第 8 卷 第 5 期
2019 年 9 月
集 成 技 术
JOURNAL OF INTEGRATION TECHNOLOGY
Vol. 8 No. 5
Sep. 2019

引文格式：
田烁, 李石明, 王蕾, 等. 一种用于加速神经视觉识别的硬件架构 [J]. 集成技术, 2019, 8(5): 58-71.
Tian S, Li SM, Wang L, et al. A hardware architecture for accelerating neuromorphic visual recognition [J]. Journal of Integration Technology, 2019, 8(5): 58-71.

一种用于加速神经视觉识别的硬件架构

田 烁[1] 李石明[1] 王 蕾[1] 徐 实[2] 徐炜遐[1]

[1](国防科技大学计算机学院 长沙 410073)
[2](国防科技创新研究院 北京 100000)

摘 要 深度学习的广泛应用带来了视觉分析中许多类似人类认知任务的实现。HMAX 是基于视觉皮层的生物启发模型，已在多类物体识别中被证明优于标准计算机视觉方法。但是，由于神经形态算法的高复杂性，在边缘设备上实现 HMAX 模型仍然面临巨大挑战。已有研究表明，HMAX 的 S2 阶段是运行最耗时的阶段。该文提出了一种基于脉动阵列的新架构来加速 HAMX 模型的 S2 阶段。仿真结果表明，与基准模型相比，HMAX 模型最耗时的 S2 阶段执行时间平均减少了 14.65%、内存所需的带宽减少了 3.34 倍。

关键词 深度学习；HMAX；脉动阵列；加速器
中图分类号 TP 391.9 **文献标志码** A doi: 10.12146/j.issn.2095-3135.20190729003

A Hardware Architecture for Accelerating Neuromorphic Visual Recognition

TIAN Shuo[1] LI Shiming[1] WANG Lei[1] XU Shi[2] XU Weixia[1]

[1](*College of Computer Science and Technology*, *National University of Defense Technology*, *Changsha* 410073, *China*)
[2](*National Innovation Institute of Defense Technology*, *Beijing* 100000, *China*)

Abstract The widespread application of deep learning has led to the realization of many human-like cognitive tasks in visual analysis. HMAX is a visual cortex-based bio-inspired model that has proven superior to standard computer vision methods in multi-class object recognition. However, due to the high complexity of neural morphology algorithms, implementing HMAX models on edge devices still faces significant challenges. Previous experimental results show that the S2 phase of HMAX is the most time-consuming stage. In this paper, we propose a novel systolic array-based architecture to accelerate the S2 phase of the HAMX model. The simulation results show that compared with the baseline model, the execution time of the most time-consuming S2 phase of

收稿日期：2019-07-29 **修回日期**：2019-08-16
基金项目：超级计算机处理器研制项目(2017ZX01028103)；国家自然科学基金项目(61802427)
作者简介：田烁(通讯作者)，博士研究生，研究方向为人工智能硬件加速，E-mail：tianshuo14@nudt.edu.cn；李石明，博士研究生，研究方向为人工智能硬件加速；王蕾(通讯作者)，博士，副研究员，硕士研究生导师，研究方向为计算机体系结构、异步电路和人工智能计算，E-mail：leiwang@nudt.edu.cn；徐实，博士，助理研究员，研究方向为集成电路设计；徐炜遐，博士，研究员，博士研究生导师，研究方向为计算机体系结构、高性能微处理器设计和人工智能。

the HMAX model is reduced by 14.65% on average, and the required memory bandwidth is reduced by a factor of 3.34 X.

Keywords　deep learning; HMAX; systolic array; accelerator

1　引　言

近年来，深度学习和人工感知算法广泛应用于许多领域，如自动驾驶汽车[1]、监视和安全[2]以及太空探索[3]等。卷积神经网络(Convolutional Neural Network，CNN)[4]是使这些应用成为可能的关键网络拓扑之一。虽然目前学者仍在研究如何增强这些算法，但灵长类动物特别是人类的视觉处理在效率和能力方面仍然优于这些算法。例如，作为一个大规模并行处理器的人脑，包含 1 000 亿个并行操作的神经元，可以提供 10^{16} FLOPS(每秒浮点运算)，同时仅仅消耗 20 W。

HMAX(Hierachical Model and X)[5]由 Riesenhuber 与 Poggio 于 1999 年提出，是一种用于物体分类的皮质模型。该模型通过以分层方式从一组简单特征构造复杂特征，主要模拟腹侧视觉路径以提供不变对象识别的方法。它主要集中在任何自上而下的效果(如注意力转移和眼球运动)之前的几百毫秒，称为“即时物体识别”，这意味着不需要对它进行训练。文献[6-7]已经证明，在几个标准数据集的分类中，使用 HMAX 作为对象分类的模型优于标准计算机视觉方法。

已有研究表明，基于图形处理单元(GPU)的 CNS-HMAX 框架[8]比基于中央处理器(CPU)的实现快 97 倍。然而，其过高的功耗和计算带宽阻碍了嵌入式或低功耗系统的部署。在 Nvidia Tesla C1060 GPU 上运行的 CNS-HMAX 框架[8]的结果表明，S2 特征的计算占据了算法总执行时间的近 85%。为解决该问题，Sabarad 等[9]针对 HMAX 模型中最耗时的 S2 层设计了可重构卷积引擎加速器 CoRe16。它是一个可重新配置的卷积引擎，可以并行执行多个可变大小的卷积，并行度高且十分灵活。实验结果表明，与基于皮质层网络模拟器(Cortical Network Simulator，CNS)的实施相比，该加速器速度提高了 5～15 倍。但其模型使用类似加法树累积的方法来计算每个像素点，这无疑会降低模型的执行速度。

在各种二维列选项中，脉动阵列架构[10]因为局部移位数据移动自然地响应原生二维卷积的本地数据流而自然地匹配 CNN。脉动阵列还可以有效地处理在深度神经网络(Deep Neural Networks，DNN)训练和运行长短期记忆网络期间发生的矩阵-矩阵和矩阵-向量运算。因为脉动阵列中操作数的移动是局部的(邻居到邻居)，所以其具有计算密度高(低面积)、功率低及控制简单等特点。结合精心设计的内存层次结构，脉动阵列架构可以利用 DNN 中的大量数据重用来实现更高的计算吞吐量，同时消耗更少的存储带宽。由于这些特性，脉动阵列已广泛应用于 Google TPU[11]、Xilinx FPGA[12]和学术研究[13]。

在对深度学习计算设计的加速器中，Sanchez 等[14]提出了一种用于实时边缘 CNN 处理的一维卷积加速器。它主要对计算密集度最高的 CNN 第一层的原始二维拓扑进行一维优化。Yang 等[15]针对深度学习算法提出了一种 32×32 脉动阵列，并讨论了可变大小的 CNN 和循环神经网络算法的映射方法。Du 等[16]将大量的共享权值放在片上静态随机存取存储器中，从而大量降低了动态随机存取存储器(Dynamic Random Access Memory，DRAM)的读写功耗。但其在获得权值后，使用广播的方法到达每个处理单元(Processing Element，PE)，在较大规模的 PE 阵

列中布局布线问题将变得十分困难。Chen 等[17]则提出了一种行停留的数据流模式，将权重以行(或列)的形式存储在每行(或列)PE 中来减少数据移动，从而减少能耗。但其输出点需要重新累加来获得，降低了执行速度。

本文所提出的 systolic 阵列中的数据流动没有通过广播，而是通过每拍传递(输入数据从左至右传递，权重数据从上至下传递)，减少了布局布线。计算过程中每个 PE 产生一个输出点，在大规模的计算过程中，可减少数据传递，提高计算速度。本文的主要贡献点如下：

(1) 设计了可以计算不同卷积大小的 HMAX 模型脉动阵列加速器；

(2) 通过设计的数据流和数据转发，可有效降低脉动阵列所需的内存带宽。

2 HMAX 模型

HMAX 是一个四阶段特征提取模型，本文使用的为 Serre 等[6]开发的 HMAX 算法的扩展。该模型紧密跟随视觉皮层的层次系统，并通过交替模板匹配和最大池操作构建越来越复杂和不变的特征表示。HMAX 模型如图 1 所示。灰度值图像首先通过 S1 层在 4 个不同方向、16 个尺度大小执行卷积操作(图中为 8 个尺度大小)；然后，C1 层为相邻两个尺度中具有相同方向的特征值提供局部最大池化(M)；在随后的下一阶段，S2 层实际上使用具有一组 patch 的径向基函数(Radial Basis Function，RBF)单元，其中这些 patch 是从一组代表性的图像中随机采样得到；最后，C2 层对 S2 层进行最大池化操作，生成 C2 值。

S1 层：灰度值输入图像首先由简单 S1 元素的多维数组处理，对应于在初级视觉皮层中发现的 Hubel 和 Wiesel 的经典简单细胞[18]。S1 层由 Gabor 函数表示[19]。Jones 和 Palmer[20]已经证明它可以为皮质单细胞感受野提供一个很好的模型，具体描述方程如下：

$$F(x,y)=\exp\left(-\frac{x_0^2+\gamma y_0^2}{2\sigma^2}\right)\times\cos\left(\frac{2\pi}{\lambda}x_0\right) \quad (1)$$

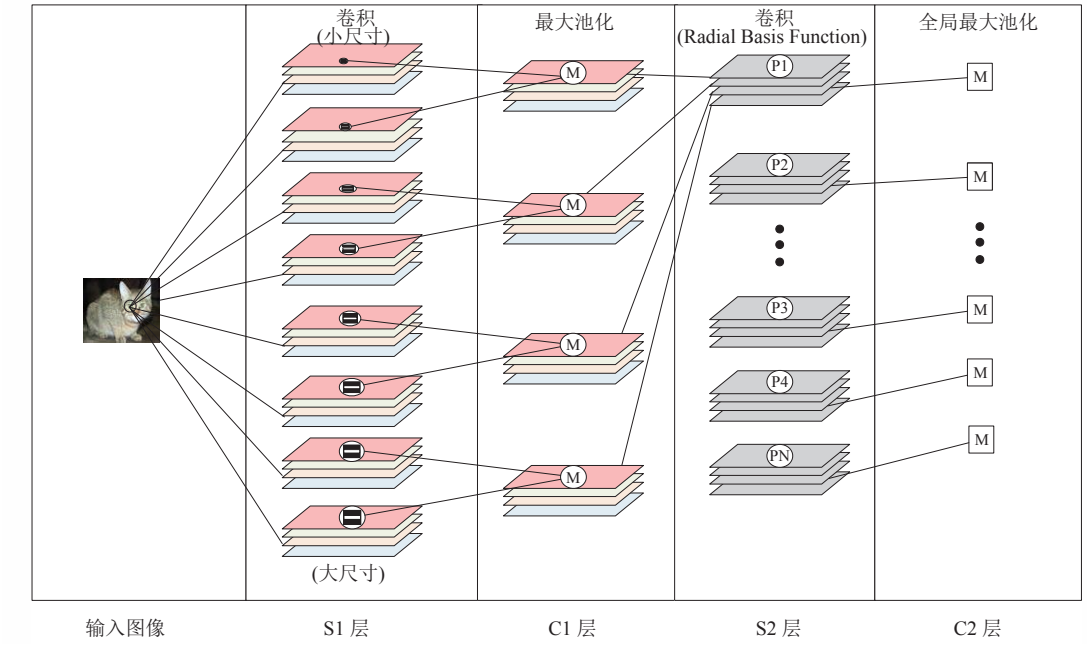

图 1 HMAX 模型概述

Fig. 1 HMAX model overview

$$x_0=x\cos\theta+y\sin\theta,\ y_0=-x\sin\theta+y\cos\theta \quad (2)$$

调整所有滤波器参数，如纵横比 $\gamma=0.3$、方向 θ、有效宽度 σ、波长和滤波器尺寸 S，使相应的 S1 单元的调谐特性与基于两组数据的大量初级视觉皮层中的简单单元相匹配[21-22]。这是通过对参数空间进行采样，并应用相应的滤波器来检测皮质细胞的刺激(即光栅、条和边)，并选择大量初级视觉皮层单细胞的调谐特性后的参数值(见表 1)。设置 S1 过滤器来制作一个比例金字塔，它跨越一系列尺寸，从 7×7 到 37×37 像素分两步。为了保持可管理单元的数量，本文选择了 4 个方向(0°、45°、90° 和 145°)，共有 64 个不同的 S1 层滤波器(16 尺度×4 个方向)。

C1 层：C1 单元汇集来自前一层的视网膜组织 S1 层的输入，对具有相同的方向和相同的组最大池化操作(表 1)。每个组包括 2 个相邻的滤波器尺寸(具有 8 个组，总共 16 个 S1 滤波器尺寸)。例如，组 1 包含大小为 7×7 和 9×9 的 S1 滤波器。S1 层的组也决定了每个 C1 层($N_S\times N_S$)的池化大小。同样，该过程是针对 4 个方向和每个组独立实现的。复杂单元的响应 r 来自前一 S1 层的 m 个最大输入响应 $(x_1,\cdots,x_m)$，使得：

$$r=\max x_j,\ j=1,\cdots,m \quad (3)$$

例如，考虑第一个组：$S=1$。对于每个方向，它包含 2 个 S1 映射：一个使用大小为 7×7 的滤波器获得，另一个使用大小为 9×9 的滤波器获得(如表 1 所示)。映射具有相同的维度，但它们是不同过滤器的结果。C1 单元响应通过使用池大小 $N_S\times N_S=8\times 8$ 对这些图进行次采样来计算。从每个池中，通过取所有 64 个元素的最大值来完成。作为最后一个阶段，C1 层从相同的空间邻域中获取最大值，只记录两张地图中的最大值。需要注意的是，C1 响应不是在每个可能的位置计算的，而只是在 Δ_S 的数量上重叠，这使得下一阶段的计算更加有效。调整控制池化操作的参数(表 1)，可使 C1 单元的调整与实验测量的复杂组细胞的调整相匹配[20]。

S2 层：在 S2 层中，单元在所有 4 个方向上

表 1　S1 和 C1 层参数总结

Table 1　Summary of the S1 and C1 parameters

C1 层			S1 层		
组数 S	池化网格（$N_S\times N_S$）	重叠 Δ_S	滤波器大小 s	Gabor σ	Gabor γ
组 1	8×8	4	7×7	2.8	3.5
			9×9	3.6	4.6
组 2	10×10	5	11×11	4.5	5.6
			13×13	5.4	6.8
组 3	12×12	6	15×15	6.3	7.9
			17×17	7.3	9.1
组 4	14×14	7	19×19	8.2	10.3
			21×21	9.2	11.5
组 5	16×16	8	23×23	10.2	12.7
			25×25	11.3	14.1
组 6	18×18	9	27×27	12.3	15.4
			29×29	13.4	16.8
组 7	20×20	10	31×31	14.6	18.2
			33×33	15.8	19.7
组 8	22×22	11	35×35	17.0	21.2
			37×37	18.2	22.8

聚集来自局部空间邻域的输入 C1 单元。S2 单元采用径向基函数 RBF 单元[23]的形式。每个 S2 单元的响应取决于新输入和存储的权重(patch)之间的欧几里德距离。也就是说，对于来自先前 C1 层的特定比例 S 的图像映射 X，对应 S2 单元的响应 $R(X,P_i)$ 由下式给出：

$$R(X,P_i)=\exp\left(-\beta\|X-P_i\|^2\right) \tag{4}$$

其中，β 定义了调谐的清晰度；P_i 是从一组代表性图像中随机抽取的 n 个特征(RBF 单元的中心)之一。在运行时，针对 8 个组中每一个组的所有位置计算 S2 特征图。为每个(0～4 000 个)P_i 计算一个这样的多尺度图。

C2 层：最后一组移位和尺度不变的 C2 响应是通过对整个 S2 矩阵上每个 S2 类型的所有尺度和位置的全局最大池化计算得到的，即 S2 层存储的 patch 组 P_i 与每个位置和尺度的输入图像之间相匹配，且只保留最佳匹配值，而丢弃其余的。最终结果是 n 个 C2 值的向量，其中 n 对应于 patch 的数量。C2 响应 R_{C2} 可表示为：

$$R_{C2}=\max\{R(X,P_i)\} \tag{5}$$

或者，可以将 S2 和 C2 计算重写为：

$$R(X,P_i)=\left(\beta\|X-P_i\|^2\right) \tag{6}$$

$$R_{C2}=\exp\{\min R(X,P_i)\} \tag{7}$$

这种重新公式化允许将指数运算转移到 C2 计算中。因此，C2 操作是全局最小值而不是全局最大值，而指数只需要根据最小值而不是整个数组来计算，即只需要对 C2 向量直接执行一次指数操作。同样，在方程中也考虑了常数。这有助于减少计算资源(乘法器)，进而加速卷积运算。

3 加速器结构

在本节中，将介绍硬件加速器的体系结构 SHALE(Systolic HMAX Accelerator)。SHALE 加速器由脉动阵列组成，主要针对 HMAX 最耗时的 S2 阶段，具有高度的模块化空间和时间规律性。

3.1 脉动阵列结构

如图 2 所示，模型 SHALE 是典型的脉动阵列。其中，Patch 组从上到下流动，输入数据从左向右流动。在模型中，输出特征图的每个像素都分配到给定的 PE。假设资源不受限制，只需要与输出像素一样多的 PE，并在每个周期中流式传输所需的操作数。就产生给定像素而言，在计算元件数量有限的实际情况下，资源是时分复用的。一旦给定的 PE 生成输出像素，

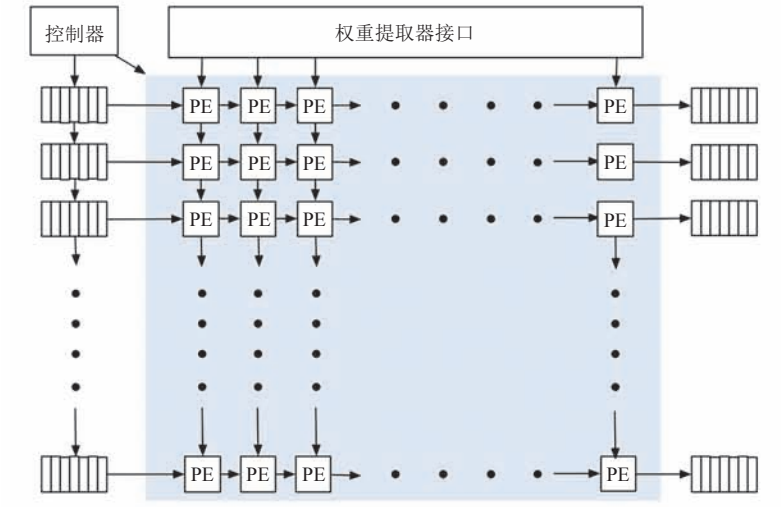

图 2 SHALE 模型结构

Fig. 2 The structure of SHALE

结果就会传输到内存，并分配 PE 来计算下一个像素。

图 3 为数据流的示意图，这里 kernel 大小为 4，S2 大小为 60。数据从阵列的左边缘和上边缘馈送。其中，左边缘中流动的为 C1 输入特征值，上边缘流动的为 patch 组值。在给定列中，每行中的 PE 负责输出相同组中不同 patch 组的像素。但是，每列产生对应于相同 patch 组的不同输出像素。本文使用 16×16 来形成加速模型，以便在不同形状的脉动阵列中实现最佳性能(详情见 4.4 小节)。

3.2　处理单元

处理块是加速器中的基本计算元件，并且被定制用于计算 S2 层中的 RBF 特征。PE 的简化框图如图 4 所示。从左到右到激活寄存器的 PE 输入的是 C1 特征向量，它们是来自 C1 中相同组的不同方向的特征。然后它们以延迟(1 个周期)的值传递给正确的 PE。类似地，从上到下到权重寄存器的 PE 输入是不同的 patch，并且延迟(1 个周期)值被传递到下面的 PE。减法器的数值流入乘数器，以计算 C1 特征和 patch 之间的欧几里德距离的部分结果。该部分乘积由加法器累加。累积和的数量取决于正在处理的 S2 kernel 的大小和方向的数量，确保每个 PE 计算 S2 的一个像素。所有操作和传输均由控制寄存器控制。

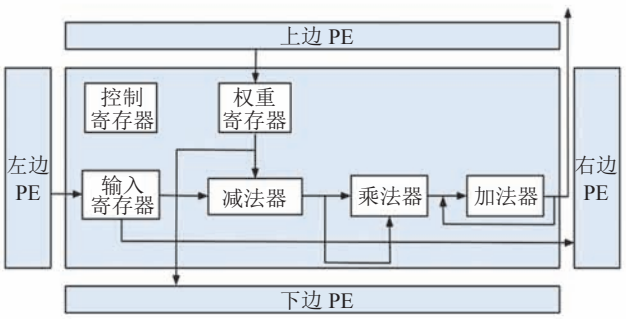

图 4　单个处理单元结构

Fig. 4　Schematic diagram of a single processing element

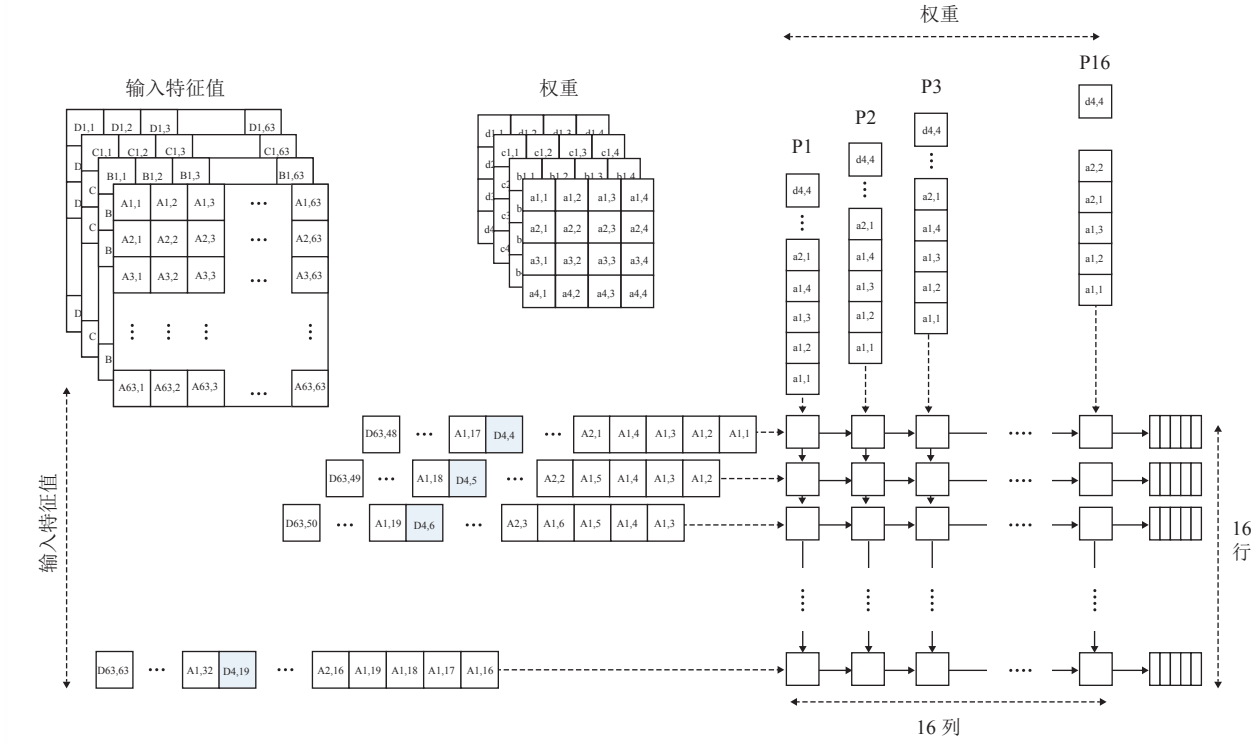

图 3　SHALE 模型数据流示意图

Fig. 3　Data flow diagram of SHALE

4 实验结果及分析

在本节中，主要使用 Matlab 仿真方法模拟 HMAX 模型的计算和运行时间来比较性能。Matlab 在 Mac OS Mojave 下运行，CPU 为 2.6 GHz Intel Core i5 和 8 GB DDR3 1600 MHz。本文主要使用 Serre 等[6]的 HMAX 模型，因为它是 HMAX 模型的经典实现，具有优异的识别性能，并且在包括许多不同对象类别的各种图像数据集上优于标准的系统。HMAX 模型(S1、C1、S2、C2)的基本实现可以在 Serre 等[6]的研究中找到。示例数据集是 Wild 数据库中标记面部的 10 张图像[24]。

本文还模拟 Sabarad 等[9]的研究作为基准线。实验中主要截取 Serre 等[6]研究中的 C1 数据作为输入，并且 patch 组使用从随机自然图像中提取的一组 8 种不同大小的通用 patch。为了与基准线中的 kernel 数量相同，本文使用其中的 4 个(kernel＝4,8,12,16)。本文将减法运算设置为 1 个节拍，乘法运算设置为 5 个节拍，并将加法运算设置为 1 个节拍。为了更好地计算模型运行时间，本文为每个 PE 单元添加一个计时器，以计算运算的周期数。此外还添加了寄存器以传递脉动阵列中的数据。关于存储带宽要求，本文通过计算输入 C1 数据流来获得周期中所需的不同数据的数量。SHALE 模型和基准线的运算结果与 Serre 等[6]研究中的 S2 数据完全一致，验证了模型计算的正确性。

以本文采用的 16×16 大小的 systolic 阵列为例，如图 3 所示。在第 1 个节拍到来时，最左上角的 PE 对数据第一行的(A1,1)以及 patch 组 P1 的(a1,1)进行减以及乘累加计算。在第 2 个节拍到来时，左边第一行第二列 PE 对传递过来的第一行的(A1,1)以及 patch 组 P2 的(a1,1)进行计算，同时左边第二行第一列的 PE 对第二行的数据(A1,2)以及传递过来的 patch 组 P1 的(a1,1)开始计算。第 3 个节拍到来时，左边第一行第三列、第二行第二列、第三行第一列开始计算。以此类推，当达到 31 个节拍时，最右下角的 PE 开始计算，即可灌满整个阵列。本文以最右下角最后一个 PE 计算完成作为整个运算的结束。

计算过程的循环分两部分：输入数据的循环和 patch 组的循环。其中，输入数据的循环次数与 S2 大小相关，patch 组的循环次数与 patch 组数量相关。在此 S2 大小取 60。因为 systolic 阵列中每一列对应一个 patch 组输出的 16 个像素点，所以数据循环的次数为 60×60÷16＝225，即一次数据循环中可得出一个 patch 组中 16 个输出点。一次 patch 组循环有 16 个 patch 组，其中 patch 数量取 400，所以 patch 组一共循环 400÷16＝25 次。PE 中输出的像素点时间与 kernel 大小、方向以及计算时间相关。这里 kernel 大小取 4，方向大小为 4，计算时间为一次减、乘累加占用的时间，为 7 个节拍。所以总的计算时间为 225×25×4×4×4×7＋31＝2520031 个节拍。

4.1 计算延迟的比较

图 5 为 S2 大小对不同 kernel 大小的总延迟的影响。总延迟在一定程度上反映了模型的性能。其中，总延迟越小，计算速度越快。从图 5 可以看出，本文所提出的模型 SHALE 和 CoRe16[8]的总延迟随着 S2 尺寸的增加而增加。这是因为随着 S2 尺寸的增加，模型需要计算的节点逐渐增长，总延迟增加。几乎在所有情况下，本文模型比 CoRe16 表现更好，这主要是因为他们的模型使用加法树累积来计算每个像素，这降低了模型的速度。SHALE 可以通过设计的数据流计算每个 PE 中的相应像素点。在最好的情况下(kernel＝12)，本文模型总延迟比 CoRe16 减少了 46.6%。

图 6 显示了与基准线相比模型运行时间减少的百分比。其中，S2 尺寸的大小为 1～60。当

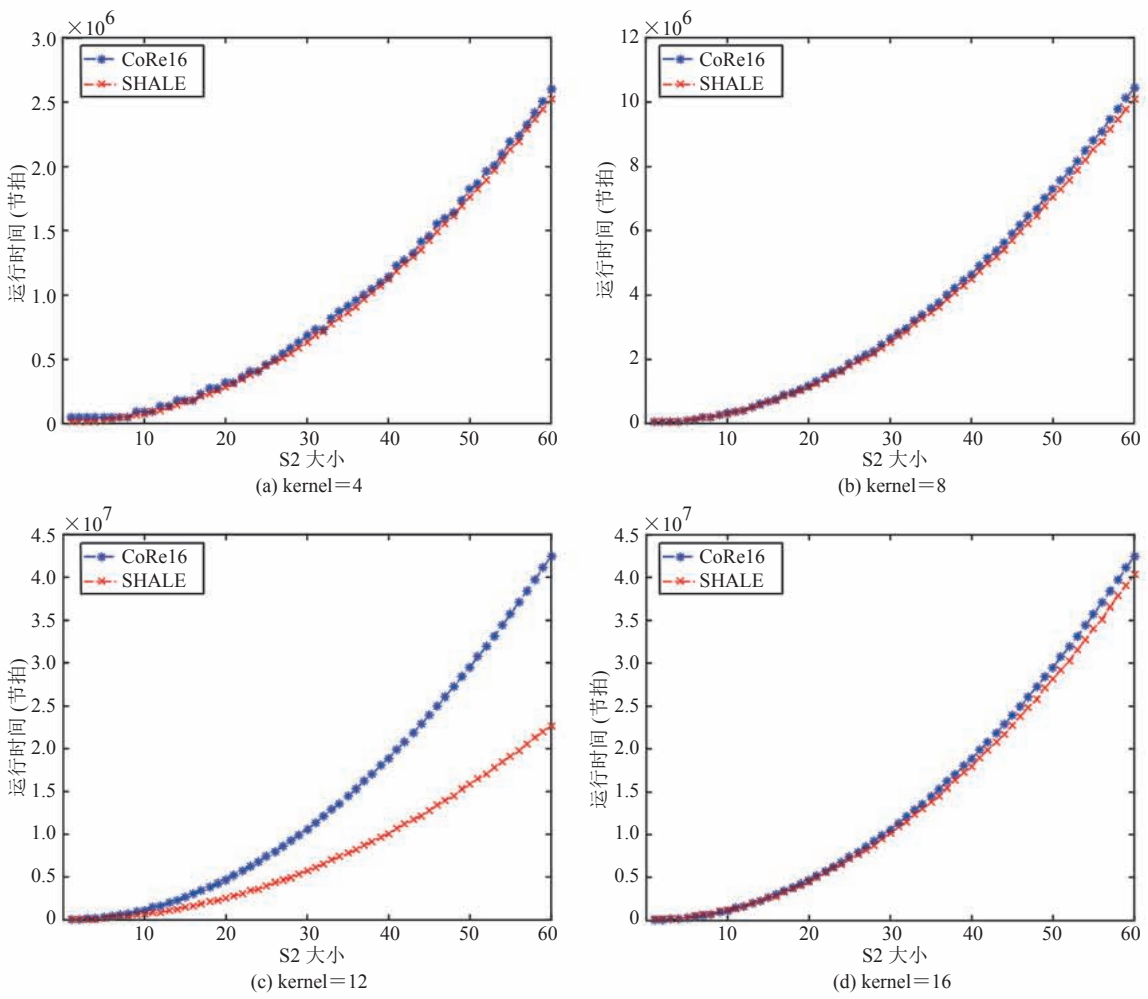

图 5　S2 大小对总时延影响

Fig. 5　Effect of S2 size on total latency

kernel 尺寸从 4 到 16 时，平均总延迟分别减少了 11.18%、3.44%、39.67% 和 2.5%。可以看出，在大多数情况下，加速器 SHALE 可以获得更好的性能。但当 kernel＝12 或 16 且 S2 尺寸很小(小于 10)时，SHALE 加速效果不如 CoRe16 的好。这主要是因为 CoRe16 使用加法树的累加来计算每个像素，当 S2 尺寸较小时，其单元利用率较高(参见 4.3 节)。

与图 6 中 S2 尺寸取值(1～60)不同的是，大多数计算集中在 S2 尺寸为 10～60。因此，本文也进一步取 S2 尺寸为 10～60 时对 SHALE 运行时间与基准线做对比(图 7)。从图 7 可以看出，随着 S2 尺寸的增加，与 CoRe16 相比，本文模型

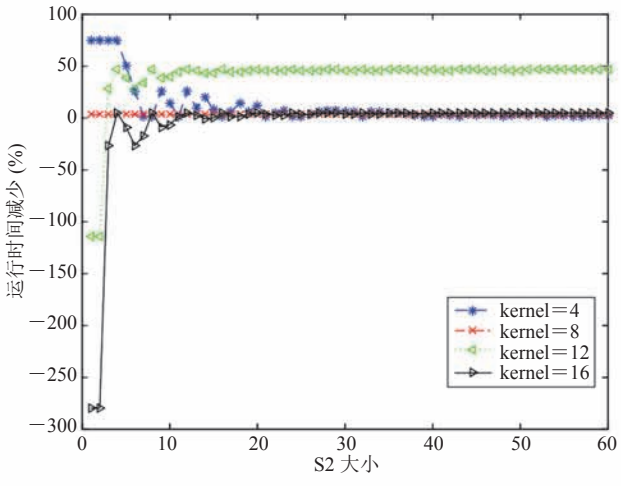

图 6　SHALE 模型和基准线运行时间减少百分比(1≤S2≤60)

Fig. 6　Percentage of SHALE run time reduction compared to baseline (1≤S2≤60)

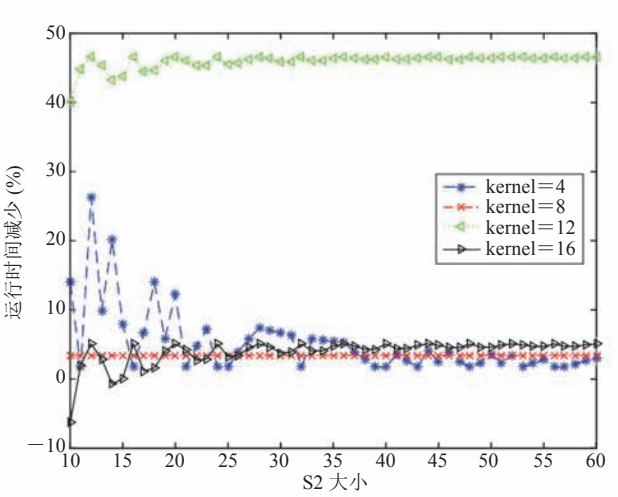

图 7 SHALE 模型和基准线运行时间减少百分比(10≤S2≤60)

Fig. 7 Latency reduction compared to baseline (10≤S2≤60)

运行时间减少与基准线相比的百分比逐渐稳定，但几乎在所有情况下，它仍然优于基准线。在不同的 kernel 大小下，平均延迟减少了 14.65%。

图 8 为不同 kernel 大小的 patch 组数量对总延迟的影响。这里 S2 大小为 60，权重组数量从 400 到 4 000，总共 10 组。从图 8 可以看出，本文所提出的模型和 Core16 的总延迟随着 patch 数量的增加而增加。这是因为随着 patch 组数量的增加，模型需要计算的节点数量逐渐增加，因此总延迟也会增加。几乎在所有的情况下，SHALE 模型都比 CoRe16 表现更好。这主要是因为 SHALE 加速器重复使用不同 patch 组中的数

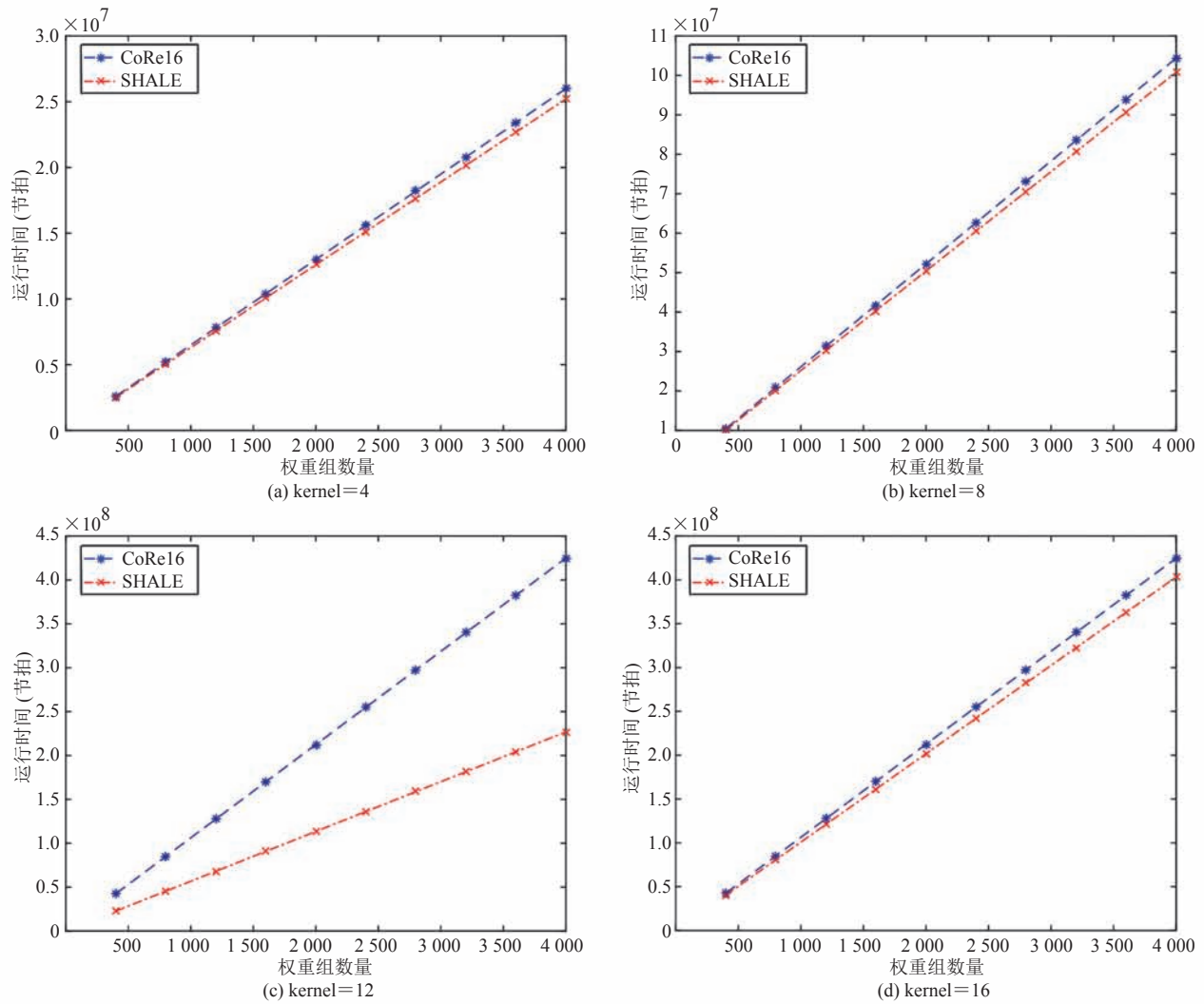

图 8 不同 patch 组对总时延的影响

Fig. 8 Effect of patches on latency

据并选择适当的脉动阵列大小(参见 4.4 小节中的分析)，而一次只计算一个 patch 组的 CoRe16 只重用 patch 组，而不重用 C1 数据流。

图 9 为不同 patch 组下 SHALE 运行时间减少与基准线相比的百分比。从图 9 可以看出，随着 patch 组数量的增加，与基准线相比，模型运行时间减少的百分比在不同的 kernel 大小下是稳定的。同时，随着 kernel 尺寸的增加，SHALE 运行时间减少与基准线相比的百分比也在稳步提高。这主要是因为 Sabarad 等[9]使用加法树累积方法来计算较大 kernel 的像素点。kernel 越大，累积操作所需的时间越长。在本文所提出的模型中，每个固定的 PE 单元产生相应的 S2 像素，并且 kernel 大小仅影响单个 PE 为每个像素计算的时间。在 kernel 大小为 12 的情况下，与基准线相比的最大减少为 46.6%，这主要是由于 SHALE 模型阵列单元的高利用率(参见 4.3 小节)。在不同的 patch 数量下，SHALE 的平均计算延迟比基准线低 14.55%。

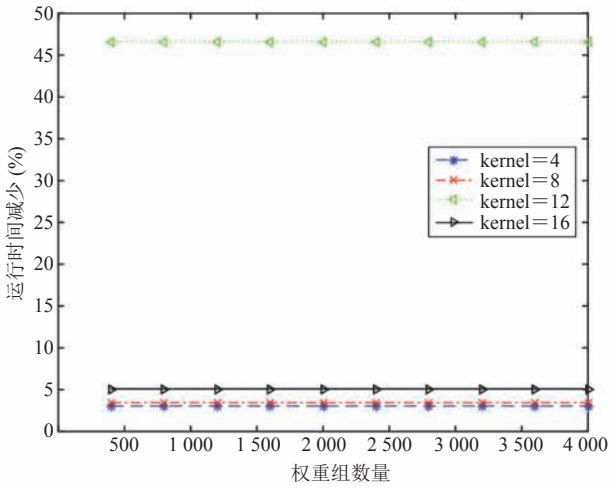

图 9　不同 patch 组下与基准线运行时间减少的百分比

Fig. 9　Latency reduction compared to baseline in different patches

4.2　存储带宽的比较

存储带宽为 HMAX 模型在单个节拍计算所需的数据量。在这里，本文将数据位宽设置为 32 位，patch 组为 400，S2 大小为 60。

图 10 为模型 SHALE 在不同 kernel 下和基准线的存储带宽要求。可以看出，随着 kernel 大小的增加，SHALE 模型所需的存储带宽越来越小，而 CoRe16 几乎没有变化(kernel 为 12 时除外)。这主要是因为随着 kernel 大小的增加，单个数据复用的次数也就越多，SHALE 模型在一个周期内(与 kernel 大小相关)，需要从内存中读取的数字越少。CoRe16 通过 PE 单元与 kernel 大小为 4 的组合来计算更大的 kernel 像素点。完美组合匹配(即 kernel 大小为 8 或 16)时，带宽几乎不变。当 kernel 大小为 12 时，其单元利用率会降低，所需的存储带宽也会降低。

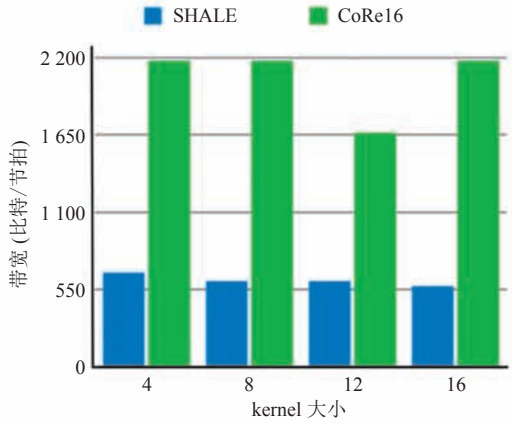

图 10　存储带宽需求

Fig. 10　Storage bandwidth requirement

这主要是通过在输入数据的一个节拍中观察一列 16 行数据流而获得的。由于本文提出的模型是 16×16 脉动阵列，因此一个节拍内所需的最大数据量为(16＋16)×32＝1024 位。但是数据可以重复使用，因此只需要从内存中加载不同的数据，并且可以通过转发(广播)来实现相同的数据。如数据流(图 3)所示，kernel 大小为 4，(A1, 4)可以在一个周期内重复使用 4 次(等于 kernel 大小)，而对于内存只能重复一次。本文实验从加速器中提取了数据流。对于 16 个 C1 输入，只需要在每个周期加载 5×32＝160 位。对于 16 行输入 patch，每个 patch 组不同，每个循环需要

16×32＝512 位。因此，每个周期加载共 512＋160＝672 位数据。

类似地，对于 CoRe16，由于它是通过累加加法树实现的，因此 CoRe16 中 1×4 处理模块之一的一个周期所需的最大数据量是(1＋4)×32＝160 位。虽然基准线由 64 个这样的处理模块组成，但每个周期所需的最大数据量为 160×64＝10240 位。然而，C1 数据的 CoRe16(基准线)中的每个组合处理单元(Composed Processing Blocks，CPB)(4 个 1×4 个处理模块)在每个周期仅需来自存储器的 4×32＝128 位。这是因为 C1 数据的其他 PE 输入可以通过广播来实现。CoRe16 总共包含 16 个 CPB，因此每个周期共需要 128×16＝2048 位。关于 patch 组，每循环一个 CPB 组需要 4×32＝128 位(对应于所需的 C1 数据)，其余 15 个 CPB 使用相同的 patch 组，这可以通过转发来实现。因此，CoRe16 每个周期总共需要 2048＋128＝2176 位。与基准线相比，SHALE 存储带宽减少了 3.24 倍。在不同 kernel 下，存储带宽平均可减少 3.34 倍。

4.3 单元利用率的比较

图 11 为模型 SHALE 和不同 kernel 下的基准线阵列单元的利用率的比较。其中，patch 组为 400，S2 大小为 1～60，其他参数不变。可以看出，随着 S2 尺寸的增加，SHALE 的单元利用率逐渐稳定，可以达到近 100%。这主要是因

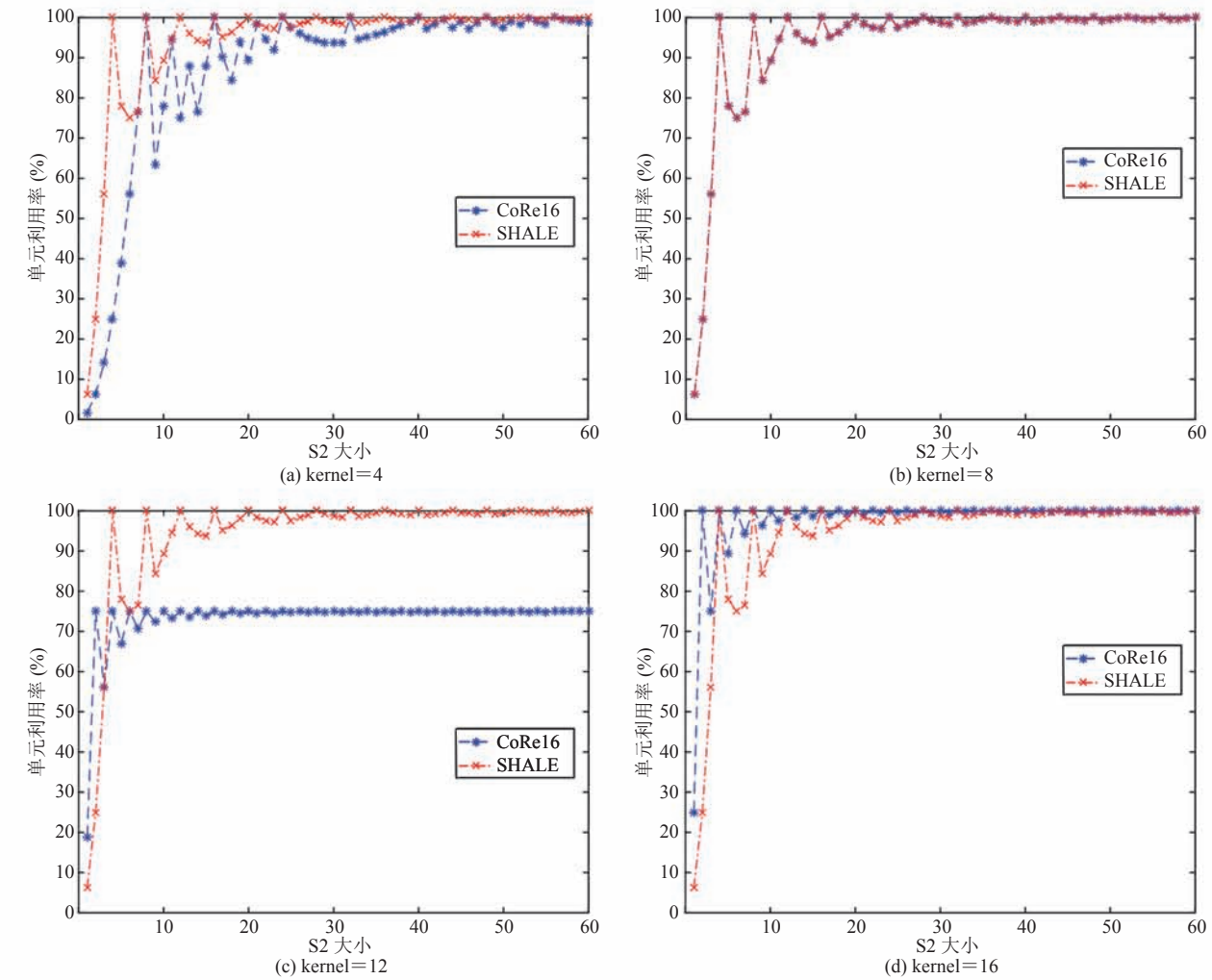

图 11 单元利用率比较

Fig. 11 Unit utilization comparison

为 S2 尺寸越大，输入的 C1 数据流越多，模型 SHALE 在满载时运行的次数越多(一次至少 256 个像素)，未使用单元占用的比例越少。在不同的 kernel 下，其单元利用率是不变的。这主要是因为本文所提出的模型通过每个 PE 单元来计算像素点。因此，kernel 的大小仅影响 PE 单元中单个像素的计算时间。

对于基准线，随着 S2 尺寸的增加，单元利用率逐渐变得稳定。在不同的 kernel 下，CoRe16 的单元利用率不同。这是因为基准线模型主要通过加法树积累来计算像素点。Kernel 值越大，模型计算的像素越少，相对单元阵列的利用率越高。

图 12 为本文模型单元利用率在不同 kernel 下与基准线相比提高的百分比。可以看出，在大多数情况下，SHALE 模型单元利用率将更好或更接近基准线。当 kernel 等于 12 或 16 且 S2 很小(小于 10)时，其利用率不如 CoRe16。这主要是因为 SHALE 模型为每个 PE 单元计算一个像素点，而在小数据流的情况下，整个模型的循环数量很小，并且未使用的单元阵列的数量相对较大。基准线使用较小的 kernel 来累积较大 kernel 的像素。因此在 S2 尺寸较小的情况下，可以在 CoRe16 中使用更多的单位计算。

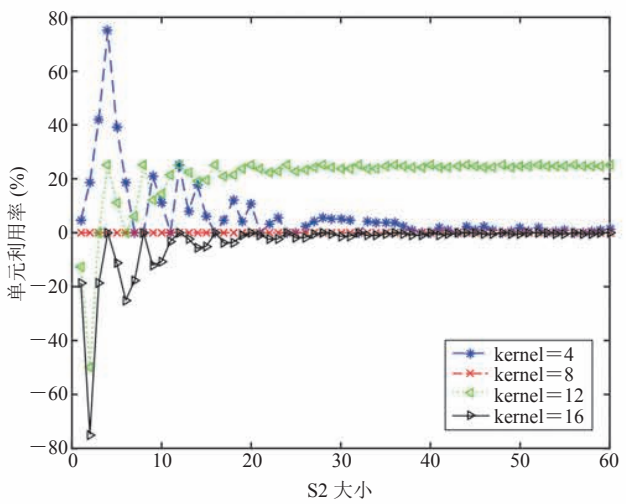

图 12　SHALE 模型单元利用率在不同 kernel 下与基准线相比提高的百分比

Fig. 12　Percentage of SHALE unit utilization improvement compared to baseline under different kernels

4.4　不同形状阵列的运行时间

图 13 为不同形状脉动阵列的运行时间。此处固定 PE 的总数为 256(与基准线相同)，patch 组为 400，S2 大小为 60，kernel 大小为 4，方向大小取 4。在 systolic 阵列大小为 1×256 中，前者 1 表示输入阵列的大小，后者 256 表示 patch 组阵列大小。从图 13 可以看出，当阵列大小为 16×16 时，运行时间是最短的，与 1×256、2×128、4×64、8×32 相比，分别降低 21.9%、21.8%、10.7%、3.8%；与 256×1、128×2、64×4、32×8 相比，分别降低 6.2%、3.0%、1.3%、0.4%。这主要是因为方形的 systolic 阵列能够正好利用 patch 组的循环，即 patch 组数对 16 能够整除，同时数据的循环也可以很好地利用，阵列的单元利用率较高。同时，可以看出当阵列越接近正方形时，运行时间越短。

图 13　不同形状的脉动阵列的运行时间

Fig. 13　Runtime of our systolic array in different shapes

4.5　能耗评估

在本小节中，评估了 HMAX 模型在脉动阵列上每张图像的能耗。其中，图像尺寸为 250×250。S1 层使用了有 16 个组和 4 个方向，S2 层有 4 000 个 patch 组。时钟频率是 100 MHz。能耗评估分为两部分：计算能耗和 DRAM 访存

能耗。本文首先通过 Vivado 仿真单个 PE 的计算并获得波形图，验证计算功能的正确性；然后，通过寄存器转换级电路(RTL)综合来计算脉动阵列的功耗。SHALE 模型的计算能耗通过：脉动阵列功耗×一个像素所需的计算次数×阵列完成整个操作所需的计算次数×每个时钟的时间得到。

Scale-sim[25]是一个可以精确到每个时钟的可配置脉动阵列 DNN 模拟器。本文使用 Scale-sim 来获取输入和 patch 组权重的读取轨迹，以及 HMAX 模型的每一层中输出特征 DRAM 的写入轨迹。然后使用 DRAMsim2[26]来评估模型的 DRAM 访存能耗。DRAMsim2 中的示例轨迹主要分为 3 个部分：存储器访问地址、类型(读或写)和存储器访问时间。在示例轨迹中，假设只保留一个读取的访存，可以获得 DDR3 中的一个读取功耗。在 DRAMsim2 中，一次读取的数据大小是 64 字节。在本文中，数据的大小定义为 4 个字节。本文将其除以 16 以获得 DRAM 单个数据的读取功耗。写功耗采用相同的方法得到。之后本文使用 HMAX 模型对由 Scale-sim 实现的每一层 DRAM 进行读写跟踪，以获得 DRAM 的读写访问次数。两者相乘后再乘以时钟时间即可得到每层的访存能耗。

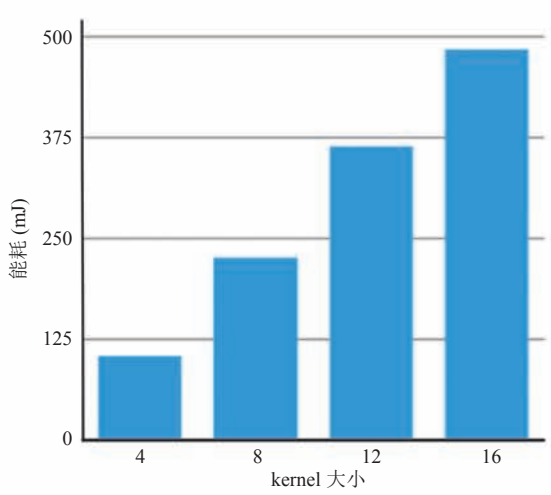

图 14 HMAX 模型在不同 kernel 尺寸下的能耗

Fig. 14 Energy consumption in different kernel sizes for HMAX model

图 14 为每张图像在 HMAX 模型中使用 4.1 节数据流的计算和 DRAM 访存能耗。可以看到，随着 kernel 大小的增加，每张图像的计算和内存消耗逐渐增大。这主要是因为随着 kernel 大小增加，计算 S2 层中的单个像素所需的计算量大大增加，并且所需的计算能耗和访存能耗也逐渐提高。

5 结 论

本文提出了一种基于 HMAX 的多类目标识别算法的脉动加速器。加速器的核心是通过使用设计的数据流计算每个 PE 中的相应像素点。仿真结果表明，与基准线模型[9]相比，HMAX 模型最耗时 S2 阶段的执行时间平均减少了 14.65%，内存所需带宽减少了 3.34 倍。本文将在基于现场可编程门阵列(FPGA)的 HMAX 硬件上实施提出的脉动加速器模型 SHALE，为边缘智能系统提供高速、高能效的解决方案。

参 考 文 献

[1] Cerri P, Soprani G, Zani P, et al. Computer vision at the hyundai autonomous challenge [C] // International IEEE Conference on Intelligent Transportation Systems, 2011: 777-783.

[2] Collins RT, Lipton AJ, Kanade T, et al. A system for video surveillance and monitoring [R]. CMU-RI-TR-00-12, 2000.

[3] Matthies L, Maimone M, Johnson A, et al. Computer vision on mars [J]. International Journal of Computer Vision, 2007, 75(1): 67-92.

[4] Farabet C, Martini B, Akselrod P, et al. Hardware accelerated convolutional neural networks for synthetic vision systems [C] // International Symposium on Circuits and Systems (ISCAS), 2010: 257-260.

[5] Riesenhuber M, Poggio T. Hierarchical models

of object recognition in cortex [J]. Nature Neuroscience, 1999, 2(11): 1019-1025.

[6] Serre T, Wolf L, Bileschi S, et al. Robust object recognition with cortex-like mechanisms [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29(3): 411-426.

[7] Mutch J, Lowe DG. Object class recognition and localization using sparse features with limited receptive fields [J]. International Journal of Computer Vision (IJCV), 2008, 80(1): 45-57.

[8] Mutch J, Knoblich U, Poggio T. CNS: a GPU-based framework for simulating cortically-organized networks [R]. MIT-CSAIL-TR-2010-03, 2010.

[9] Sabarad J, Kestur S, Park MS, et al. A reconfigurable accelerator for neuromorphic object recognition [C] // The 17th Asia and South Pacific Design Automation Conference, 2012: 813-818.

[10] Kung HT. Why systolic architectures? [J]. Computer, 1982, 15(1): 37-46.

[11] Jouppi NP, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit [C] // 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), 2017: 1-12.

[12] Wang C, Gong L, Yu Q, et al. DLAU: a scalable deep learning accelerator unit on FPGA [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2016, 36(3): 513-517.

[13] 刘勤让, 刘崇阳, 周俊, 等. 基于线性脉动阵列的卷积神经网络计算优化与性能分析 [J]. 网络与信息安全学报, 2018, 4(12): 20-28.

[14] Sanchez J, Soltani N, Chamarthi R, et al. A novel 1D-convolution accelerator for low-power real-time CNN processing on the edge [C] // 2018 IEEE High Performance Extreme Computing Conference (HPEC), 2018, doi: 10.1109/HPEC.2018.8547530.

[15] Yang ZJ, Wang L, Ding D, et al. Systolic array based accelerator and algorithm mapping for deep learning algorithms [C] // IFIP International Conference on Network and Parallel Computing, 2018: 153-158.

[16] Du ZD, Fasthuber R, Chen TS, et al. ShiDianNao: shifting vision processing closer to the sensor [J]. ACM SIGARCH Computer Architecture News, 2015, 43(3): 92-104.

[17] Chen YH, Emer J, Sze V. Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks [J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 367-379.

[18] Hubel DH, Wiesel TN. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex [J]. The Journal of Physiology, 1962, 160: 106-154.

[19] Gabor D. Theory of communication [J]. Journal of the Institution of Electrical Engineers-Part I: Genera, 1947, 94(73): 58.

[20] Jones JP, Palmer LA. An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex [J]. Journal of Neurophysiology, 1987, 58: 1233-1258.

[21] De Valois R, Albrecht DG, Thorell LG. Spatial frequency selectivity of cells in macaque visual cortex [J]. Vision Research, 1982, 22(5): 545-559.

[22] De Valois RL, Yund EW, Hepler N. The orientation and direction selectivity of cells in macaque visual cortex [J]. Vision Research, 1982, 22(5): 531-544.

[23] Poggio T, Bizzi E. Generalization invision and motor control [J]. Nature, 2004, 431: 768-774.

[24] Learned-Miller E, Huang GB, RoyChowdhury A, et al. Labeled faces in the wild: a survey [M] // Advances in Face Detection and Facial Image Analysis, 2016: 189-248.

[25] Samajdar A, Zhu YZ, Whatmough P, et al. SCALE-Sim: systolic CNN accelerator [C] // International Symposium on Performance Analysis of Systems and Software, 2019.

[26] Rosenfeld P, Cooper-Balis E, Jacob B. Dramsim2: a cycle accurate memory system simulator [J]. IEEE Computer Architecture Letters, 2011, 10(1): 16-19.