第36卷 第15期 计 算 机 工 程 2010年8月
Vol.36 No.15 Computer Engineering August 2010
·人工智能及识别技术· 文章编号：1000—3428(2010)15—0179—03 文献标识码：A 中图分类号：TP193

基于CUDA技术的卷积神经网络识别算法

张佳康，陈庆奎
(上海理工大学光电信息与计算机工程学院，上海 200093)

摘 要：针对具有高浮点运算能力的流处理器设备 GPU 对神经网络的适用性问题，提出卷积神经网络的并行化识别算法，采用计算统一设备架构(CUDA)技术，并定义其上的并行化数据结构，描述计算任务到 CUDA 的映射机制。实验结果证明，在 GTX200 硬件架构的 GPU 上实现的并行识别算法的平均浮点运算能力峰值较 CPU 上串行算法提高了近 60 倍，更适用于神经网络的相关应用。

关键词：流处理器；单指令多线程；GTX200 硬件架构；CUDA 技术；卷积神经网络

CUDA Technology Based Recognition Algorithm of Convolutional Neural Networks

ZHANG Jia-kang, CHEN Qing-kui
(School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093)

【Abstract】For the problem whether Graphic Processing Unit(GPU), the stream processor with high performance of floating-point computing is applicable to neural networks, this paper proposes the parallel recognition algorithm of Convolutional Neural Networks(CNNs). It adopts Compute Unified Device Architecture(CUDA) technology, definites the parallel data structures, and describes the mapping mechanism for computing tasks on CUDA. It compares the parallel recognition algorithm achieved on GPU of GTX200 hardware architecture with the serial algorithm on CPU. It improves speed by nearly 60 times. Result shows that GPU based the stream processor architecture are more applicable to some related applications about neural networks than CPU.

【Key words】stream processor; Single-Instruction Multiple-Thread(SIMT); GTX200 hardware architecture; Compute Unified Device Architecture (CUDA) technology; Convolutional Neural Networks(CNNs)

卷积神经网络(Convolutional Neural Networks, CNNs)的识别算法是高密集型计算，使用图形处理单元(Graphic Processing Unit, GPU)的流计算模型[1]来实现能将其性能发挥到更高的水平。

1 计算统一设备架构

硬件日渐成熟，在开发领域，计算统一设备架构(Compute Unified Device Architecture, CUDA)[2]技术的出现，开创出了 GPU 计算[3]的新时代，GPU 已经逐渐成为计算机中新的计算资源[3]。CUDA 编程模型见文献[4]。在模型中，基于 CUDA 开发的程序代码在实际执行中分为运行在 CPU 上的宿主代码(Host Code)和运行在 GPU 上的设备代码(Device Code)。不同类型的代码由于其运行的物理位置不同，能够访问到的资源不同。GPU 是一个能够处理大量并行线程的协处理器。线程被组织成线程块(Thread Blocks)，Block 再组织成 Grid，每一个 Block 拥有一个二维的 ID 作为标识，而一个线程可以拥有一个最多三维的 ID 作为标识，用为寻址使用。各线程在设备上执行同一个程序，它就是流计算模型中的“计算节点”，即 Kernel。而 Block 中的线程又会以 32 个为一组来创建、执行和调度，这 32 个线程被称为 Warp。Warp 块以单指令流多线程(Single-Instruction Multiple-Thread, SIMT)[4]架构管理运行各种不同程序的众多线程。

GTX280 在 NV GTX200 硬件架构[4]的支持下，此 GPU 具有 240 个流处理器(Stream Processors, SPs)。理论峰值浮点运算能力为 933 GFLOPS。其双精度 64 bit 理论峰值浮点运算能力为 933/8=116.625 GFLOPS，而实际运用中大概在 90 GFLOPS 左右。在存贮器层次结构上，具有 4 个不同类型的片上存储器。

2 五层卷积神经网络

体现卷积神经网络特征的 3 个基本概念为特征映射(Feature Map)、权重共享和子抽样[5-6]。五层 CNNs 结构如图 1 所示。此神经网络应用于手写数字识别。其中，前三层均由若干特征映射组成，每一层的特征映射的尺度较前一层都有所缩减，数量增加。各特征映射共享一个接受域和一个偏置。

C1 层为输入层，输入 29×29 的数字手写图像对应的{0,1}矩阵，所以在 C1 层就有 29×29 个神经元，即 841 个典型神经元组成。

C2 层是一个卷积层，分布了 6 个特征映射，使用一个 5×5 的接受域对 C1 层进行卷积。从 C1 层采样，C2 层的每个特征映射是由 13×13 个神经元组成的。因此，有 13×13×6=1 014 个节点(或神经元)，它到 C1 层的所有权重数量为

基金项目：国家自然科学基金资助项目(60573108)；上海教委发展基金资助项目(09YZ428)；上海教委科研创新基金资助重点项目(08ZZ76)；上海市重点学科建设基金资助项目(S30501)

作者简介：张佳康(1985－)，男，硕士研究生，主研方向：并行计算；陈庆奎，教授、博士生导师

收稿日期：2010-04-25 **E-mail：**shinson@126.com

(5×5+1)×6=156 个，有 1 014×(5×5+1)=26 364 条连接。

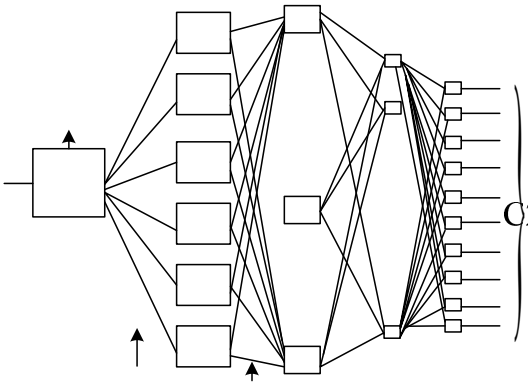

图 1　五层卷积网络拓扑图

C3 层也是一个卷积层，由比 C2 层尺寸更小、数量更多的特征映射组成，共有 50 个。每个特征映射包含一个 5×5 像素的图像，仍然使用一个 5×5 的接受域对 C2 层进行卷积，不同的是特征映射上的每一个像素(或神经元)是 C2 层中 6 个特征映射相应区域结合的一个 5×5 卷积核。在这层有 5×5×50=1 250 个神经元，它到 C2 层的所有权重数量为(5×5+1)×6×50= 7 800 个，有 1 250×(5×5+1)=32 500 条连接。

N1 层是一个具有 100 个典型神经元的全连通层，且不含有特征映射，仍共享一个偏置用于分类，与 C3 层的 1 250 个神经元都有连接，即 100×(1 250+1)=125 100 条连接，100×(1 250+1)=125 100 个权重。

N2 层是输出层，输出手写数字图像对应数字 0~9 的相似度集。它与 N1 层是完全连通的，具有 10 个神经元，用于二次分类和输出结果。有 10×(100+1)=1 010 个权重，以及 10×(100+1)=1 010 条连接。

其中，除 N2 层外，其他各层都使用双曲正切激活函数[2]：

$$\phi(x)=1.715\ 9\times\tanh(\frac{6x}{9}) \tag{1}$$

3　卷积神经网络并行识别算法

3.1　基于CUDA的并行化数据结构

如下是算法中涉及到的数据结构向 CUDA 映射后的具体描述，通过 ***blockIdx***(x,y)和 ***threadIdx***(x,y,z)这 2 个一维索引向量来寻址。

(1)卷积层特征映射

五层卷积神经网络可描述为一个由 2 个二维数组构成的数据结构(接受域、特征映射)，即 $\boldsymbol{FM}_{ij}=\{\boldsymbol{P}_{ij}[N][N], \boldsymbol{F}_{ij}[M][M]\}$，其中，$\boldsymbol{FM}_{ij}$ 表示第 i 层第 j 个特征映射；$\boldsymbol{P}_{ij}[N][N]$表示第 i 层第 j 个接受域为 $N\times N$ 二维数组 $\boldsymbol{P}_{ij}$；$\boldsymbol{F}_{ij}[M][M]$表示第 i 层第 j 个特征图为 $M\times M$ 二维数组 $\boldsymbol{F}_{ij}$。

以层为单位，映射到 CUDA 上后，构成一个一维的数组集，如下式所示：

$$\_\boldsymbol{FM}_i=\{\_\boldsymbol{P}_i[N\times N],\ \_\boldsymbol{F}_i[(M\times M)\times\sum_1^j j]$$
$$\{\_\boldsymbol{F}_{i1}[M\times M],\ \_\boldsymbol{F}_{i2}[M\times M],\ \cdots,\ \_\boldsymbol{F}_{ij}[M\times M]\}\} \tag{2}$$

其中，$\_\boldsymbol{P}_i=\{Pixel\}_{N\times N}$；$\_\boldsymbol{F}_{ij}=\{Neuron_{CNN}\}_{M\times M}$，$i$=1,2,3。$\_\boldsymbol{FM}_i$ 表示第 i 层特征映射集，每层特征映射共享一个 $N\times N$ 接受域 $\_\boldsymbol{P}_i$，接受来自上层特征图的采样值，$\_\boldsymbol{F}_{ij}$ 表示第 i 层第 j 个特征图，且为一维数组，由 $M\times M$ 个卷积神经元组成，它们组成 $\_\boldsymbol{F}_i$，表示所有第 i 层特征图组成的一维数组。j 的范围取决于该层的特征图数量。但 i=1 时，不包含接受域 $\_\boldsymbol{P}_i$。

(2)各层与各层之间的连接权重

卷积层的每个特征映射相对于上层特征映射的权重可表示为 $\boldsymbol{W}_{ij}\{\boldsymbol{wd}_{ij}[N][N],\ e_{ij}\}$，$\boldsymbol{wd}_{ij}$ 表示与接受域相对的 $N\times N$ 第 i 层第 j 个特征映射链接权重的二维数组，每个特征映射共享一组权重数和偏置数 e_{ij}，它们组合成一维数组 $\_\boldsymbol{W}_i$'映射到 CUDA，而对于普通层的权重集，定义如下式所示：

$$\_\boldsymbol{W}_i'=\{\{\_e_{i1},\bigcup_{j=1}^{n}\_w_{ij}^1\},\{\_e_{i2},\bigcup_{j=2}^{n}\_w_{ij}^2\},\cdots,$$
$$\{\_e_{is},\bigcup_{j=s}^{n}\_w_{ij}^s\}\}\qquad j\leqslant n,\ s\leqslant m,\ i=3,4 \tag{3}$$

(3)2 种神经元行为定义

卷积神经元行为如下式所示：

$$Neuron_{CNN}=f(\sum_{j=1}^{N\times N}\_w_{ik}[j]pixel_j+\_e_{ik})$$
$$\_w_{ik}[j]\in\_\boldsymbol{wd}_{ik}[N\times N],\ pixel_j\in\_\boldsymbol{P},\ \_e_{ik}\in\_\boldsymbol{W}_i' \tag{4}$$

其中，N 从接受域 $\_P$ 中取得；k 为第 k 个特征图。

典型神经元行为如下式所示：

$$Neuron_{is}=f(\_e_{is}+\sum_{j=1}^{n}\_w_{ij}^s u_j)$$
$$\_w_{ij}^s\in\_\boldsymbol{W}_i',u_j\in Neuron_{i-1,s},\ \_e_{is}\in\_\boldsymbol{W}_i' \tag{5}$$

其中，u_j 为链接始点对应的神经元计算结果，即上层的输出。

3.2　线程设置

GTX280 的计算能力为 1.3 的规范[4]，参照规范，从上到下各层之间的 Kernel 具体线程设置情况如表 1 所示。

表 1　线程设置和流处理器分配情况

内核 (Kernel)	Grid 维度 (dim3)	Block 维度 (dim3)	总线程数 /个
1	(6, 1)	(13, 13)	1 014
2	(50, 1)	(5, 5)	1 250
3	(100, 1)	(1, 1)	100
4	(10, 1)	(1, 1)	10

3.3　待识别个体识别结果结构分析

经过上述过程，待识别个体最后被分类成一组与 0~9 数字相关的相似度集 $\boldsymbol{U}\{u_i\}$, i=0,1,⋯,9，由待识别个体数字 0~9 的相似度 u 组成，u 可正可负。最终由 MAX{$\boldsymbol{U}$}得出识别结果。

4　关键算法描述

设 *dimGrid* 和 *dimBlock* 分别为 Grid 维度和 Block 维度。

4.1　卷积计算算法CC

卷积计算算法 CC 如下：

(1)内核启动 Kernel<<<dimGrid,dimBlock>>>

(2)For i=1 To n×n DO(并行地)

在 Device 的共享内存内初始化卷积结果 r

(3)__shared__ double r=0;

(4)索引数据位置，blockID=blockIdx.x;

(5)取_W_i中偏置 e_{is};

(6)根据接受域_P_i的大小循环执行 1)~4)步

1)根据_P_i从矩阵_FM_i中采样;

2)按照式(4)执行卷积;

3)调用 4.2 节算法数据收集;

4)使用式(1)的激活函数对神经元的输出幅度进行调整;

End For

(7)线程同步__syncthreads()

(8)算法结束

4.2　数据收集算法DC

数据收集算法 DC 如下：

(1)For idx=1 To n Do(并行地)

For idy=1 To n Do

使用共享存储器(shared memory)收集数据

Thread[idx*pitch+idy]执行_FM_i[g(idx, idy)]=(__shared__ double) result;

End For

End For

(2)算法结束

4.3 识别分类算法DCL

识别分类算法 DCL 如下:

(1)For i=1 To n×n DO(并行地)

在 Device 的共享内存内初始化分类结果 cr

(2)__shared__ double cr=0;

(3)索引数据位置，blockID=blockIdx.x;

(4)按照式(5)执行分类;

(5)调用 4.2 节算法数据收集;

(6)使用式(1)的激活函数对神经元的输出幅度进行调整;

End For

(7)数据返回用户区

(8)算法结束

5 对比实验

测试在 NVIDIA GeForce GTX280 上进行，版载全局内存为 1 GB。此 GPU 搭载在配备了 Intel Core2 E8400 3.0 GHz 的 PC 机上。

为了使实验结果具有可比性，使用 MINST[6]手写数字字符库和自建库分别在 CPU 上和 GPU 上进行了比较。实验使用 Mike O'Neill 采用 MNIST 库的训练方法得到的权重数据来进行。结果显示，自建库和 MNIST 库的准确率分别在 93%和 95%左右，如表 2 所示。

表 2 数字识别检测结果

识别库	图像数	正确检测数	正检率/(%)
MNIST	10 000	9 570	95.7
		9 450	94.5
		9 620	96.2
自建库	1 000	935	93.5
		941	94.1
		927	92.7

在计算精度上，CUDA 技术和面向 x86 架构 CPU 的高级语言在技术处理上有所不同。在 CPU 中，目前大多数高级语言(包括 C)都按照 IEEE-754 标准来规定浮点数的存储格式。在 CUDA 中，计算设备也遵循单精度的二进制浮点数 IEEE-754 标准，不同的是(此处仅部分列举，具体见文献[4]):

(1)加法和乘法通常被合并成一个乘加指令(FMAD);

(2)除法通过非标准兼容的倒数实现;

(3)平方根通过非标准兼容的平方根倒数实现;

(4)不支持直接舍入到正/负无穷;

(5)没有动态配置的舍入模式;

(6)没有浮点异常的监测机制，浮点异常总是被记录的;

(7)一个操作的结果包含一个或多个 NaN，NaN 的位模式是 0x7FFFFFFF。

GPU 和 CPU 最后输出的手写数字相对于 0~9 数字的相似度有微小误差，由图 2 可以看到各个数字在 CPU 和 GPU 上 1 000 次识别输出的相似度的标准方差的数量级都在 10^{-7}，计算误差足够小，而且识别正确与否取决于各个数字相似度之间差值程度，所以 GPU 上识别检测正确率与 CPU 的完全相同，而速度上却要相差 2 个数量级。图 3 为此应用在 CPU 和 GPU 的浮点运算能力的比较，GPU 和 CPU 的平均浮点运算能力峰值相差最高达到 60 倍左右，随着识别次数的增加，浮点运算能力表现平稳，呈线性态势。

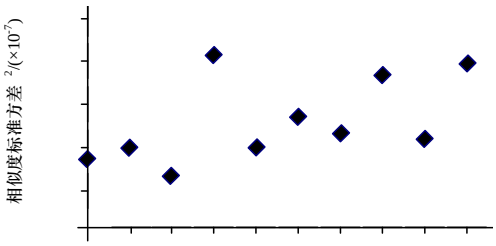

图 2 1 000 次识别输出“相似度”的标准方差

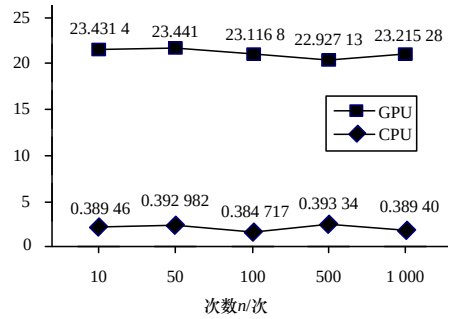

图 3 *n* 次识别的平均浮点运算能力比较

6 结束语

卷积神经网络虽然拓扑结构简单，但仍然需要巨大计算量。NVIDIA 的 GPU 凭借基于流处理器的硬件架构，在 CUDA 编程模型的支持下对基于卷积神经网络的手写识别性能提升明显，相对于 CPU 发挥出了惊人的优势，实验表明流处理器架构适合卷积神经网络。然而，由于在输入比较大的情况下，介于设备数据传输带宽的限制，可能会成为流处理器的一个瓶颈，进一步地提高流处理器的利用率、合理地调度和分配数据能更好地优化神经网络的各种应用。

参考文献

[1] Zhang Ying, Yang Xuejun, Wang Guibin, et al. Scientific Computing Applications on a Stream Processor[C]//Proc. of IEEE Int'l Symp. on Performance Analysis of Systems and Software. Austin, Texas, USA: [s. n.], 2008: 105-114.

[2] Luebke D. CUDA: Scalable Parallel Programming for High-performance Scientific Computing[C]//Proc. of the 5th IEEE Int'l Symp. on Biomedical Imaging: From Nano to Macro. Paris, France: [s. n.], 2008: 836-838.

[3] 田 文, 徐 帆, 王宏远, 等. 基于 CUDA 的尺度不变特征变换快速算法[J]. 计算机工程, 2010, 36(8): 219-221.

[4] NVIDIA Corporation. NVIDIA CUDA Compute Unified Device Architecture Programming Guide 2.0[EB/OL]. (2008-06-07). http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf.

[5] Lawrence S, Giles C L, Tsoi A C. Convolutional Neural Networks for Face Recognition[C]//Proc. of IEEE Computer Society Conference on CVPR. San Francisco, California, USA: [s. n.], 1996: 217-222.

[6] 肖柏旭, 张丽静. 基于分流抑制机制的卷积神经网络人脸检测法[J]. 计算机应用, 2006, 26(z1): 46-48.

编辑 任吉慧