改进的基于嵌入式 SoC 卷积神经网络识别模型

孙 磊 肖金球 夏 禹 顾敏明
(苏州科技大学电子与信息工程学院 江苏 苏州 215009)
(苏州科技大学苏州市智能测控工程技术研究中心 江苏 苏州 215009)

摘 要 针对当前在 FPGA 上实现卷积神经网络模型时卷积计算消耗资源大,提高 FPGA 芯片性能代价较大等问题,提出一种改进的基于嵌入式 SoC 的优化设计方法。对卷积计算的实现方法和存储访问通道加以优化,以提高并行计算性能;将 32 位位宽的浮点数量化为 16 位定点数,加快前向传播的数据传输;结合硬件描述软件的高层次综合技术,将卷积神经网络映射到硬件平台成为一种同步数据流模型从而加快计算速度。通过实验证明,该方案较现有设计节约了 89% 的 BRAM 和 72% 的 LUT,在工作频率为 100 MHz 的测试中,其处理速度比单独使用 Cortex-A9 的方案提升了 42 倍。

关键词 卷积神经网络 嵌入式系统 FPGA 定点数量化

中图分类号 TP391 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2020.03.043

IMPROVED CONVOLUTIONAL NEURAL NETWORK RECOGNITION MODEL BASED ON EMBEDDED SOC

Sun Lei Xiao Jinqiu Xia Yu Gu Minming
(*School of Electronics and Information Engineering, Suzhou University of Science and Technology, Suzhou 215009, Jiangsu, China*)
(*Intelligent Measurement and Control Engineering Technology Research Center, Suzhou University of Science and Technology, Suzhou 215009, Jiangsu, China*)

Abstract Aiming at the problems that convolution computing consumes a lot of resources and costs a lot to improve the performance of the FPGA chip when implementing the convolutional neural network model on FPGA, we propose an improved design method based on embedded SoC. The implementation method and storage access channel of convolutional computing were optimized to improve the performance of parallel computation. Then, the 32-bit-wide floating-point number was quantified to 16-bit fixed-point number to speed up the data transmission of forward propagation. We combined the high-level synthesis technology of hardware description software, and the convolutional neural network was mapped to the hardware platform to become a synchronous data flow model to speed up the calculation speed. The experimental results show that the proposed optimization scheme saves 89% of BRAM and 72% of UT, compared with the existing design. The processing speed of the proposed scheme is 42 times faster than the scheme using Cortex-A9 alone in the 100 Mhz.

Keywords CNN Embedded systems FPGA Fixed-point quantization

0 引 言

利用 FPGA 并行计算的特点可以加速网络的运算,但是需要消耗较多的逻辑资源。RTL 的设计方法即使在资源配置和计算性能上有一定的优势,但开发周期长、难度大、门槛高。Venieris 等[1]设计了一种将卷积神经网络映射到 FPGA 平台的自动化设计方法,大大减少了设计难度。卢冶等[2]提出一种面向边缘计算的嵌入式 FPGA 平台的构建方法,与多种平台比较

收稿日期:2019-05-09。江苏省产学研前瞻性联合基金项目(BY2011132);江苏省研究生创新与教改项目(09150001)。
孙磊,硕士生,主研领域:机器学习,嵌入式开发。**肖金球**,教授。**夏禹**,硕士生。**顾敏明**,实验师。

显现出优势和可行性。仇越等[3]提出了基于 Zynq 加速器的实现方法,并利用高层次综合技术和嵌入式 SoC 对卷积神经网络进行优化设计。Stornaiuolo 等[4]提出快速部署 Overlay 的设计方法,以 Xilinx 公司的 Vivado 套件和 PYNQ(Python Productivity for Zynq)为实验平台进行高层次综合设计,进一步加快了深度学习网络在嵌入式 SoC 上的部署速度。

本文以 PYNQ 为硬件实现平台,将卷积神经网络进行离线训练得到权重、偏置参数的模型文件,并设计定点数量化的算法;在嵌入式系统下调用 FPGA 资源,设计卷积神经网络和加速计算,对已有的加速器进行改进和优化,将实验数据与 ZynqNet[3] 实验方法比较得出了更佳的效果,与单 CPU 处理器计算卷积神经网络比较具有更快的计算速度。

1 卷积神经网络和嵌入式 SoC

1.1 卷积神经网络模型

通常卷积神经网络包括卷积层、池化层、激活函数(非线性 ReLU)和全连接层。它们依次作用于特征图,每一层从前一层读取、执行和输出。图 1 所示为经典 LeNet-5 手写识别模型。

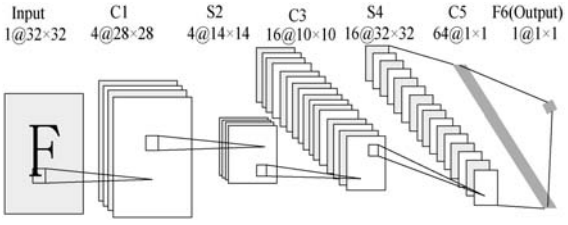

图 1 LeNet-5 手写识别模型

卷积层通过输入特征图和权重组成的卷积核进行二维卷积操作如式(1),输出通常直接连接到非线性单元,进行非线性激活。式(2)指池化层采用最大值采样或者均值采样,可以显著降低网络的计算复杂度。式(3)是全连接层在输入特征图和权重之间执行点积。全连接层被放置在网络的末端,并减小输出值。ReLU 是执行线性激活,阈值为 0。ReLU 在卷积神经网络中应用广泛,可以显著加快随机梯度下降的收敛速度,减少训练时间,如式(4)所示。

$$f_i^{\text{output}} = \sum_{j=1}^{n_{in}} f_j^{\text{in}} \times w_{i,j} + b_i \quad 1 \leqslant i \leqslant n_{\text{out}} \tag{1}$$

$$f_{i,j}^{\text{output}} = \max_{p\times p}\begin{pmatrix} f_{m,n}^{\text{input}} & f_{m,n+1}^{\text{input}} & \cdots & f_{m,n+p-1}^{\text{input}} \\ f_{m+1,n}^{\text{input}} & f_{m+1,n+1}^{\text{input}} & \cdots & f_{m+1,n+p-1}^{\text{input}} \\ \vdots & \vdots & & \vdots \\ f_{m+p-1,n}^{\text{input}} & f_{m+p-1,n+1}^{\text{input}} & \cdots & f_{m+p-1,n+p-1}^{\text{input}} \end{pmatrix} \tag{2}$$

$$f^{\text{output}} = \sum_{j=1}^{n} f_j^{\text{in}} w + b \tag{3}$$

$$f^{\text{output}} = MAX(f^{\text{in}},0) \tag{4}$$

根据链式法则和反向传播原理,以损失函数的方式对输入、输出和准确值进行对比评估,从而对权重、偏差等参数进行调整。本文采用先经标定好的数据集进行训练,卷积神经网络模型的权重和偏置参数是离线预训练得到的。因此前向传播过程是本文设计实现目的。

1.2 嵌入式 SoC 的设计方法

基于 FPGA 的传统 RTL 设计方法,有难度大、周期长等缺点。藉此提出的高层次综合技术是以 C、C ++ 等语言编写高级接口程序,并将其转换成 HDL 代码进行 FPGA 部署的一种设计方法。在此技术上,为了提高效率和简化难度,利用嵌入式系统的优势引入具有高度可读性的 Python 解释器语言,并且利用 Web 服务器基于浏览器的工具包进行访问,其中就包含了结合开源框架 Jupyter-notebook 在 ARM 处理器上运行交互式 Python 内核实现 SoC 编程操作的设计方法。

本文根据卷积神经网络层与层之间的数据依赖关系,对前向传播的卷积神经网络以同步数据流方式进行设计;对 32 位浮点数进行量化压缩,在精确率可接受的情况下实现 16 位定点数转换;最后利用 Overlay 的设计方法部署 FPGA 部分,协调使用嵌入式 SoC 的逻辑资源。

2 模型设计

2.1 系统设计

本文设计的整体架构为 ARM 处理器部分和 FPGA 逻辑资源部分。带有 Python 解释器以及 Jupyter-notebook 工具包的 Linux 系统,可以调用高级接口来控制特征图加载到 DDR 内存上,经过卷积神经网络模型计算后的数据交换由 AXI_DMA 模块负责,并将输出结果显示在电脑上。

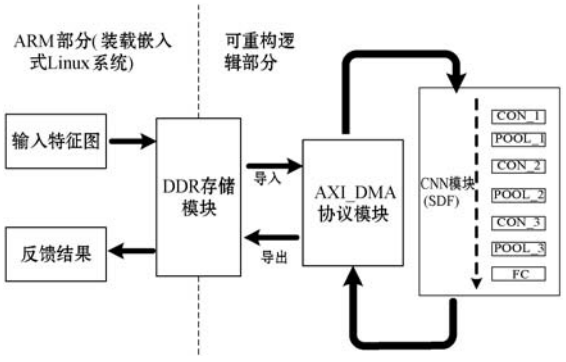

图 2 系统整体结构框图

2.2　定点数量化

Han 等[5]提出的深度压缩即通过裁剪训练量化和霍夫曼编码实现方法,实现难度大,不适合小规模计算样本。蔡瑞初等[6]设计了动态定点量化小数位数和精确率之间的平衡的方法,即反向传播时对网络进行微调,计算量有一定减少,但对于识别网络的计算模型研究缺乏。以上量化方法对于嵌入式 SoC 下的卷积神经网络的计算没有较好的解决方法。本文分析了文献[7]在嵌入式 SoC 平台下实现一个卷积层时消耗资源情况,如表 1 所示,可以看出这样的设计在目前的硬件平台上不可能实现整个卷积神经网络。

表 1　浮点数卷积层消耗资源展示

模块	使用	资源	百分比/%
BRAM	54	140	38.6
DSP	76	220	34.5
LUT	1 803	53 200	3.39

本文继续研究发现,通常用位宽、步长和动态范围 3 个参数来表示浮点转定点的依赖关系:

$$Range \approx Stepsize \cdot 2^{Bitwidth} \tag{5}$$

于是提出当给出定点的位宽后,在大的动态范围下和小的分辨率之间权衡量化误差关系是解决问题的关键的想法。文献[8-9]给出了各种输入分布的最优对称均匀量化器的步长表,分析表中内容可知随着输入分布的峰值增加量化效率降低,和高斯分布具有相似性。所以在对预训练的浮点数据分析后,确定每层的权重、偏置和激活函数,本文提出一种浮点数量化为定点数的方法:

$$s = \xi \cdot Stepsize(\beta) \tag{6}$$

$$n = -\lfloor \log_2 s \rfloor \tag{7}$$

式中:$Stepsize(\beta)$ 对应量化位宽的最佳步长,ξ 是量化有效标准差(因为在理想零均值高斯分布下量化值的标准偏差为 σ,并且 $\xi \geqslant \sigma$,本文采用 $\xi = 3\sigma$ 的准确率数值),s 是计算所得的步长,n 表示小数位数。式(7)以 2 为底计算对数,应用舍入函数取舍,计算小数位数,对最终结果进行移位操作将 32 位转换成 16 位。

在实验中得出与 32 位浮点算法相比,FPGA 上的定点算法需要较少的 DSP 和 LUT,并且简单的定点算法操作可以在一个时钟周期内执行完成。

2.3　卷积层设计和数据流优化

将复杂的卷积运算转换成乘加法运算,可以在 FPGA 中有效执行。为了减少缓冲区的使用和提高计算的吞吐量,本文提出一种交错计算的方法,如图 3 所示。

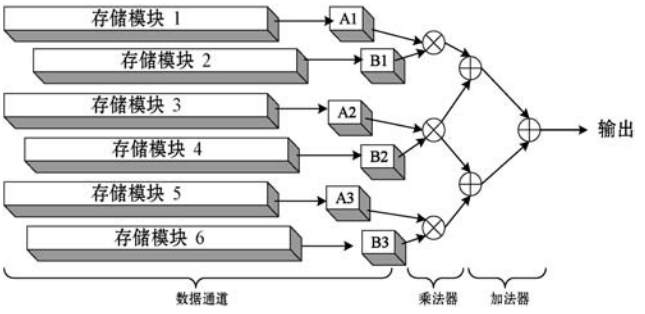

图 3　FPGA 实现矩阵乘法原理图

在 3×3 的划窗操作后,本文设计将特征图提取在存储单元奇数模块,权重值提取到存储单元偶数模块。在每个时钟周期的激励下,将运算数串行输出到寄存器 Ai 和 Bi 中。普通单个矩阵乘法器负责 9 个卷积数和 9 个权重数的乘加树计算,因此计算会消耗较多的 DSP 资源。本文提出不改变滑窗计算过程并且减少 DSP 使用个数的乘加树计算方法。图 3 可理解为单个计算单元设计了 6 个 DSP 组成的 3 层乘加树。第 1 层是 3 个乘法器将权值和卷积数进行乘法计算。第 2 层则是级联第一层的计算结果,并且共享中间的计算值进行累加计算。第 3 层单一的加法器负责将上一层输出的两个值求和。最终结果由 ReLU 函数激活送至特征图缓冲区。如此在有限的类似设计单元中完成卷积计算。而且在 FPGA 中每个阶段都是由寄存器组成,不需要等待当前计算完成后将内存的数据提取到通道 A 或 B,所以在每个时钟周期同时进行输入和输出操作大大加速了运算过程。

考虑到把所有数值都提取到 BRAM 上进行计算是不实际的,但将数据从外部存储提取再计算会影响计算速度,所以本文提出在设计当中将输入的特征图值存储在基于 LUT 的存储模块中,权重值存储在 BRAM 的存储单元里,如此可以加速计算和节省资源。伪代码如下:

```
static ap_int < wordwidth > A[A_COL_MAX][A_ROW_MAX] B[B_COL_MAX][B_ROW_MAX];
#pragma HLS RESOURCE variable = A
core = RAM_S2P_LUTRAM
#pragma HLS RESOURCE variable = B core = RAM_S2P_BRAM
for (int ia = 0;ia < A_COL_MAX; ++ia)
    for (int ib = 0;ib < B_COL; ++ib)
        for (int ic = 0;ic < B_ROW_MAX/FACTOR; ++ic)
            for (int id = 0;id < FACTOR; ++id){
                sum +=
                    A[ia][id * B_ROW_MAX/FACTOR + ic] *
                    B[ib][id * B_ROW_MAX/FACTOR + ic].
```

通常卷积神经网络表示为一系列的层,这些层构

成一个有向无环图模型,这些层的连接是通过高级描述方案完成。所以本文研究了将有向无环图模型映射到硬件的模型即同步数据流模型。类似于搭积木将每个层例化成模块后相互连接。实验表明计算过程实现了不间断流水操作,并且输入即输出的节点运算方法一定程度上提高了系统的鲁棒性,因为每个层的设计都是独立地驱动数据流,从而形成异构流模式,这种输入输出模式,输出数据立即流出,而不是缓冲在片上存储器上,节省整个网络的内存占用。

3 实验与结果分析

3.1 实验环境

实验采用 Xilinx PYNQ-Z2 嵌入式 SoC 实验平台,该平台搭载的是 xc7z020clg400SoC FPGA 芯片,片上有双核 Cortex-A9 处理器。FPGA 部分主要有 BRAM、DSP 以及 LUT 可编程阵列[10]。实验中的硬件编程环境为 Vivado 2017.4,高层次综合系统 Vivado 高层次综合技术 2017.4;卷积神经网络的训练和执行采用 Theano 框架。实验的测试数据集是 CIFAR-10 中的 500 幅图片,每幅图片大小为 32 × 32 × 3。对比实验中,采用 Intel i5-4200U 低功耗的 CPU。在 Windows 10 操作系统下,以 Google Chrome 浏览器执行与嵌入式系统的可视化交互编程。

3.2 实验结果

在主频为 100 MHz 的工作频率下,该方法优化了对片上资源的使用,设计实现了 16 位定点量化,虽然有一些分类精度的损失,但是比 Zynqnet 的设计节约了 88% BRAM、75% 的 DSP 和 73% LUT。本文设计对比 ZynqNet 中对资源使用如表 2 所示。

表 2 资源消耗以及对比分析

模块	资源使用		本文使用比/%	节约比/%
	本文设计	ZynqNet[3]		
BRAM	108	996	77	89
DSP	192	771	87	75
LUT	43 068	155 196	81	72

在 3 个计算平台下,对 500 幅 32 × 32 × 3 大小的图片进行识别,计算速度如表 3 所示。可以看出,相同工作频率下 ARM 处理器的识别速度提高了 42 倍,本文方法的准确率可达 73.7%,对比使用 CPU 计算浮点数的准确率 75.2%,这是可以接受的。

表 3 不同平台的实验数据对比

平台	图象数/幅	识别时间/s	准确率/%
Cortex-A9[7]	500	82.4	73.7
xc7z020	500	3.84	73.7
i5-4200	500	41	75.2

4 结 语

本文提出一种改进的基于嵌入式 SoC 卷积神经网络 CIFAR-10 的识别模型。该模型使用的资源少,功耗低。为了提高 FPGA 加速方案的准确率和性能,未来将优化 FPGA 的部署和训练更精确的网络模型,以及研究卷积神经网络针对层的优化设计。

参 考 文 献

[1] Venieris S I, Bouganis C S. fpgaConvNet: A framework for mapping convolutional neural networks on FPGAs[C]//2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, 2016: 40 - 47.

[2] 卢冶,陈瑶,李涛,等. 面向边缘计算的嵌入式 FPGA 卷积神经网络构建方法[J]. 计算机研究与发展,2018,55(3): 551 - 562.

[3] 仇越,马文涛,柴志雷. 一种基于 FPGA 的卷积神经网络加速器设计与实现[J]. 微电子学与计算机,2018,35(8): 68 - 72,77.

[4] Stornaiuolo L, Santambrogio M, Sciuto D. On how to efficiently implement Deep Learning algorithms on PYNQ platform[C]//2018 IEEE Computer Society Annual Symposium on VLSI(ISVLSI). IEEE Press, 2018: 587 - 590.

[5] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[EB]. arXiv preprint arXiv:1510.00149,2015.

[6] 蔡瑞初,钟椿荣,余洋,等. 面向"边缘"应用的卷积神经网络量化与压缩方法[J]. 计算机应用,2018,38(9):2449 - 2454.

[7] Wang E, Davis J J, Cheung P Y K. A PYNQ-based framework for rapid CNN prototyping[C]//2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines(FCCM). IEEE Press,2018:223.

[8] Lin D, Talathi S, Annapureddy S. Fixed point quantization of deep convolutional networks[C]//Proceedings of the International Conference on Machine Learning. 2016: 2849 - 2858.

[9] Dettmers T. 8-bit approximations for parallelism in deep learning[EB]. arXiv preprint arXiv:1511.04561, 2015.

[10] Hutchings B, Wirthlin M. Rapid implementation of a partially reconfigurable video system with PYNQ[C]//Proceedings of the 27th International Conference on Field Programmable Logic and Applications(FPL). IEEE Press, 2017: 1 - 8.