计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20190111
56(6): 1170-1181, 2019

多层神经网络算法的计算特征建模方法

方荣强[1] 王 晶[1,4] 姚治成[2] 刘 畅[1] 张伟功[3,4]
[1](首都师范大学信息工程学院 北京 100048)
[2](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)
[3](高可靠嵌入式系统技术北京市工程研究中心(首都师范大学) 北京 100048)
[4](北京成像理论与技术高精尖创新中心(首都师范大学) 北京 100048)
(zwg771@cnu.edu.cn)

Modeling Computational Feature of Multi-Layer Neural Network

Fang Rongqiang[1], Wang Jing[1,4], Yao Zhicheng[2], Liu Chang[1], and Zhang Weigong[3,4]
[1](*College of Information Engineering*, *Capital Normal University*, *Beijing* 100048)
[2](*State Key Laboratory of Computer Architecture* (*Institute of Computing Technology*, *Chinese Academy of Sciences*), *Beijing* 100190)
[3](*Beijing Engineering Research Center of High Reliable Embedded System* (*Capital Normal University*), *Beijing* 100048)
[4](*Beijing Advanced Innovation Center for Imaging Theory and Technology* (*Capital Normal University*), *Beijing* 100048)

Abstract Deep neural networks (DNNs) have become increasingly popular as machine learning technique in applications, due to their ability to achieve high accuracy for tasks such as speech/image recognition. However, with the rapid growth on the scale of data and precision of recognition, the topology of neural network is becoming more and more complicated. Thus, how to design the energy-efficiency and programmability, neural or deep learning accelerator plays an essential role in next generation computer. In this paper, we propose a layer granularity analysis method, which could extract computation operations and memory requirement features through general expression and basic operation attributions. We also propose a max value replacement schedule strategy, which schedules the computation hardware resource based on the network feature we extract. Evaluation results show our method can increase computational efficiency and lead to a higher resource utilization.

Key words neural network; features extraction; hardware accelerator; computer architecture; resource scheduling

摘 要 随着深度学习算法在语音和图像等领域中的成功运用,能够有效提取目标特征并做出最优决策的神经网络再次得到了广泛的关注.然而随着数据量的增加和识别精度需求的提升,神经网络模型的复杂度不断提高,因此采用面向特定领域的专用硬件加速器是高效运行神经网络的有效途径.然而如何根据

收稿日期:2019-03-01;**修回日期**:2019-04-23
基金项目:国家自然科学基金项目(61772350);共有信息系统装备预先研究项目(公开)(JZX2017-0988/Y300);北京市科技新星计划项目(Z181100006218093);体系结构国家重点实验室开放课题(CARCH201607);北京未来芯片技术高精尖创新中心科研基金资助项目(KYJJ2018008);北京市高水平教师队伍建设计划(CIT&TCD201704082);科技创新服务能力建设-基本科研业务费(科研类)(19530050173,02518530500)
This work was supported by the National Natural Science Foundation of China (61772350), the Common Information System Equipment Pre-research Funds (Open Project) (JZX2017-0988/Y300), Beijing Nova Program (Z181100006218093), the Open Project of State Key Laboratory of Computer Architecture (CARCH201607), the Research Fund from Beijing Innovation Center for Future Chips (KYJJ2018008), the Construction Plan of Beijing High-level Teacher Team (CIT&TCD201704082), and the Capacity Building for Sci-Tech Innovation Fundamental Scientific Research Funds (19530050173, 025185305000).
通信作者:王晶(jwang@cnu.edu.cn)

网络规模设计高能效的加速器,以及基于有限硬件资源如何提高网络性能并最大化资源利用率是当今体系结构领域研究的重要问题.为此,提出基于计算特征的神经网络分析和优化方法,基于"层"的粒度解析典型神经网络模型并提取模型通用表达,根据通用表达式和基本操作属性提取模型运算量和存储空间需求等特征.提出了基于最大值更替的运行调度算法,利用所提取的特征分析结果对神经网络在特定硬件资源下的运行调度方案进行优化.实验结果显示:所提方法能够有效分析对比网络特征,并指导所设计调度算法实现性能和系统资源利用率的提升.

关键词 神经网络;特征提取;硬件加速器;计算机体系结构;资源调度

中图法分类号 TP391

随着神经网络越来越广泛地应用于语音识别、计算机视觉、智能机器人、故障检测、市场分析、决策优化等领域[1-2],人们对网络精度的要求不断提高,网络层数越来越多,计算复杂度越来越高,如 2014 年 GoogLeNet[3]网络已经达到 22 层.日益增加的计算复杂度使得训练和推理的开销问题逐步凸显出来,当前 GPU 和 FPGA 等专用硬件加速芯片已经成为神经网络运行的重要平台.而随着人工智能技术的进步,移动设备、嵌入式设备等在计算、体积、功耗等方面受限的设备也需要应用深度学习技术.由于设备资源的约束,导致现有复杂的深度神经网络无法进行高效的计算.上述场景为计算机体系结构提出了新的挑战:如何在保持现有神经网络精度不变的情况下,使得网络模型能在资源受限的设备上高效运行,并最大化系统资源利用率.

针对上述问题可以从算法角度减小网络计算量,也可以从硬件角度优化资源利用率.现有研究从算法角度提出了大量优化方案[4]:针对神经网络运算矩阵和矩阵相乘的运算方式,可以利用奇异值分解来压缩神经网络计算量[5];针对权重矩阵往往比较稀疏的特性,可以利用矩阵稀疏编码的方式压缩神经网络[6];通过改变卷积运算算法可以加速网络执行[7-12];权值重载方式也能够有效减少片上存储开销[13].算法的优化可以实现对深度神经网络的压缩,从而削减网络的计算量和存储需求,提高网络执行的效率,但这些方法同时也都是预先静态地对算法进行优化,无论裁剪到什么程度的网络,最终都需要在实际的硬件上运行,因此从体系结构角度,在有限硬件资源上优化神经网络的运行效率是当前研究的重要问题.

研究人员从体系结构角度入手,提出多种提升神经网络执行速度的方案:利用神经网络中数据重用的特点可以优化卷积神经网络[14];文献[15]将网络权值矩阵和输入矩阵进行矢量化操作,能够大大减小运算量,提升计算效率.将神经网络从通用处理器 CPU 的执行平台,迁移到具有高并行度的 GPU 和 FPGA 平台也是有效提高运行效率的手段.利用 OpenCL 将卷积计算转化为并行度更高的矩阵乘法运算能够进一步优化 GPU 平台上卷积运算效率.FPGA 平台上利用可重构和流水化并行执行的卷积运算过程能够实现对执行速度和资源利用率的同步提升,基于 FPGA 的可重构特性能够化简乘法运算,提高系统能效性[16-19].

然而,无论哪一种神经网络加速器的设计,都需要了解算法的特征,根据不同算法的计算规模有针对性地分配计算和存储资源,才能在提高程序运行效率的同时最大化系统资源利用率.此外,对于硬件加速器设计的验证,如果运行真实的大规模神经网络将导致验证的周期和成本都大幅增加.而抽取典型计算片段不但能够保证功能验证覆盖率,还能有效降低验证成本.要达到这 2 方面目的,都需要对算法的特征进行分析,找出模型中频繁出现的层,本文称为算子(operator),了解算子的计算和访存特点,从而找到加速优化的切入方向.

因此本文提出基于基本运算的神经网络特征提取优化方法,主要贡献包括 3 个方面:

1) 针对典型神经网络进行分析,找出其中核心算子,对每个算子分析内部包含的基本运算、运算的数量和内存占用量随输入变化的关系.

2) 在算子的粒度,根据网络的描述解析网络基本结构,获得包含算子和算子顺序的模型通用表达式,并给出图形化描述.根据分析获得的算子内部特征和模型通用表达式,计算网络模型的乘加运算量、存储占用量等典型特征值.

3) 基于所获得的运算量等网络特征,结合硬件资源数量,提出基于最大值的网络运行调度优化方案,提高了神经网络的执行效率,同时最大化硬件资源的利用率.

1 基于运算操作的神经网络特征提取方法

神经网络通常由不同的层,如卷积层、池化层、全连层等以特定顺序组合而成,其中乘法和加法操作(operation)是基本的运算.为了分析神经网络运行对硬件资源的需求,首先从"层"的粒度分析网络结构,然后基于每层的运算特点分析该层所需要的乘法和加法等基本操作的次数以及所需占用的存储空间.

1.1 神经网络模型解析

深度神经网络虽然可以通过改变层级结构、神经元个数以及神经元之间的连接衍生出不同的网络模型,但网络中所涉及的核心算子种类并不多,通常包括卷积层、池化层、激活函数和全连接层等.尽管输入和参数不同,但每个算子的运算方式是确定的,因此可以通过分析得到计算每个算子的基本操作和基本资源需求公式.

模型解析器可以分析神经网络模型包含的算子以及算子的执行顺序.按照算子的种类和执行顺序,建立一个仅包含算子种类和其执行顺序但不包含操作次数的图形化通用表达式,把神经网络中各种算子看作节点,把其所需要的输入和产生的输出作为连接节点的有向边,这样便产生了一个有向无环图(DAG),这个图结构作为后续特征提取模块的输入.

模型解析器是本文所提出的网络特征提取方法的第 1 步,特征提取方法结构如图 1 所示.模型解析器基于输入的神经网络模型和其对应的参数,分析模型所包含的层数和每层的功能,将网络转化成对应的通用表达式,然后通过 TensorFlow 内嵌的 TensorBoard 可视化组件获得其图形化的显示,从而获得模型结构和算子种类、算子数量和算子顺序.

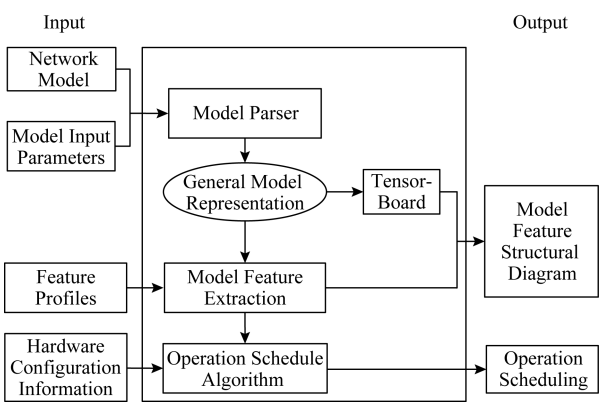

Fig. 1 Design of network feature extraction

图 1 网络特征提取整体设计

在模型解析之后,模型特征提取模块加载所获得的通用表达式以及包含基础算子的特征描述文件,获得算子内包含的乘加等基本运算操作的类型、次数、存储占用情况等特征.所获得的计算和访存特征作为运行调度模块的输入,结合系统硬件资源信息,例如 FPGA 芯片能够支持的加法器和乘法器个数,运行调度模块采用最大值更替算法计算出在给定硬件资源下最大化资源利用率的网络运行调度方案.

1.2 基于算子的模型特征提取

算子的计算量等特征是可以通过加载提前设定的配置文件(profile)来解析统计出加法和乘法等基本操作计算公式.模型特征提取模块负责根据计算公式对输入模型进行整体的特征提取与计算.模块的输入包括 2 部分:配置文件和模型通用表达式,其中模型通用表达式是模型解析器传递的包含算子种类及执行顺序的分析结果,也就是没有神经元具体执行运算和内存占用信息的图结构.算子特征描述文件定义了算法模型中具体算子的特征.

当模块得到通用模型描述文件和配置信息的输入后,遍历代表网络模型的图结构,进行模型特征信息的构建.对每个算子进行单独的特征提取,通过计算参数配置文件的算子特征公式,可以获得加法和乘法基本操作次数等神经网络的特征,并对所有特征进行统计与存储.

算子特征提取流程如图 2 所示:首先判断算子对应的类型在配置文件中是否有定义,如果有,则从模型算子节点中加载,按照配置文件中该算子的公式定义来计算出对应的乘法和加法等基本操作的次数,而后输出该算子的特征;如果配置文件中没有定义,则进行第 2 步判断,检测系统内部是否有内置特征解析方法,若有,则调用提取特征的函数对该算子进行解析,若无,则输出空特征.

特征提取模块通过配置文件管理器来管理所有的配置文件,管理器包含 2 方面的管理功能:算法模型初始化参数的管理和算法基础算子的特征管理.算法模型初始化参数的管理将每个算子对应的信息做初始化赋值,算法基础算子的特征管理记录是对每个算子预置基本操作的特定计算公式.在以 Conv2D 算子为例的算子配置文件内容中,分为 Var 和 Feature 两大部分.当解析该算子的特征时,首先利用字段 Var 中指定的方法进行变量初始化,即按顺序从 Var 中读取指定的变量提取方法,并将执行方法得到的结果存放在指定的变量中.对于算法模型,算子的计算量往往和输入数据大小、维度等有关,所以在计算"算子的计算量"之前,需要把算子

所涉及的输入输出数据样式确定下来,例如模型输入图片的大小、批大小(*batch_size*)等参数.神经网络算法模型由不同功能的算子通过不同的连接方式组成,算子不同则计算类型、计算量、数据处理大小等都不一样,所以对于"特征提取模块"而言,它需要知道用于构建模型的每个算子参数、计算特征等,例如二维卷积算法需要知道卷积核的大小、移动步长、数据维度大小等特征后才能根据具体的算法得出该算子在模型中的特征.为了方便特征提取模块获取每个算子的属性,参数配置文件管理器基于配置文件的算子管理方式,当用户需要修改或者添加算子特征时只需修改对应的配置文件.

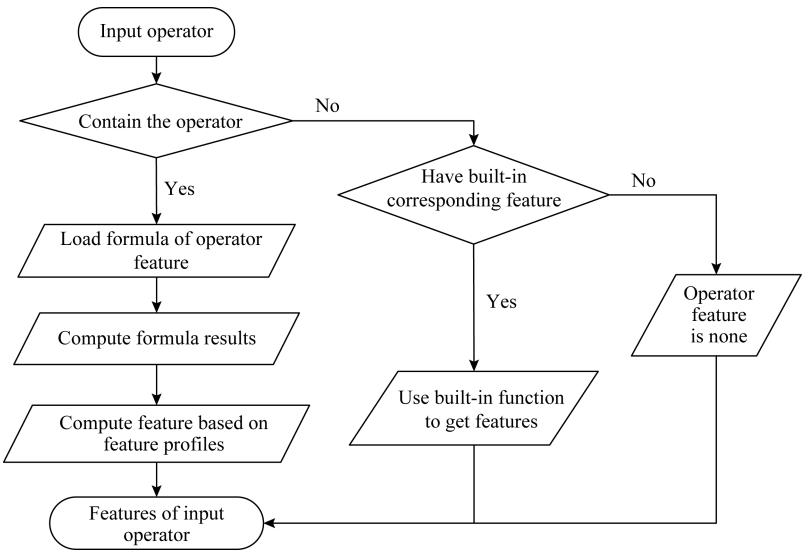

Fig. 2 Operator-based feature extraction process

图 2 基于算子的特征提取流程

1.3 算子的特征计算方法

本文选取了 10 种典型神经网络算子,根据算子的输入参数列表,如表 1 所示,以加法为例给出了每个算子计算资源需求的公式,如表 2 所示.

1) Conv2D 算子.它是 2 维卷积运算,该算子实际上是用滤波器矩阵在输入图像矩阵上沿水平或竖直方向上按步长数值滑动的同时做加权叠加的操作.卷积核单次计算加法次数为$((f_height \times f_width-1)\times in_channels)\times out_channels$,所需加法次数为$(f_height \times f_width \times in_channel-1)\times[(in_height-f_height)/strides+1]\times[(in_width-f_width)/strides+1]$,最后乘以图片的张数和输出结果的通道数就可以得到卷积算子包含的加法次数.而二维卷积所包含的乘法次数,首先计算单个卷积核单次计算的乘法次数$(f_height \times f_width \times in_channels)$,再乘以图片张数和移动步数即可获得其包含的乘法次数.

2) Avg_Pool.它为平均池化算子,该算子是通过定义一个方形窗口滤波器矩阵,让该窗口以不重叠的步长 *strides* 在输入图像上沿水平或竖直方向上运算的同时记录下该窗口矩阵范围内所有输入图像矩阵元素的平均数作为输出图像矩阵的像素值,而输出结果的图像矩阵大小取决于定义窗口矩阵的大小.该算子可以将输入的图像进行有效压缩处理以节省资源消耗.关于平均池化的加法计算,步骤 1 是计算出它的移动步数,步骤 2 则是计算出它的每步所包含的数值的个数.首先,计算窗口大小 ***ksize*** 平均值需要 $\Pi(\boldsymbol{ksize})-1$ 次加法;接着对输出矩阵 ***output*** 进行求维度运算,可以获得一个关于输出结果的维度矩阵,即 $\boldsymbol{shape}(\boldsymbol{output})$;而后对该维度矩阵求连乘积,即可获得输出矩阵中所包含的元素个数;最后将它们乘在一起就可以得到平均池化算子所包含的加法次数.而该算子所包含的乘法次数与输出矩阵中的元素个数相等,因为对于每步池化都需要做一次乘法,又因为该池化核所走的步数等于输出矩阵元素个数,因此其包含的乘法计算的次数为 $\Pi(\boldsymbol{shape}(\boldsymbol{output}))$.

3) Max_Pool.它为最大池化算子,与平均池化算子相同,都包含 ***ksize*** 参数和同样的生成规则,与之不同的仅在于它不取窗口内的平均值而取其最大值作为输出结果.其加法计算即为在最大池化核的范围内进行数据比较次数,实为减法操作次数,

Table 1 Typical Operation Input Parameters

表 1 典型算子输入参数

Operator	Involved Partial Parameters and Method	Corresponding Explanation
Conv2D	***Input***[*batch_size*, *in_height*, *in_width*, *in_channels*]	The Number/Height/Width and Channels of the Input
	Filter[*f_height*, *f_width*, *in_channels*, *out_channels*]	The Height/Width of the Filter, Input/Output Channels
	strides	The Stride of the Filter
	padding	Padding Pixels
Avg_Pool	***ksize***	Pooling Window Size
Max_Pool	***output***	Output Matrix
Bias_Add	***bias***	Offset Value Vector
MatMul	x	The Number of Rows in Multiplicand Matrix
	y	The Number of Columns in Multiplier Matrix
	m	The Number of Rows in Multiplier and the Columns in Multiplicand Matrix
LRN	$n/2$	Normalization Radius
	k	Offset Value
	a	Input Featuremap
	b	Output Featuremap
	α	Constant Hyper-parameters
	β	Constant Hyper-parameters
	$dim_1, dim_2, dim_3, dim_4$	The Values of Each Dimension in a Four-dimensional Tensor
Softmax	***logits***	Non-empty Two-dim-Tensor
Sigmoid tanh ReLU	*Output_index*	The Index of Hidden Layer Output Vector
Global Variable Built-in Method	*batch_size*	The Number of Samples Propagated Through the Network
	num_classes	The Total Classes Number
	shape()	The Dimension of a Matrix
	Π()	Matrix Multiplication

Table 2 Analysis of Add Operation Feature of Typical Operators

表 2 典型算子加法特征分析

Operator	Formula of Calculating Number of Addition Operation
Conv2D	$((f\_height \times f\_width - 1) \times in\_channels) \times batch\_size \times ((in\_height - f\_height)/strides + 1) \times ((in\_width - f\_width)/strides + 1) \times out\_channels$
Avg_Pool	$\Pi(\boldsymbol{shape}(\boldsymbol{output}) \times (\Pi(\boldsymbol{ksize}) - 1))$
Max_Pool	$\Pi(\boldsymbol{shape}(\boldsymbol{output}) \times (\Pi(\boldsymbol{ksize}) - 1))$
Softmax	$batch\_size \times num\_classes \times (num\_classes - 1)$
Bias_Add	$\boldsymbol{shape}(\boldsymbol{bias})$
MatMul	$x \times y \times (m - 1)$
LRN	$(n+3) \times dim_1 \times dim_2 \times dim_3 \times dim_4$
Sigmoid	1
tanh	2
ReLU	1

故需要比较 $\Pi(\boldsymbol{shape}(\boldsymbol{output}) \times (\Pi(\boldsymbol{ksize}) - 1))$ 次.由于该算子是将池化核内所有数据进行比较,因此不需要计算乘法.

4) Bias_Add.它是对 2 维卷积层和全连接层添加偏置值的标准化函数,该算子所包含的加法次数即为矩阵的维度,由于是直接相加,因此乘法计算次数为零.

5) MatMul.它为矩阵乘法算子,是全连层主要的运算.其输入为 $\boldsymbol{A}_{x \times m}$ 和 $\boldsymbol{B}_{m \times y}$ 的矩阵,这 2 个矩阵经 MatMul 算子运算后得到 $\boldsymbol{C}_{x \times y}$ 的矩阵,因此产生的结果中有 $x \times y$ 个新数值,结果元素可计算为

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{im}b_{mj}.$$

每个元素的计算需要 $m-1$ 个已有数值相加,因此其加法次数为 $x \times y \times (m-1)$ 次,而乘法次数也相应为 $x \times y \times m$ 次.

6) LRN.它为局部相应归一化算子,用来抑制过拟合现象,并加快收敛速度的算子[20].可计算为

$$b_{x,y}^{i} = a_{x,y}^{i} \Big/ \left(k + \alpha \sum_{j=\max(0, i-n/2)}^{\min(N-1, i+n/2)} (a_{x,y}^{j})^2\right)^{\beta},$$

其中,加法共有 $n+3$ 次.$b_{x,y}^{i}$ 再乘以矩阵的 4 个维度

参数 dim_1,dim_2,dim_3 和 dim_4 可以计算出整个算子包含的所有加法次数.在分母处涉及 $n+\beta$ 次乘法运算,因此在同一维内乘法共运算 $n+\beta+1$ 次,同样最后乘以矩阵的 4 个维度参数便得出 LRN 算子所包含的乘法次数.

7) Softmax.它为神经元激活函数算子,该算子将多分类的输出数值转化为相对概率,经常被用于分类器后的输出单元做处理,依据 Softmax 函数的计算公式:$Softmax[i,j]=\exp(logits[i,j])/sum\_j(\exp(logits[i,j]))$,可知其分母由输出层类别个数 $num\_classes$ 相加而成,因此加法计算次数为 $num\_classes-1$.又因为训练一个样本有 $num\_classes$ 个结果需要激活函数 Softmax 函数进行计算,所以加法次数为 $num\_classes\times(num\_classes-1)$,最后乘以每次训练投放的样本个数 $batch\_size$ 可以获得一次完整训练所进行的加法次数.对于分母累加中的每一项,分别有 $logits[i,j]-1$ 次乘法,共有 $num\_classes$ 项相加,故有 $(num\_classes)\times(logits[i,j]-1)$ 次乘法.又因为需要对样本中每个数据做 Softmax 激活,而每次激活时需要做一次除法.除法在机器运算实为右移操作,而乘法为左移操作,在功耗上的消耗相当,因此可看做是做一次乘法,故需要做 $batch\_size\times num\_classes+1$ 次.而 Softmax 包含的乘法次数为 $(batch\_size\times num\_classes)\times num\_classes\times[(logits[i,j]-1)+1]$.

8) Sigmoid.它也是神经元激活函数算子,其计算公式为 $Sigmoid(Output\_index)=1/(1+e^{-Output\_index})$,其加法次数为 1 次,乘法次数在分母运算了 $Output\_index$ 次,共计 $Output\_index+1$ 次.

9) tanh.它是常用的激活函数算子,也称为双曲正切函数.tanh 在特征相差明显时效果较好,在循环过程中会不断扩大特征效果.其计算公式为

$$\tanh(Output\_index)=(e^{Output\_index}-e^{-Output\_index})/(e^{Output\_index}+e^{-Output\_index}),$$

其加法次数为 2 次,乘法次数由于 e^{Output_index} 计算一遍后可以多次重用,故其乘法次数为 $Output\_index+1$ 次.

10) ReLU.它为激活函数算子,由于函数解析式简单,因此能获得更快的收敛速度.其计算公式为 $ReLU(Output\_index)=\max\{0,Output\_index\}$,该算子不包含乘法,仅有一次比较,故加法次数为 1.

本文针对以 VGG 和 AlexNet 为代表的主流神经网络,在层粒度进行了解析.所提出的静态分析方案不增加动态网络运行延迟和能耗开销.同时,本文所提出的方案给出了网络层粒度的任务分布,可以结合对网络数据流的分析,建立动态网络运行分析模型,为综合优化性能和能耗提供支持.

神经网络架构通常计算密度高、存储需求大,为了适应低功耗的移动等存储环境,裁剪和量化技术被广泛应用.量化技术通常对所有输入数据采用统一的方法,如散列和定点化进行处理,因此对于增加量化技术的神经网络,量化的计算量与输入数据规模成比例.而对于应用剪枝技术的神经网络,本方法可以计算出网络的计算量和存储量上限,结合裁剪技术的具体实现方案,通过对权值矩阵的分析统计裁剪情况,得出实际的网络计算特征.

2 基于神经网络特征的运行优化算法

支持深度神经网络运行的 GPU 和 FPGA 平台能够同时并行执行大量乘加运算,甚至可以支持多个算子的并行执行,流水化地执行多个输入可以有效提高硬件资源利用率[21].以 FPGA 为例,FPGA 芯片可以划分为多个 DSP,乘法器和加法器都由若干个 DSP 组成,此时算子所需硬件计算单元就可以转化为 DSP 数量,当算子在 FPGA 芯片上执行无法占满整个芯片时,可以根据所需 DSP 数量计算组合方案,在同一时间内可以让多个算子同时处理.

专用硬件加速器和 FPGA 通常都只能包含固定数量的加法器和乘法器,具体的数值则由它的结构和硬件工艺所确定.然而,硬件支持的乘加运算器数量和神经网络不同层次所需要的乘加操作次数难以完美匹配,也就是可以放入同一块芯片或加速单元中运行的算子有多重组合.如果没有对网络结构和拓扑进行全局性的分析,系统只能顺序或者随机地选取算子执行,不但无法保证资源利用率,往往还会影响加速性能,加大硬件开销.因此针对网络内部不同层对硬件资源的需求,结合实际计算资源的情况进行优化调度是必要的.

本文利用所提取的特征分析结果对神经网络在特定硬件资源下的运行调度方案进行优化,我们将不同的算子转化成乘法和加法操作次数,结合系统资源的划分,如 FPGA 中 DSP 的资源数量,提出最大值更替调度算法.本文以卷积层为例来进行调度的规划,算法可以推广到神经网络包含的各种算子的硬件资源调度.

神经网络的基本乘法和加法操作还可以归一化量化为基本单元数,例如量化为逻辑门的数量或

DSP 的数量,根据不同的硬件描述和实现工艺,不同的层可以用乘法和加法数量乘以乘法器和加法器的门的数量需求得到本层以基本单元数为单位的尺寸.神经网络的调度算法可以建模为不同大小的层,尽可能放入有限大小的硬件芯片上,这类似于背包问题.但与背包问题有 2 方面区别:1)背包问题只选择部分物品装满一个背包即可,而神经网络的所有操作都需要执行,如果总的操作数量大于当前硬件能够支持的计算量,则需要多次执行,也就相当于多次背包;2)背包问题只给出选择物品的数量,而调度算法需要获得调度方案和具体顺序以指导实际运行,因此我们改进了背包算法,提出基于最大值更替的调度算法.算法的描述如算法 1 所示.

算法 1. 基于最大值更替的调度算法.

输入:硬件容量 C、卷积层种数 N、第 i 种卷积层的总个数 k_i、第 i 种卷积层综合乘法开销 $W[i]$;

输出:使用的硬件总数、每个硬件分别承载的卷积层种类和其数量.

```
while (仍有卷积层未删除) {
  if (C_i − max ≥ 0) {
    C_i = C_i − max;
    k_i = k_i − 1;
    记录当前的 b_i;
    if (k_i == 0) {
      删除该卷积层;
      最大值设为当前数据中的最大值;
      if (删除的是最小值) {
        最小值更新为当前数据中的最小值;
    } }
  }
  else {
    if (max == min) {
      输出所有记录的 b_i;
      重置记录 b_i 的数组;
      算法计数器+1;
      C_i = C;
      最大值设为当前数据中的最大值;
    }
    else
      max 值更改为当前数据中的次大值;
  }
}
```

算法 1 输入分为 2 部分:1)网络模型的各卷积层所包含的乘法和加法次数列表数据,该部分数据来自特征提取模块所输出的结果;2)当前所应用的硬件资源包含的基本单元数.设网络模型中卷积层的大小有 N 种,第 $i(0\leqslant i\leqslant N)$ 种卷积层执行所需的基本单元数 $W[i]$ 由上一步的特征提取给出.在算法运行过程中记录第 i 种的 3 方面信息:该种卷积层已被规划了的次数 a_i、该种卷积层是输入当中的序号 b_i 以及该种卷积层当前的个数 k_i;同时将硬件平台所包含的基本单元数量记为 C.在规划过程中,首先判断当前基本单元数 C_i 是否大于当前最大卷积操作的资源需求 max,如果大于那么当前卷积可以执行,则 $C_i=C_i-max$ 且最大值的 $k_i=k_i-1$;此时再判断 k_i 是否为 0,如果为 0 那么从当前数据中移除该种卷积,最大值 max 重新设为当前数据中的最大值,如果此时移除的恰好是当前最小卷积操作资源需求 min,那么将该卷积移除后更新 min 值为当前数据中的最小值.如果当前基本单元数量 C_i 比卷积操作的最大值 max 小,那么就判断当前的最大值是否跟当前数据中的最小值重合,如果重合那么将当前基本单元数 C_i 重置为 C,最大值 max 重新设为当前数据中的最大值;而如果最大值和最小值还未重合,那么最大值设为当前数据中的次大值,而后再次进行 C_i 与 max 的比较.依次进行上述操作,直到所有卷积都完成规划.

3 实验评估

本文选取了 6 个被广泛应用于各个领域中的神经网络模型.

1) 图像识别类模型.通过多层卷积算子提取出图像的特征,根据提取出的特征对图像进行处理、分析和理解,以识别各种不同模式的目标和对象.常见的包括:AlexNet 模型、VGG 模型和 Inception 模型.

2) 音频识别类模型.主要作用是区分人声、动物声音或者音乐演奏等声音,典型的神经网络模型是 VGGish.

3) 视频识别类模型 SSD.SSD 是基于前向传播的神经网络、无全连接层、参数少、运行速度快、识别精度高.

4) 文本类模型 Attention.Attention 模型从网络中某些状态集合中选取与给定状态较为相似的状态,然后训练一个模型来对输入进行选择性的学习并且在模型输出时将输出序列与之进行关联.

首先,通过分析各个模型所包含的算子类型和数量,如表 3 所示;然后从模型复杂度和资源需求等方面对算子的特征进行分析;最后,采用所提出的最大值更替算法给出了调度方案,并同顺序调度方案对比资源利用率的提升效果.本文用 TensorFlow 中的图形化组件 TensorBoard,按前面的设计方法,可以提取出神经网络的图形化表示,如 AlexNet 的图形化表示,如图 3 所示.通过分析可以清楚地得到 AlexNet 网络的结构图,由 8 层组成,共有 5 个卷积层和 3 个全连接层,分别是输入层→卷积→池化→卷积→池化→卷积→卷积→卷积→池化→全连接→dropout→全连接→dropout→全连接,在每一个卷积层后都经过了降采样(pooling 处理),同时该模型中加入了 dropout 操作.我们将 6 个神经网络模型均按照上述方式进行建模,进而分析出它们的特征.

Table 3 Six Classical Neural Network Models

表 3 6 个经典神经网络模型

Application	NN Model	Operators
Image Recognition	AlexNet[20]	Conv2D, Max_Pool, MatMul, Bias_Add, LRN, ReLU
	Inception	Conv2D, Max_Pool, Avg_Pool, MatMul, Bias_Add, Softmax
	VGG19[22]	Conv2D, Max_Pool, MatMul, Bias_Add, Softmax
Audio Recognition	VGGish	Conv2D, Max_Pool, MatMul, Bias_Add, Softmax
Video Recognition	SSD[23]	Conv2D, Softmax, MatMul, Bias_Add, Max_Pool
Text Recognition	Attention[24]	MatMul, Bias_Add, tanh, Softmax, Sigmoid, Conv2D

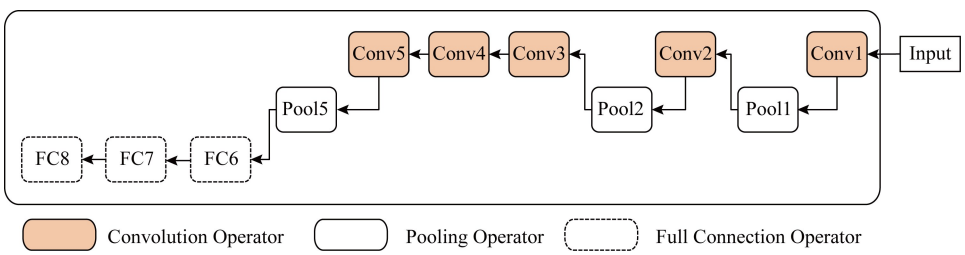

Fig. 3 The visualization model of AlexNet

图 3 AlexNet 可视化模型

3.1 经网络特征分析

加速器的选取主要从 3 个角度进行分析:从模型的总体角度统计运算量,给出针对模型选用加速器的建议;从算子角度,通过对各个算子的运算量比较,得出针对该算子的加速器设计建议;考虑时间因素判断模型各阶段应使用的加速器结构.针对这 3 个需求,我们主要从模型复杂程度、卷积算子占算子总数比例、各模型占用内存量情况以及运算量 4 个角度对网络的特征进行分类分析.

1) 模型复杂度分析

由模型复杂程度的对比,我们统计了所有算子的总数量.由于模型结构这一概念过于抽象,很难直接对模型结构进行讨论,因此对模型的复杂度采用操作总数作为比较标准.

从图 4 对 6 种网络的分析结果中可以看出,Attention 网络的算子个数最多,达到了 5 905 个,整个网络模型也最复杂;而 VGGish 的算子个数最少,仅有 178 个,其结构也就最简单.因此在模型结构复杂度角度,视频识别模型 SSD、文本识别类模型的操作数量非常庞大,因此可以在不影响模型功能的前提下从精简算子数量的角度对模型进行优化.

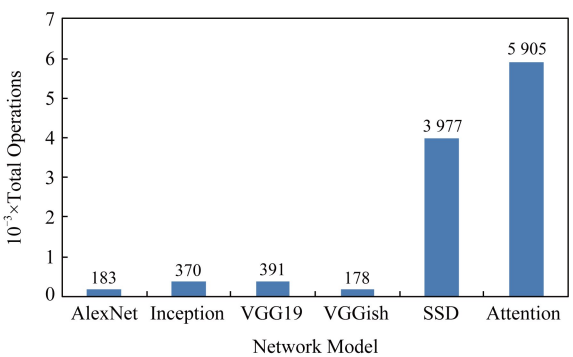

Fig. 4 Model complexity

图 4 模型复杂度

2) 卷积算子占比分析

卷积操作是神经网络众多算子中计算复杂度和能耗开销最高的算子,卷积的优化对提高网络性能有重要的影响.基于本文所提出的方法,对 6 种网络的卷积操作比例进行了统计,如图 5 所示:

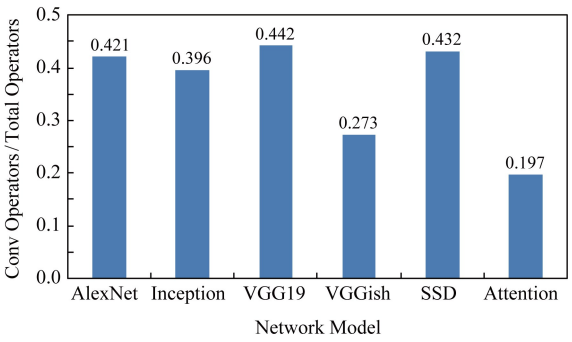

Fig. 5 Convolution operator proportion

图 5 卷积算子占比

由图 5 可知,图像、视频识别类模型、卷积算子占比明显多于文本类模型和音频模型中的卷积算子占比.这是由于卷积在网络中担当着对输入进行特征提取的工作,卷积算子越多,对输入提取出的特征就越详细.音频分析网络由于没有图片特征分析那么高的要求,故比例相对较低.根据这些分析,在加速器设计过程中对卷积占比高的神经网络进行加速,可以通过减少卷积操作方式或个数等技术达到更好的效果.

3) 内存需求分析

神经网络的执行过程中要存储大量中间结果,因此对内存需求量往往较大.我们使用各个模型总内存占用量除以运算次数得到如图 6 所示的内存占用量.

针对统计出的模型中平均每次运算占用的内存量设定阈值 0.1B/次,超过阈值的模型体现为访存密集型;小于阈值的模型则体现计算密集型.在这些网络中,AlexNet 的平均每次运算占用的内存量远超过其他的神经网络,它在这些网络模型中访存比例更高,因此调度运算顺序,减小运算中间结果的存储等优化方法能够有效提高这类网络的性能.

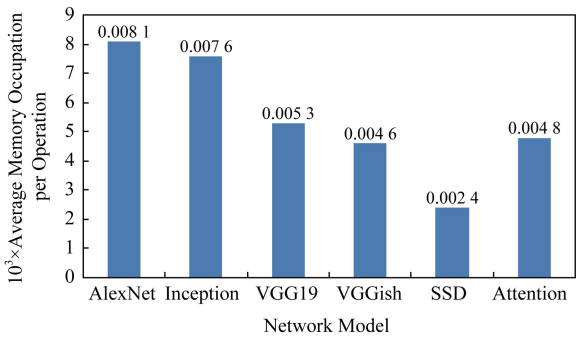

Fig. 6 Operational memory usage

图 6 运算占用内存使用量

4) 模型特征分析

基于对模型的操作数、卷积占比、平均内存量、加法次数、乘法次数 5 个方面的统计信息,通过雷达图对比不同模型的特征,如图 7 所示.结果显示每个模型都有自己的特点:SSD 和 Attention 计算占用内存方面比较小,其他资源消耗比较平衡.而

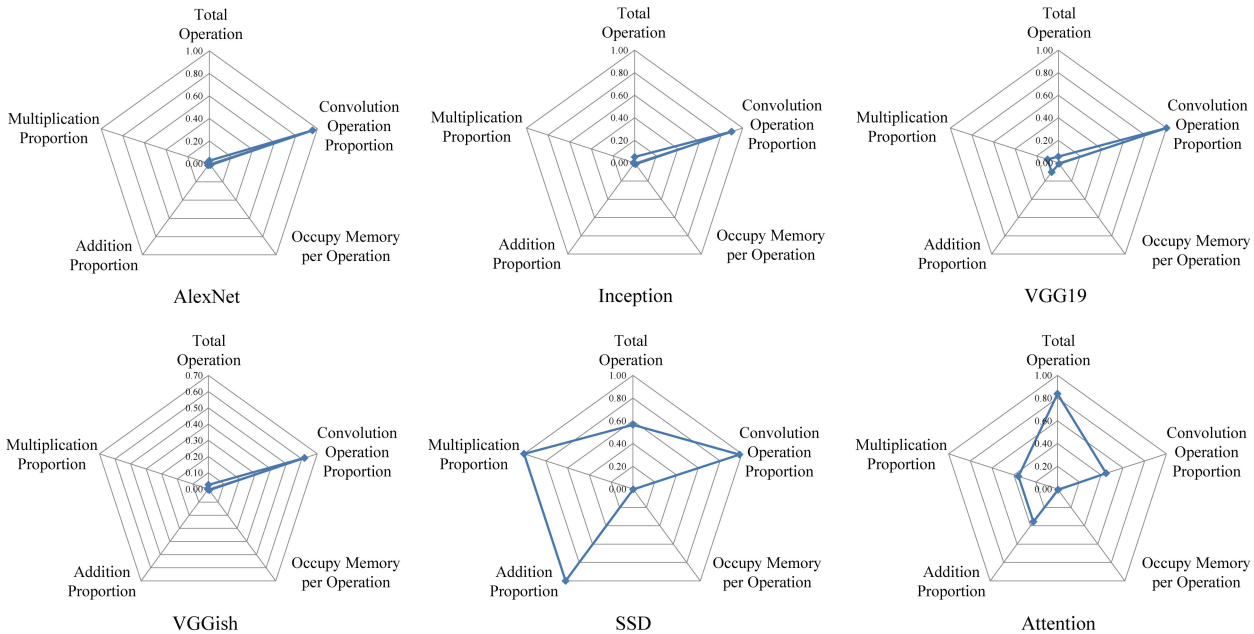

Fig. 7 Radar charts of hardware resource consumption distribution for neural network models

图 7 各神经网络模型硬件资源消耗分布雷达图

AlexNet,Inception,VGG19 和 VGGish 这 4 个模型都是卷积占比突出,因此本文针对这 4 个模型归一比较,如图 8 所示.模型在不同的维度展现出不同的特征,通过雷达图结果的分析,能够为未来加速器设计技术从加速卷积操作、减少内存占用等方面的选择提供指导.

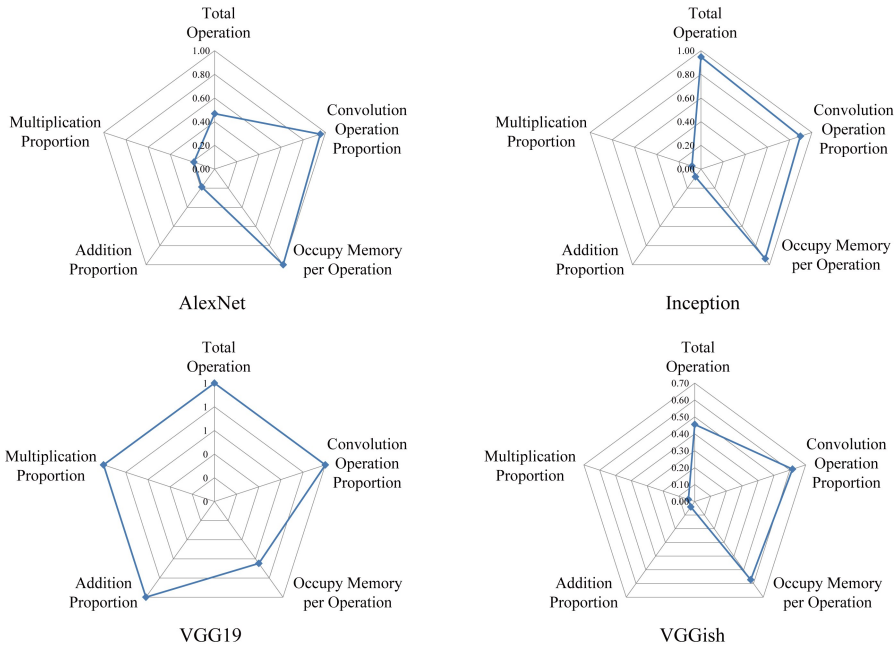

Fig. 8 Detailed comparison of AlexNet, Inception, VGG19 and VGGish resources

图 8 AlexNet,Inception,VGG19,VGGish 网络资源占比详细分析

3.2 神经网络调度优化评测

我们对比了所提出的最大值更替调度算法和顺序执行方案的资源利用率.顺序调度方案将可以同时执行的操作按从大到小的顺序依次放入芯片中.若可以放下,则更新当前芯片容量,同时减少该种算子的数量;若放不下则启用下一个芯片计数继续放置,算法描述如算法 2 所示.

算法 2. 顺序调度.

```
while (仍有卷积层未删除) {
  if (C_i − max ≥ 0) {
    C_i = C_i − max;
    k_i = k_i − 1;
    记录当前的 b_i;
    if (k_i == 0) {
      删除该卷积层;
      最大值设为当前数据中的最大值;
    }
  }
  else {
    输出所有记录的 b_i;
    重置记录 b_i 的数组;
    算法计数器+1;
    C_i = C;
    最大值设为当前数据中的最大值;
  }
}
```

本文参考 XC7VX690T 系统资源进行调度,该实验板的 DSP 数量为 3 600.规划方案中芯片的平均利用率如图 9 所示:

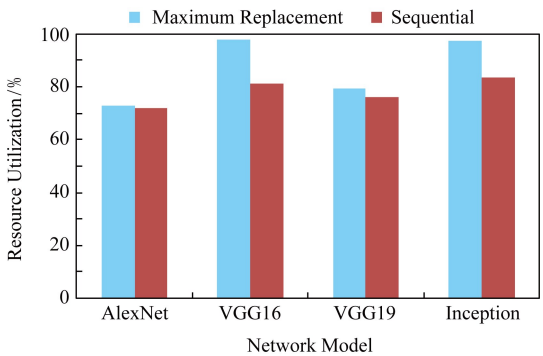

Fig. 9 Hardware resource utilization rate

图 9 硬件资源利用率

结果显示最大值更替调度算法显著地提高了 VGG16 和 Inception 网络运行时的硬件资源利用率.VGG16 网络中,相对于传统的顺序调度,芯片

利用率仅有 81.31%,而在相同条件下,采用最大值更替调度算法,可以将芯片使用率提升到 97.57%.于此类似,在 Inception 网络中,最大值更替调度算法将资源利用率提高了 13.89%.而对于 AlexNet 和 VGG19 两种算法的调度结果相差不大.从网络特性角度分析,AlexNet 网络的雷达图同 Inception 以及 VGG 都不同,而对于 AlexNet 算法顺序调度得到的结果已经相对较好,因此最大值更替算法的提升效果不明显.

而对比 VGG16 和 VGG19,最大值更替算法能否产生作用,与输入的数据分布有关.在这 2 个实验中,模型上的差异只有层数不一样,而卷积的相关参数是相同的,在这种情况下,同样的输入会使得调度大同小异,但若改变输入的数据,其调度结果会因调度算法的选择而产生较大的不同.本文所提出算法的计算复杂度为 $O(n!)$,顺序调度方式虽然计算复杂较低,但是其芯片利用率远小于基于最大值更替调度算法所提供的调度效果.

4 结 论

神经网络技术日益发展的今天,各个领域对网络运算速度和精度的需求都在不断提高.然而随着应用领域的不同,网络模型也千差万别.如何根据网络规模设计高能效的加速器,以及基于有限硬件资源如何提高网络性能并最大化资源利用率是当今体系结构领域研究的重要问题.为此,本文提出一种基于算子的模型分析方法,将神经网络的层视为算子,首先分析模型的算子种类、数量和顺序.然后基于不同算子功能分析统计算子中乘法和加法等基本操作数量以及内存占用量等特征的公式,实现基于不同输入的模型特征分析.此外,本文提出基于计算特征的最大值更替调度算法,实现基于给定硬件资源和不同模型规模的运行调度方案.实验结果显示,本文所提出的方法为从体系结构角度分析神经网络,优化硬件加速器的设计提供了参考和指导.

参 考 文 献

[1] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection [C] //Proc of the 29th IEEE Conf on Computer Vision and Pattern Recognition (CVPR). Piscataway, NJ: IEEE, 2016: 779-788

[2] Redmonand J, Farhadi A. Yolo9000: Better, faster, stronger [C] //Proc of the 30th IEEE Conf on Computer Vision and Pattern Recognition (CVPR). Piscataway, NJ: IEEE, 2017: 6517-6525

[3] Szegedy C, Liu Wei, Jia Yangqing, et al. Going deeper with convolutions [C] //Proc of the 28th IEEE Conf on Computer Vision and Pattern Recognition (CVPR). Piscataway, NJ: IEEE, 2015: 1-9

[4] Liu Sicong, Lin Yingyan, Zhou Zimu, et al. On-demand deep model compression for mobile devices: A usage-driven model selection framework [C] //Proc of the 16th Annual Int Conf on Mobile Systems, Applications, and Services (MobiSys). New York: ACM, 2018: 389-400

[5] Lane N D, Bhattacharya S, Georgiev P, et al. Deepx: A software accelerator for low-power deep learning inference on mobile devices [C] //Proc of the 15th ACM/IEEE Int Conf on Information Processing in Sensor Networks (IPSN). Piscataway, NJ: IEEE, 2016: 1-12

[6] Lane N D, Bhattacharya S. Sparsifying deep learning layers for constrained resource inference on wearables [C] //Proc of the 14th ACM Conf on Embedded Network Sensor Systems (SenSys). New York: ACM, 2016: 176-189

[7] Li Jiajun, Yan Guihai, Lu Wenyan. CCR: A concise convolution rule for sparse neural network accelerators [C] //Proc of the 2018 Design, Automation & Test in Europe Conf (DATE). Piscataway, NJ: IEEE, 2018: 189-194

[8] Liang Yun, Lu Liqiang, Xiao Qingcheng, et al. Evaluating fast algorithms for convolutional neural networks on FPGAs [C] //Proc of the 25th IEEE Symp on Field Programmable Custom Computing Machines (FCCM). Piscataway, NJ: IEEE, 2017: 101-108

[9] Liu Baoyuan, Wang Min, Foroosh H, et al. Sparse convolutional neural networks [C] //Proc of the 28th IEEE Conf on Computer Vision and Pattern Recognition (CVPR). Piscataway, NJ: IEEE, 2015: 806-814

[10] Howard A G, Zhu Menglong, Chen Bo, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017

[11] Han Song, Mao Huizi, Dally W J. Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding [C] //Proc of the 4th Int Conf on Learning Representations (ICLR). New York: ACM, 2016: Article ID: 11

[12] Changpinyo S, Sandler M, Zhmoginov A. The power of sparsity in convolutional neural networks [C] //Proc of the 5th Int Conf on Learning Representations (ICLR). New York: ACM, 2017: Article ID: 336

[13] Venieris S I, Bouganis C. Latency-driven design for FPGA-based convolutional neural networks [C] //Proc of the 27th Int Conf on Field Programmable Logic and Applications (FPL). Piscataway, NJ: IEEE, 2017: 1-8

[14] Chen Y H, Emer J, Sze V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks [C] //Proc of the ACM/IEEE 43rd Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2016: 367-379

[15] Alwani M, Chen Han, Ferdman M, et al. Fused-layer CNN accelerators [C] //Proc of the 49th Annual IEEE/ACM Int Symp on Microarchitecture (MICRO). Piscataway, NJ: IEEE, 2016: 1-12

[16] Iandola F N, Han Song, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size [C] //Proc of the 4th Int Conf on Learning Representations (ICLR). New York: ACM, 2016: Article ID: 465

[17] Altaf M S B, Wood D A. LogCA: A high-level performance model for hardware accelerators [C] //Proc of the ACM/IEEE 44th Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2017: 375-388

[18] Zhang Chen, Li Peng, Sun Guangyu, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks [C] //Proc of the 23rd ACM/SIGDA Int Symp on Field-Programmable Gate Arrays (FPGA). New York: ACM, 2015: 161-170

[19] Motamedi M, Gysel P, Akella V, et al. Design space exploration of FPGA-based deep convolutional neural networks [C] //Proc of the 21st Asia and South Pacific Design Automation Conf (ASP-DAC). Piscataway, NJ: IEEE, 2016: 575-580

[20] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks [C] //Proc of the Neural Information Processing Systems (NIPS). Cambridge, MA: MIT Press, 2012: 1097-1105

[21] Shen Yongming, Ferdman M, Milder P. Maximizing CNN accelerator efficiency through resource partitioning [C] //Proc of the ACM/IEEE 44th Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2017: 535-547

[22] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition [C] //Proc of the 2nd Int Conf on Learning Representations (ICLR). New York: ACM, 2014: Article ID: 4

[23] Liu Wei, Anguelov D, Erhan D, et al. SSD: Single shot multibox detector [C] //Proc of the European Conf on Computer Vision (ECCV). Cham: Springer, 2015: 21-37

[24] Xu K, Ba J L, Kiros R, et al. Show, attend and tell: Neural image caption generation with visual attention [C] //Proc of the 32nd Int Conf on Machine Learning (ICML). New York: ACM, 2015: 37:2048-37:2057

Fang Rongqiang, born in 1996. Master candidate. His main research interests include computer architecture, high performance computing.

Wang Jing, born in 1982. PhD, associate professor. Her main research interests include computer architecture, energy efficient computing, high performance computing, hardware reliability and variability.

Yao Zhicheng, born in 1989. Master, engineer. His main research interests include computer architecture, high performance computing.

Liu Chang, born in 1996. Bachelor, engineer. Her main research interests include computer architecture, high performance computing.

Zhang Weigong, born in 1967. PhD, professor. Member of CCF. His main research interests include computer architecture, high reliable embedded system design.