DOI:10.3969/j.issn.1000-1565.2019.01.017

基于 FPGA 的卷积神经网络加速系统

李小燕[1]，张欣[1]，闫小兵[1]，任德亮[1]，李彦青[2]，傅长娟[2]
(1.河北大学 电信与信息工程系,河北 保定 071002;2. 保定永红铸造机械厂,河北 保定 072150)

摘 要:以在现场可编程门阵列(FPGA)上部署卷积神经网络为背景,提出了卷积神经网络在硬件上进行并行加速的方案.主要是通过分析卷积神经网络的结构特点,对数据的存储、读取、搬移以流水式的方式进行,对卷积神经网络中的每一层内的卷积运算单元进行展开,加速乘加操作. 基于 FPGA 特有的并行化结构和流水线的处理方式可以很好地提升运算效率,从对 ciafr-10 数据集的物体分类结果看,在不损失正确率的前提下,当时钟工作在 800 MHz 时,相较于中端的 Intel 处理器,可实现 4 倍左右的加速.卷积神经网络通过循环展开并行处理以及多级流水线的处理方式,可以加速卷积神经网络的前向传播,适合于实际工程任务中的需要.

关键词:现场可编程门阵列(FPGA);卷积神经网络;并行化;流水线;分类;加速
中图分类号:TP391 **文献标志码**:A **文章编号**:1000-1565(2019)01-0099-07

Convolutional neural network acceleration system based on FPGA

LI Xiaoyan[1], ZHANG Xin[1], YAN Xiaobing[1], REN Deliang[1], LI Yanqing[2], FU Changjuan[2]
(1. College of Telecommunications and Information Engineering, Hebei University, Baoding 071002, China; 2. Baoding Yonghong Foundry Machinery Factory, Baoding 072150, China)

Abstract: In this paper, the convolutional neural network is deployed on the Field Programmable Gate Array (FPGA). As a background, a convolutional neural network is proposed to accelerate hardware. The paper analyzes the structural characteristics of convolutional neural networks, stores, reads, and moves data in a stream-style manner. Next, the convolution unit in each layer of the convolutional neural network is expanded to speed up the multiplication and addition operations. Based on the (FPGA) unique parallel structure, pipeline processing method can effectively improve the efficiency of the operation. From object classification results for the ciafr-10 dataset, at 800MHz operating frequency and without loss of accuracy, FPGA compared to General purpose processor can achieve 4 times speed up, Convolutional neural network through parallel process and multi-stage pipeline process can accelerate forward propagation of convolutional neural networks, being suitable for the demand of practical engineering tasks.

Key words: field programmable gate array (FPGA); convolutional neural network; parallelization; stream-style; classification; accelerate

收稿日期:2018-09-02
基金项目:国家自然科学基金资助项目(61674050)
第一作者:李小燕(1994—),女,湖北十堰人,河北大学在读硕士研究生,主要从事忆阻器等新型电子器件集成和用于集成的逻辑控制嵌入式电路设计研究.E-mail:18612969742@163.com
通信作者:张欣(1966—),男,河北承德人,河北大学教授,主要从事机器视觉和图像处理方向的研究.
E-mail: zhangxin@hbu.edu.cn

现有的绝大多数 CNN 算法早期一直部署在通用芯片 CPU 或 GPU 上,在 CNN 网络结构中,单独每层内的计算是相互独立不相关的,而层与层之间是以一个流水的结构进行连接. CPU 由于自身体系架构的限制[1],在处理 CNN 这种并行化程度高、简单运算次数较多的网络时并不是很理想. GPU 本身是利用堆叠 CUDA 核心来实现并行化处理的,核心数目的增多带来的是功耗和体积的增加. 随着摩尔定律无以为继,要获得更高性能、更低功耗,芯片只能做得越来越专用[2]. 现场可编程门阵列(field programmable gate array, FPGA) 作为一种可以反复自定义布局和布线的硬件器件[3],结构灵活、资源丰富,将算法映射到 FPGA 的硬件模块上,各个模块相互连接,并行执行,特有的流水结构可以很好与 CNN 算法相匹配,充分挖掘卷积神经网络[4]中固有的并行性. 在深度学习的 inference 方面,FPGA 正被越来越多的研究者所关注.

本文从 CNN 的网络结构入手,简单介绍卷积神经网络 CNN,重点在挖掘网络结构中存在的并行性,并在 FPGA 上实现硬件加速方案, 最后采用 cifar-10 数据集[5]作为整个实验的分类效果检测的样本.

1 FPGA 方案设计

1.1 总体设计

系统总体采用软硬件结合的方式,对输入的目标图像进行分类,图 1 表示系统的总体结构.系统首先在 PC 端的 linux 系统下,搭建卷积神经网络框架,并对其进行多次的训练和调优,得到合适的权重值和偏置值[6],并将其提取出来.同时系统在 FPGA 端搭建卷积神经网络硬件平台,将 PC 端提取的权重值和偏置值送入 FPGA 中进行应用.最终在硬件平台上实现对目标图像的准确分类,并实现加速的效果.

1.2 卷积神经网络结构设计与分析

为了能更好地在硬件上实现,本文所采用的 CNN 模型结构是由 1 个数据输入层 input、1 个输出层 output,2 层卷积层以及 1 个全连接网络 Softmax 组成. 卷积层之间通过激活函数层衔接,用到的激活函数是 ReLU,模型结构如图 2 所示. 在本文实验中,用到的图像数据库是 cifar-10. 数据库中图像尺寸为 32×32 像素点,其中 kernel 代表卷积核,都采用大小为 5×5 的卷积核对特征图像进行卷积操作,num output 表示卷积核的数量,stride 表示卷积核在图像上每次平移的步数. 具体的网络结构如图 2.

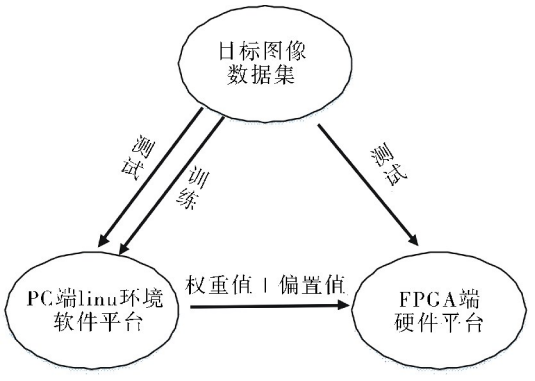

图 1 系统整体结构

Fig.1 Overall structure of the system

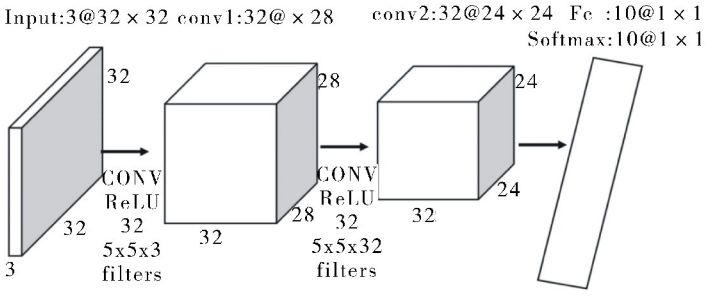

图 2 cifar-10 图像分类的网络模型结构

Fig.2 cifar-10 image classification -model structure

Input: 32×32×3, batch_size:100

Convolution1:

num_output:32

kernel_size: 5×5×3 stride: 1

ReLU 1: relu1

Convolution 2:

num_output: 32

kernel_size: 5×5×32, stride: 1

ReLU 2: relu2

Softmax:output:10 vector

(classification result)

在上述模型中,权重系数的数目 w = 卷积核的数目×单个卷积核的 w 参数的数目.

1)conv1:$w = 32\times(5\times5\times3+1)$, (+1 for bias);

2)conv2:$w = 32\times(5\times5\times32+1)$,(+1 for bias);

3)Fc:$w = 10\times(24\times24\times32+1)$, (+1 for bias).

经过计算,以上 3 层的参数总和为 212 394 个,如果采用 float 来存储这些参数的话,每个参数占 32 位,则总共消耗的存储空间约为 830 kB,这些参数在硬件里固定的存储在内存中,通过 DMA 读取进行卷积操作.

1.3　硬件平台搭建

FPGA 主要做深度学习的 inference,通过分析上述模型以及实现方式,可以看出 CNN 算法中在进行前向传播时层与层之间是顺序执行[7]的,每一层的输出结果作为下一层的输入,直至输出最终的分类结果. FPGA 的架构不同于传统的通用处理器,它自身的硬件模块是可以进行复制以及搬移使用的. CNN 算法中卷积层中存在很多可以复用、并行的部分[8],因此可以利用 FPGA 的硬件资源加速实现.

本文的硬件系统中,采用的 Xilinx 公司的 zynq 系列的芯片. Zynq 芯片采用 ARM+FPGA 的异构方式. ARM 端也称为 PS(Programming System)端,FPGA 端也称为 PL(Programming Logic)端,中间通过 AXI 总线连接. 图像数据的输入是通过 CMOS 摄像头采集后以 DMA 的方式写入 DDR 中,CNN 算法中涉及到的参数也是存储在 DDR 中,数据的读写以及传输都是以数据流的方式传递[9]. PS 端主要负责指挥协调整个算法的执行,PL 端主要负责利用硬件实现部分算法的加速[10],具体的硬件架构如图 3 所示.

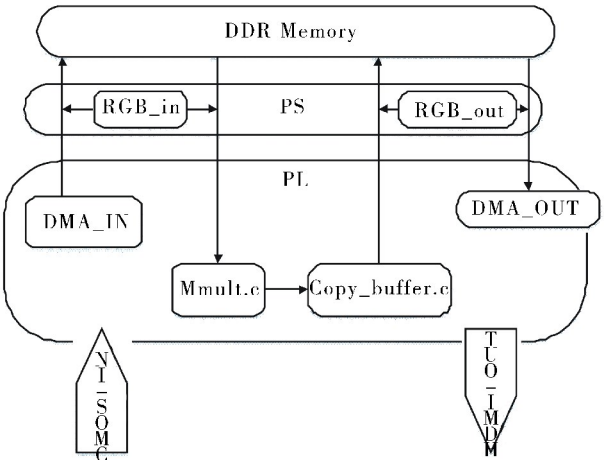

图 3　硬件部署 CNN 算法

Fig. 3　CNN on hardware

2　核心加速单元设计

在结合算法分析上述硬件系统时可以发现,时间消耗主要是在于内存中图像数据和参数的访问,以及多

层次的卷积运算. 利用 FPGA 中的丰富资源[11],可以针对上述 2 个问题作出如下优化:

1)内存在进行数据的访问搬移时,从整体上来讲可以分为 3 个流程,即数据的读取、数据的计算、数据的存储. 采用 PIPELINE[12]的方式,缩短整体的时间,假设每一步的操作都需要 1 个指令周期,则数据的搬移的时延为 3 个指令周期,循环时延需要 6 个指令周期. 以插入 1 级流水为例,数据的搬移的时延为 3 个指令周期,循环时延为 4 个指令周期,具体示意图如图 4 所示.

卷积核每执行 1 行的卷积操作,都需要从内存中读取 2 个区域的参数,1 个是图像的像素,1 个是卷积核的参数. FPGA 可以通过使用 ARRAY_PARTITION 操作把大面积的数据分拆成多组较小的数组块或者独立的寄存器,用于增加访问的并行性,消除 Block RAM 访问的瓶颈. 这样单个周期就可以从内存中读取 25 个像素值和 25 个参数值,将速度通过处理器提高了 25 倍.

2)在进行每一层卷积时,主要需要 4 级 for 循环结构来实现:第 1 级 for 循环是为了遍历所有的卷积核;第 2 级 for 循环是为了遍历图像的所有通道数;第 3 级 for 循环实现遍历输出图像的每一像素点;第 4 级 for 循环实现遍历卷积核中的每一点.

①通过分析算法,第 4 级算法主要进行的是卷积运算[13],以 5×5 卷积为例,每卷积 1 次,需要进行 25 次乘法和 24 次加法,前 3 级的 for 循环都是在复用这个卷积运算. FPGA 中具有丰富的 DSP4148 内核,可以利用硬件去完成乘加运算. FPGA 在调用重复的硬件模块之前加入 INLINE 申明,可以实现函数的内联,同时实现跨越函数边界的逻辑优化,直接复制硬件单元,降低函数调用的开销. 卷积层 1 是 32×32 图像与 5×5 卷积核进行卷积,得到 28×28 的特征图像. 卷积层 2 是 28×28 图像与 5×5 卷积核进行卷积,得到 24×24 的特征图像. 为了在 2 个卷积层中都可以复用乘加模块 Mmult,需要对第 1 次输出的特征图做补 0 操作,具体效果如图 5 所示.

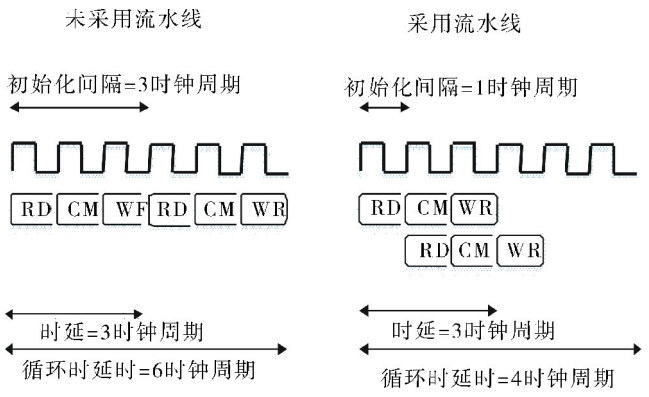

图 4 PIPELINE 一级流水示意

Fig. 4 PIPELINE First-grade stream

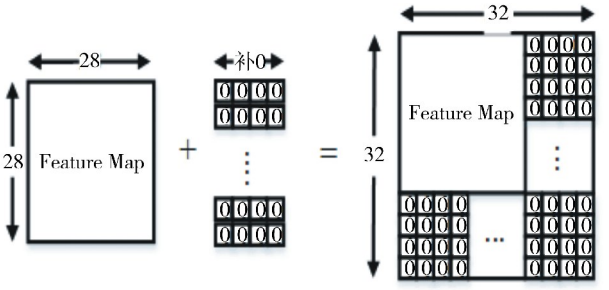

图 5 特征图补零操作

Fig. 5 Feature map zero-fill operation

②乘加模块 Mmult 可以实现复用之后,通过分析硬件电路中数据流[14]的特点,在通用处理器中,1 个周期只可以执行 1 次乘法或者加法操作. 以 5×5 卷积为例,每卷积 1 次,需要进行 25 次乘法和 24 次加法,也就是 49 个周期. 在 FPGA 中利用 UNROLL 指令,可以将 for 循环展开,提高程序在执行循环迭代时的并行性. 采用空间换时间的方法,1 个指令周期,就可以将多次的乘加运算执行完毕,极大地提高了代码的执行效率.

3 实验结果及分析

3.1 分类结果验证

本文实验主要分为 2 部分:第 1 部分是利用 cifar-10 数据集,在 PC 端 ubuntu 系统中搭建 caffe 环境进

行训练生成权值与偏置参数的过程;第 2 部分是在 eaglego 系列的 FPGA 开发板上进行算法的移植应用. 在移植算法时用到的工具是 xilinx 官方正在推出的 SDSOC. SDSOC 包括 Vivado&SDSOC 工具,Vivado 主要进行 PL 部分的设计,SDSOC 主要进行 PS 部分的设计.

图 6 表示在 PC 端训练完成后测试的效果,可以看到测试的图片是一张 cat,PC 端结果中 cat 的计算得到的概率值为 0.535 5,在 cat、truck、deer、frog、horse 这 5 类结果中所占比例最大,也就输出了最终的测试结果.

图 7 表示在 FPGA 开发板上的测试结果. FPGA 开发板通过摄像头拍摄目标图像,并将结果输出至 HDMI 显示,同时将分类结果显示在图像的左上角处,可以看到分类的结果是正确的.

图 6　PC 端测试结果

Fig. 6　Test results on PC

图 7　FPGA 端测试结果

Fig. 7　Test results on FPGA

3.2　分类速度验证

在进行实验的过程中,FPGA 的开发工具选用的是 Xilinx 公司的 SDSoc 软件, FPGA 芯片选用的是 Xilinx 公司的 ZYNQ7020, FPGA 的频率设置为 100 MHz.摄像头的分辨率为 1 080 P,在实验中采用 zoom 函数将图像缩放至 32×32.算法在 SDSoc 工具编译后会生成 FPGA 的硬件资源消耗情况,如表 1 所示.

表 1　FPGA 硬件资源消耗情况

Tab. 1　FPGA Hardware resource consumption

模块单元	使用情况/个	资源总量/个	资源使用率/%
DSP	45	220	20.45
BRAM	72	140	51.43
LUT	20 555	53 200	38.64
FF	19 600	106 400	18.42

在上述网络的情况下,FPGA 的资源中 BRAM 的利用率最高是 50%左右,可见网络参数及图像像素的读取与搬移[15]也是限制网络层数的一个主要因素,如果在硬件上部署更深层次的网络,DSP 内核增加的同时,带宽也要随之增加.

在实验中的 ZYNQ 系列的芯片中含有一个双核的 ARM,内核是 Cortex-A9. 同时加速方案是 ARM+FPGA 的这种异构形式,对照实验是在 PC 端完成的. PC 端处理器是 Intel core-i5 系列,4 核,主频

2.5 GHz,系统是 window7 下 64 位的虚拟机中安装了 Ubuntu16.04,分配硬盘空间 20 G,运行内存 1 G. 在以上 3 种器件中运行基于 cifar-10 的分类网络,通过人工秒表计时的方式,记下对应平台的多帧图像的识别的起始和终止时间,求出时间差的平均值,进而求出单帧图像的识别时间.对时间进行统计整理成表,具体性能对比如表 2 所示.

表 2 cifar-10 物体分类时间对比

Tab.2 cifar-10 Object classification time comparison

平台	图像迭代/次	单帧时间/s
ARM Cortex-A9	10 000	5.60
Intel core- i5	10 000	0.15
FPGA-7020	10 000	0.27

上述结果表明,在 100 MHz 的情况下,使用 Xilinx 公司最低端的 zynq-7020 芯片就可以达到 PC 端处理器的一半水平,zynq-7020 芯片最高可以倍频到 800 MHz,性能也将提升 8 倍左右,可以实现通用处理器4 倍的加速效果.

4 结束语

重在设计一种卷积神经网络的加速器,通过压缩卷积神经网络的本身结构,设计出了层间独立,层内深度并行流水的加速方案. 通过在 PC 端进行训练得到合适的卷积核参数及偏置值,部署在 FPGA 这种专门针对可并行化算法的加速器件,在内存数据的读取、卷积运算等方面进行了优化,使得在 1 个时钟周期内就可以完成多步的乘加操作. 在 cifar-10 数据集上,在不影响识别率的情况下,对网络结构进行了裁剪,去掉了池化层,保存了更多的图像信息. FPGA 在 100 M 时钟的情况下对单张 32×32 的图像的识别时间达到了 0.27 s,可以实现 4～5 帧/s 取得了较好的效果,后续工作可以在资源更丰富、主频更高的 FPGA 上,部署更加复杂、效果更好的网络.

参 考 文 献:

[1] LI H, FAN X, JIAO L, et al. A high performance FPGA-based accelerator for large-scale convolutional neural networks [C]//IEEE, 26th International Conference on Field Programmable Logic and Applications, 2016:1-9. DOI: 10.1109/FPL.2016.7577308.

[2] ZHANG C, LI P, SUN G, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks[C]// ACM, Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2015:161-170. DOI: 10.1145/2684746.2689060.

[3] FARABET C, POULET C, HAN J Y, et al. CNP: An FPGA-based processor for convolutional networks[C]// IEEE, International Conference on Field Programmable Logic and Applications, 2009:32－37. DOI: 10.1109/FPL.2009.5272559.

[4] PERKO M, FAJFAR I, TUMA T, et al. Low-cost, high-performance CNN simulator implemented in FPGA[C]// IEEE, Proceedings of the 2000 6th IEEE International Workshop on Cellular Neural Networks and Their Applications, 2000:277-282. DOI: 10.1109/CNNA.2000.876858.

[5] FARABET C, POULET C, LECUN Y. An FPGA-based stream processor for embedded real-time vision with Convolu-

tional Networks[C]// IEEE, 12th International Conference on Computer Vision Workshops, 2009:878-885. DOI: 10.1109/ICCVW. 2009. 5457611.

[6] SANKARADAS M, JAKKULA V, CADAMBI S, et al. A massively parallel coprocessor for convolutional neural networks[C]//IEEE, 20th IEEE International Conference on Application-Specific Systems, Architectures and Processors, 2009:53-60. DOI: 10.1109/ASAP.2009.25.

[7] MA Y, CAO Y, VRUDHULA S, et al. Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks[C]//ACM, Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2017:45－54. DOI:10.1145/3020078.3021736.

[8] CHANG J W, KANG S J. Optimizing FPGA-based convolutional neural networks accelerator for image super-resolution [C]//IEEE Press, Proceedings of the 23rd Asia and South Pacific Design Automation Conference, 2018: 343-348. DOI: 10.1109/ASPDAC.2018.8297347.

[9] GARCIA C, DELAKIS M. A neural architecture for fast and robust face detection [C]//IEEE, 16th International Conference on Pattern Recognition. 2002, 2(11):44-47. DOI: 10.1109/ICPR.2002.1048232.

[10] DELAKIS M, GARCIA C. Text detection with convolutional neural networks[C]// Proceedings of the Third International Conference on Computer Vision Theory and Applications, 2008:290-294.

[11] ZHANG C, LI P, SUN G, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks [C]//ACM, Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2015:161-170. DIO: 10.1145/2684746.2689060.

[12] PEEMEN M, SETIO A A A, MESMAN B, et al. Memory-centric accelerator design for convolutional neural networks [C]//Proceedings of the 2013 IEEE 31th International Conference on Computer Design, 2013:13-19. DOI: 10.1109/ICCD.2013.6657019.

[13] LI N, TAKAKI S, TOMIOKAY Y, et al. A multistage dataflow implementation of a deep convolutional neural network based on FPGA for high-speed object recognition[C]//IEEE, 2016 IEEE Southwest Symposium on Image Analysis and Interpretation, 2016:165-168. DOI: 10.1109/SSIAI.2016.7459201.

[14] BACIS M, NATALE G, SOZZO E D, et al. A pipelined and scalable dataflow implementation of convolutional neural networks on FPGA[C]//IEEE, Parallel and Distributed Processing Symposium Workshops, 2017:90-97. DOI:10.1109/IPDPSW.2017.44.

[15] NATALE G, BACIS M, SANTAMBROGIO M D. On how to design dataflow FPGA-based accelerators for convolutional neural networks[C]//IEEE, 2017 IEEE Computer Society Annual Symposium on VLSI, 2017:639-644. DOI: 10.1109/ISVLSI.2017.126.

(责任编辑:孟素兰)