doi:10.3772/j.issn.1002-0470.2020.03.004

基于改进动态配置的 FPGA 卷积神经网络加速器的优化方法①

陈 朋②* 陈庆清** 王海霞* 张怡龙** 刘义鹏③* 梁荣华*

(*浙江工业大学计算机科学与技术学院 杭州 310000)

(**浙江工业大学信息工程学院 杭州 310000)

摘 要 卷积神经网络(CNN)已广泛应用于各种计算机视觉任务,基于 GPU 的卷积神经网络加速器往往存在功耗较高、体积较大和成本较高的问题。针对上述问题,文中提出一种基于改进动态配置的现场可编程门阵列(FPGA)卷积神经网络加速器的优化方法。使用高层次综合工具,在引入分割参数的基础上,通过在资源约束情况下基于流水线结构的层间模块复用,采用 8-16 位动态定点设计方案,以有限的硬件资源实现性能优化的卷积神经网络硬件结构,提升计算效率的同时缩短了开发周期。利用该方法在 ZCU102 平台上构建实现了 AlexNet 网络和 VGG 网络。在最大精度损失 0.63% 的条件下,将加速器性能分别从 46.3 fps 和 37.2 fps 提高到 290.7 fps 和 54.4 fps,计算能效分别达到了 TITAN-X 的 1.78 倍和 3.89 倍。实验数据充分说明,采用改进动态配置的优化方法,利用高层次综合工具进行开发的 FPGA 卷积加速器,既满足了计算实时性的要求,同时也解决了功耗和体积问题,验证了本方法的有效性。

关键词 卷积神经网络(CNN);现场可编程门阵列(FPGA);模块复用;流水线;动态定点

0 引 言

近年来,由于卷积神经网络(convolutional neural network,CNN)的结构层次越来越深,在图像分类、目标检测、目标跟踪[1]等计算机视觉领域都得到了广泛的应用,但高计算复杂度和大内存占用是硬件加速的瓶颈,需要开发更高效的硬件加速解决方案来驱动实时应用程序[2]。Li 等人[3]提出了基于 GPU 的大规模递归神经网络的高效实现。Li 等人[4]分析了 GPU 加速器潜在的性能瓶颈。随着 CNN 模型越来越大、越来越深,CNN 加速器需要更多的计算操作和数据访问,虽然 GPU 具有强大的计算能力,但也存在着功耗较高、体积较大和成本较高等问题。

现场可编程门阵列(field programmable gate arrang, FPGA)具有大量阵列形式逻辑、运算单元,在尺寸、功耗和并行运算方面都比 GPU 有优势[5],具有高性能、低功耗、使用灵活方便等优点[6]。传统构造 FPGA 的卷积神经网络的方式是基于寄存器传输级(register transfer level,RTL)描述语言设计的。郭晓丹等人[7]在 FPGA 上实现了单比特 BP 人工神经网络,采用了低环路延迟加法器、混合信号乘法器来减少硬件消耗,提高运算精度。Han 等人[8]在 FPGA 平台上设计的网络加速器,其能源效率是 CPU 的 40 倍,GPU 的 11.5 倍。林军等人[9]提出了基于数据流水线的资源管理,冯煜晶等人[10]针对流水线停顿提出了动态指令调度机制。Li 等人[11]采

① 国家自然科学基金(U1909203,61527808),浙江省属高校基本科研业务费专项资金(RF-C2019001)和浙江省重点研发计划(2019C01007)资助项目。

② 男,1981 年生,博士,副教授;研究方向:模式识别,嵌入式系统设计;E-mail:chenpeng@zjut.edu.cn

③ 通信作者,E-mail:liuyipeng@zjut.edu.cn

(收稿日期:2019-04-10)

用流水线结构达到 565.94 GOP/s 和 391 fps 的峰值性能。但传统的基于 RTL 设计的神经网络仍然具有流程复杂、周期较长和优化空间较小等问题[12]。Ma 等人[13,14]基于 RTL 设计了可扩展的自动编译器,但仍然无法避免流程复杂、周期较长等问题。

在其他基于 FPGA 的 CNN 加速器的设计方法中,使用基于高层次综合(high-level synthesis,HLS)工具可以将高级编程语言直接转化为硬件描述语言(hardware description language,HDL),并且对所生成的硬件结构可以通过插入优化指令进行优化,其中包括映射硬件寄存器、循环、接口等操作[15]。Zhang 等人[16]通过 HLS 工具,使得基于 FPGA 的加速器的速度是 NVIDIA K80 的 4.75 倍。卢治等人[17]使用 HLS 工具证明了 FPGA 平台在不同的网络模型下效能远高于 GPU 平台。由此可见,HLS 工具设计的 FPGA 神经网络加速器同样具有良好的性能,而且在设计周期上远远短于传统方法,具有良好的可扩展性。

另一方面,在数据规模较大的分类场景下,并不需要网络的高精度,需要的是更高实时性。在 FPGA 上,实现定点运算的效率比浮点运算高得多。Gysel 等人[18]和 Qiu 等人[19]分别在 GPU 平台和基于 RTL 设计的 FPGA 平台提出动态定点数据量化方法,并且 Qiu 等人[19]只引入了 0.4% 的精度损失,实现了 4.45 fps 的帧率。在可接受的精度损失的条件下,对 FPGA 卷积加速器采用定点设计,可以有效减少硬件开销,从而提升加速器的性能。

综上所述,本文提出改进动态配置的 FPGA 卷积神经网络加速器的优化方法。该方法采用 C ++ 作为编程语言,将加速器任务划分到处理器系统与可编程逻辑上。处理器系统上使用基于模块复用的流水线方式,多个模块之间通过片内总线实现高速互连。在可编程逻辑上将引入分割参数,使用 HLS 工具设计具有可扩展性卷积加速器并实现其功能。同时采用 8-16 位动态定点方案,将浮点数据改为动态定点数据,引入量化参数,以适配不同网络层量化配置。本文将 ImageNet 数据库作为数据集,在 FPGA 上搭建了 CNN 网络加速器,验证了方法的有效性。本文的主要工作如下。

(1) 引入分割参数后,以流水线的方式将网络中相同层进行模块复用来实现并行工作,并建立冲突解决机制来解决输入数据的竞争问题。

(2) 根据 FPGA 的计算特性,采用动态定点数据量化方法,将浮点数据改为动态定点数据。并在 HLS 工具设计过程中引入量化参数,以适配不同网络层量化配置。

1　FPGA 卷积神经网络加速器的优化方法

1.1　系统任务的软硬件划分

本文对优化对象进行划分,将不同的优化对象分别划分到处理器系统(processing system,PS)与可编程逻辑(programmable logic,PL)上,优化示意图如图 1 所示。在 PS 部分上实现基于流水线结构的层间模块复用,并建立对应的冲突处理机制来解决复用造成的信号冲突;在 PL 部分上利用 HLS 工具设计优化卷积加速器,通过仿真报告来验证资源占用比。完成设计后,对整个系统进行联合仿真,验证结果的准确性,通过实验来验证设计方法的有效性。

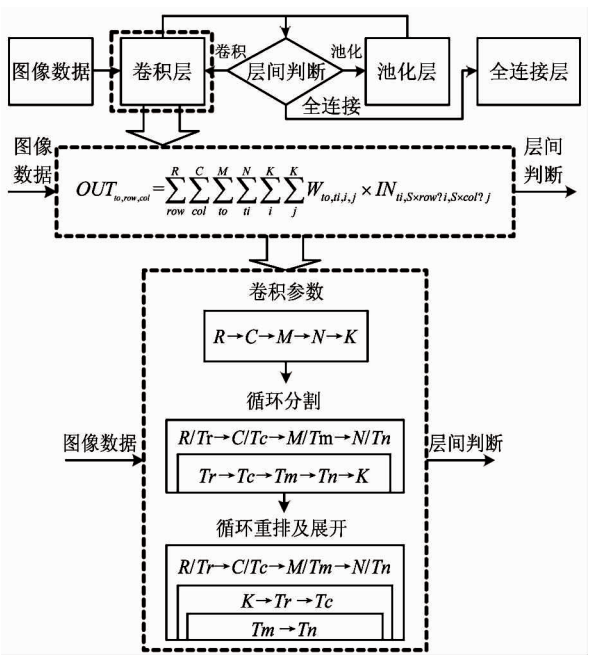

图 1　优化示意图

1.2　具有可扩展性的卷积加速器设计

卷积神经网络的特征提取阶段往往由多个计算

层组成，例如 AlexNet 网络，它的特征提取阶段可以由 5 个卷积层和 3 个池化层组成。卷积运算如图 2 所示。卷积层接收特征图作为输入，每个输入特征图通过卷积核映射生成输出特性，输出特征图将形成下一个卷积层的输入特征图集。N 表示特征图的个数，W 和 H 分别表示输入特征图的宽与高，K × K 表示卷积核的窗口大小，S（通常小于 K）表示窗口的平移步长，M 表示输出特征图的个数和卷积核的个数，C 和 R 分别表示输出特征图的宽与高。

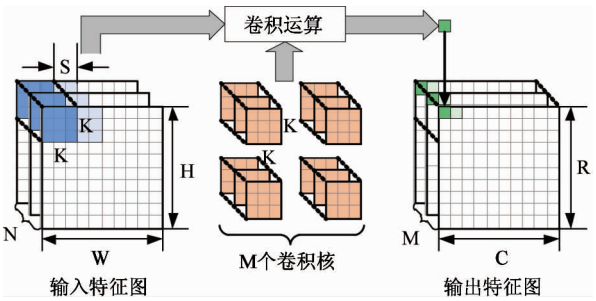

图 2　卷积运算示意图

池化层的目标是减少前一个卷积层产生的数据的大小，同时保持最相关的特性。实际上，池化层通常插入到 2 个卷积层之间。因为卷积层和池化层拥有相似的内存访问模式，因此它们在本文中使用相同的内存和优化设计方案，可以用式（1）表达。其中 *OUT* 表示输出特征图，*W* 表示权重系数，*IN* 表示输入特征图。

$$OUT_{to,row,col} = \sum_{row}^{R}\sum_{col}^{C}\sum_{to}^{M}\sum_{ti}^{N}\sum_{i}^{K}\sum_{j}^{K} W_{to,ti,i,j} \times IN_{ti,S\times row+i,S\times col+j} \tag{1}$$

在一般情况下，卷积神经网络所处理的输入集输出集均为 3 维数据，需要和各个层的卷积核大小和步长一起作为函数变量进行传递。但如果循环内含有变量，就不能利用 HLS 工具的循环展开、循环流水线等优化方式进行优化。

为了提高数据处理的并行度，引入 < *Tm*, *Tn*, *Tr*, *Tc* > 分割参数，对输出特征图深度、输入特征图深度、输出特征图宽和长进行分割。在 Zhang 等人[20]的工作中，不同循环迭代之间的数据共享关系可以分为 3 类：无关、独立、依赖。循环参数 *Tr*、*Tc* 和输入集 *IN* 是依赖关系，和输出集 *OUT* 是独立关系；*Tm* 和权重集 *W*、输出集 *OUT* 均是独立关系；*Tn* 和输入集 *IN*、权重集 *W* 均是独立关系。

在卷积层加速器进行高层次综合之前，需要确定分割参数 < *Tm*, *Tn*, *Tr*, *Tc* > 的数值，即将其作为固定的加速器参数。根据数据共享关系，对循环进行分割及重排后的卷积结构如式（2）所示。其中 $F(x)$ 表示循环展开，$L(x)$ 表示循环流水线。在经过优化后，卷积层中的卷积运算根据分割参数的设定，展开为多条处理通道进行并行计算。

$$P_{Tr,Tc,Tm,Tn} = \sum_{trr=row}^{Tr}\sum_{tcc=col}^{Tc} L\Big(\sum_{i=0,j=0}^{K} F\Big(\sum_{too=to}^{Tm} F\Big(\sum_{tii=ti}^{Tn} W_{too,tii,i,j} \times IN_{tii,S\times trr+i,S\times tcc+j}\Big)\Big)\Big)$$

$$OUT = \sum_{row=0}^{R/Tr}\sum_{col=0}^{C/Tc}\sum_{to=0}^{M/Tm}\sum_{ti=0}^{N/Tn} P_{Tr,Te,Tm,Tn} \tag{2}$$

1.3　资源约束下基于流水线结构的层间模块复用

对于大规模的应用场景，图像数据往往是具有连续性的。根据 CNN 数据处理特性，可知 CNN 内部的数据处理过程是依照不同处理层顺序进行的，即网络中前后相邻的层之间，数据具有依赖关系，无法进行并行处理。对于 GPU 来说，卷积神经网络的每一层输入都是上一层的输出，层间的数据关系紧密相关，是一种串行结构，无法做到层间并行运算。相比于 GPU，FPGA 的硬件结构可以采用流水线结构，用以减少运算的时间开销，提高资源利用率。

本文中的流水线由多个卷积神经网络的层级结构组成。由于流水线中的不同层级模块具有不同的计算时间，计算周期最长的层级模块是流水线中瓶颈段，实际加速比也由瓶颈段所决定的。实际加速比 *S* 如式（3）所示：

$$S = \frac{m\sum_{i=0}^{n} t[i]}{\sum_{i=0}^{n} t[i] + (m-1)\Delta T} \tag{3}$$

其中 ΔT 为瓶颈段的计算周期，$\Delta T = \max_{i=0}^{n}(t[i])$，$t[i]$ 表示各个模块的计算时间，n 表示模块的数量，m 表示流水线任务的个数。

在资源约束情况下，想要保持卷积神经网络良好的计算性能，需要充分考虑其并行运算的情况。由于相邻层之间具有数据依赖关系，其数据处理无法并行进行，为达到对资源的高效利用，本文采用各个不同网络层间模块复用的方案，如图 3 所示，以实

现对硬件资源的节约并提高单个加速器的计算效能。即其中所有卷积层采用同一个卷积加速器进行计算,所有池化层采用同一个池化层加速器进行处理,所有全连接层采用同一个全连接层加速器进行处理。

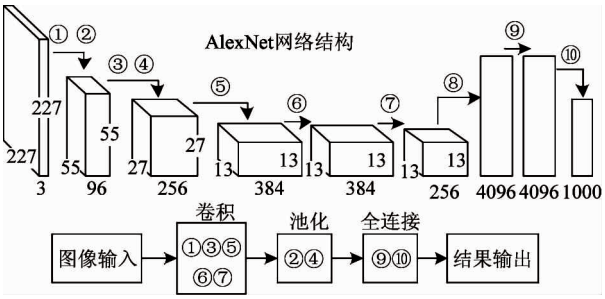

图 3　网络层间模块复用示意图

模块复用的关键在于解决输入数据的竞争问题。模块重用之前,各个模块的输入和输出是相互独立的。模块重用之后,多个输入需要整合成一个输入,重用模块的输出是根据输入选择的。t 时刻多个输入数据的冲突信号 C 是由输入数据的有效信号 V 组成的,如式(4)所示,其中 v 表示输入数据个数。

$$C[t] = \sum_{i=0,\ j=0,\ i\neq j}^{v} V[i] \wedge V[j] \tag{4}$$

如果重用模块在 t 时刻检测到冲突($C \geqslant 1$),就需要建立冲突解决机制来避免冲突。我们通过增加流水线 cache 的方式,避免数据同时到达重用模块的输入端口。具体操作是在后一个模块数据输入前增加 FIFO 来缓存输入数据,直到冲突解决。

对于流水线中的第 k 个模块,它在 t 时刻输入信号 V_α 如式(5)所示。当 $V_\alpha = 0$ 时,说明输入无效。当 $V_\alpha > 0$ 时,则说明输入有效。

$$V_\alpha[k] = (t - \sum_{j=0}^{k-1} t[j]) \% \Delta T \tag{5}$$

在增加了 x 级的 cache 后,模块的流水线在 t 时刻输入信号 V_β 如式(6)所示,此时冲突信号 C 如式(7)所示。其中 V_x 表示在增加了 x 级的 cache 后,重用模块的输入数据的有效信号。

$$V_\beta[k] = (t - x - \sum_{j=0}^{k-1} t[j]) \% \Delta T \tag{6}$$

$$V_x = \begin{cases} 0 & V_\beta = 0 \\ 1 & V_\beta > 0 \end{cases}$$

$$C[t] = \sum_{i=0,\ j=0,\ i\neq j}^{v} V_x[i] \wedge V_x[j] \tag{7}$$

1.4　动态定点数据量化

卷积神经网络的复杂性集中在两部分,卷积层中存在着大量的运算,而网络权重通常集中于全连接层。例如 AlexNet,超过 90% 的运算集中在卷积层,超过 90% 的网络权值集中在全连接层中,如图 4 所示。

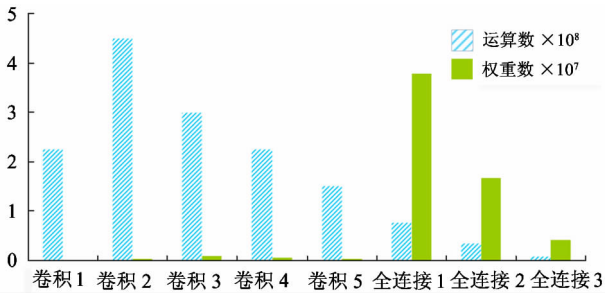

图 4　AlexNet 中不同层的运算次数和权重数

传统的卷积神经网络模型在 CPU 和 GPU 的训练过程中得到的都是 32 位的浮点型数据,但是在测试或者分类阶段,并不需要这样的高精度。FPGA 卷积神经网络加速器的设计目标是以最高性能实现最佳精度。在 Zhang 等人[20]的工作中,实现模型采用 32 位的浮点型数据进行计算。然而在 FPGA 上,实现定点运算的效率比浮点运算高得多[21]。在尽可能不影响精度的前提下,如果将网络进行量化,使用定点运算来代替浮点运算,直接降低存储需求以及存储传输所消耗的能量,从而提升加速器的性能。网络的准确性往往取决于用于表示特征图和训练参数的精度参数,需要合理的量化参数来保证模型量化后的网络准确性。

对传统的静态定点方案而言,只能满足大部分的权重取值范围,而动态定点方案可以通过调整数据格式来满足不同网络层的精度需求。在动态定点方案中,每个数字都可以用式(8)表示:

$$(-1)^S \cdot 2^{-F} \sum_{i=0}^{B-2} 2^i \cdot x_i \tag{8}$$

其中 B 表示位宽,S 表示符号位,F 表示小数长度,x 表示当前位的值。动态精度量化与静态精度量化不同之处在于 FL 对于不同网络层的特征图集是动态的,而在一个图层中是静态的,这样做是为了将每个

网络层的误差最小化。

由于动态定点中每个网络层的量化参数各有差异，特别是针对层间模块复用的情况，因此在使用 HLS 工具设计网络层加速器时，需引入量化参数 $<B_{in}, B_{out}, B_w, F_{in}, F_{out}, F_w>$ 来适配各种量化配置的情况。$<B_{in}, B_{out}, B_w>$ 表示输入集、输出集、权重集所需的位宽，$<F_{in}, F_{out}, F_w>$ 表示输入集、输出集、权重集所需的小数位。

在完成训练之后需要对网络进行量化来获取量化参数，并根据网络精度变化来选取最优的量化参数。在选取量化参数时应满足式(9)的条件：

$$F = arg\ min \sum | R - R_{B,F} | \tag{9}$$

其中，R 为训练时得到的定点数据，$R_{B,F}$ 为给定条件下的定点格式。

2　实验结果和分析

本文实验采用了 Xilinx 公司的 Xilinx SDSoC 2017.4 软件环境进行硬件开发，使用的硬件平台为该公司的 ZCU102 开发板，其芯片型号为 ZU9EG，使用的工作频率为 200 MHz，采用了 8-16 位动态定点方案。实验采用的数据集为 ImageNet 数据集，对比的 GPU 平台为 NVIDIA 公司的 TITAN-X，本实验测试的网络结构为 AlexNet 网络和 VGG 网络。

卷积层加速器内部的乘加运算需调用数字信号处理器(digital signal processor, DSP)来进行，在本文中，对卷积操作的循环进行了展开，卷积层加速器所消耗的 DSP 数量与循环分割参数 Tm 和 Tn 的关系如图 5 所示，可见卷积层加速器的 DSP 的消耗量与分割参数 $Tm \times Tn$ 呈线性关系，循环展开越大，DSP 消耗量也越大。

卷积层加速器内部的片上缓存主要是调用双极随机存取存储器(bipolar random access memory, BRAM)来进行存储的，在本文中卷积层加速器所消耗的 BRAM 与循环分割参数 Tr 和 Tc 的关系如图 6 所示，可见 BRAM 的消耗量与分割参数 $Tr \times Tc$ 基本呈线性关系，便于之后的拓展设计。

AlexNet 网络和 VGG 网络在动态定点量化前后的 Top-1 和 Top-5 的准确率如表 1 所示。AlexNet 网络量化前后 Top-1 和 Top-5 的精度损失为 0.4% 和 0.14%，VGG 网络量化前后 Top-1 和 Top-5 的精度损失为 0.63% 和 0.53%，与其他加速算法的精度损失对比如表 2 所示。Naveen 等人[22]和 Wei 等人[23]采用 8-16 位定点，Zhang 等人[24]采用 16 位定点，其

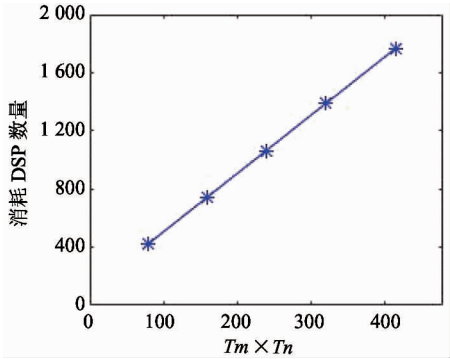

图 5　不同参数配置下的 DSP 消耗量

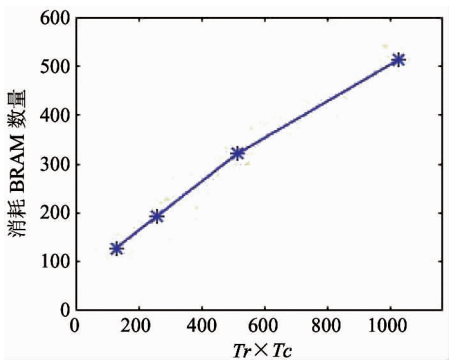

图 6　不同参数配置下的 BRAM 消耗量

表 1　量化前后网络模型精度比较

准确率	32 位浮点		8-16 位动态定点	
	Top-1	Top-5	Top-1	Top-5
AlexNet	63.87%	85.39%	63.47%	85.25%
VGG	71.59%	90.07%	70.96%	89.54%

表 2　不同加速器的精度损失比较

精度损失	AlexNet		VGG	
	Top-1	Top-5	Top-1	Top-5
本文	0.40%	0.14%	0.63%	0.53%
文献[22]	1.41%	0.97%	1.77%	0.96%
文献[23]	<2%	<2%	<2%	<2%
文献[24]	1.9%	0.87%	—	—

算法精度损失均在 1% ~2% 左右。而本文采用的 8-16 位动态定点设计方案在此基础上将精度损失减小至 0.63%。

利用循环展开、流水线等 HLS 工具的优化技术进行网络加速,表 3 显示了优化前后的单张图片运行时间和资源占用比。DSP 的资源比优化前增加 7.2 倍,其余资源也有增加,但都未超过系统所提供的硬件资源上限,同时运算速度提升了 295.7 倍。在应用层间模块复用后,硬件资源基本没有变动,但平均计算时间仅是原来的 20%。

表 3　优化前后运算时间、资源占用对比

	优化前	优化后	层间复用
运算时间	5 367.5 ms	18.15 ms	3.44 ms
DSP	10.67%	76.94%	76.94%
BRAM	85.47%	90.84%	90.84%
LUT	30.78%	49.27%	49.89%
FF	23.10%	32.13%	32.07%

针对 AlexNet 网络结构,应用本文的优化配置后,对其性能及功耗进行测量,并将结果与其他优化方法比较,其对比结果如表 4 所示。在 TIAN-X 上每幅图的计算时间为1.93 ms,本文每幅图的计算时间为3.44 ms,是 TIAN-X 的 1.78 倍,但在 TIAN-X 上每幅图所需功耗为 0.511 J,本文每幅图所需功耗仅需 0.078 J,其计算能效达到了 GPU 的 7.2 倍,并且 FPGA 的便携性远远高于 GPU。Zhang 等人[20]使用 HLS 工具,提出了基于循环迭代之间的数据共享关系的循环分割与重排。本文在此基础上进行 8-16 位的动态定点量化,并引入了基于流水线层间模块复用,将计算速度提高了 6.2 倍,同时也优于文献[22]基于 OpenCL 的开发方式。文献[9]基于 RTL 设计,用流水线结构来增加吞吐量,达到了 391 fps的峰值性能。虽然 Li 等人[9]设计的加速器速度更快,但由于功耗更大,其计算能效和本文的方法几乎相同,且开发周期较长,应对不同网络结构的可扩展性不强。

针对 VGG 网络结构的性能比较如表 5 所示,在 TIAN-X 上每幅图的计算时间为 6.60 ms,本文每幅图的计算时间为 18.37 ms,计算能效达到了 GPU 的 3.89 倍。文献[23]采用的是脉动阵列的高通量 CNN 设计,每幅图的计算时间为 26.85 ms。文献[25]针对卷积循环的内存访问和数据移动进行了定量分析和优化,每幅图的计算时间分别为 47.97 ms。文献[24]使用 RTL 提出了一个深度流水线的多 FPGA 架构。而本文结合了循环迭代优化和流水线优化,提高了计算速度的优化上限,把计算速度提高了至少 2 倍。

表 4　AlexNet 网络加速器的性能比较

	计算平台	设计方式	精度	每幅图的运行时间 (ms)	功耗(W)	每幅图的功耗(J)
Caffe GPU	TITAN-X	C + +	32 位浮点	1.93	165	0.511
文献[20]	Virtex-7 VX485T	HLS	32 位浮点	21.61 (5 CONV layers)	18.61	>0.402
文献[22]	Stratix-V GSD8	OpenCL	8-16 位定点	20.1	>19.1	>0.384
文献[11]	Virtex-7 VC709	RTL	16 位定点	2.56	30.2	0.077
本文	ZCU102 ZU9EG	HLS	8-16 位定点	3.44	22.8	0.078

表 5　VGG 网络加速器的性能比较

	计算平台	设计方式	精度	每幅图的运行时间(ms)	功耗(W)	每幅图的功耗(J)
Caffe GPU	TITAN-X	C + +	32 位浮点	6.60	247	1.630
文献[23]	Arria 10 GT 1150	OpenCL	8-16 位定点	26.85	—	—
文献[25]	Arria-10 GX 1150	RTL	8-16 位定点	47.97	—	—
文献[24]	Virtex-7 VX690T	RTL	16 位定点	151.80	—	—
本文	ZCU102 ZU9EG	HLS	8-16 位定点	18.37	22.8	0.419

3 结 论

本文利用 HLS 工具的可扩展性,提出了一种改进动态配置的 FPGA 卷积神经网络加速器的优化方法。引入循环分割参数,设计了具有可扩展性的卷积加速器。将网络层通过复用模块,组合成流水线的工作方式,采用 8-16 位动态定点方案。在 Xilinx 的 ZCU102 开发板上实现了 AlexNet 网络和 VGG 网络。在最大精度损失 0.63% 的条件下,将加速器性能分别从 46.3 fps 和 37.2 fps 提高到了290.7 fps和 54.4 fps,计算能效分别达到了 TITAN-X 的 1.78 倍和 3.89 倍。实验结果表明,本文所提出的改进动态配置的 FPGA 卷积神经网络加速器的优化方法,在计算能效上优于 GPU Titan-X,满足了计算实时性的要求,同时也解决了功耗和体积问题。

参考文献

[1] 胡硕,赵银妹,孙翔. 基于卷积神经网络的目标跟踪算法综述[J]. 高技术通讯, 2018, 28(3):207-213

[2] Li Y, Liu Z, Xu K, et al. A GPU-outperforming FPGA accelerator architecture for binary convolutional neural networks[J]. *ACM Journal on Emerging Technologies in Computing Systems*, 2018, 14(2):354-369

[3] Li B, Zhou E, Huang B, et al. Large scale recurrent neural network on GPU[C]//International Joint Conference on Neural Networks, Beijing, China, 2014: 4062-4069

[4] Li X, Zhang G, Huang H, et al. Performance analysis of GPU-based convolutional neural networks[C]//Proceedings of the 45th International Conference on Parallel Processing, Philadephia, USA, 2016:67-76

[5] 吴艳霞,梁楷,刘颖,等. 深度学习 FGPA 加速器的进展与趋势[J]. 计算机学报, 2019, 42(11): 2461-2480

[6] 原魁,路鹏,邹伟. 自主移动机器人视觉信息处理技术研究发展现状[J]. 高技术通讯, 2008,18(1):104-110

[7] 郭晓丹, 孟桥, 梁勇. 基于Σ-△调制的单比特非线性 BP 人工神经网络的硬件实现[J]. 高技术通讯, 2013, 23(12):1316-1322

[8] Han S, Kang J, Mao H, et al. ESE: efficient speech recognition engine with sparse LSTM on FPGA[C]//Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, USA, 2017: 75-84

[9] 林军,倪宏,孙鹏,等. 嵌入式系统流水线资源管理模型[J]. 高技术通讯, 2013,23(9):914-920

[10] 冯煜晶,欧焱,叶笑春,等. 基于网络负载特征感知的数据流指令调度机制研究[J]. 高技术通讯, 2018,28(11-12):885-898

[11] Li H, Fan X, Li Jiao, et al. A high performance FPGA-based accelerator for large-scale convolutional neural networks[C]//Proceedings of the 26th International Conference on Field Programmable Logic and Applications, Lausanne, Switzerland, 2016:1-9

[12] Wim M, Kristof V, Toon G, et al. An overview of todays high-level synthesis tools[J]. *Design Automation for Embedded Systems*, 2012, 16(3):31-51

[13] Ma Y, Naveen S, Cao Y, et al. Scalable and modularized RTL compilation of convolutional neural networks onto FPGA[C]//Proceedings of the 26th International Conference on Field Programmable Logic and Applications, Lausanne, Switzerland, 2016:1-8

[14] Ma Y, Cao Y, Sarma V, et al. An automatic RTL compiler for high-throughput FPGA implementation of diverse deep convolutional neural networks[C]//Proceedings of the 27th International Conference on Field Programmable Logic and Applications, Ghent, Belgium, 2017:1-8

[15] Chen Y, Nguyen T, Chen Y, et al. FCUDA-HB: hierarchicaland scalable bus architecture generation onFPGAs with the FCUDA flow[J]. *IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems*, 2016, 35(12):2032-2045

[16] Zhang X, Liu X, Anand R, et al. High-performance video content recognition with long-term recurrent convolutional network for FPGA[C]//Proceedings of the 27th International Conference on Field Programmable Logic and Applications, Ghent, Belgium, 2017:1-4

[17] 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

[18] Gysel P, Pimentel J, Motamedi M, et al. Ristretto: a framework for empirical study of resource-efficient inference in convolutional neural networks[J]. *IEEE Transactions on Neural Networks and Learning Systems*, 2018, 29(11):5784-5789

[19] Qiu J, Wang J, Yao S, et al. Going deeper with embedded FPGA platform for convolutional neural network[C]//Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, USA, 2016:26-35

[20] Zhang C, Li P, Sun G, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks [C] // Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, USA, 2015:161-170

[21] 刘勤让, 刘崇阳. 利用参数稀疏性的卷积神经网络计算优化及其 FPGA 加速器设计[J]. 电子与信息学报, 2018, 40(6):102-108

[22] Naveen S, Vikas C, Ganesh D, et al. Throughput-optimized openCL-based FPGA accelerator for large-scale convolutional neural networks [C] // Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, USA, 2016:161-180

[23] Wei X, Yu H, Zhang P, et al. Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs[C] // Proceedings of the 54th Annual Design Automation Conference, Austin, USA, 2017:1-6

[24] Zhang C, Wu D, Sun J, et al. Energy-efficient CNN implementation on a deeply pipelined FPGA cluster[C] // Proceedings of the 2016 International Symposium on Low Power Electronics and Design, San Francisco, USA, 2016: 326-331

[25] Ma Y, Cao Y, Sarma V, et al. Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks [C] // Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, USA, 2017:45-54

Optimization method of FPGA convolutional neural network accelerator based on improved dynamic configuration

Chen Peng*, Chen Qingqing**, Wang Haixia*, Zhang Yilong**, Liu Yipeng*, Liang Ronghua*

(*College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310000)

(**College of Information Engineering, Zhejiang University of Technology, Hangzhou 310000)

Abstract

Convolutional neural network (CNN) has been widely employed for various computer vision tasks. GPU-based convolutional neural network accelerators often have problems of high-power consumption, large size and high cost. Aiming at the above problems, this paper proposes an optimization method of field programmable gate array (FPGA) convolutional neural network accelerator based on improved dynamic configuration. High-level synthesis tools are used to achieve performance optimization with limited hardware resources and the 8-16 bit dynamic fixed-point, and utilizes the pipeline structure-based inter-layer module multiplexing under resource constraints, which improves the computational efficiency and shortens the development cycle. This method is used to build and implement the AlexNet network and VGG network on the ZCU102 platform. With 0.63% accuracy loss, the accelerator performance is improved from 46.3 fps and 37.2 fps to 290.7 fps and 54.4 fps respectively, and the calculation energy efficiency reaches 1.78 times and 3.89 times compared to TITAN-X respectively. The experimental data fully demonstrates that the FPGA convolution accelerator developed by the high-level synthesis tool adopts the improved dynamic configuration optimization method, which not only satisfies the requirements of real-time calculation, but also solves the power consumption and volume problem, and verifies the effectiveness of the proposed method.

Key words: convolution neural network (CNN), field programmable gate array (FPGA), module multiplexing, pipeline, dynamic fixed