基于层间特征图压缩的卷积神经网络加速器

谢良波[1*],陈 林[1],周 牧[2],卜文杰[1]
(1. 重庆邮电大学通信与信息工程学院,重庆 400065;2. 重庆邮电大学电子科学与工程学院,重庆 400065)

摘 要: 随着深度学习技术的飞速发展,卷积神经网络(Convolutional Neural Network,CNN)在图像识别与处理任务中展现出卓越的性能。然而,随着网络深度的增加,海量的中间数据传输给硬件加速器的片上存储和访存带宽带来了巨大的压力,“访存墙”问题日益凸显,严重制约了系统的整体吞吐量与能效比。针对该问题,现有的层间特征数据压缩方法主要分为两类。一类侧重于硬件实现的轻量化,其面积开销虽小,但受限于算法复杂度,压缩率较低,难以有效缓解高吞吐场景下的片外带宽压力。另一类追求压缩性能,导致过高的硬件面积开销,难以在资源受限的边缘设备上部署。针对上述挑战,本文提出了一种面向CNN层间特征图的统计感知混合压缩方法,核心设计目标是实现高压缩率和低硬件开销,解决压缩性能和资源消耗难以兼顾的问题。该方法通过深入挖掘数据的稀疏性与分布特征,结合“离线分析-在线压缩”的软硬件协同机制,实现了硬件友好的数据编码。离线分析阶段,对CNN层间特征数据进行统计分析,生成所需编码表及基准值。在线压缩阶段,对特征数据进行分类,划分为零值数据与非零值数据,对零值数据,采用结合熵编码的增强型零游程编码;对非零数据,采用动态基准-增量编码。该差异化编码机制在维持高压缩率的同时,将硬件面积开销降低了58.7%~72.9%,解决了传统压缩算法硬件复杂度高的问题。基于AlexNet、VGG16、ResNet34和MobileNetV2四种具有代表性的CNN层间特征图压缩实验,对本文所提方法在不同网络结构和数据格式下的压缩性能进行了系统评估。实验结果表明,相较于同类研究,本文所提数据压缩方法在INT8量化格式下的压缩率最高提升了58.5%,在FP32/FP16格式下最高提升了36.7%。在ALINX AXU5EV目标平台上部署VGG16模型,基于本文数据压缩方法的加速器的推理吞吐量可达242.8 GOPS,相比无压缩基准架构,运算性能与能效比分别提升了41.4%和27.8%。实验结果表明,本文所提方法平衡了CNN层间特征图压缩的压缩率和硬件开销,为资源受限边缘场景下的CNN加速器设计提供了新的解决方案。

关键词: 现场可编程门阵列;卷积神经网络加速器;硬件加速;层间特征图;数据压缩

基金项目: 重庆市自然科学基金资助项目(No.CSTB2023NSCQ-MSX0249,No.CSTB2023NSCQ-LZX0126);重庆市教委科学技术研究基金资助项目(No.KJQN202300615)

中图分类号: TP302;TN47 **文献标识码:** A **文章编号:** 0372-2112(XXXX)XX-0001-17

电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20260134

A Convolutional Neural Network Accelerator Based on Inter-layer Feature Map Compression

XIE Liangbo[1*], CHEN Lin[1], ZHOU Mu[2], BU Wenjie[1]
(1. *School of Communications and Information Engineering*, *Chongqing University of Posts and Telecommunications*, *Chongqing* 400065, *China*;
2. *School of Electronic Science and Engineering*, *Chongqing University of Posts and Telecommunications*, *Chongqing* 400065, *China*)

Abstract: With the rapid development of deep learning technology, convolutional neural networks (CNNs) have demonstrated exceptional performance in image recognition and processing tasks. However, as the network depth increases, the massive transmission of intermediate data imposes tremendous pressure on the on-chip memory and memory access bandwidth of hardware accelerators. The increasingly prominent “memory wall” problem has severely constrained the overall throughput and energy efficiency of the system.To address this issue, existing inter-layer feature data compression methods are mainly divided into two categories. The first category focuses on lightweight hardware implementation: despite low area overhead, their compression ratio is limited by algorithm complexity, making it difficult to effectively alleviate the off-chip bandwidth pressure in high-throughput scenarios. The second category pursues superior compression performance, but incurs excessive hardware area overhead, which is hard to deploy on resource-constrained edge devices.Aiming at the above challenges, this paper proposes a statistic-aware hybrid compression method for CNN inter-layer feature maps, with the core design goal of achieving high compression ratio and low hardware overhead to resolve the difficulty in balancing compres-

收稿日期:2026-02-14;录用日期:2026-03-19;责任编委:苏钰淇
*通信作者:谢良波

sion performance and resource consumption. By deeply exploiting the sparsity and distribution characteristics of the data, this method realizes hardware-friendly data coding combined with a hardware-software co-design mechanism of "offline analysis-online compression". In the offline analysis stage, statistical analysis is performed on the CNN inter-layer feature data to generate the required coding tables and baseline values. In the online compression stage, the feature data are classified into zero-value data and non-zero-value data. For zero-value data, an enhanced zero run-length encoding combined with entropy coding is adopted; for non-zero data, dynamic baseline-delta encoding is applied. This differentiated coding mechanism reduces the hardware area overhead by 58.7% to 72.9% while maintaining a high compression ratio, which solves the problem of high hardware complexity in traditional compression algorithms. We conduct a systematic evaluation of the compression performance of the proposed method under different network structures and data formats, based on compression experiments on inter-layer feature maps of four representative CNNs: AlexNet, VGG16, ResNet34, and MobileNetV2. Experimental results show that, compared with similar studies, the proposed data compression method achieves a maximum improvement of 58.5% in compression ratio under the INT8 quantization format, and a maximum improvement of 36.7% under FP32/FP16 formats. When deploying the VGG16 model on the ALINX AXU5EV target platform, the accelerator based on the proposed data compression method reaches an inference throughput of 242.8 GOPS. Compared with the compression-free baseline architecture, the computing performance and energy efficiency are improved by 41.4% and 27.8%, respectively. The experimental results demonstrate that the proposed method balances the compression ratio and hardware overhead for CNN inter-layer feature map compression, and provides a new solution for the design of CNN accelerators in resource-constrained edge scenarios.

Keywords: field programmable gate array; convolutional neural network accelerator; hardware accelerator; interlayer feature map; data compression

Foundation Item(s): Chongqing Natural Science Foundation Project (No.CSTB2023NSCQ-MSX0249, No. CSTB2023NSCQ-LZX0126); Science and Technology Research Program of Chongqing Municipal Education Commission (No.KJQN202300615)

0 引言

卷积神经网络(Convolutional Neural Network, CNN)是深度学习领域的一个分支,具有很强的特征提取能力,被广泛用于处理图像分类[1]、视频跟踪[2]和文本分析[3]等任务,为计算机视觉和自然语言处理等人工智能领域的发展做出了巨大的贡献。然而,由于CNN具有数据量大和计算量大等特点,其在边缘设备上的部署受到限制;同时,边缘设备通常要求处理的实时性[4],这使得CNN在边缘端的应用面临巨大挑战。

目前主流的CNN加速方案涵盖了图形处理器(Graphics Processing Units, GPU)、专用集成电路(Application Specific Integrated Circuit, ASIC)和现场可编程门阵列(Field-Programmable Gate Array, FPGA)等多种架构。其中,边缘GPU(如NVIDIA Jetson系列[5])凭借成熟的并行计算生态和强大的计算通量,显著降低了CNN的部署门槛。然而,受限于其通用的单指令多线程架构,GPU在处理访存密集型任务时存在较高的硬件冗余与功耗开销,难以完全满足严苛功耗受限场景下的加速需求。ASIC架构则通过固化的数据流优化与定制化计算阵列,实现了极致的能效比与低功耗特性。例如,Eyeriss v2在65 nm工艺下通过引入分层片上网络[6],增强了对稀疏模型及复杂访存模式的适配能力;文献[7]提出了一种面向资源受限的片上网络组播方案,采用树形组播网络降低加速器的片上通信延迟;文献[8]通过改进Booth编码器与Wallace树,在底层电路层面降低了乘累加运算的复杂度;文献[9]在ASIC上实现了基于随机计算的CNN加速器,通过全并行的随机计算架构减少了推理延迟;文献[10]提出了一种内存利用感知的ASIC架构,通过800 MHz的极高主频和定制的权重固定数据流,达到了极高的能效比。尽管如此,ASIC方案的功能在流片后难以更改,且其昂贵的流片费及长迭代时间周期限制了其对算法快速迭代的灵活性支持。相比之下,FPGA凭借其可编程逻辑与可定制的存储层级,不仅能实现高效的并行计算,更能灵活部署自定义的自适应数据压缩逻辑,成为实现软硬件协同优化的理想载体。

部署CNN的关键挑战之一是数据的内存访问问题,其在加速器系统总能效中占比超70%[11],为此,研究人员致力于通过多种途径降低内存访问量与神经网络计算复杂度。在算法层面,提出了硬件友好的网络量化与剪枝方法。通常情况下,神经网络训练与推理默认采用32位浮点数据以保证精度,而网络量化与剪枝方法旨在以更低比特表示数据或移除冗余参数,从而减轻存储与计算负担。因此,在可接受的精度损失范围内,将数据量化为定点数或整型数据,

以减少计算量和内存访问。在算法层面,文献[12]提出了均匀增量量化策略用于稀疏神经网络模型的量化,文献[13]则对MobileNetV2进行8位整数量化,以提升其硬件友好性。为消除训练与推理中的冗余计算,剪枝方法通过移除对输出影响较小的网络连接来降低计算复杂度。文献[14]提出动态细粒度剪枝方法,以减少训练开销与片上资源占用。文献[15]针对非结构化剪枝的计算不规则问题,提出重排特征图(feature map, fmap)和权重的策略以减少计算冗余。在硬件层面,研究者通过设计高效并行架构与数据重用策略来提升能效。Eyeriss采用行平稳数据流提高加速器的吞吐量[16],文献[17]则利用多计算引擎的并行架构,显著提升系统处理能力。上述这些方法均从不同角度减少了计算量与数据移动,从而有效降低了系统整体能耗。

由于CNN中广泛使用整流线性单元(Rectified Linear Unit, ReLU)作为激活函数,其将负值置零、正值保留的特性,导致层间数据具有较高的稀疏性。为减少存储与传输开销,研究者提出了多种面向稀疏数据的压缩方法。文献[18]提出零游程编码,通过记录连续的零值来压缩数据。文献[19]则利用非零值的位置掩码及其数值进行压缩。Cnvlutin[20]和EIE[21]采用类似压缩稀疏行(Compress Sparse Rows, CSR)格式,记录非零元素的偏移量与数值。文献[22]将CSR和基于位置掩码的方法结合在一起,根据特征图不同的稀疏度灵活切换压缩方法。然而,上述方法仅针对零值进行压缩,未能有效压缩非零值,因而整体压缩率有限。EBPC[23]结合零游程编码(Zero Run-Length Encoding, Zero-RLE)和位平面压缩(Bit Plane Compress, BPC)同时压缩零值和非零值,使得整体的压缩效果更好,但BPC需要利用数据的相关性进行压缩,对于相关性不大的数据压缩效果会下降。文献[24]通过分析浮点数的数据特性,对浮点数的指数位进行增量压缩,但对于零值仍采用Zero-RLE,压缩程度有限,且未考虑不同网络层间数据的分布差异。

为利用特征图高稀疏性缓解边缘CNN部署的存储与功耗瓶颈,本文提出了一种基于离线统计分析的层间特征图高效混合压缩架构。本文的主要贡献概括为:(1)提出了一种基于统计感知的层间特征图混合压缩策略。通过深入分析CNN特征图的数据分布规律与稀疏特性,设计了差异化的编码策略,实现零值和非零值的高效压缩。(2)设计了一种低开销的软硬件协同压缩架构,将高开销任务移至离线编译阶段,极大地简化了片上压缩/解压模块的逻辑设计。(3)完成了基于FPGA的加速器集成与系统级性能验证。实测结果显示,该方案有效突破了片外存储带宽瓶颈,使系统推理吞吐量达到242.8 GOPS,运算性能与能效比相比无压缩基准设计分别提升了41.4%和27.8%。

1 相关工作

1.1 零游程编码

游程编码(Run-Length Encoding, RLE)是一种基于数据连续重复特性的简单无损压缩技术。其核心思想是:将连续出现的相同符号序列用一个数据对来替代,从而消除数据中的空间冗余或时间冗余。它对高度同质化的数据能取得显著的压缩效果。数学上,对于一个由符号s连续出现l次构成的游程,RLE将其编码为有序对(s,l)。原始数据D可表示为

$$D=s_1^{l_1}s_2^{l_2}\cdots s_k^{l_k} \tag{1}$$

其中,$s_i\neq s_{i+1}$。编码后的数据C即为序列:

$$C=(s_1,l_1),(s_2,l_2),\cdots,(s_k,l_k) \tag{2}$$

游程解码(Run-Length Decoding, RLD)过程为,顺序读取每个(符号,长度)对,将符号重复输出指定的长度,即可无损恢复原始数据。在卷积神经网络中,经过ReLU激活后的特征图通常具有高度稀疏性,往往包含大量连续的零值,使用RLE对零值进行编码的方法就被称为零游程编码。Zero-RLE具备极低的算法和硬件实现复杂度,在FPGA中仅需比较器与计数器即可实现,几乎不占用额外的片上存储资源。该编码方式具备完美的流式处理能力,它不需要预先读取整个文件来建立统计模型,这使得Zero-RLE非常适合对延迟敏感的实时数据流压缩。

1.2 基准-增量编码

基准-增量编码(Base-Delta EnCoding)是一种基于数值平移的无损数据预处理技术。其核心思想是通过减去一个固定的参考值,将原始数据的数值范围平移至更靠近零的区域,从而减少表示每个数据值所需的比特位数。从数学角度看,对于原始数据序列$X=\{x_1,x_2,\cdots,x_n\}$,基准编码将其转换为新的序列$B=\{b_1,b_2,\cdots,b_n\}$:

$$b_i=x_i-X_{\text{ref}},\ i=1,2,\cdots,n \tag{3}$$

其中,X_{ref}是预先选定的基准值。解码时只需执行逆运算:

$$x_i=b_i+X_{\text{ref}} \tag{4}$$

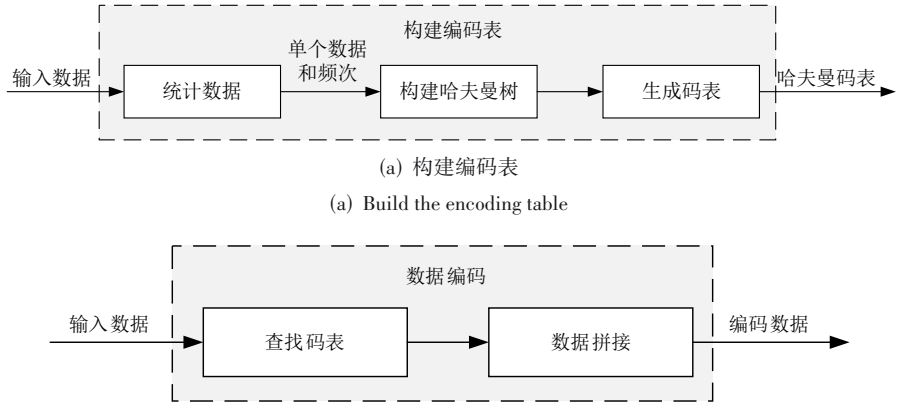
从硬件实现角度来看,基准增量编码最主要的优势是计算逻辑简单,编码与解码过程仅涉及加法与减法运算,适合在FPGA中利用查找表(Look-Up Table, LUT)或数字信号处理(Digital Signal Processing, DSP)单元高效实现。由于计算逻辑简单,其资源开销低,无需复杂控制逻辑或大规模查找表,因此编码模块可实现为流水化结构。其编码过程主要基于前一时刻

数据的递推关系，编码与解码过程可与计算模块并行执行，不影响硬件整体的吞吐量。

1.3 哈夫曼编码

哈夫曼编码是一种用于无损数据压缩的熵编码算法。其基本原理是依据符号出现的统计特性进行变长编码。通过为出现频率较高的符号分配较短的编码长度，而对低频符号采用较长的编码表示，该方法能够有效降低整体数据表示的冗余度。对于符号重复性较强的数据场景，哈夫曼编码通常能够获得较为理想的压缩效果。

哈夫曼编码的整体处理流程如图1所示，主要包括编码表构建（图1(a)）和数据编码（图1(b)）两个阶段。在编码表构建阶段，编码器首先对输入数据流中的符号进行统计分析，计算各字符的出现频率并生成频率统计表；随后依据该统计结果构建哈夫曼树，并由此生成对应的编码表。进入数据编码阶段后，编码器按照输入数据流的顺序，查找编码表并将各字符映射为其唯一对应的二进制码字，最终通过码字拼接形成连续的编码比特流，从而有效去除数据冗余并输出压缩后的编码结果。



(a) 构建编码表

(a) Build the encoding table

(b) 数据编码

(b) Data coding

图1 哈夫曼编码流程图

Figure 1 Huffman coding flowchart

哈夫曼编码树的示例构建过程如图2所示。假设输入的文本数据流为：ABECBDEECDECDBECECDBCACEEDDECCDCBBCCEECA。接下来统计数据流中出现的字符种类，并将其存放在字符集合S中，S={A, B, C, D, E}，统计每个字符出现的频次将其存放在频次集合F中，F={3, 6, 13, 7, 11}，之后将字符集合S和频次集合F输入到构建哈夫曼树的算法中以构建哈夫曼树。其中，左子树的前缀为0，右子树的前缀为1，从哈夫曼编码树的根结点走向叶子结点，经过码字组合的编码即为对应符号的哈夫曼编码。整理后的编码表如表1所示。

在得到输入数据流的字符编码表之后，将字符与二进制编码对应再拼接就能得到最后的编码数据流为000001…10000，由于任意码字均不会成为其他码字的前缀，所以在解码过程中只需按顺序依次读取比特并对应上字符即可完成解码。

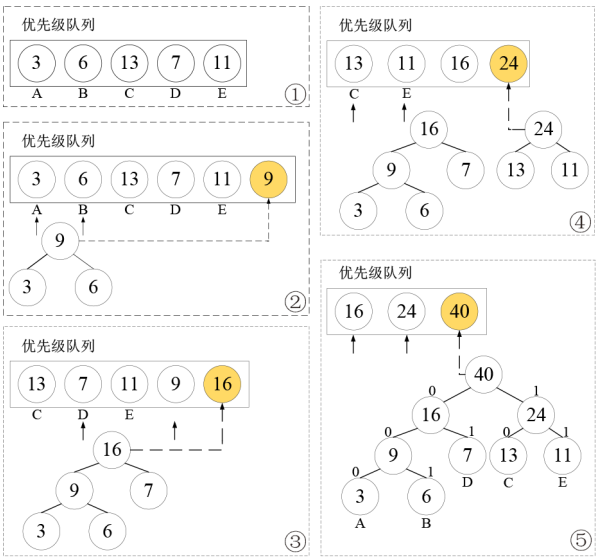


图2 示例树构建过程图

Figure 2 Diagram of the example tree construction process

2 高效混合压缩策略

图3展示了本文提出的统计感知混合压缩架构。如图3(a)所示，在离线统计分析阶段，通过对卷积神经网络层间特征图的零值与非零值分布进行统计分析（具体分析方法将在后文详细阐述），生成优化的零值编码表及逐层最优基准值。如图3(b)所示，在

表1 字符编码表

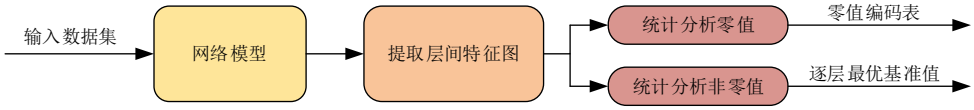
Table 1 Character encoding table

字符	二进制编码	长度
A	000	3
B	001	3
C	10	2
D	01	2
E	11	2

片上数据编码阶段,输入的特征数据流通过零值比较器实现流分离,划分为稀疏零值流与非零值数据流。针对稀疏零值流,系统调用预存的编码表执行熵编码增强的Zero-RLE;针对非零值数据流,采用动态基准-增量编码,通过逐层更新基准值,以大幅缩减数据的动态范围与有效位宽。最后对编码数据进行位对齐处理将变长编码值打包输出。

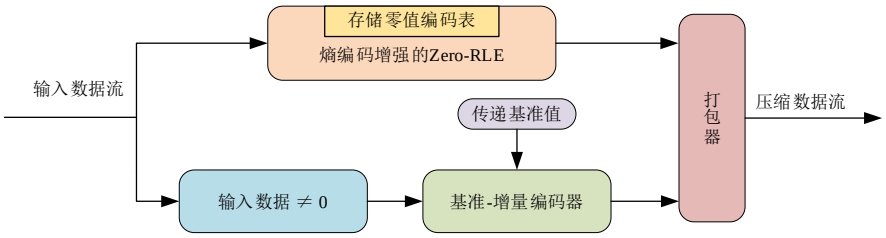
2.1 熵编码增强的Zero-RLE零值压缩方法

现有的稀疏压缩方案通常在压缩率与硬件效率之间进行权衡。位图编码虽然提供了良好的并行性,但无法利用零值的长程相关性进一步降低存储密度;而传统的游程编码虽然利用了空间连续性,但其定长的计数器表示在处理短游程时效率欠佳。本文提出的方法通过引入基于统计先验的哈夫曼编码,对Zero-RLE的输出进行二次熵压缩,有效地结合了Zero-RLE处理长序列的能力与熵编码处理非均匀分布的优势。



(a) 离线统计分析总体框图

(a) Block diagram of offline statistical analysis



(b) 片上数据编码总体框图

(b) Block diagram of on-chip data encoding

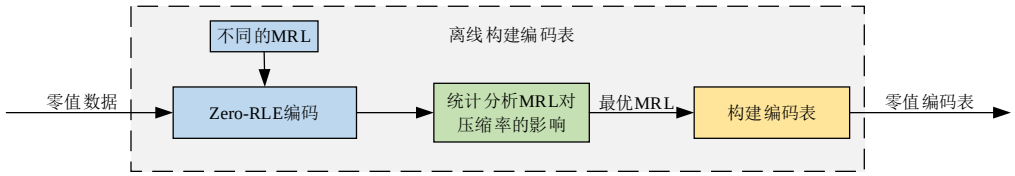
图3 压缩方法总体框图

Figure 3 Overall block diagram of the compression method

基于熵编码增强的零游程编码流程如图4所示。在编码表构建阶段,如图4(a)所示,采用离线统计分析的方法。首先针对稀疏零值数据流实施Zero-RLE编码预处理。在此过程中,最大游程长度(Maximum Run-Length,MRL)的设定至关重要,它作为硬约束限制了计数器的位宽与数值范围。为了确定最优的MRL,需要在压缩增益与硬件资源开销之间进行多维度的权衡分析。最终,基于优化后的游程长度分布,采用哈夫曼算法生成变长码表,以实现熵编码层级的二次压缩。在数据编码阶段,如图4(b)所示,只需要先进行Zero-RLE编码,再通过多路选择器选择零值编码值,即可完成稀疏零值的压缩。

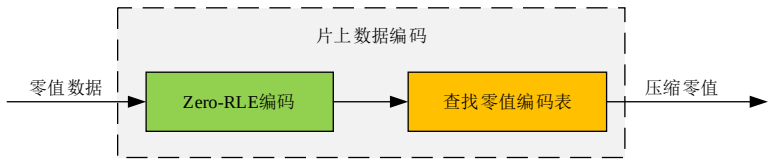
图5展示了在单精度浮点数(32-bit Floating Point, FP32)下,不同网络的零值压缩率随MRL增大的变化情况(半精度浮点数(16-bit Half-Precision Floating Point, FP16)与8位整数(8-bit Integer,INT8)格式下趋势相似)。随着MRL的增大,各网络的压缩率均持续上升;当MRL=13时,ResNet34与MobileNetV2的压缩率增长已趋于饱和,而VGG16和AlexNet的增长趋势也显著放缓。由于减少编码值的种类有利于节省片上存储资源,综合权衡压缩率与硬件开销后,最终选取MRL=13作为最优值。

VGG16不同零计数值(Zero-Count Value,ZCV)的分布情况如图6所示。根据数据分布结果对1~13进行哈夫曼编码,明确哈夫曼树的左子树前缀为0,右子树前缀为1,即可生成哈夫曼码表。表2汇总了不同数据类型的编码结果详情,其中ZCV展示了生成的哈夫曼码字及其位宽。除了ZCV的编码外,本文还将重点介绍零计数值位置(Position of Zero-Count Value,PZCV)、增量压缩值及未压缩值的编码方法。表中涉及的diffBit、expBit和fracBit分别代表增量值、



(a) 离线构建编码表

(a) Build the code table offline



(b) 片上数据编码

(b) On-chip data encoding

图4 零值压缩方法流程图

Figure 4 Flowchart of the zero-value compression method

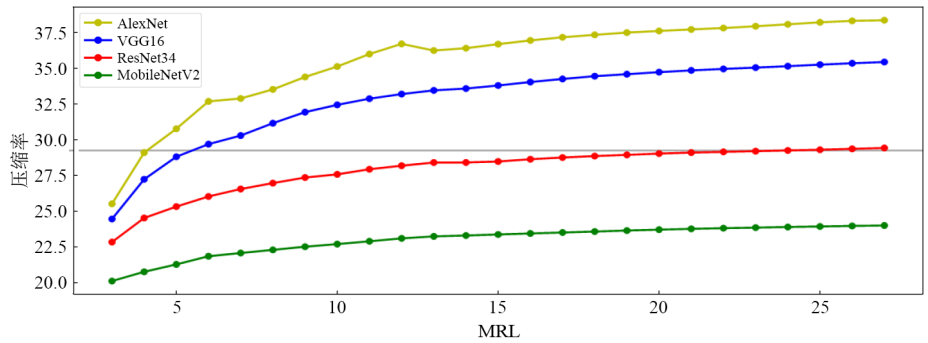


图5 MRL对不同网络的零值压缩率影响情况

Figure 5 The influence of MRL on the zero-value compression ration of different networks

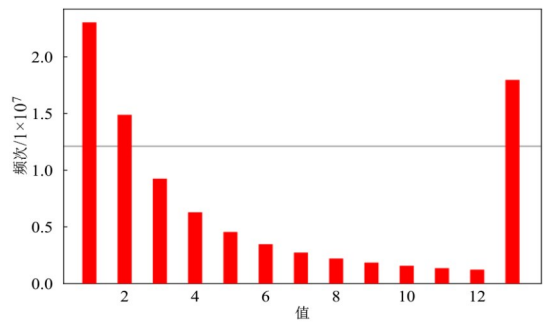


图6 零计数值分布情况

Figure 6 Distribution of zero-count values

指数位和尾数位，diffBits、expBits 和 fracBits 则分别代表其数据位宽。

2.2 基于动态基准-增量编码的非零值压缩方法

针对非零数据的压缩，本文提出了一种动态基准-增量编码方法。考虑到 IEEE 754 浮点数经 ReLU 激活后恒为非负的特性，符号位不再包含有效信息，因此压缩重心被转移至决定数值动态范围的指数域。

表2 VGG16的fmap编码表

Table 2 The fmap encoding table of VGG16

数据类型		码值(binary)	长度[bits]
ZCV	1	01	2
	2	101	3
	3	0001	4
	4	0011	4
	5	00100	5
	6	00101	5
	7	10001	5
	8	10011	5
	9	100000	6
	10	100001	6
	11	100100	6
	12	100101	6
	13	11	2
PZCV		01	2
增量压缩值		1 & diffBit & fracBit	1 + diffBits + fracBits
未压缩值		00 & expBit & fracBit	2 + expBits + fracBits

对于量化后的INT8数据,鉴于其统一的定点数值结构,可将该编码方式直接应用于整个8位整数数值,从而实现了对全字长的整体压缩。该方法的压缩流程如图7所示。

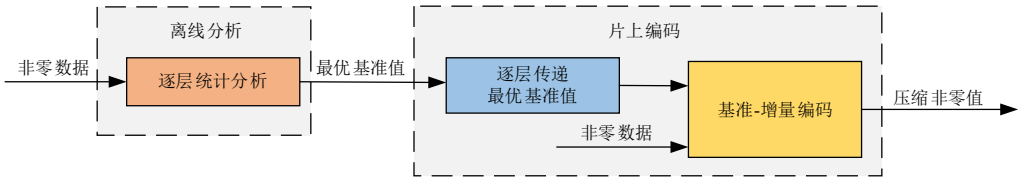


图7 非零值压缩方法流程图

Figure 7 Flowchart of non-zero value compression method

在离线统计分析阶段,通过预先对校准数据集进行推理,捕获各层特征图指数位的分布规律。基于预设的硬件存储位宽约束,为每一层选取一个最优的固定基准值。最终,仅存储原始指数值与该基准值之间的差值,从而实现对非零数据的高效压缩。用逐层统计的方法选择基准-增量压缩基准值的优势在于:1、每一层的固定比较值单独统计,整个网络的整体压缩效果更好;2、统计的基准值具有更好的压缩效果。

对于基准值的选取上,本文采用逐层统计分析的方法,针对不同层的数据特征分别确定基准值。设L表示神经网络的第L层,该层特征图数据的取值集合为$X^{(L)}$。令$H^{(L)}(v)$为数值v(在浮点数格式下为指数域值,在INT8格式下为全字长整数值)在该层中出现的频次,即直方图统计函数,其中$v\in[V_{\min},V_{\max}]$。

在动态基准-增量编码中,设b_{diff}为增量存储位宽(即diffBits),则该位宽能覆盖的连续数值范围W定义为

$$W=2^{b_{\text{diff}}} \tag{5}$$

为了寻找最优基准值β_{opt},使得该基准值所覆盖的长度为W的区间内包含的数据总量最大,从而最大化压缩率。该优化问题可表示为

$$\beta_{\text{opt}}^{(L)}=\arg\max_v\left(\sum_{k=0}^{W-1}H^{(L)}(v+k)\right) \tag{6}$$

其中,求和项$\sum_{k=0}^{W-1}H^{(L)}(v+k)$表示以$v$为起始值,长度为$W$的连续区间内的累计数据频次。

一旦确定了最优基准值β_{opt},对于任意输入数据$x\in X^{(L)}$,其压缩判决条件定义为

$$\text{Mode}(x)=\begin{cases}\text{Compressed}, & \text{if } \beta_{\text{opt}}\le v\le\beta_{\text{opt}}+W-1\\ \text{Uncompressed}, & \text{otherwise}\end{cases} \tag{7}$$

以图8统计结果为例,若增量存储位宽为2,则该层选取125作为基准值,对应数据压缩范围为125~128;其他层的基准值亦按照此方法确定。

本文压缩方法的编码输出包含两个数据流:零计数值(ZCV)流与非零值流。为区分压缩与非压缩数据,非零值的符号位被替换为前缀码,而尾数位则保持原样存储。如图9所示,压缩后的非零值数据流由三类元素构成:增量压缩值、未压缩值及零值计数值(记录1~13个连续零的个数)。以图9中VGG16的FP32特征图统计数据为例,对于可压缩的非零元素,采用1比特前缀码标识,其编码结构为[前缀码+增量值(diffBit)+尾数位(fracBit)];对不可压缩的非零元素,则采用2比特前缀码,编码结构为[前缀码+指数位(expBit)+尾数位(fracBit)]。此外,数据流中还需记录ZCV的位置(PZCV),该信息同样以2比特编码表示。解压时,算法依据前缀码识别数据类型,并根据记录的零值个数恢复原始数据。

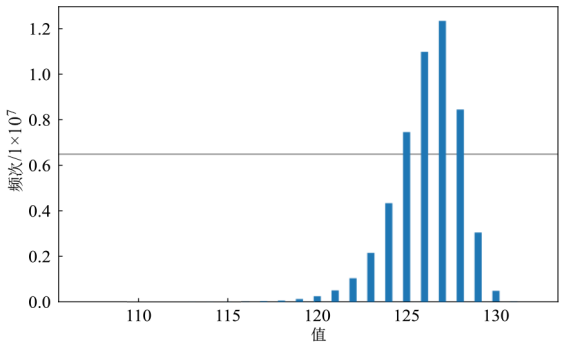


图8 VGG16第二层指数值统计图

Figure 8 The statistical chart of the fmap values of the second layer of VGG16

3 硬件架构

3.1 压缩器与解压缩器硬件架构

鉴于前文所述的压缩算法原理系基于FP32数据格式展开,为保持阐述的一致性,本小节的硬件架构设计亦以FP32格式为例进行图示与说明。需要指出的是,该架构具有良好的通用性与可扩展性,其核心逻辑与数据通路设计同样适用于FP16和INT8格式,仅需调整相应的数据位宽即可。整体硬件架构包括压缩器和解压缩器,压缩器负责将层间数据压缩以提高数据传输效率和节省存储空间,在卷积计算之前解压缩器负责解压缩数据。

压缩器主要由两部分组成,如图10所示,一部

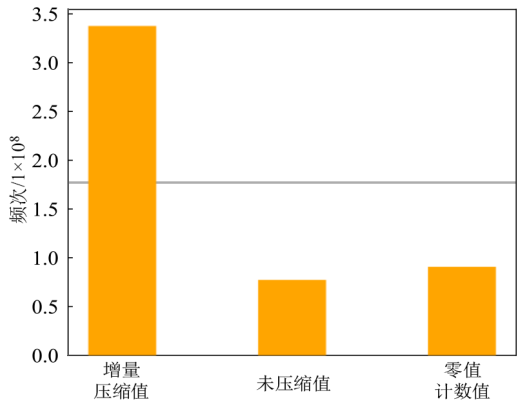


图9 VGG16的fmap数据统计图

Figure 9 The data statistics graph of the fmap of VGG16

分是熵编码增强的Zero-RLE负责压缩数据中的零值,该部分由Zero-RLE子模块级联哈夫曼编码子模块,另一部分是基准-增量编码器负责压缩数据中的非零值。整体数据流向为:原始数据流输入后,首先进入Zero-RLE子模块,完成连续零值计数与非零数据提取;提取的零计数值送入哈夫曼编码子模块,生成零计数编码值并打包输出;提取的非零数据送入基准-增量编码子模块,完成增量编码后,由打包器转换为32位定长数据输出。各子模块通过同步时钟与有效标志实现数据协同,确保时序一致性。

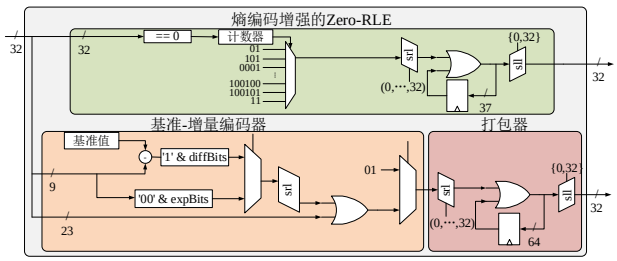


图10 压缩器架构图

Figure 10 Compressor architecture diagram

解压缩器作为压缩的逆过程,负责将压缩后的双数据流恢复为原始数据流。整体架构如图11所示,采用双输入流水线并行处理架构,由熵编码增强的Zero-RLD,解包器和基准-增量解码器协同完成解压缩任务。解压缩器接收两个独立输出流:零值编码流和非零数据压缩流,通过并行解码、时序协调,最终恢复为与原始数据完全一致的输出流。这一设计充分考虑了数据流的内在特性,通过分离处理零值和非零值两条路径,既保证了处理效率,又确保了数据的完整性和时序正确性。模块采用全同步设计,所有操作在统一的时钟域内进行,通过精心设计的控制逻辑确保数据流的正确传递和处理。各子模块之间通过有效标志和就绪信号实现握手通信,形成了完整的流水线处理结构。

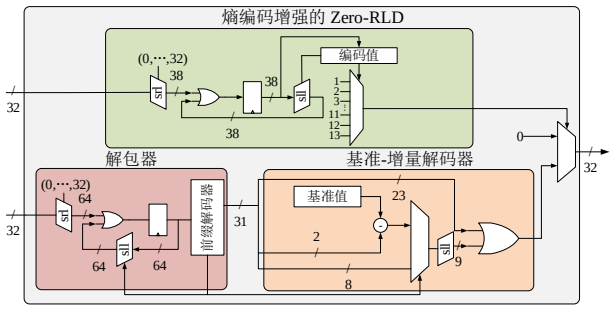


图11 解压缩器架构图

Figure 11 Decompressor architecture diagram

3.2 加速器硬件架构

图12为所设计的加速器架构示意图,整体分为处理系统端(Processing System,PS)端的控制调度部分和可编程逻辑(Programmable Logic,PL)端的推理计算部分。PS端主要包括安全数码(Secure Digital,SD)卡、CPU和双倍数据速率同步动态随机存储器(Double Data Rate,DDR)。SD卡中存储网络推理所需的数据,CPU负责对整个加速流程进行控制,DDR主要负责存储推理过程中产生的中间数据。PL端主要由控制模块、缓存模块和计算模块三部分组成。控制模块负责接收PS端发送过来的控制指令并进行PL端的整体数据分配、计算逻辑控制和状态管理。缓存模块包括输入特征图缓存,权重缓存,输出特征图缓存以及其余片上缓存区。缓存模块根据存储要求,缓存推理时需要的数据。计算模块包括处理引擎(Processing Engine,PE)计算阵列,输出累加模块,激活模块和池化模块。计算模块根据计算需要,完成CNN的推理计算过程。

3.2.1 PE微架构优化

本设计针对卷积计算的稀疏特性,利用DSP单元自带的时钟使能端(Clock Enable,CE)来实现基于数据有效性的动态功耗管理。

如图13所示,当检测到输入特征数据为零时,门控电路通过拉低DSP48E2计算单元的时钟使能信号,阻断时钟网络对计算逻辑的驱动。该策略在保持数据流水线节拍严格对齐的前提下,最大限度地减少了无效数据的电路翻转,从而在不损失并行计算性能的基础上降低了系统的动态功耗。

3.2.2 DSP计算方式优化

为充分挖掘片上DSP资源的计算潜力,本设计利用DSP48E2输入端口27 bits位宽优势,采用了基于输入打包的单指令多数据流计算架构,具体表现为用一条乘法指令控制两次乘法运算。在卷积神经

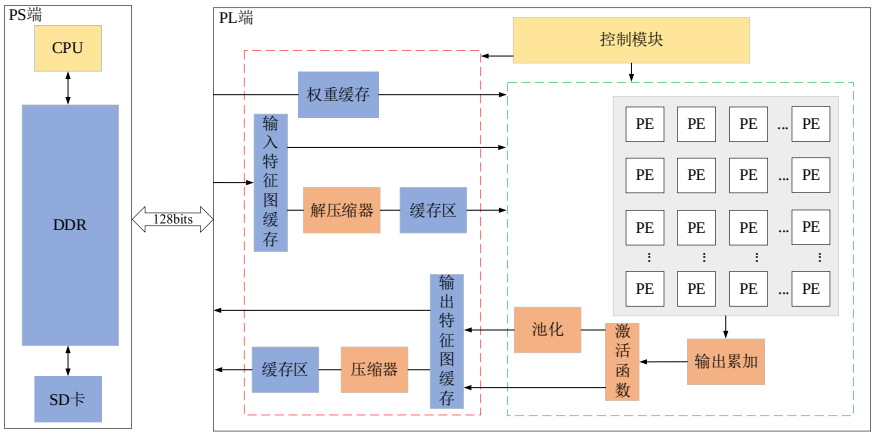


图 12　卷积神经网络加速器架构

Figure 12　CNN accelerator architecture

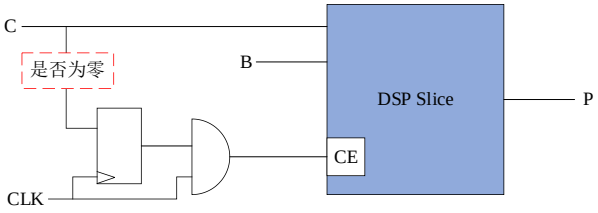


图 13　门控 DSP 架构

Figure 13　Gate-controlled DSP architecture

网络的推理过程中,存在同一个特征数据与不同权重相乘,也存在同一权重与不同特征数据相乘的情况。结合前面提到的门控电路设计,本文采用共享特征数据策略。通过将两组 8 bits 权重预先打包至 C 端口的高低位段,并与输入端口 B 的公共特征数据进行乘法运算,实现了在单个 DSP 周期内同时完成两路卷积乘法的操作,具体加速方式如图 14 所示。

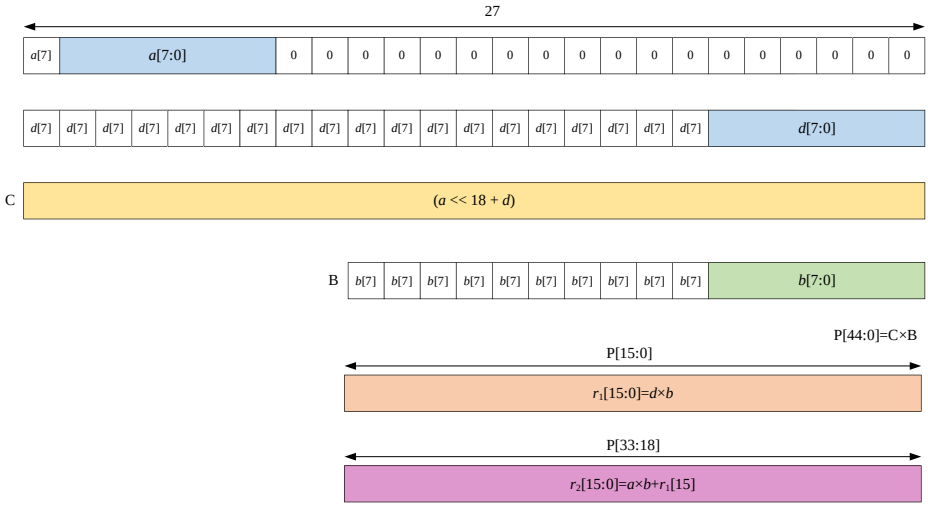


图 14　DSP 计算方式优化

Figure 14　Optimization of DSP calculation method

将两个 INT8 权重表示为 $a[7:0]$ 和 $d[7:0]$,公共特征数据表示为 $b[7:0]$。在计算 $a\times b$ 和 $d\times b$ 时,先将 a 和 d 的位宽扩展到 27 bits,b 的位宽扩展到 18 bits,具体扩展方式为,a 进行符号位填充变成 9 位,同时低 18 位全部填充零,d 和 b 分别进行符号位扩展变成 27 位和 18 位。之后将扩展后的 a 和 d 相加送入 C 端口,b 送入 B 端口,DSP 即可完成乘法计算。乘法计算后的结果送到输出端口 P,将 P[15:0] 记为 $r_1[15:0]$,P[33:18] 记为 $r_2[15:0]$,则前者保存的是 $d\times b$ 的值,后者保存的是 $a\times b$ 与 $r_1[15]$ 相加的值。若 r_1 是正数,正数的符号位为 0,那么 r_2 就是 $a\times b$ 的计算结果。若 r_1 是负数,负数的符号位为 1,那么 r_2 就是 $a\times b-1$ 的计算结果。结合这两种情况,$a\times b$ 的结果就为 $r_2[15:0]+r_1[15]$。

3.2.3　并行计算与数据复用

本设计的计算阵列由 8×72 个 PE 单元构成,每个

PE单元负责并行计算输出特征图中连续两个像素点的数据。为充分发挥压缩策略的带宽优势，加速器采用权重固定的数据流架构，即卷积核权重被预加载并暂存于PE单元内部，在卷积窗口滑动计算期间保持复用，而层间特征数据则在阵列中持续流动。图15展示了PE阵列的数据输入模式。图中F(i,j)表示特征图卷积窗口中的数据位置，W(i,j)表示卷积核在不同输入通道上的权重位置，Ci_m与Co_n分别代表第m个输入通道和第n个输出通道。在运算过程中，相同位置的两个权重数据与对应的特征图数据被传入PE单元，由单元内的DSP执行两次INT8乘加运算。在阵列排布上，横向每9个PE单元划分为一组，对应一个3×3的卷积窗口。每一组可完成2个输出通道的计算，因此单行72个PE单元即可实现16个输出通道的并行运算。整个阵列共8行，从而实现了8个输入通道与16个输出通道的混合并行运算。

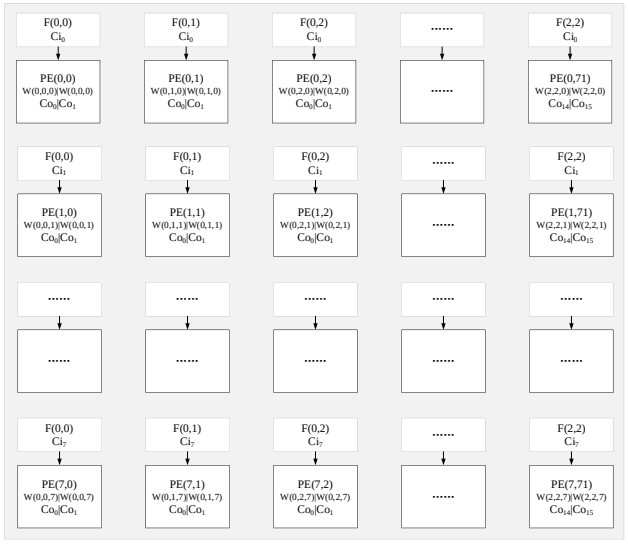


图15 PE阵列

Figure 15 PE array

4 实验与结果分析

4.1 实验环境与模型部署

关于实验环境的构建与加速器验证，本文遵循软硬件协同设计的开发流程。在模型准备阶段，基于PyTorch深度学习框架构建并训练卷积神经网络模型，随后完成INT8量化及推理精度验证。在硬件系统实现阶段，利用Vivado开发环境完成PL端的RTL编码、逻辑综合及布局布线；同时，利用Vitis集成开发平台完成PS端驱动程序与调度逻辑的开发。实验涉及的核心软件工具链及其版本配置汇总于表3。

本文选定ALINX AXU5EV开发板作为硬件验证载体，其核心芯片为Xilinx Zynq UltraScale+ MPSoC EV系列(XCZU5EV)。该平台集成了高性能的ARM Cortex-A53处理器与丰富的可编程逻辑资源，确保了VGG16层间大规模数据的实时缓存与压缩。其硬件实物及实验平台如图16所示。

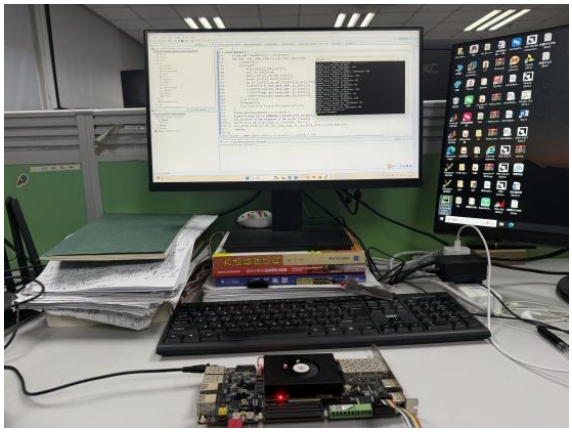
表3 实验开发工具

Table 3 Experimental development tools

工具	型号
操作系统	Windows11
PyTorch	1.9.1
Python	3.10.1
Vivado	2022.2
Xilinx Vitis	2022.2

(a) AXU5EV开发板

(a) AXU5EV Development board



(b) 实验平台

(b) Experiment platform

图16 实验环境

Figure 16 Experimental environment

图17展示了本文设计的卷积神经网络加速器在异构平台上的部署流程。该流程主要分为三个阶段，第一个阶段为离线参数提取，利用Python和PyTorch框架对预训练的VGG16模型执行INT8量化和参数提取。在此阶段，不仅提取模型权重与偏置，还需确定

熵编码映射表和各卷积层的最优基准值。熵编码映射表后续固化于硬件压缩模块中,而模型参数及层相关的动态配置信息则存储于SD卡中,作为系统的离线输入数据。第二个阶段为硬件架构实现,根据第三节介绍的卷积神经网络加速器设计架构,采用Verilog语言实现核心算子与压缩逻辑。在Vivado开发环境下完成加速器IP核的封装,并通过Block Design工具构建包含AXI4总线、DDR控制器及加速器IP的硬件系统。经过综合与布局布线,生成用于配置PL端的比特流文件(BIT流文件),并导出硬件描述文件(XSA文件)。第三个阶段为驱动与逻辑控制,基于XSA文件,在Xilinx Vitis环境下利用C语言编写控制程序。该程序负责硬件IP初始化、内存映射配置及数据调度流程,编译生成PS端的可执行文件(ELF文件)。在VGG16的部署中,控制程序(ELF)通过循环调度,逐层更新权重参数与IP核中的配置信息,从而实现了深层网络在有限硬件资源下的串行流水计算。最后,系统通过加载BIT流文件与运行ELF文件,实时读取SD卡中的VGG16权重与图像数据,在FPGA开发板上完成端到端的推理验证与性能评估。

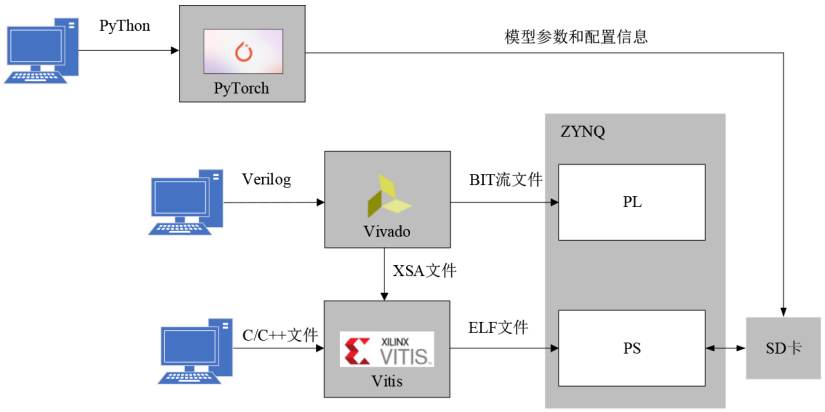


图17 加速器部署流程

Figure 17 Accelerator deployment process

4.2 参数选择

由于不同卷积神经网络在结构设计和特征分布上具有显著差异,本文选取了多种具有代表性的CNN模型以全面评估所提出压缩方法的适用性与鲁棒性。AlexNet[25]作为早期的标准卷积网络,参数冗余度较高,其层间特征图通常呈现出较强的稀疏性特征,适合用于验证压缩方法在高冗余场景下的有效性;VGG16具有大量特征通道,生成的特征图宽度较大,能够充分体现所提方法在大规模特征图上的压缩性能;ResNet34[26]通过引入跳跃连接改变了特征图的数值分布特性,可用于验证所提方法对残差结构的适应能力;而MobileNetV2[27]采用深度可分离卷积和倒残差结构,其特征图分布与传统卷积网络存在显著差异。因此,本文选用AlexNet、VGG16、ResNet34以及MobileNetV2作为基准模型,对压缩方法进行综合性能评估。

在实验过程中,选用ImageNet Large Scale Visual Recognition Challenge[28](ILSVRC)2012验证集作为输入数据集,并从中随机选取1 000张图像用于特征图的统计分析。所有CNN模型均采用PyTorch框架中提供的预训练权重。在模型前向传播过程中,本文从ReLU激活函数的输出端提取各层特征图,用于后续的统计建模与压缩实验。此外,本文采用xEyF的形式表示浮点数格式,其中x表示指数位数,y表示尾数位数。具体而言,FP32格式采用8E23F表示,而FP16格式采用5E10F表示。

由于不同数据格式下的基准-增量压缩差值存储位宽会影响压缩算法的效果,因此,表4、表5和表6分别统计分析了三种格式下不同存储位宽的压缩性能。根据统计结果,可以发现在FP32格式下,AlexNet和VGG16的差值存储位宽取3 bits压缩效果最好,ResNet34和MobileNetV2取2 bits压缩效果最好。在FP16和INT8格式下,四个网络的增量压缩差值存储位宽取2 bits压缩效果最好。

4.3 压缩性能评估

图18展示了不同浮点格式下各神经网络层间数

表4 FP32格式下差值存储位宽对压缩率的影响

Table 4 The influence of the bit width of difference storage in FP32 formats on compression rate

CNN	指数位存储长度			
	1	2	3	4
AlexNet	2.703	2.863	**2.920**	2.873
VGG16	2.309	2.426	**2.440**	2.368
ResNet34	2.006	**2.104**	2.092	2.027
MobileNetV2	1.714	**1.784**	1.783	1.729

表5 FP16格式下差值存储位宽对压缩率的影响

Table 5 The influence of the bit width of difference storage in FP16 formats on compression rate

CNN	指数位存储长度		
	1	2	3
AlexNet	2.579	**2.703**	2.689
VGG16	2.229	**2.318**	2.268
ResNet34	1.955	**2.026**	1.957
MobileNetV2	1.680	**1.727**	1.677

表6 INT8格式下差值存储位宽对压缩率的影响

Table 6 The influence of the bit width of difference storage in INT8 formats on compression rate

CNN	存储长度			
	1	2	3	4
AlexNet	5.086	**5.391**	5.034	5.012
VGG16	4.350	**4.568**	4.198	4.076
ResNet34	3.786	**3.972**	3.948	3.826
MobileNetV2	3.448	**3.589**	3.556	3.484

据的压缩率。与Zero-RLE及文献[19]方法相比，本文方法将压缩率分别提升了13.5%~36.6%和15.2%~36.7%。这主要是由于本方法不仅对非零值进行压缩，还通过熵编码增强零值压缩，从而获得更高的整体压缩效率。相较于同样压缩零值与非零值的文献[24]和EBPC方法，本文算法进一步将压缩率提高了1.6%~16.7%与0.2%~20.8%。由于本文对零值采用熵编码以减少编码冗余，并对非零值引入动态基准-增量编码机制，因此相比已有方法取得了更好的压缩效果。

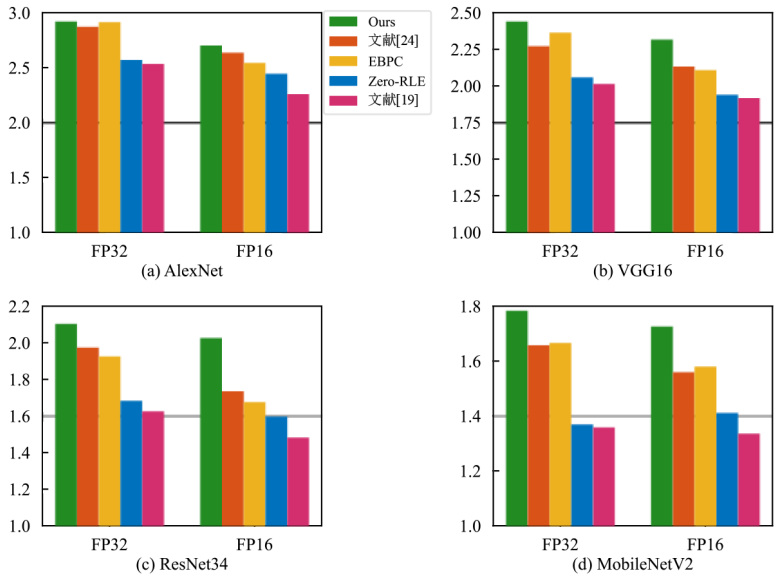


图18 主流CNN的FP32 fmap和FP16 fmap压缩率对比

Figure 18 Comparison of compression ratios of FP32 fmap and FP16 fmap in mainstream CNNs

除了支持常见的浮点格式外，本文提出的压缩算法可以推广到INT8格式下的层间特征图，适用范围更广；而EBPC只支持浮点格式和定点数格式，文献[24]仅支持浮点格式。如图19所示，在INT8格式下，本文的算法与Zero-RLE和文献[19]相比分别提高了10.5%~36.7%和32.1%~58.5%。

图20展示了本文所提出算法在零值数据压缩率方面与Zero-RLE以及文献[19]方法的对比结果。现有零值压缩方案大多采用Zero-RLE或文献[19]中的压缩策略，在对比过程中，这些方法均未引入前缀码区分机制，仅对零值数据进行独立编码。实验结果表明，相较于Zero-RLE，本文算法的零值压缩率提高了8.2%~21.1%；相较于文献[19]，零值压缩率提高了21.3%~40.5%。

图21对比了本文方法与其他非零值压缩方法的压缩率。相较于文献[24]，本文方法取得了0.6%~16.6%的提升，这主要得益于所采用的动态基准-增量编码机制，有效提高了非零值的压缩效率。在与EBPC的对比中，本文方法在AlexNet FP32下略低3.3%。这是由于EBPC依赖于数据相关性，在数据分布集中时表现更优。然而，在其他网络与数据格式下，本文方法的性能均接近或优于EBPC，体现出更稳健的综合压缩能力。

4.4 硬件开销分析

本文基于TSMC 40 nm工艺，使用Synopsys Design Compiler对所提出的加速器进行了综合。表7对比了本文方法、EBPC及文献[24]在16位与32位数据位宽

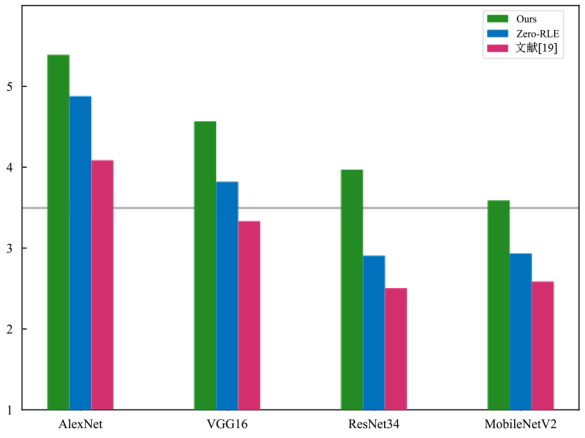


图19 主流CNN的INT8压缩率对比

Figure 19 Comparison of INT8 fmap compression ratios of mainstream CNNs

下的面积与门计数。当特征图(fmap)位宽从16 bits增至32 bits时,本文压缩器与解压缩器的面积分别增加69.4%与75.6%。该增长主要源于非零数据存储与计算单元的面积扩展,而零值处理部分保持不变。与EBPC相比,在16 bits下,本文方法与文献[24]的压缩器面积分别减少58.7%与63.4%,解压缩器减少63.8%与68.8%;在32位下,对应降幅分别为62.1%与65.8%(压缩器)以及66.5%与72.9%(解压缩器)。本文面积略高于文献[24],原因在于本设计将数据流压缩为两个独立流进行存储,其打包与解包模块需分别设计,且解压缩器需额外逻辑实现双流同步。尽管如此,本文在压缩率上优于文献[24],并支持INT8格式的特征图压缩,在面积与压缩性能之间取得了良好平衡。此外,得益于对零值的进一步压缩,本文方法在稀疏网络上具有更好的适用性。

4.5 系统性能与能效提升

本文所设计的加速器有8×72个PE单元,每个PE单元可以计算两次乘累加操作,一次乘累加过程包含一次乘法和一次加法,一共四次运算,设定加速器的工作频率为200 MHz,则加速器的理论峰值吞吐量为8×72×4×200 000 000=460.8 GOPS。表8展示了在相同架构下应用压缩方法和没有应用的加速器性能对比。可以看到应用压缩方法后的加速器吞吐量和计算密度提升了41.39%,更加接近加速器的理论峰值算力,这是因为压缩之后每次片上与片外数据交互时传输的数据量增加,使得PE阵列的计算能力得到了释放。应用压缩方法后的加速器虽然功耗增加了12.43%,但是能效比提升了26%,这表明应用压缩方法带来的性能收益远高于其带来的功耗开销。

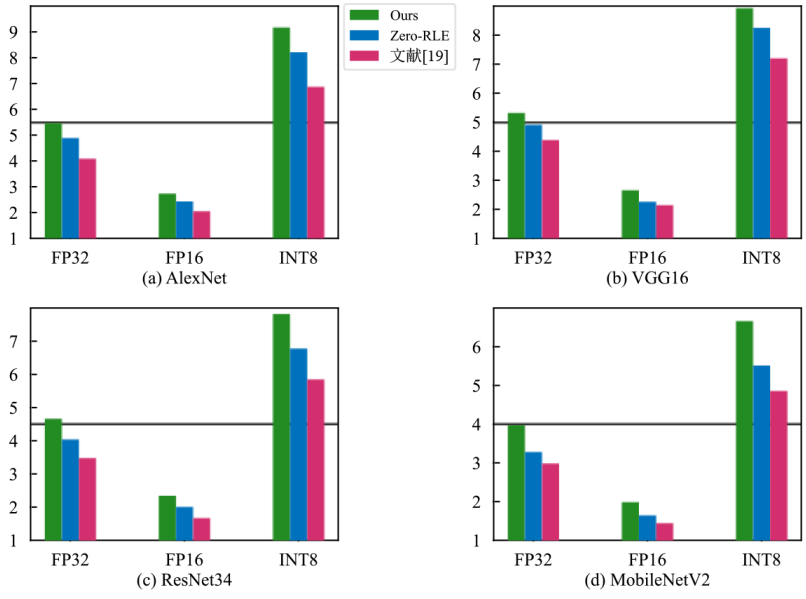


图20 主流CNN的fmap零值压缩率对比

Figure 20 Comparison of zero value compression ratios for the mainstream CNNs

表9将本文的加速器与GPU和ASIC架构下的CNN加速器进行对比。对比结果显示,在与GPU对比时,本文所提出加速器的吞吐量是GPU的6.12倍,能效比是GPU的119倍。性能提升的核心原因在于,本文的加速器使用了高效的层间特征图压缩方法,减少了数据的传输时间,并且加速器通过优化DSP计算方式提高了吞吐量。在ASIC架构下,与文献[9]相比,尽管随机计算架构通过极简的逻辑门设计实现了极低的静态功耗,但其计算性能高度依赖于位流长度,且在处理VGG16等深度网络时面临精度损失和

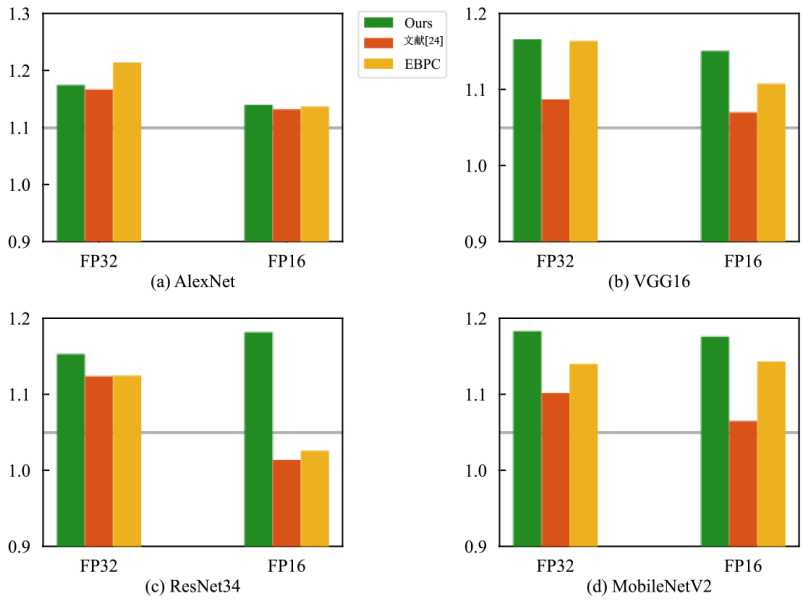


图21 主流CNN的fmap非零值压缩率对比

Figure 21 Comparison of non-zero value compression ratios for the mainstream CNNs

表7 压缩架构面积对比

Table 7 Comparison of compression architectural area

方法	Fmaps Bits	压缩器		解压缩器	
		面积/μm^2	门数量/个	面积/μm^2	门数量/个
Ours	16	1 389	1 969	1 416	2 007
	32	2 354	3 336	2 487	3 525
EBPC	16	6 880	4 778	7 986	5 546
	32	12 611	8 792	15 160	10 528
文献[24]	16	8 916	1 751	8 803	1 729
	32	15 300	3 005	14 530	2 853

注:面积转换为两输入与非门的数量。1Gate: 0.705 6 μm^2(TSMC 40 nm);1Gate:1.44 μm^2(UMC 65 nm);1Gate:5.092 2 μm^2(TSMC 130 nm)。

推理延迟较大的挑战。相比之下,本文方案在维持标准二进制计算精度的基础上,通过层间数据压缩从根本上缓解了边缘计算中的“访存墙”难题。与最先进的ASIC定制化方案文献[10]相比,尽管专用集成电路凭借物理层面的全固化电路设计在能效比指标上具有先天优势,但本文方案展现了另一种高效优化路径。通过统计感知的层间数据压缩逻辑,本文在通用的FPGA逻辑资源上实现了对访存带宽瓶颈的高效突破。由于ASIC架构在流片后功能无法更改,且流片费用高、迭代周期长,难以应对算法的快速迭代需求;而本文充分利用FPGA的硬件可重构特性,实现了针对特征图统计分布特性的动态压缩逻辑。这种“硬逻辑、软配置”的架构不仅提升了单位面积的算力密度,还确保了对不同深度神经网络的动态适配能力,在维持高能效的同时,极大地降低了硬件的部署及迭代成本。

表8 应用压缩算法前后性能对比

Table 8 Comparison before and after applying compression algorithm performance

方法	DSP使用数/个	吞吐量/GOPS	计算密度/(GOPS/DSP)	功耗/W	能效比/(GOPS/W)
无压缩	592	171.74	0.29	3.84	44.72
压缩	592	242.82	0.41	4.25	57.13

表9 与GPU和ASIC性能对比

Table 9 Comparison with GPU and ASIC performance

平台	GPU	ASIC		FPGA
架构	NVIDIA RTX5000[29]	文献[9]	文献[10]	本文
工作频率/MHz	1 350	200	800	200
吞吐量/GOPS	39.67	220	371.6	243
功耗/W	83.2	0.651	0.045 5	4.254
能效比/(GOPS/W)	0.48	340	8 170	57.13

表10为本文设计的加速器与其他基于FPGA的加速器的性能进行对比。为消除硬件平台差异的影响,本文选用计算密度与能效比作为核心评估指标。与文献[30]相比,其采用的切片流水线卷积技术虽降低了DRAM访问,但乘法器利用率较低,导致计算密度仅为本文的61%,能效比仅为本文的83%。文献[17]通过负载均衡与数据调度提升了PE利用率,但为追求性能使用了大量DSP资源,致使其计算密度仅为本文的71%;尽管能效比因工艺优化略高,但其硬件资源效率不及本文通过PE微架构优化所实现的性能-资源比。文献[31]采用多计算引擎并优化数据重排与层间传输,提高了计算单元利用率,但未对层间特征图进行压缩,在存储受限场景下整体效率受限,其计算密度与能效比分别为本文的89%和76%。文献[32]针对稀疏数据提出脉动阵列加速方法,但其压缩算法性能较弱且控制逻辑复杂,能效比仅为本文的65%。综上所述,本文加速器通过优化DSP计算方式、高效的层间数据压缩及数据流复用策略,在计算密度与能效比方面均优于对比方案,在资源受限条件下实现了更优的综合性能。

表10 加速器性能对比

Table 10 Comparison of accelerator performance

加速器架构	文献[30]	文献[17]	文献[31]	文献[32]	本文
硬件平台	ZCU102	VX980T	VX690T	ZCU102	ZU5EV
工作频率/MHz	250	150	180	200	200
数据精度/bit	8	8	8	8	8
网络模型	VGG16	VGG16	VGG16	VGG16	VGG16
DSP使用数	1 015	3 395	2 115	1 061	592
吞吐量/GOPS	256	1 000	753.2	410	243
计算密度/(GOPS/DSP)	0.25	0.29	0.36	0.39	0.41
功耗/W	5.38	14.36	17.41	11	4.254
能效比/(GOPS/W)	47.56	69.64	43.26	37.24	57.13

5 结论

为高效利用卷积神经网络层间特征图的高稀疏性,并缓解片上与片外存储器间频繁数据交互带来的带宽压力,本文深入剖析了特征图的数据分布规律与稀疏特性,提出了一种面向层间特征图的统计感知混合压缩架构。该方法采用了差异化编码机制:针对零值数据,利用融合熵编码的增强型零游程算法以最大化压缩增益;针对非零数据,则设计了动态基准-增量编码以利用局部数据相关性。实验验证表明,该方案在保证高压缩率的同时,极大地降低了编解码硬件模块的面积开销,实现了压缩性能与硬件资源消耗之间的最佳折中。相比于同架构的基准设计,集成该压缩策略的加速器在吞吐量和能效比上均实现了显著提升,充分验证了该方法的有效性与优越性。

参考文献

[1] Chen Junliang. CNN or RNN: Review and experimental comparison on image classification[C]//2022 IEEE 8th International Conference on Computer and Communications. Piscataway: IEEE, 2022: 1939-1944.

[2] Bolme D S, Beveridge J R, Draper B A, et al. Visual object tracking using adaptive correlation filters[C]//2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2010: 2544-2550.

[3] Wang Jin, Yu L C, Lai K R, et al. Tree-structured regional CNN-LSTM model for dimensional sentiment analysis[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2020, 28: 581-591.

[4] Wu Di, Fan Xitian, Cao Wei, et al. SWM: A high-performance sparse-Winograd matrix multiplication CNN accelerator[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2021, 29(5): 936-949.

[5] Mittal S. A survey on optimized implementation of deep learning models on the NVIDIA Jetson platform[J]. Journal of Systems Architecture, 2019, 97: 428-442.

[6] Chen Y H, Yang T J, Emer J, et al. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2019, 9(2): 292-308.

[7] 欧阳一鸣, 王奇, 汤飞扬, 等. MRNDA: 一种基于资源受限片上网络的深度神经网络加速器组播机制研究[J]. 电子学报, 2024, 52(3): 872-884.
Ouyang Yiming, Wang Qi, Tang Feiyang, et al. MRNDA: A multicast mechanism for resource-constrained NoC-based deep neural network accelerators[J]. Acta Electronica Sinica, 2024, 52(3): 872-884. (in Chinese)

[8] Lee S S, Nguyen T D, Meher P K, et al. Energy-efficient

high-speed ASIC implementation of convolutional neural network using novel reduced critical-path design[J]. IEEE Access, 2022, 10: 34032-34045.

[9] Frasser C F, Linares-Serrano P, de los Ríos I D, et al. Fully parallel stochastic computing hardware implementation of convolutional neural networks for edge computing applications[J]. IEEE Transactions on Neural Networks and Learning Systems, 2023, 34(12): 10408-10418.

[10] Li Jixuan, Li Ke, Un K F, et al. An 800-MHz 8.17-TOPS/W 0.63-TOPS/mm^2 memory-utilization-aware CNN accelerator featuring a memory stationary dataflow[J]. IEEE Journal of Solid-State Circuits, 2025, 60(8): 3033-3042.

[11] Xiong Feng, Tu Fengbin, Shi Man, et al. STC: Significance-aware transform-based codec framework for external memory access reduction[C]//2020 57th ACM/IEEE Design Automation Conference. Piscataway: IEEE, 2020: 1-6.

[12] Yuan Tian, Liu Weiqiang, Han Jie, et al. High performance CNN accelerators based on hardware and algorithm co-optimization[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2021, 68(1): 250-263.

[13] Jiang Weixiong, Yu Heng, Yajun Ha. A high-throughput full-dataflow MobileNetv2 accelerator on edge FPGA[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023, 42(5): 1532-1545.

[14] Wu Bi, Yu Tianyang, Chen Ke, et al. Edge-side fine-grained sparse CNN accelerator with efficient dynamic pruning scheme[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2024, 71(3): 1285-1298.

[15] Yang Chen, Meng Yishuo, Huo Kaibo, et al. A sparse CNN accelerator for eliminating redundant computations in intra- and inter-convolutional/pooling layers[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2022, 30(12): 1902-1915.

[16] Chen Y H, Krishna T, Emer J S, et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks[J]. IEEE Journal of Solid-State Circuits, 2017, 52(1): 127-138.

[17] Huang Wenjin, Wu Huangtao, Chen Qingkun, et al. FPGA-based high-throughput CNN hardware accelerator with high computing resource utilization ratio[J]. IEEE Transactions on Neural Networks and Learning Systems, 2022, 33(8): 4069-4083.

[18] Parashar A, Rhu M, Mukkara A, et al. SCNN: An accelerator for compressed-sparse convolutional neural networks[C]//2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture. Piscataway: IEEE, 2017: 27-40.

[19] Aimar A, Mostafa H, Calabrese E, et al. NullHop: A flexible convolutional neural network accelerator based on sparse representations of feature maps[J]. IEEE Transactions on Neural Networks and Learning Systems, 2019, 30(3): 644-656.

[20] Albericio J, Judd P, Hetherington T, et al. Cnvlutin: Ineffectual-neuron-free deep neural network computing[C]//2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture. Piscataway: IEEE, 2016: 1-13.

[21] Han Song, Liu Xingyu, Mao Huizi, et al. EIE: Efficient inference engine on compressed deep neural network[J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 243-254.

[22] Chen Yuechen, Louri A, Liu Shanshan, et al. A balanced sparse matrix convolution accelerator for efficient CNN training[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2024, 71(10): 4638-4651.

[23] Cavigelli L, Rutishauser G, Benini L. EBPC: Extended bit-plane compression for deep neural network inference and training accelerators[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2019, 9(4): 723-734.

[24] Yan B K, Ruan S J. Area efficient compression for floating-point feature maps in convolutional neural network accelerators[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2023, 70(2): 746-750.

[25] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84-90.

[26] He Kaiming, Zhang Xiangyu, Ren Shaoqing, et al. Deep residual learning for image recognition[C]//2016 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2016: 770-778.

[27] Sandler M, Howard A, Zhu Menglong, et al. MobileNetV2: Inverted residuals and linear bottlenecks[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 4510-4520.

[28] Russakovsky O, Deng Jia, Su Hao, et al. ImageNet large scale visual recognition challenge[J]. International Journal of Computer Vision, 2015, 115(3): 211-252.

[29] 龚贵川, 谢良波, 黄倩, 等. 基于ZYNQ的高效卷积神经网络加速器设计[J]. 电讯技术, 2026, 66(2): 259-266.
Gong Guichuan, Xie Liangbo, Huang Qian, et al. Design of an efficient convolutional neural network accelerator

based on ZYNQ[J]. Telecommunication Engineering, 2026, 66(2): 259-266. (in Chinese)

[30] Kuo J T, Wu C B, Chen Yiyuan. Implementation of a tile-grained pipeline architecture for CNN accelerator[C]// 2023 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia). Piscataway: IEEE, 2023: 1-4.

[31] Foroumandi R, Mashoufi B, Fathi A. High-efficiency FPGA-based CNN accelerator with optimized data handling for convolution and fully connected layers[J]. IEEE Access, 2025, 13: 211235-211250.

[32] Sun Wenhao, Liu Deng, Zou Zhiwei, et al. Sense: Model-hardware codesign for accelerating sparse CNNs on systolic arrays[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2023, 31(4): 470-483.

作者简介

谢良波 男,1986年1月出生于四川省成都市。现为重庆邮电大学副教授、硕士生导师。主要研究方向为无线感知、混合信号集成电路、FPGA信号处理及神经网络加速器。

Email: xielb@cqupt.edu.cn

周 牧 男,1984年2月出生于四川省自贡市。现为重庆邮电大学教授、博士生导师。主要研究方向为量子精密测量、无线定位与感知技术等。

E-mail: zhoumu@cqupt.edu.cn

陈 林 男,2001年8月出生于四川省宜宾市。现为重庆邮电大学硕士研究生。主要研究方向为FPGA神经网络加速器。

E-mail: s230131005@stu.cqupt.edu.cn

卜文杰 男,1999年11月出生于四川省广元市。现为重庆邮电大学硕士研究生。主要研究方向为FPGA神经网络加速器。

E-mail: s240132223@stu.cqupt.edu.cn