doi:10.3772/j.issn.1002-0470.2022.05.001

基于 FPGA 的浮点可分离卷积神经网络加速方法①

张志超②* ** *** ****　王　剑* ** ***　章隆兵* ** ***　肖俊华* ** ***

(* 计算机体系结构国家重点实验室(中国科学院计算技术研究所)　北京 100190)

(** 中国科学院计算技术研究所　北京 100190)

(*** 中国科学院大学　北京 100049)

(**** 中国电子科技集团公司第十五研究所　北京 100083)

摘　要　针对可分离卷积神经网络在星载飞机目标型号分类应用中存在的速度瓶颈以及功耗限制等问题,提出了一种基于现场可编程门阵列(FPGA)数据流调度的浮点深度分离卷积神经网络加速方法,对通用 MobileNet 的图像分类模型进行加速。采用基于乘法矩阵与前向加法树的深度分离卷积计算阵列设计,解决了深度分离卷积浮点加速的线速吞吐瓶颈。实验结果表明,基于 FPGA 的目标分类速度为 633 FPS,功耗为 22.226 W,运算性能为 236.04 GFLOPS,计算速度达到了 Titan Xp GPU 的 1.10 ~ 2.61 倍,计算效能是 Titan Xp GPU 的 7.44 ~ 18.66倍。在同类基于 FPGA 的浮点卷积加速方案中,该方法在运算性能及能效比上达到了最优。同时,该方法提供了与原模型一致性的图像分类准确率,解耦合了软硬件协同开发流程,降低了应用开发人员使用 FPGA 加速计算的门槛。

关键词　深度可分离卷积;现场可编程门阵列(FPGA);数据流调度;加速;图像分类

0　引　言

深度学习技术[1]不仅在图像处理应用中取得了较大的发展,在遥感影像目标图像识别中也取得了广泛的应用。目前,基于卷积神经网络的深度学习技术逐渐向轻量化模型[2]发展,使得轻量化模型能够在低处理能力的处理器上使用。在解决了轻量化模型的准确率问题后,还可以进一步采用硬件加速浮点卷积网络的方法进行加速。现有的卷积神经网络的加速研究包括参数量化方法[3-4]、轻量型网络结构设计、硬件卷积加速结构设计等方法。

轻量型网络结构设计方法主要采用分层卷积、分组卷积、点卷积和小卷积核等方式降低卷积的计算量,减少特征图跨通道提取特征的计算量。MobileNet[5]采用点卷积以及深度分离卷积结构降低模型的计算量。ShuffleNet[6]采用通道洗牌策略促进通道间特征融合,从而提升准确率。轻量型网络结合 Yolo[7]检测算法也取得了高准确率与低处理时延[8-10]。轻量型网络结构设计从软件卷积结构层面降低模型复杂度,适合现场可编程门阵列(field programmable gate array, FPGA)定制硬件处理核直接加速浮点卷积模型获取加速能力。

硬件结构设计包括两种方法,一种是基于专用集成电路(application specific integrated circuit,ASIC)设计卷积计算单元和矩阵运算单元等方式获取加速能力[11-13]。另一种是基于 FPGA 的加速浮点卷积计算方法,主要通过高层次综合(high-level synthesis, HLS)[14-15]或者开放运算语言(open computing language, OpenCL)[16-18]加速的方式,采用循环展开优化以及缓存设计方式提升处理能力。基于浮点的加

① 国家自然科学基金(61432016)和国家重点研发计划(2018YFC0832306,2018YFC0831203,2018YFC0831206)资助项目。

② 男,1990年生,博士生;研究方向:计算机系统结构;联系人,E-mail: zhangzhichao@ict.ac.cn。

(收稿日期:2021-04-23)

速方式能够更好地去耦合软硬件协同开发。

本文针对星载飞机目标型号快速准确分类的需求,提出了一种基于 FPGA 的浮点可分离卷积加速方法,达到了图像分类高准确率、高吞吐量、高效能计算的效果。本文的主要贡献如下。

(1)提出了一种基于 FPGA 的浮点可分离卷积加速方法,结合轻量型卷积网络高准确率、低计算复杂度的特点,以及浮点可分离卷积网络加速技术,提供高吞吐量、高准确率的计算模式支撑。

(2)利用数据流调度技术以及浮点可分离卷积乘累加设计,解决了可分离卷积的浮点加速线速吞吐问题,同时也解耦合了软硬件协同开发。

(3)利用多核协同计算的方式,对多层卷积计算协同加速,避免过大的卷积核带来的超宽线宽问题造成的硬件时序收敛难题,进一步提升整个加速器的吞吐量。

1　卷积神经网络模型设计

针对遥感影像飞机目标型号分类任务快速低功耗处理的需求,需要在遥感影像机场区域中快速识别飞机目标的型号,对存在敏感的飞机目标型号的热点区域图像回传给数据中心后,进一步进行高精度目标处理分析。本文算法结合应用需求,一方面采用通用的轻量型网络 MobileNet,从软件层面解决目标型号分类精度问题,同时降低卷积运算的复杂度;另一方面,从 FPGA 加速浮点深度可分离卷积网络层面提升运算吞吐量。二者维持一个浮点卷积模型的编程接口,使得软硬件开发去耦合。

1.1　基于可分离卷积的飞机目标型号分类算法

基于深度可分离卷积(depthwise separable convolution, DSC)的 MobileNet 算法,通过点卷积(pointwise convolution, PC)和深度分离卷积(depthwise convolution, DC)提取特征图,大幅降低了卷积过程的运算量,且在通用数据集图像分类任务上达到了较高的准确率。基于 MobileNet 卷积网络模型在 ImageNet 数据集图像分类任务 TOP-1 准确率达到了70.6%[5],在CPU 平台上的处理时间为113 ms[1],相对于其他模型准确率较高且模型计算复杂度适中,适合将该网络模型通过迁移学习的方式解决机场区域内飞机目标的型号分类。

1.2　可分离卷积网络结构及复杂度分析

(1)基于 MobileNet 的可分离卷积网络结构描述

MobileNet 网络中的可分离卷积与二维卷积的使用方式对比如图 1 所示,图 1 表示一个常规二维卷积运算需要一个深度分离卷积加一个点卷积协同运算。

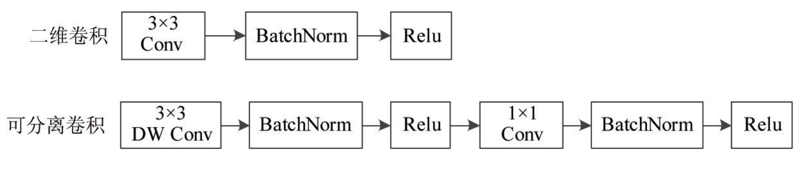

图 1　二维卷积与可分离卷积使用方式对比

MobileNet 飞机目标型号分类卷积网络的输入为 128×128 像素的 3 通道卫星影像,输出为 1000 个目标型号的类别信息,共 28 个卷积层,1 个平均池化层,1 个全连接层,能够最多分类 1000 个目标。具体的 MobileNet 每层的卷积或池化参数配置如表1所示。

(2)模型复杂度分析

通过卷积参数量计算以及卷积计算乘累加数计算,其中池化层没有参数,乘累加数不统计,该模型的权重数量为 4.24×10^6,总大小为 16.96 MB,总乘累加操作数为 0.186 GMAC。由此可见,该分类模型为一个深层轻量化的目标分类网络,适于在 FPGA 上实现全浮点精度的深度可分离卷积网络运算加速。

2　可分离卷积神经网络加速设计

为了降低软硬件协同加速MobileNet的开发耦合度,采用浮点原始模型参数的方式进行加速,保持了与原模型一致性的输出与分类准确率;采用基于

表 1　MobileNet 卷积神经网络结构

名称	卷积类型	卷积核大小	卷积步进	输入通道数	输出通道数	输入维度
cnv0	conv	3	2	3	32	128
cnv1_1	conv dw	3	1	32	32	64
cnv1_2	conv	1	1	32	64	64
cnv2_1	conv dw	3	2	64	64	64
cnv2_2	conv	1	1	64	128	32
cnv3_1	conv dw	3	1	128	128	32
cnv3_2	conv	1	1	128	128	32
cnv4_1	conv dw	3	2	128	128	32
cnv4_2	conv	1	1	128	256	16
cnv5_1	conv dw	3	1	256	256	16
cnv5_2	conv	1	1	256	256	16
cnv6_1	conv dw	3	2	256	256	16
cnv6_2	conv	1	1	256	512	8
cnv7_1	conv dw	3	1	512	512	8
cnv7_2	conv	1	1	512	512	8
cnv8_1	conv dw	3	1	512	512	8
cnv8_2	conv	1	1	512	512	8
cnv9_1	conv dw	3	1	512	512	8
cnv9_2	conv	1	1	512	512	8
cnv10_1	conv dw	3	1	512	512	8
cnv10_2	conv	1	1	512	512	8
cnv11_1	conv dw	3	1	512	512	8
cnv11_2	conv	1	1	512	512	8
cnv12_1	conv dw	3	2	512	512	8
cnv12_2	conv	1	1	512	1024	4
cnv13_1	conv dw	3	1	1024	1024	4
cnv13_2	conv	1	1	1024	1024	4
avg_pool	pool			1024	1024	4
fc	fc	1	1	1024	1000	1

数据流调度的浮点深度可分离卷积加速设计方案,利用多层卷积网络流水线协同的方式并行加速计算,提升最终的运算吞吐量。

2.1　基于数据流调度的 MobileNet 加速总体架构

针对基于卷积神经网络的目标区域筛选算法,设计基于数据流计算的卷积神经网络加速器。Xilinx UltraScale + 系列新品的内部物理结构分布为 3 个超级逻辑区(super logic region,SLR),每个逻辑区覆盖芯片内部的一个 Die,整颗芯片由 3 颗 Die 通过基底层互联。

考虑逻辑分区问题,将整个数据流计算分为 3 大块,第 1 块超级逻辑区(SLR0)覆盖卷积网络的前 13 层卷积层计算,即从 cnv0 到 cnv6_2 的计算;第 2 块超级逻辑区(SLR1)覆盖中间 10 层卷积层计算,即从 cnv7_1 到 cnv11_2 的计算;第 3 块超级逻辑区(SLR2)覆盖后 5 层卷积层计算,包括全连接计算,全连接计算可抽象为 1 × 1 的卷积计算,即从 cnv12_1 到 fc 的计算。

超级逻辑区之间通过 Xilinx 堆叠硅片互联(stacked silicon interconnect,SSI)技术互联,在超级逻辑区边缘有特定的专用寄存器进行跨逻辑区通信,基底层有超长线(super long line,SLL)来连接专用传输寄存器。超级逻辑区的内部访存和跨逻辑区通信需求均通过 SLL 互联实现。具体的基于数据流计算的可分离卷积神经网络加速总体架构如图 2 所示。

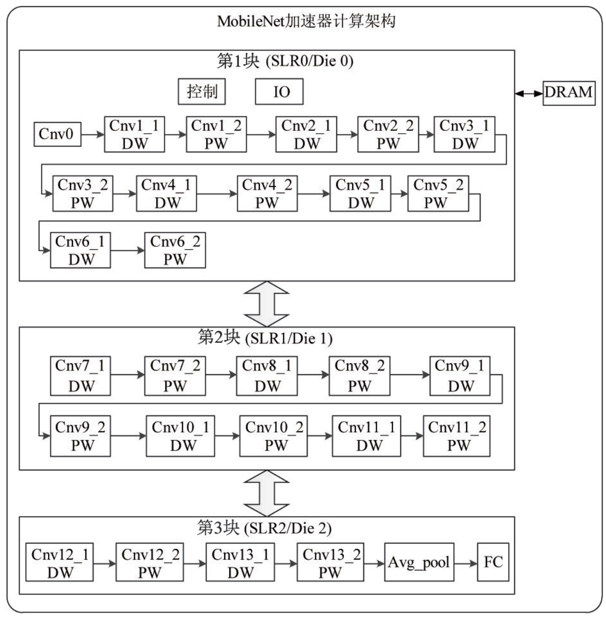

图 2　基于数据流调度的 MobileNet 加速总体架构

2.2　基于数据流调度的点卷积层设计

本节阐述基于数据流的点卷积层计算,主要包括卷积层权重输入、权重缓存、卷积输入数据流宽度变换、卷积计算阵列、批归一化处理、非线性响应等部分。具体的基于数据流的卷积层初始化与计算流程如图 3 所示,其中权重初始化只在权重缓存初始化的时候运行一次,后续的卷积计算不再需要从动态随机存取存储器(dynamic random access memory, DRAM)中调度缓存数据,降低存储带宽的开销,避免卷积计算阵列需要等待权重数据。

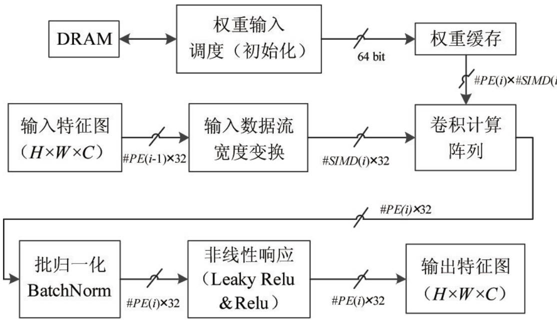

图 3　点卷积层初始化与计算流程

与常规卷积层计算流程相比,点卷积不需要进行输入数据流调度,只需要将输入宽度为处理单元(process element, PE)个数的浮点数据流转换为输入宽度为单指令多数据(single instruction multiple data, SIMD)个数的浮点数据流,因为点卷积进行的卷积运算核为 1 ×1,通常的特征图维持的 $H \times W \times C$ 即可满足 1 ×1 卷积的计算需求,不需要在特征图上跨行调度数据,因此简化了输入数据流调度为输入数据流宽度变换。卷积计算阵列依然采用基于前向累加链的卷积计算阵列设计,批归一化操作与非线性响应操作均无变化。

2.3　基于数据流调度的深度分离卷积层设计

本部分阐述基于数据流的深度分离卷积层计算,主要包括卷积层权重输入、深度分离卷积层权重缓存、深度分离卷积输入数据调度、深度分离卷积计算阵列、批归一化处理、非线性响应等部分。具体的基于数据流的深度分离卷积层初始化与计算流程如图 4 所示,其中权重初始化只在权重缓存初始化的时候运行一次,后续的深度分离卷积计算不再需要从 DRAM 中调度缓存数据,降低存储带宽的开销,避免深度分离卷积计算阵列需要等待权重数据。

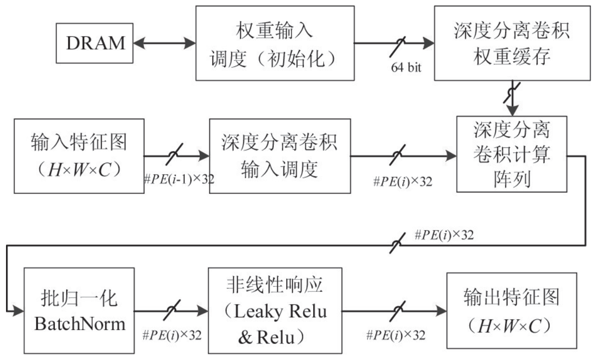

图 4　深度分离卷积层初始化与计算流程

与常规卷积层计算流程相比,深度分离卷积的批归一化操作与非线性响应操作均无变化。

(1)深度分离卷积层数据流调度

基于数据流的深度分离卷积计算要求输入数据以及权重都能够达到线速吞吐,这样深度分离卷积计算单元的流水周期不会出现空转,计算效率达到最高。基于线速吞吐的数据流调度要求,设计多端口的权重缓存,能够提供 PE 组权重,每组权重有 1 个权重 32 位浮点数;设计多端口深度分离卷积输入缓存,能够提供 1 个 32 位浮点数特征图输入;同时引入与深度分离卷积 s 行同等大小的输入先入先出队列(first input first output, FIFO),与深度分离卷积输入缓存构成异构“乒乓”缓存,去耦合上个阶段的计算输出调度和本阶段的深度分离卷积计算输入调度。输入特征图的输入宽度为上一层 PE 个数量的 32 位浮点数,通过输入 FIFO 转换宽度为 1 个 32 位浮点数据。卷积计算单元的输出为 PE 个 32 位浮点数据计算输出。具体的卷积层数据流调度如图5

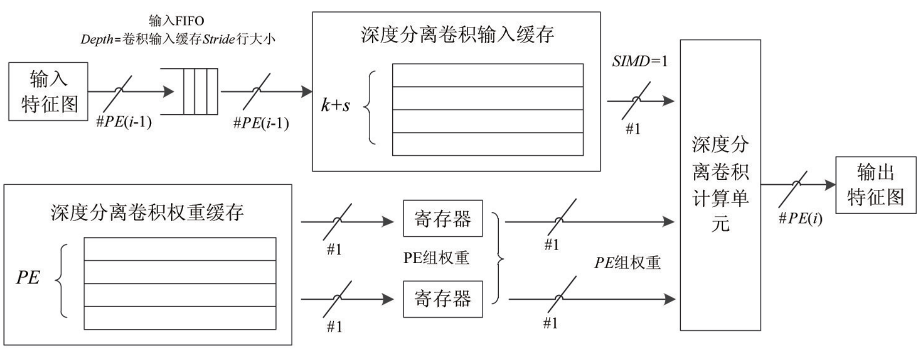

图 5　深度分离卷积层数据流调度

所示,实现了一组数据计算多个深度分离卷积输出,利用了输入数据的局部性。

(2)深度分离卷积层输入缓存设计

深度分离卷积层输入缓存的 *Stride* 行(*Stride* 为深度分离卷积滑窗步进)大小与输入 FIFO 的大小一致、结构不同,卷积层输入缓存将 $H \times W \times C$ 的输入特征图转换为卷积核所需的 $k \times k$ 的输入,通过在多组寄存器滑窗,为深度分离卷积计算阵列提供计算深度分离卷积所需的线性输入。具体的深度分离卷积输入缓存设计如图 6 所示。

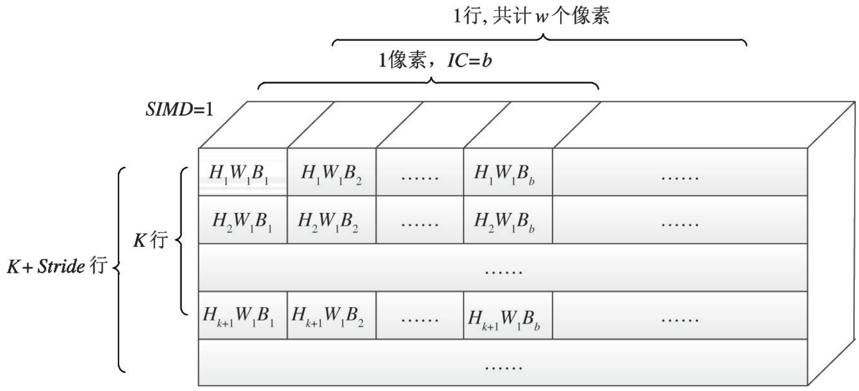

图 6　卷积层输入缓存设计

深度分离卷积层输入缓存包括 $k \times Stride$ 行寄存器组,其中 k 为深度分离卷积核大小,*Stride* 为深度分离卷积滑窗的步进大小。每行寄存器缓存输入特征图多个通道一行的数据,即 $W \times C$ 个数据,W 为输入特征图的宽度,*IC* 为输入特征图的通道数。每个寄存器行都可以按照 $SIMD = 1$ 个浮点数的宽度输出数据,一个像素的多个通道数据按照 *SIMD* 的宽度分为 b 个 Block,每个 Block 按照顺序放在寄存器行里。由于深度分离卷积的空间访存特性,需要从多个寄存器行里顺序读取数据,按照卷积核行列计算的要求顺序输出 *SIMD* 宽度的输入特征图数据流。

多个寄存器组分开存放,可以将深度分离卷积输入特征图的输出调度和输入调度分开,除了初始化读入 k 行输入特征图像素,后续可以按照 *Stride* 行补充输入特征图数据。虽然访问的同一个深度分离卷积输入缓存,利用多行深度分离卷积输入缓存寄存器,由此可以将输入数据与输出数据调取去耦合,避免数据依赖和数据污染,实现无锁的数据输入输出同时调度,满足深度分离卷积计算单元 *SIMD* 个浮点数据的输出线速调度要求。

(3)深度分离卷积层权重缓存设计

针对在片上缓存该层深度分离卷积计算所需的全部卷积,并且权重缓存的输出能按照深度分离卷积计算阵列 $PE \times 1$ 的计算模式输出面向多个处理单元 *PE* 输出每组 1 个浮点权重数据等要求,设计深度分离卷积层权重缓存。

深度分离卷积层参数按照式(1)维度顺序进行存放,其中 *PE* 为深度分离卷积计算单元的处理单元个数,每个深度分离卷积处理单元输入数据的宽度 *SIMD* 为 1,每个数据均为 32 位浮点数据。

$$W = PE \times T \tag{1}$$

深度分离卷积权重缓存 T 的计算如式(2)所示,其中 *OC* 为深度分离卷积核输出通道数量,k 为深度分离卷积核大小。

$$T = k^2 \times (OC/PE) \tag{2}$$

深度分离卷积层权重缓存具有 *PE* 组独立的缓存行,每个行的输出宽度为 1 个浮点数,由于卷积计算阵列的多个 *PE* 输出为输出特征图的一个像素的连续属性,故需要在权重缓存里将深度分离卷积权重按照多个 *PE* 行寄存器交叉存储,保证输出的 *PE* 组权重通过深度分离卷积计算阵列得到的是连续的输出特征图一个像素内部的连续属性值。具体的存放顺序如图 7 所示。

(4)深度分离卷积阵列计算单元设计

针对深度分离卷积计算高吞吐量的计算需求,设计深度分离卷积计算阵列。每个时钟周期能够处理 *PE* 个乘累加计算,其中 *PE* 为深度分离卷积阵列

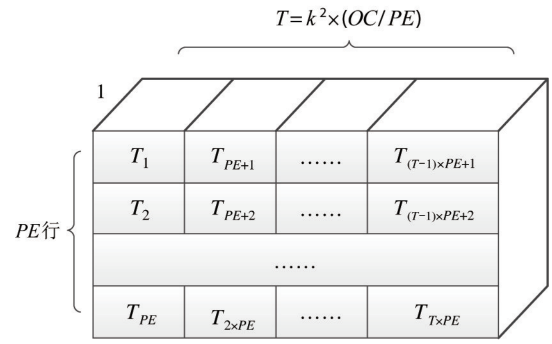

图 7　深度分离卷积层权重缓存设计

的行数，深度分离卷积输入特征图数据的个数为 1，输出为 *PE* 个深度分离卷积计算输出，*PE* 个浮点数据属于同一个输出特征图像素的连续属性值。

考虑浮点累加器调度延迟问题，主要在与累加器输出要反馈到累加器输入上，这个延迟决定了累加器的吞吐速率，如果能够打破累加器的后向反馈，则可以使累加器性能达到线速，进而深度分离卷积计算阵列达到线速。

卷积的累加次数是固定的，且累加次数随着每个深度分离卷积层参数的变化而变化。基于数据流的卷积计算阵列支持面向不同的深度分离卷积层分别配置深度分离卷积计算参数，主要是深度分离卷积阵列大小，为此本文设计了一个基于前向累加链的深度分离卷积累加器，深度分离卷积计算单元的每个 *PE* 输出 1 个数据，通过加法树 1 和 2 分别汇聚 k 个数累加。基于前向加法链的深度分离卷积累加器共需 N 个加法器，计算如式(3)所示。通过前向加速树的方式避免了卷积累加计算的负反馈过程，使得累加成为一个开环的设计，既完成了累加计算，又能够达到线速吞吐的目的。具体的基于前向加法树的深度分离卷积累加器设计如图 8 所示。

$$N = 2(k - 1) \tag{3}$$

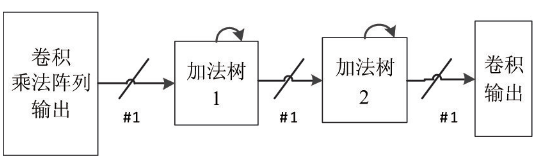

图 8　基于前向加法树的深度分离卷积累加器设计

基于乘法矩阵与前向加法树的卷积计算阵列包含两部分，一是乘法矩阵，完成 *PE* 个数的乘法运算；二是加法树矩阵，将多个周期的 $k\times k$ 个数汇聚成一个数，完成同一个深度分离卷积计算的累加功能。基于前向加法树的卷积阵列设计，打破了浮点数累加的后向反馈，不依赖于加法器的具体延迟，通过延长流水线的方式达到了深度分离卷积计算的线速累加和输出。具体的基于乘法矩阵以及前向加法链的深度分离卷积计算阵列如图 9 所示。

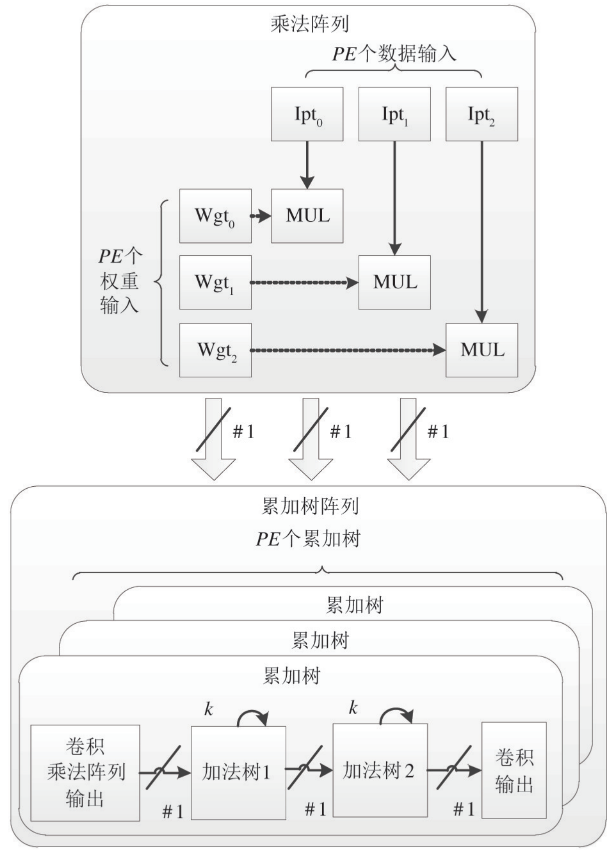

图 9　基于乘法矩阵与前向加法树的深度分离卷积计算阵列

3　实验结果和分析

3.1　加速器性能设计与分析

基于流式的卷积网络加速器可以配置不同的 *PE*、*SIMD* 参数，设计分析模型，可以获得不同的加速器性能。本节通过分析模型计算硬件预估性能，结合 HLS 高级综合资源占用分析，选取性能最高的 MobileNet 加速器进行分阶段设计，以适应 VU9P FPGA 芯片的分超级逻辑区放置资源。由于 MobileNet 模型可以调节不同的模型神经元通道深度 α 值来配置不同通道数的模型网络，但是对于硬件设

计单元来讲,可以针对不同的神经元通道深度 α 值来设计 MobileNet 加速器,对应的模型计算复杂度以及加速器处理吞吐量则随之有增减。

(1)MobileNet 卷积网络加速器配置

针对 MobileNet 模型 α 值为 1,结合硬件加速器各层 *PE*、*SIMD* 配置进行设计。MobileNet _ α _ 1 卷积网络加速器配置如表 2 所示,处理速度达到了 725.64 FPS,硬件卷积处理资源性能达到了 308.588 GFLOPS,相关吞吐量数据在214 MHz主频下得到。MobileNet _ α _ 0.75 加速器和 MobileNet _ α _ 0.5 加速器的配置与 MobileNet _ α _ 1 加速器大致相同,不同的地方在于 MobileNet _ α _ 0.75 加速器 cnv1 _ 2 层的 *SIMD* 参数设置为 12,MobileNet 各 α 值的硬件计算吞吐量大致相同,由于不同的模型神经元深度,吞吐量则按比例缩放。

表 2 MobileNet _ α _ 1 卷积网络加速器配置

名称	*PE*	*SIMD*	卷积循环次数	处理速度/FPS	硬件峰值吞吐量/MFLOPS
cnv0	4	3	294 912	725.64	5136
cnv1 _ 1	4	1	294 912	725.64	1712
cnv1 _ 2	2	16	262 144	816.34	13 696
cnv2 _ 1	2	1	294 912	725.64	856
cnv2 _ 2	2	16	262 144	816.34	13 696
cnv3 _ 1	4	1	294 912	725.64	1712
cnv3 _ 2	4	16	262 144	816.34	27 392
cnv4 _ 1	1	1	294 912	725.64	428
cnv4 _ 2	2	16	262 144	816.34	13 696
cnv5 _ 1	2	1	294 912	725.64	856
cnv5 _ 2	4	16	262 144	816.34	27 392
cnv6 _ 1	1	1	147 456	1451.28	428
cnv6 _ 2	2	16	262 144	816.34	13 696
cnv7 _ 1	1	1	294 912	725.64	428
cnv7 _ 2	4	16	262 144	816.34	27 392
cnv8 _ 1	1	1	294 912	725.64	428
cnv8 _ 2	4	16	262 144	816.34	27 392
cnv9 _ 1	1	1	294 912	725.64	428
cnv9 _ 2	4	16	262 144	816.34	27 392
cnv10 _ 1	1	1	294 912	725.64	428
cnv10 _ 2	4	16	262 144	816.34	27 392
cnv11 _ 1	1	1	294 912	725.64	428
cnv11 _ 2	4	16	262 144	816.34	27 392
cnv12 _ 1	1	1	73 728	2902.56	428
cnv12 _ 2	2	16	262 144	816.34	13 696
cnv13 _ 1	1	1	147 456	1451.28	428
cnv13 _ 2	4	16	262 144	816.34	27 392
avg _ pool	-	-	-	-	-
fc	1	16	46 080	4644.09	6848
总计				725.64	308 588

(2)基于分析模型的 MobileNet 加速器性能分析

对不同神经元深度 α 值的 MobileNet 加速器分析模型进行对比分析,具体如表 3 所示,相关吞吐量数据在 214 MHz 主频下得到。MobileNet _ α _ 0.5 加速器处理速率最高,达到了1451.28 FPS,性能是 MobileNet _ α _ 1 的 1 倍,各加速器的硬件计算吞吐量大致相同。

表 3 基于分析模型的 MobileNet 加速器性能对比分析

名称	处理速度/FPS	硬件峰值吞吐量/MFLOPS
MobileNet _ α _ 1	725.64	308 588
MobileNet _ α _ 0.75	967.52	305 164
MobileNet _ α _ 0.5	1451.28	308 588

3.2 基于 Vivado _ HLS 高级综合工具的加速器资源占用分析

(1)MobileNet 卷积网络加速器资源占用分析

基于 Vivado _ HLS 高级综合工具的 MobileNet 卷积网络加速器资源占用分析如表 4 所示,其中 BRAM 为块随机存储器,DSP 为数字信号处理器,FF 为触发器,LUT 为查找表,URAM 为超级随机存储器。从 SLR 资源用量占比来看,不同 α 值的 MobileNet 加速器所需资源占用超出单个 SLR 的资源总量,该配置中 URAM 资源占用率最高,达到了 SLR 资源用量占比的 184%,影像后续的综合实现过程,需要分割成多个阶段进行配置。从片上资源用量占比来看,各个 α 值的 MobileNet 加速器所需各类资源大致相同,虽然各个加速器的 *PE*、*SIMD* 配置大致相同,但输入输出的调度、加法链的长度以及加法树大小不同,导致了资源占比略有不同,但都未超过片上资源总量的 62%,表明基于分析模型的 MobileNet 加速器参数设置合理,适合在片上进行综合仿真。

表 4　MobileNet 卷积网络加速器资源占用

名称	α	BRAM	DSP	FF	LUT	URAM
片上资源总量	-	4320	6840	2 364 480	1 182 240	960
SLR 资源总量	-	1440	2280	788 160	394 080	320
用量	1	2129	4296	1 069 457	651 951	590
	0.75	2631	4246	1 021 983	642 095	502
	0.5	2386	4134	952 490	648 419	378
片上资源用量占比	1	49%	62%	45%	55%	61%
	0.75	60%	62%	43%	54%	52%
	0.5	55%	60%	40%	54%	39%
SLR 资源用量占比	1	147%	188%	135%	165%	184%
	0.75	182%	186%	129%	162%	156%
	0.5	165%	181%	120%	164%	118%

基于高级综合工具的 MobileNet 卷积网络加速器资源占用对比分析如图 10 所示。MobileNet _ α _ 1 加速器的资源占用最高的为 DSP 资源，达到了片上可用资源占比的 62%，使得加速器规模不能进一步扩大规模，同时也预留了资源给其他应用处理模块。各类资源的用量均已超出单个 SLR 资源的总量，如不对 MobileNet 各 α 值的加速器按照 3 个 SLR 进行切分设计加速器核，则不能满足 Vivado 综合的时序收敛要求。

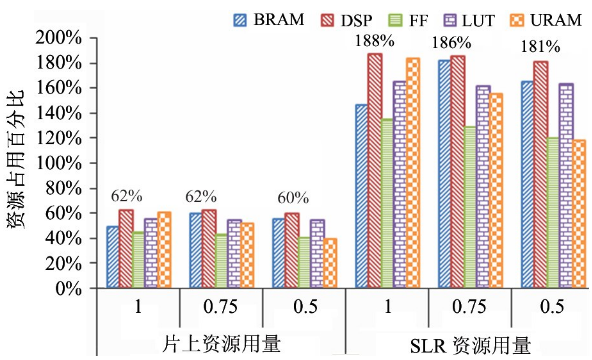

图 10　MobileNet 卷积网络加速器资源占用对比

(2)分阶段设计加速器资源占用对比分析

由于VU9PFPGA芯片内含3个逻辑分区，整体的 MobileNet 加速器占用的资源超过了一个逻辑分区，同时该设计需要一个较高的主频设计以期获得较高的吞吐能力，需要将 MobileNet 加速器切割为 3 个加速器核，分别进行仿真设计。具体的 MobileNet 分阶段设计加速器资源占用对比如图 11 所示，MobileNet _ α _ 0.5 加速器各项资源占用均值在 24% ~71% 之间，MobileNet _ α _ 0.75 加速器各项资源占用均值在 47% ~82% 之间，MobileNet _ α _ 1 加速器各项资源占用均值在 30% ~100% 之间，均能充分利用片上资源进行综合，同时又避免对布线资源过度使用造成拥塞，避免了高频时钟加速器设计的综合实现时序不收敛的问题。

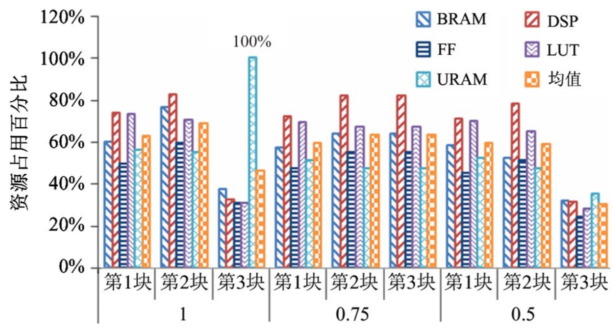

图 11　分阶段设计加速器资源占用对比

3.3　基于 Vivado 的加速器资源占用与功耗分析

可分离卷积网络加速器使用高级语言综合形成处理核以及 Vivado 板卡仿真结合的方式获得最终可执行的比特流，Vivado _ HLS 版本为 2019.1，Vivado 版本为 2019.1，基于 Vivado 综合工具的 MobileNet 可分离卷积网络加速器资源占用分析如图 12所示。其中，LUTRAM 为基于查找表的随机存储器，IO 为输入输出，GT 为千兆位收发器，BUFG 为一般时钟缓存，MMCM 为混合模式时钟管理器，PLL 为锁相环。通过分析，基于 Vivado 的资源占用与基于 HLS 的资源占用大致相当，不同 α 值的 MobileNet 加速器设计在 IO、GT、BUFG、MMCM 以及 PLL 资源占用情况一致，在 LUT、LUTRAM、FF、BRAM、URAM 以及 DSP 资源占用情况略有差别，符合设计的要求。最大的资源占用率为 URAM，约为 70%，其他的资源占用均低于 70%，使得加速器能够在有限的布线资源约束下时序收敛。

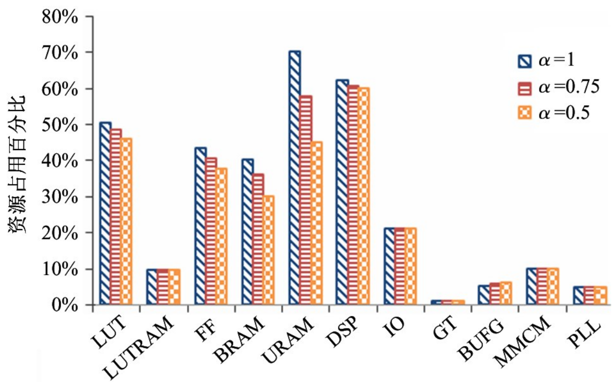

图 12　加速器综合实现后资源占用对比

基于 Vivado 仿真工具的 MobileNet 卷积网络加速器功耗分析如表 5 所示,MobileNet _ α _ 1 加速器片上功耗为 22.226 W,基于板卡测得处理速度为 633.69 FPS;MobileNet _ α _ 0.75 加速器片上功耗为 23.144 W,基于板卡测得处理速度为 855.12 FPS;MobileNet _ α _ 0.5 加速器片上功耗为20.974 W,基于板卡测得处理速度为 1283.90 FPS。不同 α 值的加速器功耗大致相当,由于设计的片上计算吞吐量大致相同,不同 α 值按比例缩减了模型的计算复杂度,其在板卡处理速度上按比例增加。

表 5　基于 Vivado 的 MobileNet 卷积网络加速器功耗对比

名称	速度/FPS	片上功耗/W	结温/℃
MobileNet _ α _ 1	633.69	22.226	36.5
MobileNet _ α _ 0.75	855.12	23.144	37.0
MobileNet _ α _ 0.5	1283.90	20.974	35.9

3.4　模型准确率与性能测试

基于开源 Google Earth 卫星遥感影像数据,本文采样了多时相的 0.5 m 分辨率的全球多个机场影像数据,并标注了 30 类具体的飞机型号分类数据作为模型训练数据。具体的飞机型号标注样本数据集如图 13 所示,包括 30 类常见飞机型号类型,共计 7882 张图片,每个类别的数量从 76 到 820 张不等。本文将该部分数据集做成 2 个训练数据集,并在 MobileNet 分类模型上进行训练得出相应的模型准确率。从数据量的降序排列的前 15 类别抽取数据,构成 15 类飞机型号分类数据集,称为 Plane15 数据集,共计 6017 张数据,采用该数据集训练出的模型称为 Plane15 模型;全部 30 类别数据构成 30 类飞机型号分类数据集,称为 Plane30 数据集,采用该数据集训练出的模型称为 Plane30 模型。

训练参数的设置包括批处理大小为 16,最大迭代次数为 200 000 次,优化器选择 Adam 优化器,初始学习速率为 0.0001。训练平台采用 Titan Xp 进行训练,软件框架采用 TensorFlow 1.5.0。训练时对数据进行 4 倍扩充,包括旋转 0 °和 90 °,饱和度增强 0 倍和 1 倍,训练、验证、测试数据集比例按照 8∶1∶1划分。

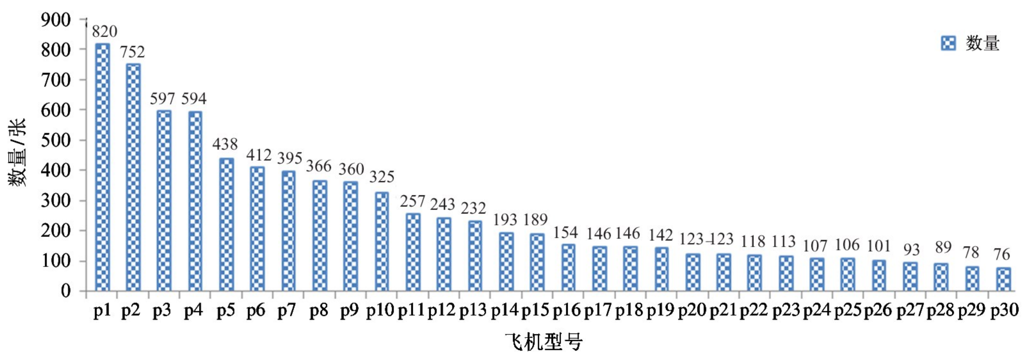

图 13　飞机型号标注样本数据集

分别用 Plane15 数据集和 Plane30 数据集训练 MobileNet 模型 Plane15 模型,训练过程如图 14 所示,经过 30 000 次迭代后模型基本收敛,在 70 000 次迭代时的损失与准确率曲线基本稳定。在 20 万次迭代中,选择各个数据集不同 α 值网络的最优模型进行准确率测试。Plane15 _ α _ 1 模型经过 200 000批次迭代训练后验证准确率为 98.9% ,测试 Top-1 准确率为 98.9% ;Plane15 _ α _ 0.75 模型经过 200 000 批次迭代训练后验证准确率为 98.2% ,测试 Top-1 准确率为 97.8% ;Plane15 _ α _ 0.5 模型

经过200 000批次迭代训练后验证准确率为98.3%，测试Top-1准确率为98.5%；Plane30_α_1模型经过200 000批次迭代训练后验证准确率为96.9%，测试Top-1准确率为97.2%；Plane30_α_0.75模型经过200 000批次迭代训练后验证准确率为97.2%，测试Top-1准确率为97.3%；Plane30_α_0.5模型经过200 000批次迭代训练后验证准确率为96.3%，测试Top-1准确率为96.1%。Plane30相关数据集模型在验证集和测试集上的准确率比Plane15数据集相关模型稍低一些，但也表现了相当高的准确率，这证明谷歌的MobileNet模型在其他应用上也有很好的学习与泛化能力，具体的测试结果如表6所示。

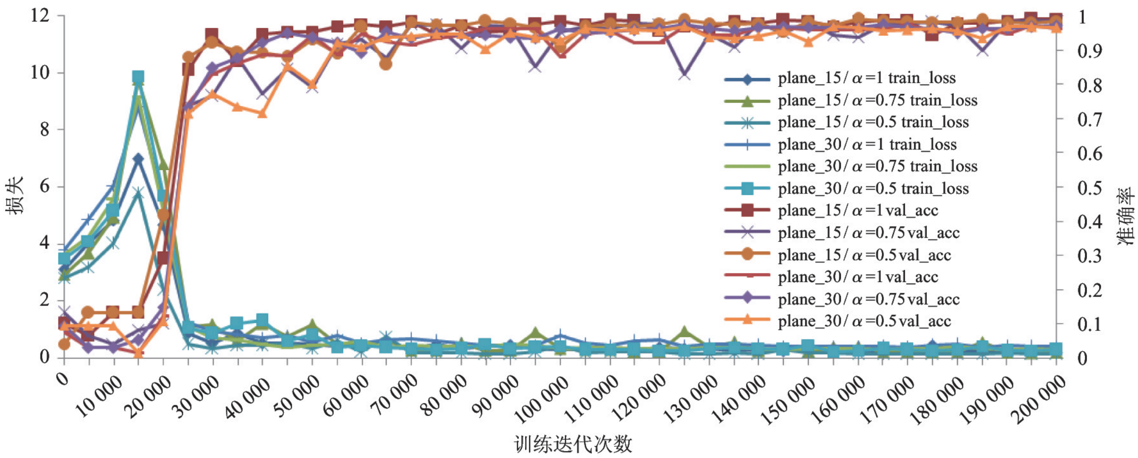

图14 MobileNet模型训练过程

表6 模型准确率实验结果

模型	α	验证	测试	
			Top-1	Top-5
Plane_15	1	98.9%	98.9%	99.9%
	0.75	98.2%	97.8%	99.9%
	0.5	98.3%	98.5%	99.9%
Plane_30	1	96.9%	97.2%	99.5%
	0.75	97.2%	97.3%	99.7%
	0.5	96.3%	96.1%	99.5%

经过Vivado综合工具综合后，在VCU 1525板卡对MobileNet模型进行了吞吐量测试，并将其结果与GPU模型进行性能对比，具体的模型性能对比如表7所示。GPU的速度测试从batch为1、2、4、8、16中挑出最高的性能进行对比，batch为8时，MobileNet模型的GPU性能最高。Plane15与Plane30模型都是基于MobileNet的GPU分类实例，测试吞吐量大致相当，功耗较高。基于FPGA实现的MobileNet_α_0.5模型达到了最高的吞吐量，为1283.90 FPS，是GPU模型的2.61、2.27倍，功耗远低于GPU平台，计算效能是GPU的18.66、16.24倍。MobileNet加速器相对于GPU的加速比为1.10～2.61倍，能效比为7.44～18.66倍，具体的加速比与能效比结果如图15所示。

表7 加速器性能对比

模型	平台	处理速度/FPS	功耗/W	处理效率/(FPS/W)
Plane_15_α_1	Nvidia Titan Xp	504.42	～150	3.36
Plane_15_α_0.75		502.50	～150	3.35
Plane_15_α_0.5		491.99	～150	3.28
Plane_30_α_1		574.54	～150	3.83
Plane_30_α_0.75		570.23	～150	3.80
Plane_30_α_0.5		565.13	～150	3.77
MobileNet_α_1	FPGA VCU 1525	633.69	22.226	28.51
MobileNet_α_0.75		855.12	23.144	36.95
MobileNet_α_0.5		1283.90	20.974	61.21

3.5 与通用FPGA加速卷积计算模型对比

将MobileNet_FPS 725的性能与其他工作对比，如表8所示，文献[16,18]的工作采用了基于硬核浮点DSP的Intel Arria 10器件，吞吐量较高，除了

采用浮点硬核加速的相关工作,本文算法的运算吞吐量达到了最高值,为 236.04 GFLOPS;运算效能也达到了最高值,为 10.62 GFLOPS/W,GOPS/DSP 指标也取得了较好结果。实验结果表明,基于乘法矩阵与前向加法树的深度分离卷积计算阵列设计有效地解决了浮点可分离卷积计算的线速吞吐瓶颈,提高了 DSP 资源的利用率,进一步提升了加速器的吞吐量与效率。

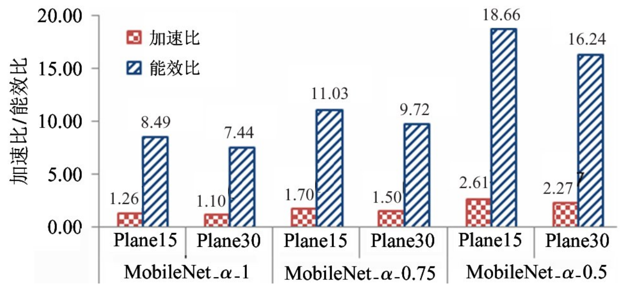

图 15　MobileNet 加速器与 GPU 加速比与能效比

表 8　与其他浮点加速卷积网络工作对比

名称	年份	位宽	描述语言	设备	主频/MHz	吞吐量/GOPS	功耗/W	LUT(k)	DSP	GFLOPS/W	GOPS/DSP
文献[14]	2015	Float 32	HLS	Virtex7 VX485T	100	61.62	18.61	186	2240	3.31	0.03
文献[15]	2016	Float 32	HLS	Virtex7 VX485T	100	75.16	33.93	28	2695	2.22	0.03
文献[16]	2017	Float 32	OpenCL	Arria10 GX 1150	370	866.00	41.70	437	1320	20.77	0.66
文献[17]	2016	Float 32	OpenCL	Stratix-V GXA7	181	33.90	27.30	-	162	1.24	0.21
文献[18]	2017	Float 32	OpenCL	Arria10 GX 1150	271	360.40	-	350	1290	-	-
文献[19]	2019	Float 32	-	ZC 706	150	22.74	-	145	596	-	-
文献[20]	2020	Float 32	Verilog	XCZU9EG	100	42.13	24.00	124	784	1.76	0.05
文献[21]	2019	Float 32	HLS	ZYNQ 7100	100	17.11	4.08	142	1926	4.19	0.01
MobileNet_α_1(本文算法)	-	Float 32	HLS	VU + VU9P	214	236.04	22.23	598	4273	10.62	0.06
MobileNet_α_0.75(本文算法)	-	Float 32	HLS	VU + VU9P	214	182.67	23.14	577	4155	7.89	0.04
MobileNet_α_0.5(本文算法)	-	Float 32	HLS	VU + VU9P	214	127.67	20.97	546	4109	6.09	0.03

4　结　论

本文面向遥感影像飞机型号分类应用,采用轻量型网络 MobileNet 构建型号分类模型,并通过基于 FPGA 的数据流调度与浮点模型进行加速,突破了浮点加速可分离卷积的线速吞吐问题,达到了加速模型与原始模型准确率一致性的要求,使得软硬件协同开发去耦合,软件算法人员专注于训练模型以及模型认知不确定性研究。通过在 Xilinx VCU1525 开发板上构建浮点可分离卷积图像分类网络加速器,运算性能最高为 236.04 GFLOPS,型号分类处理速度为 633 FPS,功耗为 22.226 W。实验结果表明,本文方法计算速度达到了 Titan X GPU 的 1.10 ~ 2.61倍,计算效能是 Titan X GPU 的 7.44 ~ 18.66 倍。通过与同类的 FPGA 加速浮点卷积网络方案对比,本文方法计算吞吐量以及计算能效比达到了最优。后续可就基于轻量型网络的多尺度目标检测加速进一步开展研究。

参考文献

[1] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, USA, 2015: 1-9

[2] SANDLER M, HOWARD A, ZHU M, et al. MobileNetv2: inverted residuals and linear bottlenecks[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 4510-4520

[3] GUPTA S, AGRAWAL A, GOPALAKRISHNAN K, et al. Deep learning with limited numerical precision[C]// International Conference on Machine Learning, Lille, France, 2015: 1737-1746

[4] COURBARIAUX M, HUBARA I, SOUDRY D, et al. Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or -1 [EB/OL]. https://arXiv. org/pdf/1602. 02830. pdf: arXiv, (2016-03-17),[2021-04-23]

[5] HOWARD A G, ZHU M, CHEN B, et al. MobileNets: efficient convolutional neural networks for mobile vision applications [EB/OL]. https: // arXiv. org/pdf/1704. 04861. pdf:arXiv,(2017-04-17),[2021-04-23]

[6] ZHANG X, ZHOU X, LIN M, et al. ShuffleNet: an extremely efficient convolutional neural network for mobile devices[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 6848-6856

[7] REDMON J, FARHADI A. YOLO9000: better, faster, stronger[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, USA, 2017:7263-7271

[8] HUANG R, PEDOEEM J, CHEN C. YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers[C]//2018 IEEE International Conference on Big Data, Seattle, USA, 2018: 2503-2510

[9] HUANG R, GU J, SUN X, et al. A rapid recognition method for electronic components based on the improved YOLO-V3 network[J]. *Electronics*, 2019, 8(8): 1-18

[10] LI X, QIN Y, WANG F, et al. Pitaya detection in orchards using the MobileNet-YOLO model[C]//The 39th Chinese Control Conference, Shenyang, China, 2020: 6274-6278

[11] CHEN T, DU Z, SUN N, et al. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning[J]. *ACM SIGARCH Computer Architecture News*, 2014, 42(1): 269-284

[12] JOUPPI N P, YOUNG C, PATIL N, et al. In-data center performance analysis of a tensor processing unit[C]// Proceedings of the 44th Annual International Symposium on Computer Architecture, Toronto, Canada, 2017: 1-12

[13] JOUPPI N P, YOON D H, KURIAN G, et al. A domain-specific supercomputer for training deep neural networks [J]. *Communications of the ACM*, 2020, 63(7): 67-78

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

[15] RAHMAN A, LEE J, CHOI K. Efficient FPGA acceleration of convolutional neural networks using logical-3D compute array[C]//2016 Design, Automation and Test in Europe Conference and Exhibition, Dresden, Germany, 2016: 1393-1398

[16] ZHANG J, LI J. Improving the performance of OpenCL-based FPGA accelerator for convolutional neural network [C]//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, USA, 2017: 25-34

[17] WANG D, AN J, XU K. PipeCNN: an OpenCL-based FPGA accelerator for large-scale convolution neuron networks[EB/OL]. https: // arxiv. org/pdf/1611. 02450. pdf:arXiv, (2016-11-08),[2021-04-23]

[18] WEI X, YU C 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

[19] 蹇强, 张培勇, 王雪洁. 一种可配置的CNN协加速器的FPGA实现方法[J]. 电子学报, 2019, 47(7): 1525-1531

[20] 赵烁, 范军, 何虎. 基于FPGA的CNN加速SoC系统设计[J]. 计算机工程与设计, 2020, 41(4): 939-944

[21] LIU B, ZOU D, FENG L, et al. An FPGA-based cnn accelerator integrating depthwise separable convolution[J]. *Electronics*, 2019, 8(3): 1-18

FPGA based floating point separable convolutional neural network acceleration method

ZHANG Zhichao* ** *** ****, WANG Jian* ** ***, ZHANG Longbing* ** ***, XIAO Junhua* ** ***
(* State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Science, Beijing 100190)
(** Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)
(*** University of Chinese Academy of Sciences, Beijing 100049)
(**** The 15th Research Institute of China Electronics Technology Group Corporation, Beijing 100083)

Abstract

In order to solve the problems of speed bottleneck and power limitation in the application of separable convolutional neural network in space-borne aircraft target classification, a floating point depthwise separable convolution neural network acceleration method is proposed based on field programmable gate array (FPGA) data stream scheduling to accelerate the general MobileNet image classification model. The design of depthwise separable convolution computation array based on multiplication matrix and forward addition tree is adopted to solve the bottleneck of line speed throughput in floating point acceleration of depthwise separable convolution. Experimental results show that the target classification based on FPGA has a speed of 633 FPS, a power consumption of 22.226 W, and a computing performance of 236.04 GFLOPS. The computational speed is 1.10 – 2.61 times higher than that of Titan Xp GPU, and the computational efficiency is 7.44 – 18.66 times higher than that of Titan Xp GPU. In the same kind of FPGA-based floating-point convolution acceleration scheme, the proposed method achieves the best performance and energy efficiency ratio. At the same time, the proposed method provides image classification accuracy consistent with the original model, decouples the software/hardware collaborative development process, and reduces the threshold for application developers to use FPGA to accelerate calculation.

Key words: depthwise separable convolution, field programmable gate array (FPGA), data stream scheduling, acceleration, image classification