核规模变化的 FT-matrix2000 与 CPU/6678 的加速比

Fig. 8 Speedup of FT-matrix2000 and CPU/6678 based on kernel scale change

神经网中卷积层的计算,采取的数据计算规模不大。从图 8、图 9 中可以看出,当矩阵卷积计算的规模较小时可以取得较大的加速比,而当矩阵规模增大时加速比有所下降,并基本保持稳定。可见,此两种计算模式,对程序的实现性能有一定的影响。

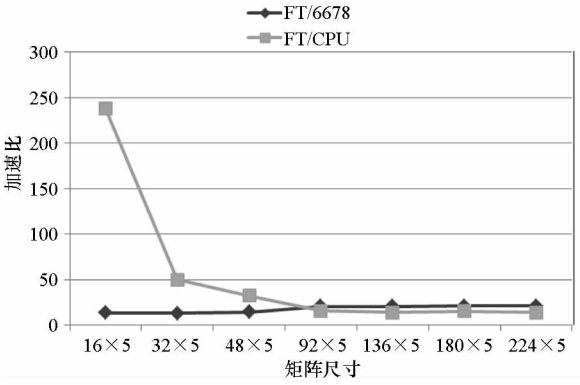


图 9 基于卷积矩阵变化的 FT-matrix2000 与 CPU/6678 的加速比

Fig. 9 Speedup of FT-matrix2000 and CPU/6678 based on matrix scale change

图 10 为 FT-matrix2000 在基于卷积核规模变化的模式下相对 GPU 的加速比。图 11 为 FT-matrix2000 在基于输入特征图规模变化的模式下相对 GPU 的加速比。从图 10 中可以看出,当输入特征图矩阵不变时,随着卷积核矩阵规模的增大,FT-matrix2000 相对 GPU 的加速比呈下降趋势。这主要因为 FT-matrix2000 属于嵌入式向量处理器,而 NVIDIA 1080TI 属于服务器级高性能处理器,拥有 3584 个 cuda 核心,单精度峰值性能高达 11.5 TFLOPS。但当矩阵卷积规模较小的时候,GPU 由于计算负载不足,其并没有完全发挥应有的计算性能,同时由于在使用 GPU 计算的时

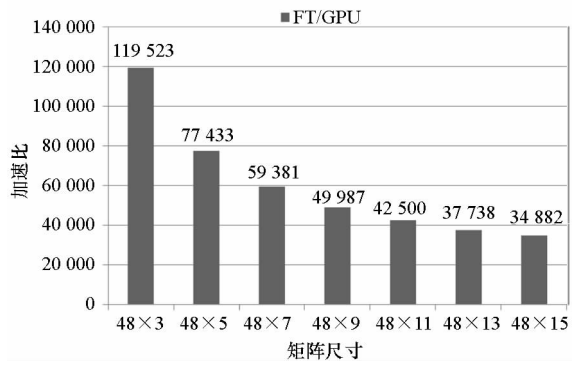


图 10 基于卷积核规模变化的 FT-matrix2000 相对 GPU 的加速比

Fig. 10 Speedup of FT-matrix2000 and GPU based on kernel scale change

候,需要通过使用 *cudaMemcpy*( ) 函数将需要计算的数据由 CPU 内存拷贝到 GPU 显存。因此,当数据量较小的时候,数据的拷贝要占据大部分时间,影响了计算效率,故加速比很高;而当计算规模增大的时候,加速比下降,主要因为数据拷贝占用时间比重降低,计算占比增加,此时数据量越大越能发挥 GPU 的众核优势。图 11 中的整体趋势同图 10 类似,只是加速比并没有图 10 高,主要是因为 FT 平台的算法实现中,内核循环是由卷积核的尺寸控制,卷积核尺寸越大,内核程序通过软件流水所获得的性能优势越明显,FT 平台该模式下矩阵卷积的算法实现效率越高。

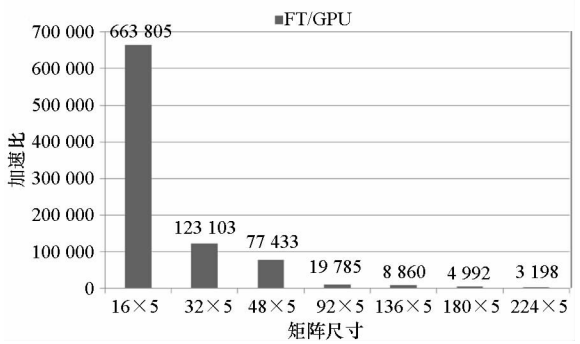


图 11 基于输入特征图规模变化的 FT-matrix2000 相对 GPU 的加速比

Fig. 11 Speedup of FT – matrix2000 and GPU based on matrix scale change

5 结论

本文提出一种基于混洗的二维矩阵卷积向量化实现方法,通过配置不同的混洗模式可以实现卷积核不同移动步长的卷积计算,同时通过对输入特征图数据的移位操作可以充分利用已取数据,大大提高数据的利用率,针对大规模的矩阵卷

积提出了一种多核实现方案,将 L1D 设置成 SRAM 模式,用双缓冲的方式平衡多级存储体之间的数据搬移,将内核计算和 DMA 搬移重叠起来,进而提高算法的计算效率。基于主流 CPU、TI6678、高性能 GPU 进行了算法的性能统计和分析。实验结果显示了 FT-matrix2000 比 CPU、TI6678 及 GPU 具有更好的计算优势,相比 CPU 最高可加速 238 倍,相比 TI6678 最高可加速 21 倍,相比 GPU 可加速663 805倍。

参考文献(References)

[1] Deng L, Yu D. Deep learning: methods and applications[M]. USA: Now Publishers Inc, 2014.

[2] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. Computer Science, 2014.

[3] Graves A, Fernández S, Gomez F, et al. Towards end-to-end speech recognition with recurrent neural networks[C]// Proceedings of the 23rd International Conference on Machine Learning, 2014: 1764 - 1772.

[4] Liu S J, Yang N, Li M, et al. A recursive recurrent neural network for statistical machine translation[C]//Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014: 1491 - 1500.

[5] Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278 - 2324.

[6] Lecun Y, Boser B, Denker J S, et al. Backpropagation applied to handwritten zip code recognition[J]. Neural Computation, 1989, 1(4): 541 - 551.

[7] Hinton G, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets[J]. Neural Computation, 2006, 18(7): 1527 - 1554.

[8] Salakhutdinov R, Hinton G. Deep boltzmann machines[J]. Journal of Machine Learning Research Proceedings Track, 2009, 9(1): 448 - 455.

[9] 刘建伟, 刘媛, 罗雄麟. 波尔兹曼机研究进展[J]. 计算机研究与发展, 2014, 51(1): 1 - 16.
LIU Jianwei, LIU Yuan, LUO Xionglin. Research and development on Boltzmann machine[J]. Journal of Computer Research and Development, 2014, 51(1): 1 - 16 (in Chinese)

[10] Bourlard H, Kamp Y. Auto-association by multilayer perceptrons and singular value decomposition[J]. Biological Cybernetics, 1988, 59(4/5): 291 - 294.

[11] Miao Y, Gowayyed M, Metze F. EESEN: end-to-end speech recognition using deep RNN models and WFST-based decoding[C]//Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding, 2015: 167 - 174.

[12] Liu S L, Du Z D, Tao J H, et al. Cambricon: an instruction set architecture for neural networks[J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 393 - 405.

[13] Shen T, Fei H U. Acceleration of CNN on GPU[J]. Application of IC, 2017, 34(6): 18 - 22.

[14] Shi Y, Lan Q, Fang H, et al. Accelerating CNN's forward process on mobile GPU using OpenCL[C]// Proceedings of Eighth International Conference on Digital Image Processing, 2016.

[15] Meloni P, Deriu G, Conti F, et al. Curbing the roofline: a scalable and flexible architecture for CNNs on FPGA[C]// Proceedings of the ACM International Conference on Computing Frontiers, 2016: 376 - 383.

[16] Han X S, Zhou D J, Wang S H, et al. CNN-MERP: an FPGA-based memory-efficient reconfigurable processor for forward and backward propagation of convolutional neural networks[C]//Proceedings of IEEE International Conference on Computer Design, 2016: 320 - 327.

[17] Chen T S, Du Z D, Sun N H, et al. DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning[J]. ACM Sigarch Computer Architecture News, 2014, 49(4): 269 - 284.

[18] Liu D F, Chen T S, Liu S L, et al. PuDianNao: a polyvalent machine learning accelerator[C]//Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems, 2015: 369 - 381.

[19] Du Z D, Fasthuber R, Chen T S, et al. ShiDianNao: shifting vision processing closer to the sensor[C]// Proceedings of the 42nd Annual International Symposium on Computer Architecture, 2015: 92 - 104.

[20] 刘仲, 田希, 陈磊. 支持原位计算的高效三角矩阵乘法向量化方法[J]. 国防科技大学学报, 2014, 36(6): 7 - 11, 47.
LIU Zhong, TIAN Xi, CHEN Lei. Efficient vectorization method of triangular matrix multiplication supporting in-place calculation[J]. Journal of National University of Defense Technology, 2014, 36(6): 7 - 11, 47. (in Chinese)

[21] 刘仲, 陈跃跃, 陈海燕. 支持任意系数长度和数据类型的 FIR 滤波器向量化方法[J]. 电子学报, 2013, 41(2): 346 - 351.
LIU Zhong, CHEN Yueyue, CHEN Haiyan. A vectorization of FIR filter supporting arbitrary coefficients length and data types[J]. Acta Electronica Sinica, 2013, 41(2): 346 - 351. (in Chinese)

[22] 张抢强, 王春霞, 刘振宇. 基于分块卷积的大图像输入卷积神经网络加速[J/OL]. 北京: 中国科技论文在线(2016 - 04 - 12)[2017 - 01 - 21]. http://www.paper.edu.cn/releasepaper/content/201604 - 138.
ZHANG Qiangqiang, WANG Chunxia, LIU Zhenyu. Accelerating large-scale convolutional neural networks based on convolution in blocks[J/OL]. Beijing: Science & Technology Magazine Online(2016 - 04 - 12)[2017 - 01 - 21]. http://www.paper.edu.cn/releasepaper/content/201604 - 138. (in Chinese)

[23] Dongarra J J. An extended set of FORTRAN basic linear algebra subprograms[J]. ACM Transactions on Mathematical Software, 1988, 14(1): 18 - 32.