CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 40 卷第 1 期 2018 年 1 月
Vol. 40, No. 1, Jan. 2018

文章编号:1007-130X(2018)01-0034-06

CNN 卷积计算在移动 GPU 上的加速研究*

王湘新[1],时　洋[2],文　梅[2]

(1. 武警湖南省消防总队信息中心,湖南 长沙 410205;2. 国防科技大学计算机学院,湖南 长沙 410073)

摘　要:卷积神经网络(CNN)凭借其优秀的表现正在诸如图像分类、语音识别等领域里扮演着越来越重要的角色,已经有一些研究人员想要将这个深度学习过程复制到手机上。但是,由于 CNN 巨大的计算量,移植程序的性能一直难以令人满意。为了探讨如何解决这一问题,借助 MXNet 这样一个深度学习的框架在手机上实现了 CNN 的前向过程,并且将注意力放在了使用手机上另一个强大的计算设备——GPU 上。最终选择使用 OpenCL 通用编程框架将前向过程中最耗时的卷积操作利用矩阵乘来完成,并转移到 GPU 上进行。在此基础之上还针对手机 GPU 做了一些优化。最终,实验结果显示我们成功地将前向过程的时间降低到了原来时间的一半。

关键词:CNN;手机;移动 GPU;快速算法;OpenCL

中图分类号:TP391.4　　**文献标志码**:A

doi:10.3969/j.issn.1007-130X.2018.01.005

Accelerating CNN on mobile GPU

WANG Xiang-xin[1], SHI Yang[2], WEN Mei[2]

(1. Information Center of Armed Police Fire Center, Changsha 410205;
2. College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: Convolutional Neural Networks (CNNs) are playing an increasingly important role in areas such as image classification and speech recognition because of their excellent performance. Some researchers have already wanted to apply this deep learning process on mobile phones, but the performance of the porting program is unsatisfactory due to the huge amount of computation of CNN. In order to explore how to solve this problem, this paper uses a deep learning framework named MXNet to realize the forward process of CNN on mobile phones and focuses on the use of GPU that is another powerful computing device on the mobile phone. Based on the OpenCL common programming framework, we use matrix multiplication to compute the most time-consuming convolution in the forward process and move it to the GPU. Besides, serval improvements are made to achieve better performance. Finally, the experimental results show that we succeed in reducing the time of the forward process to half of the original time.

Key words: CNN; mobile phone; mobile GPU; fast algorithm; OpenCL

1　引言

关于卷积神经网络 CNN(Convolution Neural Network)的相关研究在过去几十年中取得了很多的成果,CNN 作为一种强有力的方法在图像分类[1]、语音识别[2]以及目标检测[3]等相关领域中所取得的结果明显优于传统方法。在图像分类领域,

* 收稿日期:2016-11-08;修回日期:2017-02-15
基金项目:国家自然科学基金(61272145)
通信地址:410205 湖南省长沙市武警湖南省消防总队信息中心
Address: Information Center of Armed Police Fire Center, Changsha 410205, Hunan, P. R. China

CNN 方法甚至已经具有了击败人类的能力。微软在 ImageNet 的数据集上使用 CNN 将分类错误率降至 3.57%[4],而人类自己眼睛分类的错误率大约是 5.1%。CNN 应用的不断普及也使得研究人员开始尝试在移动设备上对其进行使用。

从较高的层次来看,CNN 的工作过程分为两个阶段。首先是利用大量数据让网络进行学习的训练阶段,接下来的前向阶段则是利用网络来进行推断工作。

对于移动设备上的 CNN 应用来说,往往只需要前向阶段。CNN 在移动设备上得到普及的最大制约因素就是巨大的计算量。以图像分类的前向过程为例,在常见的手机移动设备上,一张图像的分类结果就需要数秒的时间,这些时间主要被用来进行 CNN 中的卷积计算。如何处理好复杂的卷积计算是 CNN 在移动设备上取得实际应用的关键。

本文提出了一种新的方法来处理 CNN 的卷积过程,以实现更好的性能。我们的基础是 MXNet[5] 所提供的在手机 CPU 上的 CNN 基础实现[6]。我们注意到,手机上的图形处理单元 GPU(Graphics Processing Unit)相对于 CPU 来说,计算性能更为强大。因此,本文利用 OpenCL(Open Computing Language)[7] 将卷积计算转化为并行度更高的矩阵乘法运算,并将其转移到 GPU 端来进行。之后,本文又采用了诸如优化任务分配、利用片上存储、向量化和循环展开等操作来进行优化,最终,卷积过程取得了 16.39 倍的加速比,前向分类过程取得了 2.1 的加速比。

2 背景与相关研究介绍

2.1 卷积层实现方式

虽然各种各样的 CNN 具有不同的网络结构,但是它们都是由卷积层、池化层等网络层连接组成。在这些网络层中,CNN 绝大部分的计算量都集中在卷积层。卷积层通过将卷积核在图像数据上滑动计算,来提取图像中的高维特征。在 CNN 的具体实现中,对于卷积计算有两种主流实现方法:一是按照卷积定义来进行计算。这种方法的优点是原理简洁,缺点则是计算访存混乱,容易触发内存速度瓶颈。另一种方式则是将计算过程转化为矩阵相乘来实现,优点是访存更加规整,可以利用已有的数学运算库,缺点则是增加内存消耗。

在本文的卷积加速实现中,考虑到移动 GPU 的功能特点,我们采用的是第二种矩阵乘的方法,可以更加充分地利用 GPU 的并行计算性能,同时有更多的优化空间。具体的转化过程在 Kumar 的论文[8] 中有详细的介绍,示例过程如图 1 所示。原理就是将图像数据以及卷积核数据进行复制重排,构造两个大矩阵,通过这两个大矩阵的相乘来得到卷积计算的最终结果。

2.2 移动 GPU 与 OpenCL

图形处理单元(GPU)[9] 首先应用在计算机图形应用程序比如游戏之中,后来,人们又利用它的多核架构来进行通用并行计算。不同于人们在桌面 GPU 上长达几十年的尝试与研究,在移动 GPU

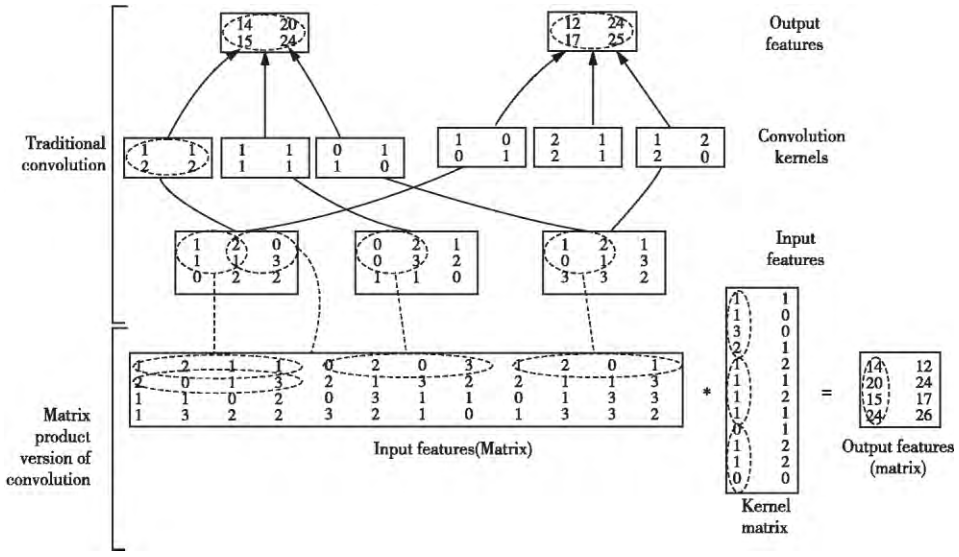

Figure 1 Convert the convolution to matrix multiplication

图 1 矩阵乘法计算卷积

这一领域,直到最近几年才有人开始尝试开发。得益于类似 OpenCL 这样的编程框架的兴起,我们才可以方便地使用移动 GPU 进行通用编程。

OpenCL 是一个开放的、无版权的跨平台并行编程标准,它提供了一个标准接口来给程序员进行编程。由制造计算设备的硬件厂商来实现这个接口并将其具体实现封装在设备之中。在 OpenCL 中,主机处理器(通常是 CPU)管理 OpenCL 的运行环境上下文以及计算设备,程序员需要进行选择来将计算密集的任务编程成为一个 kernel 文件,然后选择一个计算设备来执行这个任务。在一个计算设备中,可以划分为许多的工作组,而每个工作组又可以划分为许多的计算单元。每个计算单元都会独立地执行编写的 kernel 程序。

一个典型的移动 GPU 是由数个计算单元组成的,在每个单元内有数个算术逻辑单元 ALU (Arithmetic Logic Unit)。这里我们以高通公司生产的 Adreno 330 GPU[10] 为例:这款移动 GPU 具有 4 个计算单元和总计 128 个 ALU,在每个计算单元中有 16 线程可以并行。与桌面 GPU 不同,移动 GPU 的存储器与 CPU 芯片共享相同的内存,但是每个计算单元具有不能由其他计算单元使用的本地存储器。之前很少有人研究如何利用移动 GPU 来加速应用。龚若皓[11]在移动 GPU 平台上进行了关于加速静态、动态图像解码算法的尝试,取得了 4 倍的加速比;曾宝国[12]借助移动 GPU 给离散傅里叶变换的高效实现提供了一种新的思路,可以大大提高信号处理的实时性。对于移动 GPU 利用的探索,还是一个很新的研究课题,具有一定的研究意义。

3 实现与优化

3.1 卷积神经网络(CNN)

在 MXNet 框架中,卷积计算是通过矩阵乘法来实现的。其中,矩阵 A 表示输入图像数据,矩阵 B 是卷积内核,矩阵 C 存储计算的结果。这里设 A 具有 M 行和 K 列,B 具有 K 行和 N 列,C 具有 M 行和 N 列。在卷积计算过程中,三个矩阵均使用列主序来进行存储。在矩阵乘法的实现方式上,MXNet 选择的是 OpenBLAS[13] 数学运算库中的 SGEMM 函数。

本文目标是在移动设备上利用 GPU 的并行计算能力来对卷积层进行加速实现。为此,本文首先利用 OpenCL 编程框架在移动 GPU 上实现了矩阵乘法的多重循环朴素算法。这里设定局部大小为{1,1}和全局大小为{M,N},在此条件下,我们有 $M * N$ 个工作线程,每个工作组包含一个线程,每个线程计算 C 的一个元素,这是我们的基本内核。接下来,本文将讨论一些优化手段来提升程序性能。

3.2 优化方法

在讨论具体的优化策略之前,我们需要对矩阵的数据存储结构进行调整。首先,由于 GPU 的结构决定了它在处理 ALU 运算时速度快,处理条件分支时却相当缓慢,我们将归一化矩阵规模,以减少由于矩阵尺寸所引起的条件分支数量。在这里,我们用零填充矩阵使之行列数都是 32 的倍数。

第二,我们需要恢复矩阵 A 为行主序存储而不是列主序存储。在矩阵乘法计算中,计算 C 中的一个元素,需要 A 的一整行和 B 的一整列。如 3.1 节所述,初始矩阵被存储在列主序中。换句话说,在 B 的某一列中的元素可以被连续地访问,但 A 某一行中的元素则不能。所以,这种调整有助于提升矩阵乘法中对内存的访问连续性。接下来,我们重点介绍两种加速 GPU 上卷积实现的方法。

3.2.1 优化任务划分并使用局部存储

在本文所使用的 Adreno 330 GPU 中有 4 个计算单元,而且每一个计算单元中都有 32 个 ALU,并支持 16 个线程同时运行。因此,前文中的朴素内核对于 GPU 的计算能力利用并不充分。当某一工作组被装载到 GPU 某个计算单元时,这个工作组中仅有一个线程运行,并且这个线程只负责计算 C 中的一个元素。这种计算模式下的线程划分会造成线程数目以及空闲 ALU 单元过多的问题。为了提升 GPU 的利用率,本文采取分块矩阵乘的方法来优化线程任务划分。

该方法的示意图如图 2 所示,为了计算矩阵 C 的一个子块,需要矩阵 A 相应的行和矩阵 B 相应的列。现在,如果我们利用分块(记为 *Asub* 和 *Bsub* 以及 *Csub*)来划分 A 和 B,我们可以通过反复计算 *Asub* 与 *Bsub* 的乘积进行累加来得到 *Csub* 的值。

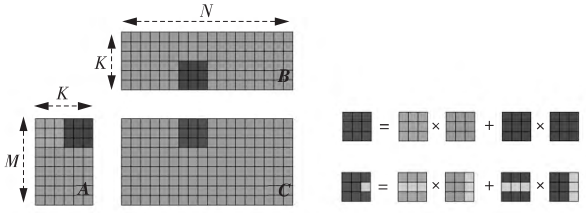

Figure 2 Principe of block matrix multiplication

图 2 矩阵乘分块算法原理

经过对于矩阵分块规模的测试,本文最终选择局部大小为{1,32}以及全局大小为{M /32, N}。这意味着现在每 32 个线程将组成一个工作组,总计拥有 $M * N$ /1024 工作组。工作组中的每个线程将计算包含 32 个元素的一列。矩阵子块中的每一列数据将由工作组中的一个线程负责计算。通过矩阵分块,本文可以解决之前内核所造成的线程数量过多的问题。

在目前卷积计算的加速实现中,每当需要矩阵 ***A***、***B*** 的数据时,都会从全局内存中进行读取。频繁的全局内存访问会造成程序执行缓慢。值得注意的是,在移动 GPU 中,每个计算单元具有一块独立的存储空间,称为本地存储器,本地存储器中的数据访问速度高于全局存储器,并且在本地存储器中的数据可以在此工作组中的线程之间实现共享。因此,本文将利用本地存储器来优化程序的访存速度。

在 Adreno 330 GPU 中本地存储器块的大小为 8 192 KB,即可容纳 2 048 个单精度浮点数。所以本文将分块矩阵实现中 ***A*** 和 ***B*** 的两个 32 * 32 子块(***Asub*** 与 ***Bsub***)迭代加载到本地存储器,线程从本地的存储器读取数据,而不是从全局存储器来读取计算的数据。通过这种访存优化,在本地存储器中的每个元素将被用于工作组中的 32 个线程。也就是说,本文成功地将程序全局内存访问量减少到只有原来的 1/32。

3.2.2 循环展开与向量化

在这一小节里,将讨论两种可以加速程序循环体执行的方法:循环展开以及向量化。

循环展开是一种循环变换技术,可以加快整个循环执行的速度,其代价则是代码二进制文件的大小增加。由于 GPU 对于条件分支的执行很慢,而循环展开则会减少甚至消除分支的数量,所以循环展开技术在 GPU 上将带来性能上的提升。而另一方面,更大的循环体会给编译器提供更多优化的机会。值得一提的是,并不是越大的循环体就会取得越好的性能,这里面还有一个高速缓存交换的因素也要考虑。在本文中,经过对性能以及代码文件大小的综合考量,我们选择以 8 为距离进行循环展开。

现在在我们的内核中,所操作的数据类型为 4 个字节的单浮点类型。本文提出对于循环执行进行向量化操作,增加对访存部件宽度的利用率。在 Adreno 330 的硬件层次不支持向量运算(如向量乘或向量加),但他们确实有专为全局和本地存储器特殊设计的更宽的读取和存储指令。这将使我们得到更快的速度,因为更宽的数据类型减少了加载/存储指令的数目。OpenCL 还支持简单的向量数据类型,这使得我们也不必将矩阵转换成向量类型就可以享受到向量化带来的收益。

经过对于 float4、float8 以及 float16 三种向量宽度的测试,本文选择了表现最好的 float4 向量来进行向量化。图 3 显示了我们如何利用这些优化来从存储器加载数据。

```
for i=0 to 31{
    local_block A[i]=A[i];
    local_block B[i]=B[i]; ⇒
    i+=1;
}
```

```
for i=0 to 31{
  (*(float4*&local_block A[i]))=(*(float4*&A[i]));
  (*(float4*&local_block A[i+4]))=(*(float4*&A[i+4]));
  (*(float4*&local_block B[i]))=(*(float4*&B[i]));
  (*(float4*&local_block B[i+4]))=(*(float4*&B[i+4]));
  i+=8;
}
```

Figure 3 Optimization for data fetch stage

图 3 数据读取阶段优化

4 实验结果

本文中所涉及的实验在小米 note[14] 上完成。这款手机使用的是高通 801 处理器,在这个处理器上集成了一颗 Adreno 330 的 GPU,其单精度 32 位浮点计算峰值为 84.6 GFlops/s,访存速度为 2.84 GB/s。

本文首先使用 AlexNet[1] 来测试卷积实现中矩阵乘法使用不同内核的执行时间,以检验本文 GPU 实现以及优化手段的加速效果。结果展示如图 4 所示。从实验结果可以看到,利用 GPU 比使用 CPU 获得了明显的加速效果。对于移动 GPU,带宽是非常紧缺的资源,所以优化策略应着眼于内存访问。我们使用本地存储器后速度提升了 3 倍;使用向量加载/存储指令将性能提升了 1.25 倍;而循环展开又将性能提升了 1.16 倍。对于最终的卷积内核,程序访存速度为 2.24 GB/s,达到了 GPU 访存峰值的 79.9%,对 GPU 进行了比较充分的利

用。图5是手机上运行我们应用程序的一个屏幕截图,可以看到图像经过CNN网络被正确分类(文本框中是分类的结果)。

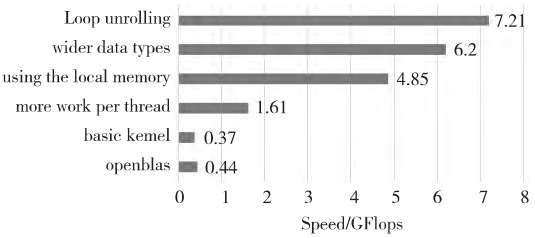

Figure 4 Comparison of the speed of matrix multiplication

图4 矩阵乘法速度比较

Figure 5 Screen capture of the App

图5 程序运行截图

尽管使用移动GPU来进行卷积计算会带来额外的开销,比如说数据的重新组织以及传输等,但是对于整体前向过程的测试结果显示,在整体时间上,本文的实现也取得了很好的加速效果。在此本文选取了不同的三种CNN网络来测试GPU实现在整体前向过程上的加速效果以及加速稳定性:AlexNet、Inception-Full Net以及Inception-Sub Net。AlexNet是首个在图像分类领域击败传统方法的CNN网络;Inception Net是由Szegedy[15]提出的一个比AlexNet更加复杂精确的网络,计算量也更大;这个网络有一个简化的、应用更加广泛的版本为Inception-sub。前向整体时间的测试结果显示在表1中。

Table 1 Forward time of different networks

表1 不同网络的前向时间

CNN类型	AlexNet	Inception-Sub Net	Inception-Full Net
CPU版本时间/s	1.30	12.05	21.93
GPU版本时间/s	0.61	5.66	10.31
加速比	2.131	2.129	2.127

可以看到,对于不同结构的CNN网络,本文的卷积实现都能取得大约2.1倍的前向过程加速比,因此程序对于不同的网络结构具有稳定性。

5 结束语

针对CNN网络中卷积计算负载较大的问题,本文在移动GPU上采取矩阵乘方法对卷积算法进行了实现。同时,通过探索利用优化任务划分、局部存储单元、向量化以及循环展开等技术来加速程序的执行速度。优化之后,矩阵乘法的执行速度由CPU版本的0.44 Gflops提升了16.39倍,达到了7.21 Gflops;同时实验结果还表明,本文的优化实现可以将整体前向的时间减少到只有原来的一半。

一个可以继续研究的问题依然是关于处理时间的。今后的工作重点将是利用CPU和GPU协同工作,以得到更快的速度。

参考文献:

[1] Krizhevsky A,Sutskever I,Hinton G E. Imagenet classification with deep convolutional neural networks[C]// Proc of International Conference on Advances in Neural Information Processing Systems, 2012: 1097-1105.

[2] Abdel-Hamid O, Mohamed A,Jiang H,et al. Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition[C]// Proc of 2012 IEEE International Conference on Acoustics,Speech and Signal Processing (ICASSP),2012: 4277-4280.

[3] Chen Y N,Han C C,Wang C T,et al. The application of a convolution neural network on face and license plate detection [C]// Proc of the 18th International Conference on Pattern Recognition,2006:552-555.

[4] He K,Zhang X,Ren S,et al. Deep residual learning for image recognition[C]// Proc of 2016 IEEE Conference on Computer Vision and Pattern Recognition, 2016:1.

[5] Chen T,Li M,Li Y,et al. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems[J]. arXiv preprint arXiv:1512.01274,2015.

[6] Leliana/WhatsThis[EB/OL]. [2015-11-11]. https://github.com/Leliana/WhatsThis.

[7] Munshi A. The openCL specification[EB/OL]. [2016-03-10]. http://www.khronos.org/opencl.

[8] Chellapilla K,Puri S,Simard P. High performance convolutional neural networks for document processing[C]// Proc of the 10th International Workshop on Frontiers in Handwriting Recognition,2006:1.

[9] Owens J D, Houston M, Luebke D, et al. GPU computing[J]. Proceedings of the IEEE, 2008, 96(5): 879-899.

[10] Qualcomm Inc. Qualcomm Adreno GPU[EB/OL]. [2013-01-11]. https://www.qualcomm.com/news/onq/2013/01/11/inside-snapdragon-800-series-processors-new-adreno-330-gpu.

[11] Gong Ruo-hao. Image codec parallel optimization based on embedded mobile GPU[D]. Chengdu: Southwest Jiaotong University, 2015. (in Chinese)

[12] Zeng Bao-guo, Yang Bin, et al, Parallelization of DFT based on embedded mobile GPU[J]. Microcontrolles & Embedded Systems, 2016: 16(1): 12-15. (in Chinese)

[13] Zhang Xian-yi, Wang Qian, Saar W. OpenBLAS library[EB/OL]. [2015-12-09]. https://www.openblas.net.

[14] XiaoMi. XiaoMi note[EB/OL]. [2015-01-15]. https://www.mi.com/minote.

[15] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[J]. arXiv preprint arXiv: 1409.4842, 2014.

附中文参考文献:

[11] 龚若皓. 基于嵌入式移动 GPU 的图像编解码并行优化[D]. 成都:西南交通大学, 2015.

[12] 曾宝国, 杨斌. 基于嵌入式移动 GPU 的离散傅里叶变换并行优化[J]. 单片机与嵌入式系统应用, 2016, 16(1): 12-15.

作者简介:

王湘新(1973-),男,辽宁海城人,硕士,高级工程师,研究方向为图像处理和云计算。**E-mail**:694558029@qq.com

WANG Xiang-xin, born in 1973, MS, senior engineer, his research interests include image processing, and cloud computing.

时洋(1992-),男,湖北丹江口人,硕士生,研究方向为深度学习和高性能计算。**E-mail**:shiyang9211@gmail.com

SHI Yang, born in 1992, MS candidate, his research interests include deep learning, and high performance computing.

文梅(1975-),女,湖南常德人,博士,研究员,博士生导师,研究方向为高性能微处理器体系结构和并行处理技术。**E-mail**:meiwen@nudt.edu.cn

WEN Mei, born in 1975, PhD, research fellow, PhD supervisor, her research interests include high performance micro-processor architecture, and parallel processing technology.