DOI: 10.13382/j.jemi.B2508595

面向 FPGA 的 CNN 软硬协同加速方法研究综述

郭楚亮[1] 娄 越[1,2] 岳豪帅[1,2] 季 拓[1,2] 程谘哲[1,2] 彭 宇[1]
(1. 哈尔滨工业大学电子与信息工程学院 哈尔滨 150001;2. 哈尔滨工业大学郑州高等研究院 郑州 450000)

摘 要:卷积神经网络(convolutional neural networks, CNN)凭借卓越的视觉特征提取能力,在深度学习领域获得了广泛应用。然而,随着网络复杂度提升及边缘计算场景对计算性能、能源效率的严苛需求,硬件加速面临巨大挑战。在低批量推理、需确定性低延迟以及硬件难以更新的特定应用场景中,通用中央处理器(central processing unit, CPU)和图形处理器(graphics processing unit, GPU)往往难以满足需求。在此背景下,基于现场可编程门阵列(field programmable gate arrays, FPGA)的 CPU-FPGA 异构计算平台,凭借其定制计算数据流、可重构性,以及确定性低延迟等优势,为实现边缘端低批量、低延迟的 CNN 推理任务提供了有效路径。从软硬件协同设计视角,系统性地探讨面向 CPU-FPGA 平台部署的 CNN 轻量化与计算加速方法:首先介绍降低网络复杂度的网络轻量化技术;其次介绍计算单元、计算阵列以及数据访存的层次化设计硬件优化策略;然后介绍基于高层次综合(high-level synthesis, HLS)的敏捷开发流程如何赋能加速器系统的快速迭代与验证。最后,从复杂芯片前端验证、确定性低延迟计算以及算法敏捷迭代 3 个维度,展望 FPGA 在人工智能新时代的技术定位、发展前景与挑战。

关键词: 现场可编程门阵列;卷积神经网络;深度学习;敏捷开发;高层次综合;神经网络轻量化
中图分类号: TN791 **文献标识码**: A **国家标准学科分类代码**: 520.4020

Survey of hardware-software co-design acceleration for CNNs on FPGAs

Guo Chuliang[1] Lou Yue[1,2] Yue Haoshuai[1,2] Ji Tuo[1,2] Cheng Xuzhe[1,2] Peng Yu[1]
(1. School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China;
2. Zhengzhou Advanced Research Institute, Harbin Institute of Technology, Zhengzhou 450000, China)

Abstract: Convolutional neural networks (CNN) have gained widespread application in the field of deep learning due to their exceptional visual feature extraction capabilities. However, with increasing model complexity and the stringent demands of edge computing scenarios for computational performance and energy efficiency, hardware acceleration faces significant challenges. In specific application scenarios requiring low-batch inference, deterministic low latency, and difficulty in updating hardware, general-purpose central processing units (CPU) and graphics processing units (GPU) often fall short of the requirements. In this context, CPU-FPGA heterogeneous computing platforms based on field programmable gate arrays (FPGA) offer an effective pathway for implementing low-batch, low-latency CNN inference tasks at the edge. Their advantages lie in customized computational dataflow, reconfigurability, and deterministic low latency. This paper systematically explores CNN compression and computational acceleration methods for deployment on CPU-FPGA platforms from a hardware-software co-design perspective. First, we introduce network compression techniques for reducing model complexity. Second, we present hierarchical hardware optimization strategies for computational units, computational arrays, and data memory access. Third, we describe how an agile development flow based on high-level synthesis (HLS) enables rapid iteration and verification of accelerator systems. Finally, from the three dimensions of complex chip front-end verification, deterministic low-latency computing, and agile algorithm iteration, we explore the technological positioning, development prospects, and challenges of FPGAs in the new era of Artificial Intelligence.

Keywords: FPGA; CNN; deep learning; agile development; HLS; neural network compression

收稿日期: 2025-07-24 Received Date: 2025-07-24

0 引 言

深度学习是基于深度神经网络的机器学习方法,卷积神经网络(convolutional neural networks, CNN)是深度学习中最具代表性的前馈视觉网络之一,其卷积、非线性激活与池化等结构性约束带来强归纳偏置,使有效的视觉表征在较少训练甚至无预训练条件下也可涌现[1]。与支持向量机等浅层机器学习方法相比,CNN 在高维数据特征提取中具有显著优势,从而在图像分类[2]、目标检测[3]、语义分割[4]等图像处理任务中得到广泛应用。然而,随着算法复杂度、数据量、网络尺寸和网络深度不断增长,硬件算力成为支撑 CNN 训练与推理的关键:在通用高性能服务器端和计算集群中,训练和推理通常依赖中央处理器(central processing unit, CPU)的数据预处理和任务调度控制,以及图形处理器(graphics processing unit, GPU)的密集通用矩阵乘法计算加速。尤其在大批量(batch size)场景下,CPU 的通用性和 GPU 的单指令流多数据流(single instruction multiple data, SIMD)架构的并行计算能力,可显著提高 CNN 训练和推理的吞吐性能。

然而,在诸如实时图像采集与处理、自动驾驶感知、工业机器人控制等低批量、低延迟的边缘端应用场景中,常伴随着传感器实时采集流式数据(即批量大小为 1)[5]及网络计算负载具有数据依赖性[6]。面对这些挑战,经典的 CPU 和 GPU 架构因其固定的数据流模式和高昂的调度开销导致在低批量推理任务中计算单元利用率低下,从而引发高延迟和低能效的问题。尽管专用集成电路(application specific integrated circuit, ASIC)能提供极致的能效与吞吐量,但其高昂的开发成本与固化的电路结构难以适应算法的快速迭代。相对地,现场可编程门阵列(field programmable gate array, FPGA)则借助大规模的可重构逻辑资源和定制化的数据通路设计能力,允许根据神经网络的特定结构和计算特性来构建高度定制化的硬件加速器,从而在低延迟、高能效场景下展现出显著优势。同时,FPGA 的可重构性使其能够灵活适应算法快速演进的神经网络,对多网络、低批量应用的边缘部署需求,可快速进行原型验证和设计迭代[7]。

尽管近年来已有研究对 FPGA 平台上的 CNN 计算加速方法进行了综述,但或侧重于通用性能评估框架[8],或聚焦于底层硬件实现技术[9-10],仍缺乏从软硬件协同设计视角,系统性地探讨面向 CPU-FPGA 异构系统的软硬协同设计方法。软硬件协同设计是一种在早期阶段就将软件和硬件作为整体联合设计和优化的方法,以实现单独优化无法达到的系统级性能、能效和成本目标。为此,从算法、架构与实现层面的联合优化角度出发,聚焦典型且广泛的 CPU-FPGA 异构平台,系统性地综述:作为算法基础的网络轻量化技术;作为硬件核心的数字架构优化策略;连接算法与硬件的系统级协同设计方法;加速设计流程的敏捷开发范式。通过层次化分析,旨在为该领域的研究者和工程师提供一个清晰的路线图,展望未来发展方向。

1 背景

CNN 通过卷积、归一化、非线性激活函数、池化层和全连接层从各类数据中提取多层次抽象特征,其大量的算术计算与数据访存开销,与 FPGA 有限的片上逻辑、计算和存储资源形成了显著矛盾。分析算法特征与硬件资源的适配关系,明确计算或带宽的性能瓶颈,是开展后续网络轻量化与软硬协同优化的必要前提。

1.1 CNN 关键算子

卷积、归一化、非线性激活函数、池化,以及全连接是 CNN 中常见的 5 类关键算子,其结构参数配置由任务需求与硬件资源约束共同决定。

1)卷积

卷积层的乘累加(multiply-accumulate, MAC)操作可占据 CNN 总计算量的绝大部分,是推理延迟和计算功耗的主要来源[11]。由于计算机采用二进制数据存储,卷积需通过对输入信号与卷积核进行均匀采样,将连续形式转换为离散卷积。根据不同的任务语义和计算模式,分为标准卷积[12]、转置卷积[13]、填充卷积[14]、深度可分离卷积[15]、可变形卷积[16]等多种形式。CNN 中基础形式的标准卷积的计算过程如图 1 所示。

在不同网络架构中,卷积核尺寸的设置需权衡感受野需求与算力开销。较大卷积核可在浅层快速扩大感受野[12],提升对大尺度目标的建模能力与检测效果,但在计算与访存开销上代价更高,在资源受限条件下往往通过多层小卷积[17]等效替代。为进一步降低参数量与计算量,可采用深度可分离卷积等方式分离特征图与通道的计算依赖。相较标准 3×3 卷积,可减少超过 80%的计算量,从而实现高效实现特征提取[15]。

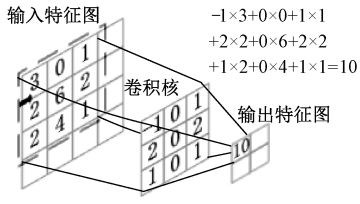


图 1 标准卷积计算示意图

Fig. 1 Schematic diagram of standard convolution calculation

2)归一化

CNN 反向传播训练阶段,不同的输入数据会使得中间特征图的内部协方差偏移,造成训练收敛速度缓慢。归一化操作在网络的训练和推理过程中,将数据变换成具有零均值和单位方差的数据,进而稳定收敛性。如图 2 所示,根据归一化维度不同,常见方法包括批量归一化[18]、层归一化[19]、实例归一化[20]、组归一化[21],以适用不同网络和任务场景。

归一化方式主要取决于网络结构、任务形式以及批量处理的片上存储需求。在内存充足的大批量训练中,批量归一化能够利用全局统计量加速收敛;而在面向边缘端小样本的场景下,批量归一化的统计误差会显著增加,此时通常切换为组归一化或层归一化,以确保网络预测稳定性[21]。归一化常可与前一层卷积进行参数折叠,从而减少额外算子与访存开销[22]。

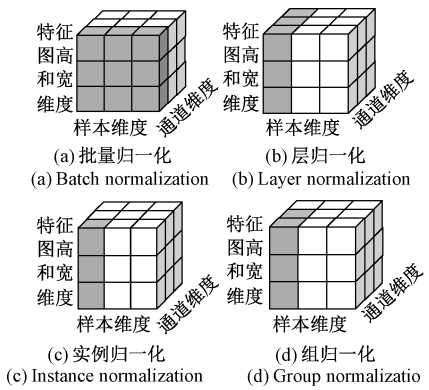


图 2　4 种归一化方法示意图[21]

Fig. 2　Schematic diagram of four normalization methods[21]

3)非线性激活函数

卷积层和全连接层均是线性算子,为提高深度网络的特征表达能力,必须为网络引入非线性。非线性激活函数通常作用于卷积层或归一化层的输出,并可在反向传播训练阶段有效地传递梯度。如图 3 所示,常用的激活函数包括 Sigmoid 函数[23]、Tanh 函数[23]、ReLU 函数[24]、Leaky ReLU 函数[25]。

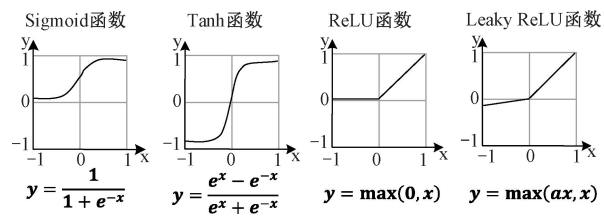


图 3　Sigmoid、Tanh、ReLU 和 Leaky ReLU 函数

Fig. 3　Sigmoid, Tanh, ReLU, and Leaky ReLU function

激活函数需考虑计算效率与任务精度的平衡。ReLU 及其变体因仅涉及线性截断而具有极高的计算效率,适用于资源受限的边缘设备与实时推理场景,能在较低计算开销下保持稳定的训练与收敛特性[12]。对于追求高任务精度的深层网络,常引入 Sigmoid、Tanh 等平滑激活函数,尽管增加了部分指数运算开销[26],但能有效提升特征提取的精细度与连续性,从而使网络在深层堆叠下仍保持较高的收敛质量与精度表现。

4)池化层

池化层通常在卷积层之后,用于下采样变换,缩减特征图尺寸。常用的池化方法包括最大池化[12]、平均池化[27],计算方法如图 4 所示。

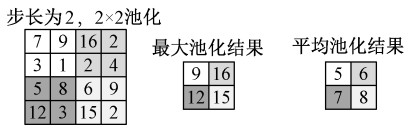


图 4　最大池化、平均池化示意图

Fig. 4　Schematic diagram of the maximum pooling and average pooling

池化窗口与步长决定下采样倍率,直接影响后续层计算量与中间特征存储规模。分类主干网络可利用池化快速降低分辨率以提升计算效率[28];而检测/分割等任务往往对小目标与边界细节更敏感,多以步长卷积替代池化,以降低空间信息损失[29]。

5)全连接层

全连接层是一种线性分类器,如图 5 所示,每个神经元与前一层所有的神经元相连,用于复杂的模式识别和概率分类任务。为缓解全连接层的过拟合以及丢失空间信息等问题,通常采用正则化[30]方法,在训练过程中随机将一部分神经元的输出设置为 0,有效地阻止神经元之间的协同适应,迫使网络学习更鲁棒的特征表示,减少过拟合。

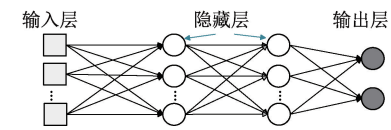


图 5　全连接层示意图

Fig. 5　Schematic diagram of the fully connected layer

全连接层通常具有较大参数量,并且相比于卷积层,呈现显著的访存密集特征。为降低参数存储与片外访问开销,许多现代 CNN 倾向于采用 1×1 卷积结合全局平均池化替代大规模全连接层,从而在资源受限与低时延推理场景获得更优的硬件复用和计算效率[27]。

1.2　CNN 算法特性与 FPGA 硬件加速挑战

分析 CNN 推理负载的计算特性与 FPGA 硬件资源约束,是硬件加速优化层次与优化方法的重要参考。

1)计算密集型与访存密集型特性

在计算形式角度方面,CNN 的推理过程基于高维张量变换。相较于经典图像处理算法,CNN 呈现出更为显著的计算密集与访存密集双重特征。

首先,卷积层的核心算子通常由六重独立的嵌套循环(涵盖批量、通道、行、列及滤波器等维度)构成,这种运算结构导致极高的算术密度。以 VGG 等典型卷积网络为例,其前向推理的运算量可达数十亿次乘加量级,且主要由多层卷积算子贡献[2],使卷积层成为推理阶段的主要性能瓶颈。

其次,大量数据的吞吐需求引发显著的访存压力。虽然卷积操作具有局部连接和权值共享特性,理论上存在极高的数据复用潜力,但在实际硬件映射中,权重参数与中间特征图的频繁搬运仍不可避免。特别地,全连接层虽为矩阵乘类算子,但在边缘实现中常受权重搬运带宽约束;而归一化、残差加法等逐元素操作算术密度更低,典型地表现为带宽受限。这种计算与访存特性的动态变化,给硬件架构的通用性设计带来了困难。

2)FPGA 资源构成及其物理约束

FPGA 硬件资源主要包括逻辑资源、存储资源、计算资源、互连与接口资源,以及时钟资源。其中逻辑资源包括查找表(look-up table, LUT)、触发器(flip-flop, FF)等,计算资源包括数字信号处理(digital signal processing, DSP)单元等,存储资源包括分布式随机存取存储器(distributed random access memory, DistRAM)、块随机存取存储器(block random access memory, BRAM)等。

DSP 是执行 MAC 操作的核心硬件单元,主要用于构建处理元素(processing element, PE)阵列;LUT 是组合逻辑单元,用于实现组合逻辑和时序控制;FF 是时序逻辑单元,用于时序逻辑和流水线设计;BRAM 等存储资源主要用于网络参数片上存储,以及作为特征图的局部缓冲区。FPGA 有限的 DSP、LUT、BRAM 资源和互连架构,以及相对受限的片外带宽等,共同构成了 CNN 加速器设计的核心硬件资源约束。

然而,随着深度神经网络复杂度的不断演进,FPGA 有限的片上资源难以满足其密集的计算与存储需求。受限于内存墙效应,仅依赖硬件规模的扩展将面临严峻的带宽受限与功耗挑战[26]。为解决上述物理约束,需从算法轻量化与硬件协同优化两个维度展开研究。

2 网络轻量化方法

尽管 FPGA 具备高度可定制的并行计算能力,但其片上存储资源和计算资源相较于 GPU 十分有限。VGG[31]、ResNet[2]、InceptionNet[27]等经典 CNN 包含数百万参数,其庞大的参数规模和计算量在硬件实现层面普遍面临内存墙(即计算单元的算力增长快于存储系统带宽与延迟改善,导致程序性能越来越受数据搬运与访存限制而非受计算本身限制)等系统瓶颈。在 FPGA 平台上,受限的片上资源使得大量中间特征和权重参数不得不频繁访问片外存储,而片外数据搬运不仅是系统能耗的主要来源[32-33],同时也是影响整体推理延迟和吞吐率的关键因素。因此,在进行硬件架构设计与加速优化之初,通过算法层面的网络轻量化来从源头上压缩网络尺寸和计算复杂度,是支撑 CNN 在 FPGA 平台上高效实现的重要基础。如图 6 所示,主流的网络轻量化方法主要包括剪枝/稀疏化、量化、低秩近似和知识蒸馏。

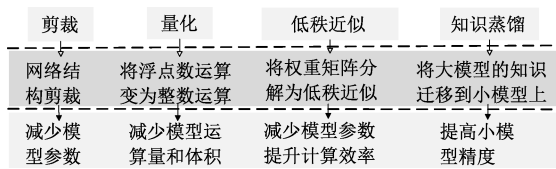


图 6 CNN 轻量化方法

Fig. 6 CNN lightweight method

2.1 剪枝

剪枝选择性移除神经网络神经元[34]、输入/输出通道[35-37]、滤波器[38]或卷积层,降低参数量,在不影响网络准确性同时,降低片外带宽和片上存储需求。

1)剪枝粒度

根据剪枝后的稀疏化子结构,可分为结构化和非结构化剪枝,如图 7 所示。非结构化剪枝以单个权重元素为单位进行剪枝,而结构化剪枝以相对紧凑、完整的网络子语义为单位进行剪枝。

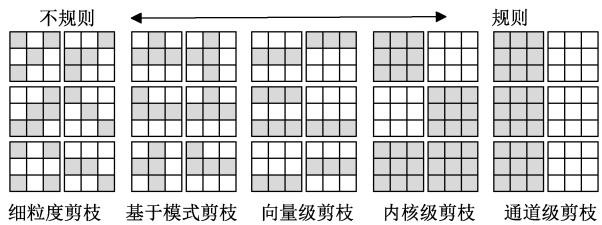


图 7 结构化剪枝与非结构化剪枝

Fig. 7 Structured pruning and unstructured pruning

非结构化剪枝[34,39-40]通常以幅值为标准,移除对任务准确率影响较低的权重元素,能够实现较高压缩率。但其产生的不规则稀疏模式对 GPU 等依赖规整数据结构的并行硬件极不友好,需要复杂的索引机制来跳过零值,这可能抵消部分计算加速带来的收益。与之相对,结构化剪枝[35,41-42]按照固定数据语义(卷积核、输入/输出

通道、卷积层、单位矩阵块等)产生的规整稀疏性,天然适配于并行计算架构,更易于在通用硬件上实现高效推理[43-44],但通常以牺牲一部分压缩率为代价。

2)剪枝策略

剪枝通过评估元素或网络语义结构对任务准确率的重要程度,利用网络的过度参数化特性,减少参数量。剪枝根据参数重要性度量方式的不同,可分为启发式剪枝[36,45]以及基于优化方法的剪枝[46-47]。

启发式剪枝通常依据经验法则,判断可移除的参数或子结构,常用的剪枝标准包括权重幅度[48]、激活值[49]和梯度信息[50]。其基本思想是,幅值较小的元素在乘累加部分和的贡献度较小,但未必与网络准确率变化最相关。因此,依赖既定规则的启发式剪枝,在网络结构较为紧凑时,可能导致次优解。

基于优化方法的剪枝,引入正则化项或约束条件将剪枝建模为优化问题,融入网络训练过程,渐进式地促使次重要的权重趋于零值。Liu 等[51]对批量归一化缩放因子使用 L_1 正则化,Lee 等[41]提出了基于核范数的正则化,以实现目标压缩率。Zhang 等[47]使用交替方向乘子法(alternating direction method of multipliers, ADMM),迭代求解两个非凸优化子问题,但具有较高的计算复杂度和延迟。Wen 等[52]在损失函数中加入 Group Lasso 正则化,实现权重结构化稀疏。

3)剪枝流程

剪枝根据产生稀疏性的不同步骤,可分为如图 8 所示的一次性剪枝[39]和迭代剪枝。

一次性剪枝根据稀疏化对象的重要性评估,一次性删除所有冗余元素或网络子结构[53-54],但即使经历多轮次微调,通常仍有准确率损失;迭代剪枝通过多轮“剪枝-微调”循环,渐进式压缩网络,逐步调整并得到最终的稀疏化结构,具有较高的准确率,是当前主流的剪枝方法[45,55-56]。

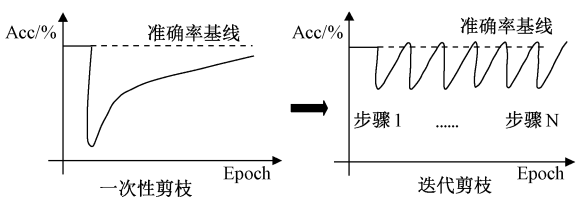


图 8　一次性剪枝和迭代剪枝

Fig. 8　One-shot pruning and iterative pruning

2.2　量化

剪枝通过移除网络中的冗余结构实现网络压缩,但在保留原有数值精度表示的情况下,其存储和算术开销仍然较高。量化将参数或特征图的数值表示从高位宽映射到低位宽,进一步降低了网络的存储需求和计算复杂度,可与剪枝互补使用[57]。当网络被量化到极低位宽的二值化[58]或三值化[59]时,可仅采用异或和计数操作,实现硬件友好的无乘法算术运算[60-61],以大幅降低硬件资源开销和计算延迟。根据量化区间划分方式、数值映射规则及在训练流程中使用的方法不同,量化方法在网络精度、训练成本以及硬件实现复杂度方面呈现出不同特性。

1)量化策略

按照步长是否均匀,量化可分为均匀量化[62]和非均匀量化[63]。均匀量化将原数值均匀映射到量化区间,实现方式简单;非均匀量化采用非等距的量化间隔,使得量化级别在数据密度较高的子区间更为密集,能更有效地表征非均匀分布的数据[48]。

按照范围是否以 0 为中心对称分布,量化可分为对称量化[64]和非对称量化[65]。对称量化中,原数值映射到以 0 为中心的对称量化区间;非对称量化中,量化范围不以 0 为中心对称分布,原数值的最小值和最大值分别映射到量化区间的最小值和最大值,对不以 0 为中心分布的数据更有效。

按照舍入规则是否具有确定性,量化可分为确定性量化[66]和随机性量化[67]。确定性量化中,每个原数值唯一地映射到一个固定量化值;随机性量化中,原数值依据某种概率分布映射,因其能避免梯度累积误差保持梯度的无偏估计[68],在训练低精度网络中能取得更好的性能。

2)量化流程

量化按照训练过程中预训练量化网络参数的产生方式,分为训练后量化和量化感知训练如图 9 所示。

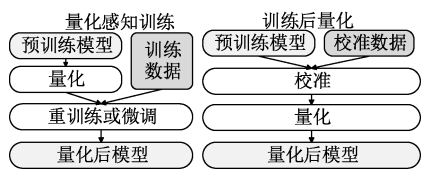


图 9　量化感知训练和训练后量化[69]

Fig. 9　Quantization-aware training and post-training quantization[69]

训练后量化采用动态或静态量化算子[70-71],将 GPU 预训练使用的单精度浮点数(FP32)转换为低位宽浮点数或定点数。为保证训练的准确率,量化感知训练采用伪量化策略,将量化噪声(包括舍入误差和截断误差)融入整个反向训练过程:网络参数在前向传播时,使用低位宽数据格式,但在反向传播时,仍保留单精度浮点数。为使产生离散数值的量化算子在反向传播的链式法则中可微,使用如图 10 所示的直通估计器的技术[72],将量化算子的导数设置为 1。

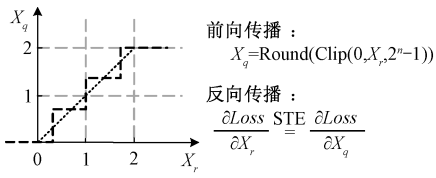


图 10 直通估计器[62]

Fig. 10 Straight through estimator[62]

在隐私数据需求场景下,量化感知训练需要完整的数据集进行网络微调的问题,可使用无需依赖真实数据的零样本量化方法[73-75],通过分析网络参数的分布规律,即可直接推导出最优的量化参数。

2.3 低秩近似

量化方法主要通过降低数值精度减少网络的存储和计算开销,但并未直接利用卷积核和权重张量在结构层面的冗余特性。低秩近似则通过多个小尺寸的低维张量,逼近原始的大尺寸高维结构,降低参数存储量并提升计算速度。根据对网络参数引入低秩结构的方式,可分为后训练分解范式[76]和结构先验范式[77]。

后训练分解范式在预训练网络上对权重张量进行低秩分解(如 SVD[17]、CP 分解[78]、Tucker 分解[79])。如图 11 所示,结构先验范式在网络设计期间引入低秩结构,将原始权重张量替换为由多个低秩卷积组合而成的近似计算结构。

结构先验范式采用正则化项,引导卷积核在训练阶段向低秩空间靠拢[80],或在损失函数中加入依赖各层秩的网络代价选择函数,精确地选择各层的最优秩数和对应的低秩矩阵[81]。后训练分解范式通过低秩分解权重实现网络压缩,但可能忽略神经元重要性而导致准确率下降[82];结构先验范式在网络设计时引入低秩结构,从架构层面构建出适合边缘部署的高效紧凑网络[82],但设计复杂性高。

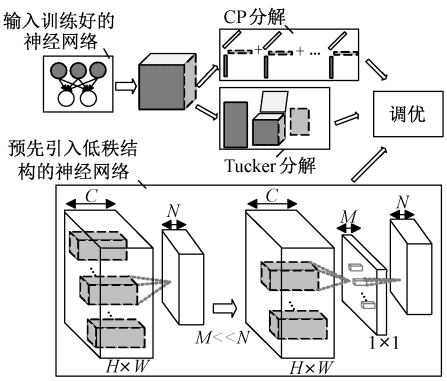


图 11 低秩分解架构

Fig. 11 Low-rank decomposition architecture

2.4 知识蒸馏

上述轻量化方法通过直接修改网络参数或计算结构实现网络压缩,而知识蒸馏则从网络训练范式出发,通过知识迁移的方式提升轻量化网络的表示能力。知识蒸馏是 Hinton 等[83]提出的一种轻量化方法,如图 12 所示,将性能强大但复杂度较高的教师网络中的知识,以软标签的形式传递给轻量化的学生网络。知识蒸馏以教师网络对样本预测的概率分布,作为附加监督信息[84],从而帮助学生网络更好地捕捉类间相似性与决策边界。

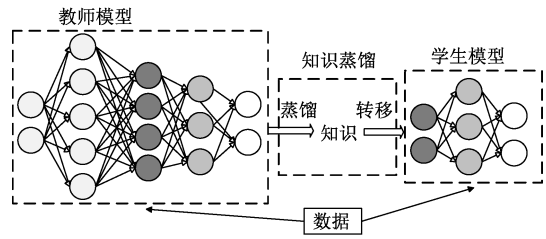


图 12 知识蒸馏框架[84]

Fig. 12 Knowledge distillation framework[84]

训练过程中,学生网络同时最小化与真实标签的交叉熵损失函数,以及与教师输出之间的库尔贝克-莱布勒(kullback-leibler, KL)散度。主流的方法类型包括使用预训练的教师网络压缩大网络,适合资源受限设备的离线蒸馏[85];同时训练教师和学生网络,提升网络泛化能力的在线蒸馏[86];依赖单一网络自学习,适用于数据稀缺场景的自蒸馏[87]。

然而,轻量化网络的实际性能,很大程度上取决于底层硬件架构的适配性。GPU 等通用计算平台在低批量、确定性延迟任务中存在明显局限,其固有的 SIMD 并行架构难以高效处理非结构化稀疏计算[48];且多层级软件栈还会引入不可控的系统调度延迟[88]。相应地,FPGA 凭借可重构特性,可针对轻量化网络的计算特征定制计算数据流通路,消除冗余控制开销并优化数据复用,从而实现更高的性能和硬件资源利用率[89]。

综上所述,网络轻量化方法旨在大幅降低 CNN 网络复杂度,使其适合边缘设备的存储资源和计算规模约束。如表 1 所示,剪枝和低秩近似通过调整网络结构与计算图,减少冗余参数和计算路径,从而压缩网络规模并降低有效计算量;量化方法则通过改变权重和特征图的数值表示形式,在保持网络结构不变的前提下降低存储位宽和算术运算复杂度;知识蒸馏不直接修改网络结构或数值表示,而是通过训练阶段的知识迁移,提升轻量化网络的表示能力。

表 1　主流 CNN 网络轻量化方法对比与 FPGA 适用性分析

Table 1　Comparison of mainstream lightweight CNN network methods and analysis of FPGA applicability

轻量化方法	代表研究	核心机制	优势特征	局限性	FPGA 适配特性
非结构化剪枝	文献[34,39-40]	按权重元素删冗余	压缩率高裁剪更灵活	稀疏不规则并行低效	需稀疏编码控制开销
结构化剪枝	文献[35,41-42,90]	按通道卷积核删冗余	规整稀疏易硬件映射	压缩受限结构约束强	规则数据通路更友好
量化	文献[58-59,62-67,70-71,73-75]	高位宽降至低位宽	存算开销显著更低	误差敏感依赖校准强	位宽可配实现更高效
低秩近似	文献[17,78-81]	低维张量组合逼近权重	降参降算挖掘冗余	分解影响精度需调参	算子更小,利于流水
知识蒸馏	文献[83-87]	教师指导训练学生网络	不改结构提升轻量精度	需教师网络训练成本	提升轻模部署后性能

3　面向 CPU-FPGA 的硬件设计优化

网络轻量化虽然在理论上显著降低了网络的计算量与存储需求,但稀疏化的权重或低位宽数据若缺乏底层硬件的针对性支持,往往难以转化为实际性能中的延迟与能效收益。将轻量化网络的潜力转化为实际的性能增益,硬件平台的选择或设计优化至关重要。如引言所述,独立的 CPU、GPU 或 ASIC 方案,在特定任务下可实现低延迟/高吞吐和高能效,但面对边缘端应用对低批量、确定性低延迟、高能效和灵活可编程性等综合需求时,往往难以完全兼顾,存在应用局限。因此,融合了 CPU 控制灵活性与 FPGA 可重构并行处理能力的 CPU-FPGA 异构系统,能够通过软硬件协同设计,精准地满足工业自动化控制、机器人视觉感知等边缘 AI 应用中低批量、确定性低延迟、高硬件利用率等不同于边缘端 GPU 高吞吐的复杂特定需求。

面向该异构平台的硬件设计优化方法涵盖性能评估的基础建模方法、加速器核心的数字架构设计范式、系统全局性能最优的协同优化策略,以及高层次综合(high-level synthesis, HLS)为代表的敏捷开发方法。

3.1　FPGA 优化目标建模

在基于 CPU-FPGA 的 CNN 加速器设计中,吞吐量和延迟不仅是衡量系统效能的核心指标,更是指导软硬件协同优化的关键依据。如 1.2 节所述,在 FPGA 资源与带宽约束显著且设计空间较大的条件下,若依赖完整实现与反复验证进行试探式优化,往往带来较高的实现与评估成本。因此,有必要采用构建成本较低且精度满足需求的性能评估模型,以为架构迭代提供快速、可靠的定量依据。

1)基于 Roofline 模型的吞吐量与瓶颈分析

Roofline 模型[91]通过关联吞吐量和带宽,可视化硬件加速器系统的计算性能。如图 13 所示,其纵轴为吞吐量(P);横轴为算术密度(I),定义为单位数据元素对应的运算次数(FLOPs/Byte);虚斜线代表由硬件物理接口决定的理论峰值带宽。然而,受限于通信协议开销与系统软件管理等因素,理论值往往难以表征真实性能。为保障模型预测的准确性,主流研究引入基于实测的微基准测试对目标平台的有效带宽进行定量表征[91],即利用不同算术强度的微基准测试进行扫描:通过低强度负载测定实际带宽(斜坡),高强度负载测定峰值吞吐(坡顶),进而拟合数据边界以确定性能拐点。基于此实测数据绘制的实斜线为实际带宽,指加速器实际可达到的最大带宽,实际传输引入的协议开销、系统软件管理开销等因素,导致其通常比理论带宽低;实水平线为实际峰值吞吐。

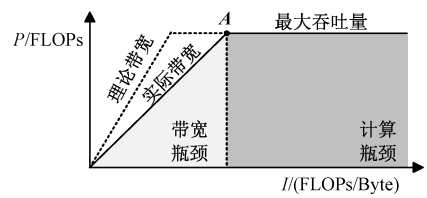


图 13　屋顶线模型[91]

Fig. 13　Roofline model[91]

系统设计的可行解落在实斜线和实水平线围绕的下方区域,实斜线下的点受限于带宽瓶颈,可进一步提高算术密度(例如提高有效计算数或降低量化位宽),直至进入计算瓶颈。理想设计应位于图 13 中的拐点 A,此时以最低的硬件计算资源开销达到峰值吞吐性能。

屋顶线模型虽能直观定位计算与带宽的宏观瓶颈,但无法表征流水线阻塞(即计算单元空闲周期)等运行时特性,使其在不同场景下的适用性存在差异。在架构设计早期,屋顶线模型凭借高可视化与低构建成本,是进行快速资源评估的首选工具。然而,在需要精确量化微观架构行为的场景下,现有研究则更倾向于采用周期精确模拟。例如,Shao 等[92]提出了基于一种高精度模拟框架,通过引入具体的硬件约束,实现了对加速器周期级行为的精确刻画;Makrani 等[93]提出的 XPPE 框架通过构建 HLS 资源特征与目标平台性能间的隐式映射关系,在无需完整物理实现的前提下,达到了较高的跨平台性能评估相关度。尽管周期精确模拟在微观行为刻画上更为准确,但受限于其高昂的开发复杂度与缓慢的仿真速度,

屋顶线模型凭借能够快速定位瓶颈并指引优化方向(如提升带宽或增大并行计算规模)的特性,依然是指导系统级架构优化的核心工具。

2)关键路径延迟建模与优化策略

不同于侧重系统整体处理能力的吞吐量指标,推理延迟是边缘实时性应用的关键指标。从理论上讲,端到端网络推理的总延迟可建模为数据传输延迟、计算延迟、控制与调度延迟等的线性叠加。其中对整体延迟贡献最大的部分被称为关键延迟,决定了时序合规时系统的最高工作频率,进而影响端到端处理的总延迟。

通过对关键延迟进行建模,可以量化关键路径延迟、分析不同优化方法对关键路径的影响[94-95]。然而,物理实现中存在的流水线填充、层间调度及动态随机存取存储器(dynamic random access memory, DRAM)刷新等隐形开销,致使简单的线性叠加模型失效,且理论预测值远优于实测值。为了确保延迟模型的高置信度,Pellizzoni等[96]聚焦于资源竞争,通过计算特定仲裁策略下内存请求的最坏情况等待时间来量化延迟边界;Shao 等[92]通过构建动态数据依赖图,利用资源受限后的关键路径长度来精确表征延迟。

若各功能模块的延迟呈现非均匀分布,且存在显著的性能瓶颈(例如,单一网络层级贡献了超过 50%的总体延迟[97]),则需聚焦于关键路径的局部优化策略,具体手段包括定制化计算单元设计[98-99]、片上数据复用[11],以及优化存储子系统带宽[100]。反之,当延迟分布相对均匀或局部瓶颈不明显时,应采用降低计算和存储冗余[48]、提高数据复用[11]等全局优化策略。

上述性能评估模型与关键路径分析为加速器设计提供了理论边界与优化方向。在明确了资源约束与性能目标后,硬件设计核心在于组织计算阵列与数据通路,以最大化逼近 Roofline 模型的理论峰值。针对不同的应用场景需求,衍生出两种主流数字架构设计范式。

3.2 数字架构设计优化

明确了性能边界与优化目标后,数字架构设计是 FPGA 加速器从理论模型走向物理部署的核心环节,涵盖系统架构与模块设计、接口通信及存储层次等多个维度,以构建满足特定功能、性能及资源约束的数字系统。最大化计算吞吐与能效时,平衡任务执行通用性,形成流式架构和单计算引擎架构两种主流范式[101]。

1)流式架构

流式架构的核心思想是层级硬件专用化,即为 CNN 中的每一层(或多层)设计专用的硬件计算模块,通过片上 FIFO 将这些模块以流水线方式连接,如图 14 所示。数据流无需回写到片外主存,显著降低了数据搬运开销,从而实现极低的端到端推理延迟[101]。

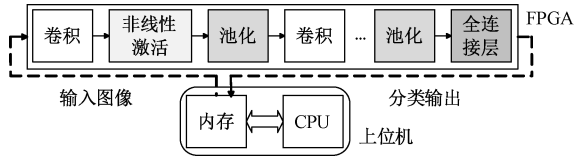


图 14 流式加速器架构[101]

Fig. 14 Streaming accelerator architecture[101]

这种架构非常适用于实时控制系统等网络结构固定且对延迟要求极为苛刻的场景[102]。例如,在边缘端实时视觉任务中(如机器人抓取分拣[103]、车载/无人机前端感知[104]与智能摄像头分析[105]等),系统多以低批量甚至单批量方式处理连续视频流[105],并强调端到端低延迟响应[106]。流式架构利用层间流水级联与数据片上直通机制,消除了中间特征图的片外回写开销,从而可以实现高度确定的低延迟性能[61]。

然而,随着网络结构向更深、更复杂的方向发展,流式架构在自动驾驶感知等资源受限的复杂边缘计算场景中的弊端日益凸显。首先,层级间的负载差异会导致资源利用率不均衡,网络若采用流式架构为每一层静态分配硬件,会导致计算量较小的池化层或激活层硬件资源闲置,造成资源浪费[107]。其次,流式架构的可扩展性受限于 FPGA 的片上物理资源,例如,在资源受限的边缘设备上部署深层网络时,若通过降低并行度以容纳完整网络,反而增加了推理延迟[108]。最后,固化的流水线设计难以适应多任务动态场景,例如,在自动驾驶等需要实时切换网络的应用中,流式架构依赖重构 FPGA 来变更网络的特性,导致其无法满足系统对灵活性的即时响应需求[109]。

2)单计算引擎架构

与流式架构不同,单计算引擎架构采用的是核心计算资源时分复用的设计理念,如图 15 所示。单计算引擎架构通常以一个通用计算引擎为核心(多采用基于脉动阵列的矩阵乘法单元,并设置片上多级缓存、数据搬运通路以及片上互连结构),由控制器按层调度,依次执行网络各层运算;同时通过软件侧配置不同层的参数与数据映射,使核心计算资源得以跨层复用,从而在有限 FPGA 资源下保持高利用率与通用性[101]。

单计算引擎架构更适用于网络与任务多样、需要频繁更新/切换、且以稳态吞吐与资源效率为主要目标的部署场景[110]。该架构常用于边缘侧通用推理平台,以及服务器/数据中心侧 FPGA 推理加速平台,此类系统通常要求硬件同时支持目标检测、语义分割等多类网络,具备网络升级、网络切换与多任务并发能力[109]。针对上述需求,单计算引擎架构通过时分复用计算阵

列,并由软件配置层参数与调度策略适配不同 CNN,从而在多网络与频繁迭代场景下具备更高通用性和资源利用率[11]。

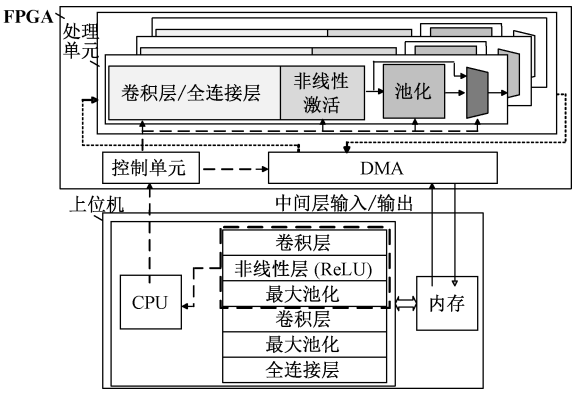


图 15　单计算引擎加速器[101]

Fig. 15　Single computing engine accelerator[101]

在此类场景中,单计算引擎架构的瓶颈往往来自层间复用引入的数据搬运与调度开销[111]。由于不同 CNN 层在同一计算阵列上顺序执行,层间切换会引入权重重载与特征图重排开销。当片上缓存无法容纳足够的权重或特征图工作集时,数据将频繁在片外主存与片上之间搬运,使性能受限于外存带宽[98]。在多任务并发或多路输入的场景下,片外主存与片上缓存成为共享资源,会导致阵列利用率下降,并进一步引起稳态吞吐降低以及端到端时延上升[94]。另一方面,在单批量且端到端延迟受严格约束的实时推理场景中,层切换与数据搬运在总时延中占比更高,单计算引擎架构在端到端时延上通常不及流式架构。但对于深层网络且资源受限的情况,单计算引擎架构往往在总延迟和吞吐量上更占优。

单计算引擎架构中脉动阵列数据流的选择、设计、优化至关重要,将直接影响数据复用效率和访存开销。常见的数据流模式包括权重固定、输出固定和行固定数据流[11]。

如图 16 所示,权重固定数据流将权重固定在 PE 中,输入特征图的数据流过 PE,与固定的权重进行运算,生成并传递部分和,适用于批量推理、深度可分离卷积等权重复用率高的场景。但对于权重复用率低、或者需要频繁更换权重(如在线学习、网络切换频繁)的场景则效率不佳。输出固定数据流减少了输出特征图数据多次写回操作,且降低了功耗,但需要频繁加载输入特征图和权重数据,并需要足够的片上存储空间来存储部分和,适用于输出通道数多且需要多次累加的场景。但对于输出通道数较少、或者片上存储资源受限的场景,其计算并行度优势不明显。

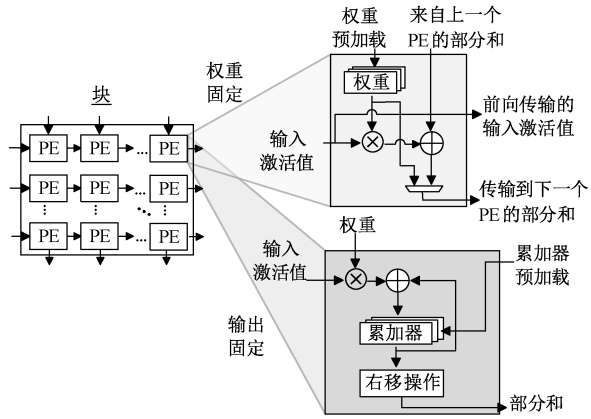


图 16　权重固定和输出固定数据流[112]

Fig. 16　Weight stationary and output stationary dataflow[112]

行固定数据流如图 17 所示,其核心思想是将输入特征图与卷积核权重的单行数据缓存于 PE 阵列,以减少数据的重复访问,适用于基础计算单元内部操作类型多样或存在数据融合需求的网络。但由于数据调度和逻辑控制相对复杂,行固定数据流难以高效用于具有极低控制逻辑复杂度要求,或片上存储资源高度受限的应用场景。

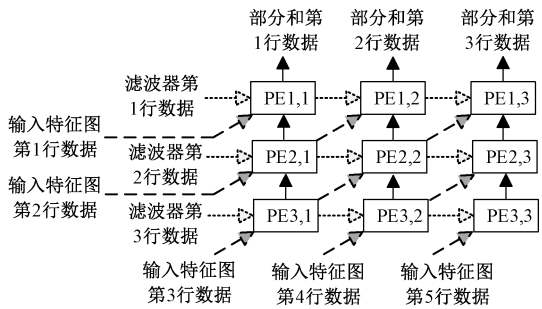


图 17　行固定数据流[11]

Fig. 17　Row stationary dataflow[11]

相比于单计算引擎架构,流式架构通用性较差,其消耗较多硬件资源,实现高计算效率、低延迟,但当网络层数增加时,单计算引擎架构往往更具优势。

尽管流式架构与单计算引擎架构在特定应用场景下各具优势,但二者均难以独立解决 FPGA 加速器所面临的共性挑战,即神经网络规模效应带来的巨量参数存储、密集数据搬运、多样化算子支持及高昂计算需求,与 FPGA 有限的片上存储及计算资源之间的矛盾。当高效的计算阵列设计确立后,系统的端到端性能往往不再受限于计算单元本身,而是取决于数据的供给效率与任务的调度延迟。这促使设计焦点从局部计算单元转向全系统架构,通过计算、存储与控制的联合优化,以实现硬件整体效能的最大化。

3.3 加速器系统级优化方法

针对上述由计算单元向系统层面转移的性能瓶颈，在确立核心数字架构的基础上，系统级设计旨在通过全局优化策略，缓解计算速度与存储带宽之间的矛盾，并解决异构任务分配不均带来的效率问题。具体而言，将围绕 3 个核心维度展开探讨：软硬件协同优化，以实现算法特性与硬件实现的深度耦合；数据管理与访存优化，以缓解内存墙瓶颈；任务分配与调度，以最大化 CPU-FPGA 异构系统的整体效率。

1）软硬件协同优化

软硬件协同优化超越了单向的“算法指导硬件”模式，强调算法与硬件之间的双向互动与共同演进。一方面，算法的内在特性可以指导硬件进行定制化设计，例如，Winograd 算法适合用 FPGA 的加法器资源替代乘法器资源，或针对 FFT 算法定制高效的蝶形运算单元。另一方面，硬件实现的约束条件亦能双向驱动算法设计，促使其朝着更适合硬件架构的方向演进，例如，FPGA 处理非结构化稀疏的效率低下，这便激励了学术界和工业界去研究更易于硬件实现的结构化剪枝方法[113]。这种双向协同使得 FPGA 充分发挥其可重构特性，能根据特定算法的计算特性进行硬件结构定制和数据流优化，从而在计算方面显著提升加速器的性能上限。

（1）快速卷积

卷积计算可采用频域卷积、Winograd 算法等快速算法大幅减少 MAC 操作数，从而降低计算延迟。FFT 和 Winograd 卷积计算过程如图 18 所示。基于 FFT 的频域卷积[114]主要分为 3 个阶段：输入特征图和卷积核进行 FFT 变换；在频域里输入特征图和卷积核转换后的矩阵进行点乘计算；进行 FFT 逆变换得到输出特征图。Winograd 卷积[115]利用多项式的代数性质，通过变换矩阵将输入特征图和卷积核转换到变换域，将空间域卷积简化为点乘运算，再转换回空间域，得到最终的卷积结果。

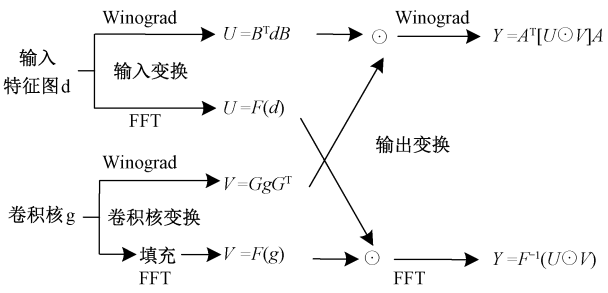


图 18 Winograd 卷积和 FFT 卷积计算流程

Fig. 18 Computation flow of Winograd and FFT convolutions

根据 FFT 和 Winograd 等快速卷积的原理，为了在 FPGA 上实现 CNN 的高效计算，通常需要设计定制化的硬件单元。Liang 等[116]针对输入/滤波器变换、逐元素矩阵乘法、输出逆变换 3 个主要阶段，分别为 FFT 和 Winograd 卷积设计了高度优化的、流水线化的 PE。Wong 等[117]针对 2×2 输出、3×3 卷积核的 Winograd 算法设计了专用硬件模块，通过简化常数运算、避免非 2 次幂分数，显著降低逻辑开销与数值误差。Liu 等[118]设计的基于变换矩阵共享的 WinoPE 架构通过标识位切换输出变换矩阵，可以实现多核尺寸硬件复用。

FFT 频域卷积适用于大尺寸卷积核、高分辨率卷积场景，可以完整保留频谱信息，但通常需要利用 Hermitian 共轭对称性，降低复数运算的计算和存储复杂度[119]。Winograd 卷积以计算开销较小的加法代替乘法，应用于 1×1 或 3×3 小卷积核时计算效率高，但在大卷积核计算的数值稳定性、准确率等方面欠佳[115]。

（2）稀疏神经网络的 FPGA 加速

非结构化剪枝导致随机分布的稀疏性，为硬件加速带来了显著的挑战，包括网络存储与检索[46]、动态稀疏计算控制[89]和负载不平衡[120]问题。

稀疏网络存储需要高效地压缩和索引非零值数据[121]。不同的 FPGA 处理单元可能分配到不同数量的非零值计算任务，导致部分单元空闲而其他单元过载，降低了整体并行效率。高效实现动态或静态的负载均衡策略，例如任务调度和数据分发机制，是充分发挥 FPGA 并行计算能力和定制数据流优势的关键[122]。

针对上述挑战，已有多种在 FPGA 上加速非结构化稀疏神经网络的方法。Han 等[33]将压缩网络完全放入片上静态随机存取存储器（static random access memory，SRAM），通过分布式存储和计算，实现了稀疏层的并行化和负载均衡。Lu 等[123]提出了分块优先数据流和压缩数据存储格式，为每个块存储权重索引数组，以跳过零值计算。Zhang 等[124]采用通道优先数据流，并使用两级部分和归约数据流来消除输出缓冲区上的访问冲突，其数据流设计思路同样可以启发 FPGA 上针对稀疏性的数据组织和计算流程优化。Deng 等[125]采用在线交集方法处理稀疏数据，动态判断和跳过零值计算，避免了对零值进行不必要的计算和访存。

结构化剪枝后的网络参数保持连续规整的排布，可以直接利用高效的通用计算库进行加速，FPGA 可以直接采用优化的矩阵乘法 IP 核或 HLS 工具设计的高性能计算单元进行加速，而无需过多关注非结构化稀疏性带来的额外开销[113]。

上述快速卷积算法和稀疏神经网络加速器两个案例都涉及到算法层面的特性（Winograd/FFT 的数学性质、稀疏网络的结构）与硬件层面的定制化设计（蝶形单元、加法器替代乘法器、稀疏计算单元）的深度耦合，主要提升了加速器的计算效率，并缓解访存瓶颈。

2)数据管理和访存优化

尽管 CNN 加速器在计算单元设计上不断演进以提高效率,然而,受限于片外 DRAM 访问的高延迟和有限带宽,其整体性能瓶颈主要表现为内存墙问题。为缓解这一问题,一系列数据管理和访存优化技术应运而生,数据压缩、分块、预取等数据访存优化方法可以减少数据访问延迟、提高吞吐量,并降低资源消耗。这 3 种方法优化流程如图 19 所示。数据压缩对数据进行量化[57]、剪枝[48]、稀疏编码[33]等操作;数据分块按照块将数据传输到片上,适配 PE 阵列的计算粒度[11];数据预取结合 FPGA 的流水线特性,通过预测数据需求来提前加载数据[126]。

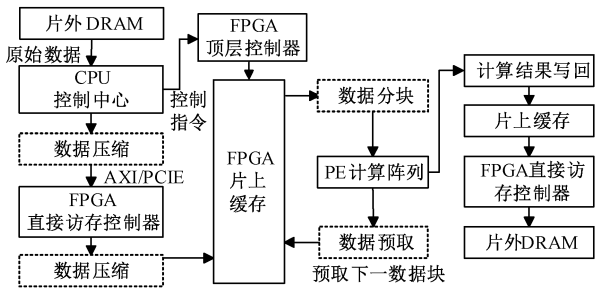
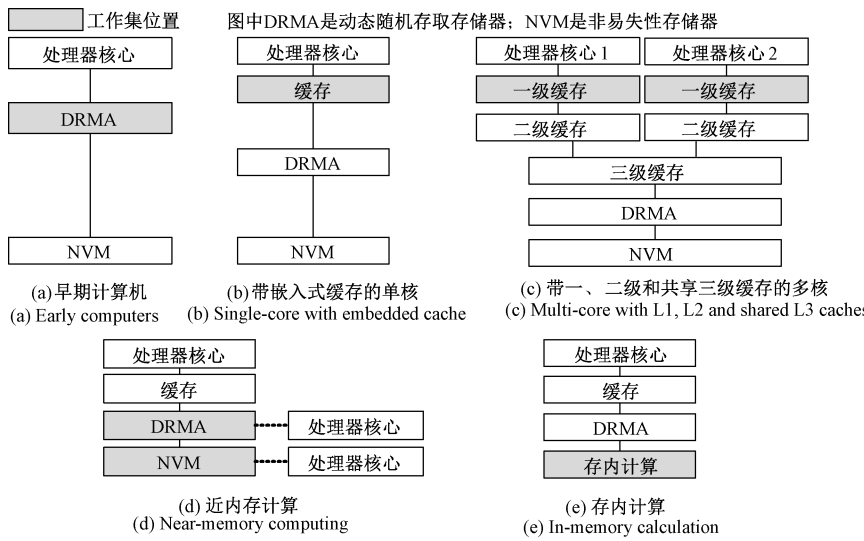


图 19　FPGA 数据压缩、分块与预取协同优化流程

Fig. 19　Collaborative optimization flow of FPGA data compression, block segmentation, and prefetching

经典冯诺依曼架构存在内存墙问题,即处理器和内存间的数据搬移主导了系统的能耗和延迟[127]。如图 20(a)~(c)所示,每次计算时需先通过数据总线从存储器中读取数据,而存储器的读取速度远低于处理器的运算速度,导致系统整体算力受限于传输带宽。虽然可以通过使用高带宽存储器(high bandwidth memory, HBM)提高数据传输[128],但无法从根本解决存算分离时频繁数据搬运导致的计算效率、延迟和能耗问题。

存算一体计算范式,通过将处理模块直接集成于存储器内,有望突破冯诺依曼架构的存算分离限制。存算一体包括存内计算(computing in memory, CiM)和近存计算(near memory computing, NMC),如图 20(d)、(e)所示,分别通过直接在内存中处理数据、缩短数据移动的距离来降低数据移动开销。CiM 神经形态计算新器件融合了计算与存储单元设计[129]。NMC 通过先进封装将内存与计算芯片紧密集成,缩短计算和存储单元的物理距离来提高带宽[130]。PiCaSO 利用 FPGA 现有的 BRAM 存储数据,并通过邻近逻辑单元执行计算[131]。Wang 等[132]为基于 SRAM 的 CiM 边缘 CNN 加速器设计提供了设计框架和评估模型,其计算阵列设计、系统级能效优化方法为基于 FPGA 的 CNN 加速器的架构设计与性能优化提供了新路径。



(a)早期计算机 (a) Early computers　(b)带嵌入式缓存的单核 (b) Single-core with embedded cache　(c)带一、二级和共享三级缓存的多核 (c) Multi-core with L1, L2 and shared L3 caches

(d)近内存计算 (d) Near-memory computing　(e)存内计算 (e) In-memory calculation

图 20　计算架构中的多级存储架构[133]

Fig. 20　Memory hierarchy in computing architecture[133]

上述数据管理和访存优化技术致力于缓解数据访存瓶颈,可以显著提高加速器计算效率。然而,在 CPU-FPGA 异构计算平台中,充分释放系统潜力还需要根据任务特性和硬件优势,对任务进行全局性的协同调度。

3)任务分配与调度

在 CPU-FPGA 异构系统中,任务的合理分配与高效调度是发挥系统潜力的关键。其核心目标是将合适的任务分配给合适的处理单元,以实现负载均衡和流水线并行。通常,计算密集型、数据并行度高的任务(如卷积、全连接层)被卸载到 FPGA 上执行;而控制密集型、逻辑复杂或串行的任务(如网络加载、非线性激活函数、任务调度本身)则由 CPU 负责[134]。

调度系统模型如图 21 所示,包含 CPU 调度模块和 FPGA 计算资源。CPU 调度模块包括调度器、布局器和加载器,FPGA 通过硬件加速 CNN 中的计算密集层,优化数据访存和计算的局部性[134]。

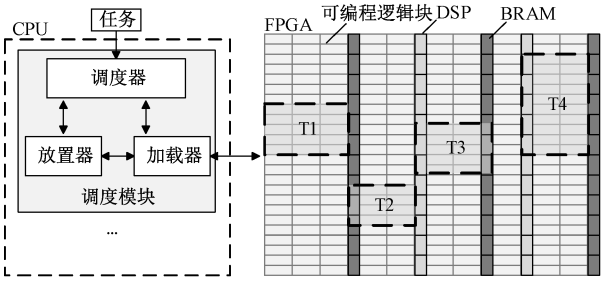


图 21 CPU-FPGA 的系统调度模型[134]

Fig. 21 Scheduling model of CPU-FPGA system[134]

计算任务可以分为静态任务和动态任务。静态任务在编译或系统启动阶段就已经确定了其在 CPU 或 FPGA 的执行位置、执行顺序和执行时间的任务,具有较高的可预测性和较低的配置开销;动态任务在 CPU 或 FPGA 的执行位置、执行顺序和执行时间在系统运行过程中根据当前的状态、数据特性或系统负载等因素动态决定,提供了更高的灵活性和负载均衡能力。

任务的静态与动态划分由硬件特性、模型特性和应用场景共同决定。CNN 网络中高计算密度且数据流规则性强的操作通常归类为静态任务,例如卷积层的 MAC 运算、池化层等[135],而控制逻辑复杂或运行时需自适应调整的操作通常归类于动态任务[136]。

上述系统级优化策略虽能最大限度释放异构平台的潜力,但随着优化维度增加,硬件设计的复杂度呈指数级上升。若继续沿用经典的寄存器传输级(register transfer level, RTL)开发模式,冗长的开发验证周期将难以适应深度学习算法的快速迭代需求。为突破这一开发效率瓶颈,基于 HLS 技术的敏捷开发范式成为了协调算法演进速度与硬件开发周期差异的关键手段。

3.4 基于 HLS 的敏捷开发方法

尽管前述优化策略显著提升了性能潜力,但经典 RTL 设计的高门槛与长周期限制了硬件的迭代效率。为此,HLS 技术通过提升设计抽象层次,为解决这一矛盾提供了关键路径[137]。HLS 允许开发者使用 C、C++等高级语言描述算法行为,并将其自动转换为 RTL 代码描述的有限状态机,从而显著提升了抽象层次,大大提高开发效率并降低设计门槛,实现了数字架构设计的敏捷开发。如图 22 所示,HLS 与 RTL 开发类似,经历仿真、综合、布局布线和后仿真等阶段,生成封装后的 IP 用于原理图设计和上位机调用。

随着敏捷开发工具的成熟,设计规模飞速增长,如何

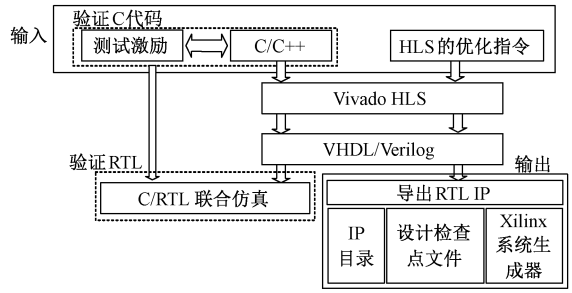


图 22 基于 Vivado HLS 的开发流程

Fig. 22 Vivado HLS development flow

寻找最优的设计参数配置组合,成为困扰硬件设计的新难题。为高效搜索复杂的参数设计空间,充分释放 HLS 在加速器敏捷开发中的潜力,需要优化原语、设计模板、设计空间探索等层次方法的支撑。

1)基于优化原语的计算阵列优化

HLS 优化原语使开发者能够显式地指导编译器将高级算法描述映射为高效的硬件结构。尽管 HLS 允许使用 C/C++描述计算逻辑,但编译器通常无法自动推断最优化的硬件实现细节,例如计算单元的并行度、流水线策略、片上存储的组织方式。循环展开、循环流水线和数组分区等关键指令,提供了在高级语言层面精确控制底层硬件设计的机制,从而避免面向底层 RTL 的繁复手工设计与调试。

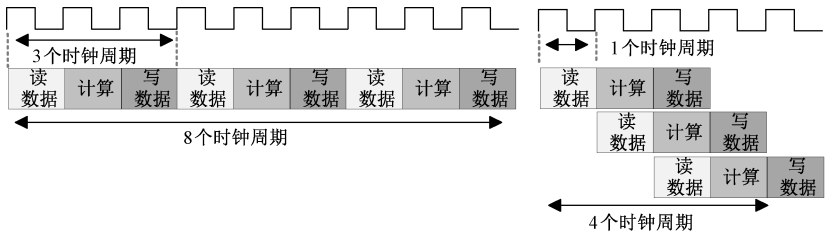
循环展开通过复制循环体以并行执行多次迭代,能够在无数据依赖时显著缩短延迟;完全展开可实现最大并行度,但往往伴随 LUT、寄存器和 DSP 资源的增加;循环流水线化则通过在循环入口处插入寄存器,使得不同循环迭代重叠执行。如图 23 所示,进行循环流水线优化可以显著提高吞吐,流水线启动间隔(initiation interval, II)是决定吞吐性能的关键因素,理想情况下最小为 1。值得注意的是,HLS 工具中流水线性能受到层次结构的影响:若顶层函数已流水线化,下层级所有循环都会自动展开;调用的子函数需单独流水线化,从而避免引入气泡周期导致吞吐量下降。

此外,由于数组通常由多组双端口 BRAM 实现,循环展开会导致同时对数组的多次读写,带来存储带宽瓶颈,需要通过数组分区指令将数组拆分为多个 BRAM 以支持并行数据访问。上述方法综合运用可使卷积运算在保持低流水线启动时间间隔的同时,实现接近理论极限的吞吐性能。

虽然优化原语可以指导 HLS 工具高效生成计算阵列,但实现有效嵌入优化原语并编写高性能 C/C++描述,现阶段仍需开发者具备一定硬件设计和优化基础。

2)基于设计模板的快速构建与部署

为进一步降低开发门槛,FPGA 厂商和社区构建了



(a) 无循环流水线
(a) Non-circulating production line

(b) 循环流水线优化
(b) Optimization of the circulating pipeline

图 23　循环流水线示意图

Fig. 23　Schematic diagram of the circulation pipeline

基于 HLS 的集成化工具链(例如 AMD Vitis AI)。此类平台通过封装预优化的 IP 核、可复用设计模板及自动化流程,简化了从算法网络到硬件实现的转化路径。这种工具链能够解析主流深度学习框架的预训练网络,并通过量化压缩、图优化、硬件指令编译等系统化步骤,实现网络至 FPGA 硬件的自动化编译与映射。

Vitis AI 是 AMD 推出的端到端 AI 推理开发部署平台,为图像分类、目标检测和分割等任务提供了高度预优化的设计模板;经过量化与编译,可直接部署于 Xilinx FPGA,显著缩短开发周期。例如,基于 ResNet 或 YOLO 的预构建设计模板可应用于快速验证场景。对于不在支持列表中的自定义层,Vitis AI 提供了运算符扩展接口,允许基于 HLS 的专有 RTL 实现无缝集成。

为进一步实现高性能 CNN 加速,还可在系统级和架构层面进行粗粒度优化。Overlay 技术通过对底层硬件资源的抽象和虚拟化,为上层应用提供更灵活的任务调度[138]。CGRA 粗粒度可重构架构[139]是 Overlay 的一种形式,通过在更大粒度上调度数据和操作,实现 CNN 加速器性能、灵活性和开发效率的多维度优化。

3)面向多目标的设计空间探索

基于 FPGA 的 CNN 加速器设计中,功耗、性能和面积等指标往往相互制约,形成庞大的设计空间。为了在资源约束下找到最优或接近最优的硬件实现方案,设计空间探索用于寻找帕累托最优解,即在不牺牲其他目标时,无法再优化当前目标,所有解的集合构成的帕累托前沿。设计空间探索方法依据目标函数复杂性、设计空间规模及可用计算资源,选择能达到或逼近帕累托前沿的设计参数配置,从而确定 CNN 加速器的最终设计规模与具体配置。

然而,利用 HLS 实现高效的多目标设计空间探索面临着显著挑战:随着设计规模扩大,HLS 优化原语及参数剧增,且指令间可能存在复杂的关联或冲突,极大增加了设计空间探索的复杂性。此外,Vivado 等工具链编译得出报告结果所需要的时间,按照设计复杂性从数小时到数天不等[140],严重限制了实践中可评估的设计方案数量,成为利用 HLS 实现敏捷开发的主要瓶颈。

为克服这一瓶颈并支撑敏捷开发流程,HLS 设计空间探索经历了自动化演进。最初,HLS 设计空间探索需要开发者凭借开发经验进行手动调整优化原语来逐步缩小开发空间,这使得高性能硬件的实现门槛极高,且效率低下[141];随后,开发者引入机器学习方法加速这一过程,通过训练网络预测不同 HLS 优化原语组合下的设计质量(例如功耗、时序、面积等)[140];近年来,领域内发展了更为自动化和智能化的搜索策略,例如利用抽象语法树对程序结构建模,结合瓶颈导向的坐标优化算法[142],实现对优化原语配置的自动推荐和探索,逐步弱化开发者对具体优化原语的使用。

基于 HLS 的敏捷开发,通过优化原语、设计模板和自动化设计空间探索方法的协同,显著提升了面向 FPGA 的 CNN 加速器设计效率。该技术路径在确保硬件性能与资源效率的前提下,大幅降低了开发迭代周期的人工成本,为边缘端 CNN 的快速部署与动态适配提供了关键技术支撑,并持续推动定制化加速器设计。

综上所述,面向 CPU-FPGA 异构平台的硬件设计优化是一个涵盖数字架构、核心算法、访存管理、任务调度及开发方法的多维协同优化过程。无论是追求极致低延迟的流式架构,还是注重通用性与资源效率的单计算引擎架构,或是各类软硬件协同与敏捷开发策略,其核心均在于依据具体的应用场景需求(如实时性、吞吐量、资源约束)在庞大的设计空间中寻找设计目标的帕累托最优解。不同的优化策略在带来性能增益的同时,也伴随着特定的资源开销或设计复杂度挑战。为直观呈现各层级优化方法的特性差异与适用边界,如表 2 所示,对前述关键技术进行了系统性梳理。该表从优化层级、核心机制与优势等维度,对当前主流的 CPU-FPGA 异构加速器硬件设计与优化策略进行了综合对比,旨在为不同约束条件下的边缘 AI 硬件架构选型与系统设计提供更具针对性的参考依据。

表 2 CPU-FPGA 异构硬件设计与优化策略对比

Table 2 Comparison of CPU-FPGA heterogeneous hardware design and optimization strategies

优化层级	技术	代表研究	核心机制	优势	局限性	适用场景
数字架构	流式架构	文献[101-102]	各层定制算子硬件,流水线级联执行	端到端延迟更低,吞吐率高且稳定	资源分配易失衡,深层网络受片上资源限制	固定网络,硬实时控制,单一任务
	单计算引擎	文献[11,101]	统一计算阵列复用,按层时分调度计算	资源利用率高,多网络切换更灵活	层间搬运开销明显,整体延迟略增大	多网络切换,深层网络,资源受限
数据流	行固定	文献[11]	输入与权重按行复用,滑窗流式推进计算	数据复用最大化,整体能效较高且均衡	控制逻辑更复杂,对片上缓存层级要求高	高吞吐能效优先,卷积主导的设计
	权重/输出固定	文献[112]	固定权重或输出驻留,复用部分和累加	显著降低带宽压力,提升阵列持续吞吐	部分和切换频繁,累加器容量需求较大	批量卷积计算,多通道输出融合场景
核心算法	Winograd	文献[115,117]	卷积变换到小域,增加加法以替代乘法	乘法数量显著减少,小核卷积速度提升	数值稳定性偏弱,变换与 LUT 开销增加	3×3 小卷积占主导,且对精度要求适中
	FFT	文献[114,116]	将卷积转频域点乘,再逆变换恢复输出	大核复杂度更低,适合超大分辨率输入	复数运算与存储开销大,小核时收益低	大于 5×5 卷积核,或超高分辨率任务
访存优化	数据分块	文献[11]	依据片上缓存容量分块,提升数据局部性	减少片外带宽需求,提高复用与吞吐稳定	边界与重叠处理复杂,实现与验证成本高	片上 BRAM 紧张,外存带宽受限系统
	数据预取	文献[126]	提前搬运后续数据,掩盖外存访问延迟	提升阵列利用率,降低等待造成的空闲	需要额外带宽与控制,预取失配会浪费	外存延迟敏感,计算易被访存拖慢
任务调度	静态调度	文献[134-135]	编译期确定映射与顺序,运行时按表执行	运行时开销近零,可预测性强便于验证	难适应动态负载变化,多分支支持较弱	结构固定,时延约束严格场景
	动态调度	文献[134,136]	运行时按负载分配任务,动态调整资源占用	负载更均衡,适配分支与稀疏等动态特征	引入管理与通信开销,控制逻辑更复杂	多分支网络,稀疏计算,负载波动大
开发流程	HLS 原语优化	文献[137]	通过展开与流水线等原语,细化算子结构	性能接近 RTL,便于快速迭代与调参优化	需较强硬件经验,调试与时序收敛较难	核心计算 IP 精细优化,极限性能敏感
	设计空间探索	文献[140,142]	自动搜索并评估参数组合,迭代优化约束	更快获得帕累托解,减少人工试错成本	搜索空间易爆炸,综合编译耗时较长	多参数协同权衡,需快速研发

4 发展前景与挑战

在大型语言模型(large language models, LLM)广泛普及的 AI 时代,计算范式正经历深刻变革。面对网络参数量的爆炸式增长,内存墙已成为制约性能的核心瓶颈,促使高端系统设计转向大规模并行 GPU 集群以及集成 HBM 的 3D 堆叠架构以提升算力密度和存储密度[143]。在此背景下,FPGA 基于其细粒度并行的架构特性,其技术定位正从通用的算子加速器,向着更具针对性的特定领域计算平台收敛。

1)复杂芯片前端开发与原型验证平台

面对张量处理器(tensor processing unit, TPU)、神经网络处理器(neural network processing unit, NPU)等 AI ASIC 日益缩短的迭代周期,FPGA 作为电路、IP 及系统设计的前端验证平台,其地位愈发重要。在 SoC 正式投片前,FPGA 提供了可运行实际软件栈的硬件环境,能够完成逻辑功能验证与原型实现,极大降低了开发风险[144]。此外,在量子计算等前沿领域,FPGA 凭借其灵活的可编程性,成为实现低温环境外围控制逻辑、微波脉冲生成与精密读出电路的首选平台[145]。这种定义算力标准的属性,使其成为芯片设计全流程中不可或缺的开发基石。

2)极低延迟与确定性计算的差异化竞争

与基于指令集架构并依赖操作系统调度的 CPU、GPU 不同,FPGA 支持裸板运行,能够实现精确至时钟周期的指令执行与数据流处理。这种高度确定性和极低延迟特性,使其在高能物理实验的数据流负载均衡与实时预处理中具有优势[146]。在 LLM 集群中,FPGA 也可承担高性能智能网卡功能,通过卸载通信协议栈与支持 GPU 直接访问技术有效缓解节点间的通信延迟[147]。

3)算法快速迭代期的领域定制计算

针对自动驾驶、人形机器人等算法尚未固化的新兴场景,ASIC 由于研发周期较长、固化后难以更改,难以适配快速演进的 Transformer 变体或多模态融合模型[148-149]。相比之下,FPGA 凭借混合精度定制位宽(如 INT4/FP8)和芯粒异构集成技术[150-151],能够以更短的周期实现实际算力性价比的提升。特别是在边缘端在线学习场景下,FPGA 可灵活实现非反向传播等新型训练范式[152-153],使设备具备“终身学习”能力,同时确保数据隐私安全[154]。

5 结 论

CNN 已成为深度学习领域一类重要的神经网络,但其日益增长的计算需求与边缘端严苛的资源、能效约束

之间的矛盾愈发凸显。基于 FPGA 的异构计算平台凭借定制化的数据流设计、高度的可重构性以及确定性的低延迟优势,为低批量、高实时性的边缘端推理任务提供了理想的硬件支撑。在探讨面向 CPU-FPGA 平台部署的轻量化与加速方法时,软硬件协同设计的视角贯穿始终。全篇在算法层面阐述了通过剪枝、量化等技术从源头压缩网络规模的必要性,在硬件架构层面分析了流式架构与单计算引擎架构的演进范式,并结合 Roofline 模型明确了系统性能优化的边界。这种从算法压缩到硬件映射的全链路优化逻辑预示着,算法轻量化与硬件定制化的深度耦合是缓解访存带宽受限与算力密度不足挑战的有效方式,也是进一步提升系统综合能效的关键所在。

在此趋势下,FPGA 不仅作为 AI ASIC 前端逻辑验证、原型设计及量子计算外围电路的核心验证基石,还凭借其无指令集裸机运行的特性,在如高能物理、工业控制等对时序要求严苛的场景中充当提供确定性低延迟的实时计算引擎。此外,针对自动驾驶、人形机器人等算法尚未固化且快速迭代的领域,FPGA 作为敏捷创新载体,有效弥补了 ASIC 研发周期长、灵活性差的短板,为新型计算范式的探索提供了高性价比的算力支持。综上所述,尽管面临内存墙与通用 GPU 生态的挑战,FPGA 凭借其独特的架构灵活性与可定制性,将在后摩尔时代的异构计算体系中持续发挥关键且不可替代的作用。

参考文献

[1] KAZEMIAN A, ELMOZNINO E, BONNER M F. Convolutional architectures are cortex-aligned de novo[J]. Nature Machine Intelligence, 2025, 7(11): 1834-1844.

[2] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016: 770-778.

[3] GUAN J, LAI R, LU Y, et al. Memory-efficient deformable convolution based joint denoising and demosaicing for UHD images[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2022, 32(11):7346-7358.

[4] 彭宇, 姬森展, 于希明, 等. 语义分割网络的 FPGA 加速计算方法综述[J]. 仪器仪表学报, 2021, 42(9): 1-12.
PENG Y, JI S ZH, YU X M, et al. Review of FPGA accelerated computing methods for semantic segmentation networks[J]. Chinese Journal of Scientific Instrument, 2021, 42(9): 1-12.

[5] HIRSCHMAN J, KAMALOV A, OBAID R, et al. At-the-edge data processing for low latency high throughput machine learning algorithms[C]. Smoky Mountains Computational Sciences and Engineering Conference. 2022: 101-119.

[6] LIN J, CHEN W-M, LIN Y, et al. MCUnet: Tiny deep learning on IoT devices[C]. Advances in Neural Information Processing Systems (NeurIPS). 2020, 33: 11711-11722.

[7] 焦李成, 孙其功, 杨育婷, 等. 深度神经网络 FPGA 设计进展、实现与展望[J]. 计算机学报, 2022, 45(3): 441-471.
JIAO L CH, SUN Q G, YANG Y T, et al. Progress, implementation and prospects of deep neural network FPGA design[J]. Chinese Journal of Computers, 2022, 45(3): 441-471.

[8] JIANG J, ZHOU Y, GONG Y, et al. FPGA-based acceleration for convolutional neural networks: A comprehensive review[J]. ArXiv preprint arXiv: 250513461, 2025.

[9] HONG H, CHOI D, KIM N, et al. Survey of convolutional neural network accelerators on field-programmable gate array platforms: Architectures and optimization techniques[J]. Journal of Real-Time Image Processing, 2024, 21(3): 64.

[10] 张坤, 高博, 冀亚玮, 等. 基于 FPGA 的卷积神经网络加速器现状研究[J]. 太赫兹科学与电子信息学报, 2024, 22(10): 1142-1153,1167.
ZHANG K, GAO B, JI Y W, et al. Research on the status of convolutional neural network accelerators based on FPGA[J]. Journal of Terahertz Science and Electronic Information Technology, 2024, 22(10): 1142-1153,1167.

[11] CHEN Y-H, 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.

[12] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84-90.

[13] LONG J, SHELHAMER E, DARRELL T. Fully convolutional networks for semantic segmentation[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015: 3431-3440.

[14] ZHAO H, SHI J, QI X, et al. Pyramid scene parsing network[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017: 2881-2890.

[15] CHOLLET F. Xception: Deep learning with depthwise separable convolutions[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017: 1251-1258.

[16] DAI J, QI H, XIONG Y, et al. Deformable convolutional networks[C]. Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2017: 764-773.

[17] MAI A, TRAN L, TRAN L, et al. VGG deep neural network compression via SVD and CUR decomposition techniques[C]. 2020 7th NAFOSTED Conference on Information and Computer Science (NICS). 2020: 118-123.

[18] IOFFE S, SZEGEDY C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]. International Conference on Machine Learning (ICML). 2015, 37: 448-456.

[19] XU J, SUN X, ZHANG Z, et al. Understanding and improving layer normalization[C]. Advances in Neural Information Processing Systems (NeurIPS). 2019, 32: 4381-4391.

[20] HUANG X, BELONGIE S. Arbitrary style transfer in real-time with adaptive instance normalization[C]. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017: 1501-1510.

[21] WU Y, HE K. Group normalization[C]. Proceedings of the European Conference on Computer Vision (ECCV), 2018: 3-19.

[22] LAN H, MENG J, HUNDT C, et al. FeatherCNN: Fast inference computation with tensorgemm on ARM architectures[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 31(3): 580-594.

[23] DING B, QIAN H, ZHOU J. Activation functions and their characteristics in deep neural networks[C]. 2018 Chinese Control and Decision Conference (CCDC). IEEE, 2018: 1836-1841.

[24] GLOROT X, BENGIO Y. Understanding the difficulty of training deep feedforward neural networks[C]. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS), 2010, 9: 249-256.

[25] MAAS A L, HANNUN A Y, NG A Y. Rectifier nonlinearities improve neural network acoustic models[C]. International Conference on Machine Learning (ICML), 2013, 28: 1-6.

[26] SZE V, CHEN Y H, YANG T J, et al. Efficient processing of deep neural networks: A tutorial and survey[J]. Proceedings of the IEEE, 2017, 105(12): 2295-2329.

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

[28] SHADOUL I, AL-HMOUZ R, HOSSEN A, et al. The effect of pooling parameters on the performance of convolution neural network[J]. Artificial Intelligence Review, 2025, 58(9): 271.

[29] SOOMRO T A, AFIFI A J, GAO J, et al. Strided fully convolutional neural network for boosting the sensitivity of retinal blood vessels segmentation[J]. Expert Systems with Applications, 2019, 134: 36-52.

[30] SRIVASTAVA N, HINTON G, KRIZHEVSKY A, et al. Dropout: A simple way to prevent neural networks from overfitting[J]. The Journal of Machine Learning Research, 2014, 15(1): 1929-1958.

[31] REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: Unified, real-time object detection[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016: 779-788.

[32] HOROWITZ M. 1.1 Computing's energy problem (and what we can do about it)[C]. 2014 IEEE International Solid-State Circuits Conference (ISSCC), 2014: 10-14.

[33] HAN S, LIU X, MAO H, et al. Eie: Efficient inference engine on compressed deep neural network[J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 243-254.

[34] MENG X, CHEN W, BENBAKI R, et al. Falcon: Flop-aware combinatorial optimization for neural network pruning[C]. International Conference on Artificial Intelligence and Statistics (AISTATS), 2024, 238: 4384-4392.

[35] YUAN T, LI Z, LIU B, et al. Arpruning: An automatic channel pruning based on attention map ranking[J]. Neural Networks, 2024, 174: 106220.

[36] CAI L, AN Z, YANG C, et al. Prior gradient mask guided pruning-aware fine-tuning[C]. Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2022, 36(1): 140-148.

[37] RUAN X, LIU Y, LI B, et al. Dpfps: Dynamic and progressive filter pruning for compressing convolutional neural networks from scratch[C]. Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2021, 35(3): 2495-2503.

[38] GUPTA A, BAU T, KIM J, et al. Torque based structured pruning for deep neural network[C]. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2024: 2711-2720.

[39] LU M, LUO X, CHEN T, et al. Learning pruning-friendly networks via frank-wolfe: One-shot, any-sparsity, and no retraining[C]. International Conference on Learning Representations (ICLR), 2022.

[40] DE RESENDE OLIVEIRA F D, BATISTA E L O,

SEARA R. On the compression of neural networks using ℓ0-norm regularization and weight pruning[J]. Neural Networks, 2024, 171: 343-352.

[41] LEE D, LEE E, HWANG Y. Pruning from scratch via shared pruning module and nuclear norm-based regularization[C]. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2024: 1393-1402.

[42] SHEN M, MOLCHANOV P, YIN H, et al. When to prune? A policy towards early structural pruning[C]. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022: 12247-12256.

[43] WANG H, FU Y. Trainability preserving neural pruning[C]. International Conference on Learning Representations (ICLR), 2022.

[44] MURTI C, NARSHANA T, BHATTACHARYYA C. Tvsprune-pruning non-discriminative filters via total variation separability of intermediate representations without fine tuning[C]. The Eleventh International Conference on Learning Representations (ICLR), 2022.

[45] CHANG J, LU Y, XUE P, et al. Iterative clustering pruning for convolutional neural networks[J]. Knowledge-Based Systems, 2023, 265: 110386.

[46] HAN S, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural network[J]. Advances in Neural Information Processing Systems, 2015, 28: 1135-1143.

[47] ZHANG T, YE S, ZHANG K, et al. A systematic DNN weight pruning framework using alternating direction method of multipliers[C]. Proceedings of the European Conference on Computer Vision (ECCV), 2018: 184-199.

[48] HAN S, MAO H, DALLY W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[C]. International Conference on Learning Representations (ICLR), 2016.

[49] HE Z, QIAN Y, WANG Y, et al. Filter pruning via feature discrimination in deep neural networks[C]. European Conference on Computer Vision (ECCV), 2022: 245-261.

[50] MOLCHANOV P, TYREE S, KARRAS T, et al. Pruning convolutional neural networks for resource efficient inference[C]. International Conference on Learning Representations (ICLR), 2017.

[51] LIU Z, LI J, SHEN Z, et al. Learning efficient convolutional networks through network slimming[C]. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017: 2736-2744.

[52] WEN W, WU C, WANG Y, et al. Learning structured sparsity in deep neural networks[J]. Advances in Neural Information Processing Systems, 2016, 29: 2082-2090.

[53] CHEN T, JI B, DING T, et al. Only train once: A one-shot neural network training and pruning framework[J]. Advances in Neural Information Processing Systems, 2021, 34: 19637-19651.

[54] HU P, PENG X, ZHU H, et al. Opq: Compressing deep neural networks with one-shot pruning-quantization[C]. Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2021, 35(9): 7780-7788.

[55] TAN C M J, MOTANI M. Dropnet: Reducing neural network complexity via iterative pruning[C]. International Conference on Machine Learning (ICML). PMLR, 2020, 119: 9356-9366.

[56] CHANG J, LU Y, XUE P, et al. Global balanced iterative pruning for efficient convolutional neural networks[J]. Neural Computing and Applications, 2022, 34(23): 21119-21138.

[57] SONG Q, DAI Y, LU H, et al. High-throughput systolic array-based accelerator for hybrid transformer-CNN networks[J]. Journal of King Saud University-Computer and Information Sciences, 2024, 36(8): 102194.

[58] HUBARA I, COURBARIAUX M, SOUDRY D, et al. Binarized neural networks[J]. Advances in Neural Information Processing Systems, 2016, 29: 4114-4122.

[59] ZHU C, HAN S, MAO H, et al. Trained ternary quantization[C]. International Conference on Learning Representations (ICLR), 2017.

[60] RASTEGARI M, ORDONEZ V, REDMON J, et al. Xnor-net: Imagenet classification using binary convolutional neural networks[C]. European Conference on Computer Vision (ECCV), 2016: 525-542.

[61] UMUROGLU Y, FRASER N J, GAMBARDELLA G, et al. Finn: A framework for fast, scalable binarized neural network inference[C]. Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), 2017: 65-74.

[62] LIU Z, CHENG K-T, HUANG D, et al. Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022: 4942-4952.

[63] OH S, SIM H, KIM J, et al. Non-uniform step size quantization for accurate post-training quantization[C]. European Conference on Computer Vision (ECCV), 2022: 658-673.

[64] ZHAO X, WANG Y, CAI X, et al. Linear symmetric quantization of neural networks for low-precision integer hardware [C]. International Conference on Learning Representations (ICLR), 2020.

[65] CHIEN J-T, CHANG S-T. Bayesian asymmetric quantized neural networks [J]. Pattern Recognition, 2023, 139: 109463.

[66] MALY J, SAAB R. A simple approach for quantizing neural networks [J]. Applied and Computational Harmonic Analysis, 2023, 66: 138-150.

[67] OSAMA A, GADALLAH S I, SAID L A, et al. Chaotic neural network quantization and its robustness against adversarial attacks [J]. Knowledge-Based Systems, 2024, 286: 111319.

[68] GUPTA S, AGRAWAL A, GOPALAKRISHNAN K, et al. Deep learning with limited numerical precision[C]. International Conference on Machine Learning (ICML). PMLR, 2015, 37: 1737-1746.

[69] GHOLAMI A, KIM S, DONG Z, et al. A survey of quantization methods for efficient neural network inference[M]. Low-Power Computer Vision. Chapman and Hall/CRC, 2022: 291-326.

[70] XIAO G, LIN J, SEZNEC M, et al. Smoothquant: Accurate and efficient post-training quantization for large language models[C]. International Conference on Machine Learning (ICML). PMLR, 2023, 202: 38087-38099.

[71] LI Y, GONG R, TAN X, et al. Brecq: Pushing the limit of post-training quantization by block reconstruction[C]. International Conference on Learning Representations (ICLR), 2021.

[72] JACOB B, KLIGYS S, CHEN B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018: 2704-2713.

[73] CAI Y, YAO Z, DONG Z, et al. Zeroq: A novel zero shot quantization framework [C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020: 13169-13178.

[74] CHEN X, WANG Y, YAN R, et al. Texq: Zero-shot network quantization with texture feature distribution calibration [J]. Advances in Neural Information Processing Systems, 2023, 36: 274-287.

[75] ZHONG Y, LIN M, NAN G, et al. Intraq: Learning synthetic images with intra-class heterogeneity for zero-shot network quantization[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022: 12339-12348.

[76] DENTON E L, ZAREMBA W, BRUNA J, et al. Exploiting linear structure within convolutional networks for efficient evaluation [J]. Advances in Neural Information Processing Systems, 2014, 27.

[77] IOANNOU Y, ROBERTSON D, SHOTTON J, et al. Training CNNs with low-rank filters for efficient image classification[C]. International Conference on Learning Representations (ICLR), 2016.

[78] PHAN A-H, SOBOLEV K, SOZYKIN K, et al. Stable low-rank tensor decomposition for compression of convolutional neural network[C]. European Conference on Computer Vision (ECCV), 2020: 522-539.

[79] LIU C, XIE K, WEN J, et al. An accuracy-preserving neural network compression via tucker decomposition[J]. IEEE Transactions on Sustainable Computing, 2024, 10(2): 262-273.

[80] WEN W, XU C, WU C, et al. Coordinating filters for faster deep neural networks [C]. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017: 658-666.

[81] IDELBAYEV Y, CARREIRA-PERPINáN M A. Low-rank compression of neural nets: Learning the rank of each layer [C]. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020: 8049-8059.

[82] SWAMINATHAN S, GARG D, KANNAN R, et al. Sparse low rank factorization for deep neural network compression [J]. Neurocomputing, 2020, 398: 185-196.

[83] HINTON G, VINYALS O, DEAN J. Distilling the knowledge in a neural network [J]. ArXiv preprint arXiv:150302531, 2015.

[84] GOU J, YU B, MAYBANK S J, et al. Knowledge distillation: A survey [J]. International Journal of Computer Vision, 2021, 129(6): 1789-1819.

[85] 于轲鑫, 琚赟. 基于知识蒸馏的边缘侧电能质量扰动识别研究[J]. 电力信息与通信技术, 2023, 21(7): 36-43.
YU K X, JYU Y. Research on power quality disturbance recognition on edge side based on knowledge distillation[J]. Electric Power Information and Communication Technology, 2023, 21(7): 36-43.

[86] ZHENG Z, PENG X. Self-guidance: Improve deep neural network generalization via knowledge distillation [C]. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022: 3203-3212.

[87] YUN S, PARK J, LEE K, et al. Regularizing class-wise

predictions via self-knowledge distillation[C]. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020: 13876-13885.

[88] KATO S, LAKSHMANAN K, RAJKUMAR R, et al. Timegraph: GPU scheduling for real-time multi-tasking environments[C]. 2011 USENIX Annual Technical Conference (USENIX ATC 11), 2011.

[89] 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 (FPGA), 2015: 161-170.

[90] 韩萍, 白继睿, 周杰龙, 等. 基于轻量化改进和模型剪枝的 SAR 图像飞机目标检测[J]. 仪器仪表学报, 2025, 46(9): 110-124.
HAN P, BAI J R, ZHOU J L, et al. Lightweight SAR image aircraft target detection based on lightweight improvement and model pruning[J]. Chinese Journal of Scientific Instrument, 2025, 46(9): 110-124.

[91] WILLIAMS S, WATERMAN A, PATTERSON D. Roofline: An insightful visual performance model for multicore architectures[J]. Communications of the ACM, 2009, 52(4): 65-76.

[92] SHAO Y S, REAGEN B, WEI G-Y, et al. Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures[J]. ACM SIGARCH Computer Architecture News, 2014, 42(3): 97-108.

[93] MAKRANI H M, SAYADI H, MOHSENIN T, et al. Xppe: Cross-platform performance estimation of hardware accelerators using machine learning[C]. Proceedings of the 24th Asia and South Pacific Design Automation Conference, 2019: 727-732.

[94] KWON H, CHATARASI P, SARKAR V, et al. Maestro: A data-centric approach to understand reuse, performance, and hardware cost of DNN mappings[J]. IEEE Micro, 2020, 40(3): 20-29.

[95] PARASHAR A, RAINA P, SHAO Y S, et al. Timeloop: A systematic approach to DNN accelerator evaluation[C]. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE, 2019: 304-315.

[96] PELLIZZONI R, SCHRANZHOFER A, CHEN J J, et al. Worst case delay analysis for memory interference in multicore systems[C]. 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010). IEEE, 2010: 741-746.

[97] CHEN T, MOREAU T, JIANG Z, et al. TVM: An automated End-to-End optimizing compiler for deep learning[C]. 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), 2018: 578-594.

[98] 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.

[99] 杨宏浩, 隋欣宇, 王一然, 等. 基于 FPGA 的高效近似乘法器设计[J]. 电子测量与仪器学报, 2025, 39(9): 224-232.
YANG H H, SUI X Y, WANG Y R, et al. FPGA-based design of high-efficiency approximate multipliers[J]. Journal of Electronic Measurement and Instrumentation, 2025, 39(9): 224-232.

[100] JOUPPI N, KURIAN G, LI S, et al. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings[C]. Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA), 2023: 1-14.

[101] VENIERIS S I, KOURIS A, BOUGANIS C-S. Toolflows for mapping convolutional neural networks on FPGAs: A survey and future directions[J]. ACM Computing Surveys (CSUR), 2018, 51(3): 1-39.

[102] CHEN H, ZHANG J, DU Y, et al. Understanding the potential of FPGA-based spatial acceleration for large language model inference[J]. ACM Transactions on Reconfigurable Technology and Systems, 2024, 18(1): 1-29.

[103] ZENG A, SONG S, LEE J, et al. Tossingbot: Learning to throw arbitrary objects with residual physics[J]. IEEE Transactions on Robotics, 2020, 36(4): 1307-1319.

[104] ZHANG X, LU H, HAO C, et al. SkyNet: A hardware-efficient method for object detection and tracking on embedded systems[J]. Proceedings of Machine Learning and Systems, 2020, 2: 216-229.

[105] ANANTHANARAYANAN G, BAHL P, BODíK P, et al. Real-time video analytics: The killer app for edge computing[J]. Computer, 2017, 50(10): 58-67.

[106] SATYANARAYANAN M. The emergence of edge computing[J]. Computer, 2017, 50(1): 30-39.

[107] LIU Z, DOU Y, JIANG J, et al. Throughput-optimized FPGA accelerator for deep convolutional neural networks[J]. ACM Transactions on Reconfigurable Technology and Systems, 2017, 10(3): 1-23.

[108] WANG E, DAVIS J J, ZHAO R, et al. Deep neural network approximation for custom hardware: Where we've

been, where we're going[J]. ACM Computing Surveys (CSUR), 2019, 52(2): 1-39.

[109] GUO K, SUI L, QIU J, et al. Angel-eye: A complete design flow for mapping CNN onto embedded FPGA[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(1): 35-47.

[110] JOUPPI N P, YOUNG C, PATIL N, et al. In-datacenter performance analysis of a tensor processing unit[C]. Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA), 2017: 1-12.

[111] SZE V, CHEN Y H, EMER J, et al. Hardware for machine learning: Challenges and opportunities[C]. IEEE Custom Integrated Circuits Conference (CICC). IEEE, 2017: 1-8.

[112] GENC H, KIM S, AMID A, et al. Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration[C]. 2021 58th ACM/IEEE Design Automation Conference (DAC). IEEE, 2021: 769-774.

[113] PRADELS L, FILIP S-I, SENTIEYS O, et al. FPGA-based CNN acceleration using pattern-aware pruning[C]. 2024 IEEE 6th International Conference on AI Circuits and Systems (AICAS). IEEE, 2024: 228-232.

[114] MATHIEU M, HENAFF M, LECUN Y. Fast training of convolutional networks through FFTs[J]. ArXiv preprint arXiv:1312.5851, 2013.

[115] LAVIN A, GRAY S. Fast algorithms for convolutional neural networks[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016: 4013-4021.

[116] LIANG Y, LU L, XIAO Q, et al. Evaluating fast algorithms for convolutional neural networks on FPGAs[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(4): 857-870.

[117] WONG Y, DONG Z, ZHANG W. Low bitwidth CNN accelerator on FPGA using winograd and block floating point arithmetic[C]. 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 2021: 218-223.

[118] LIU X, CHEN Y, HAO C, et al. WinoCNN: Kernel sharing winograd systolic array for efficient convolutional neural network acceleration on FPGAs[C]. 2021 IEEE 32nd International Conference on Application-Specific Systems, Architectures and Processors (ASAP). IEEE, 2021: 258-265.

[119] ABTAHI T, SHEA C, KULKARNI A, et al. Accelerating convolutional neural network with FFT on embedded hardware[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2018, 26(9): 1737-1749.

[120] WANG Y, QIN Y, DENG D, et al. Trainer: An energy-efficient edge-device training processor supporting dynamic weight pruning[J]. IEEE Journal of Solid-State Circuits, 2022, 57(10): 3164-3178.

[121] PARASHAR A, RHU M, MUKKARA A, et al. SCNN: An accelerator for compressed-sparse convolutional neural networks[J]. ACM SIGARCH Computer Architecture News, 2017, 45(2): 27-40.

[122] LU L, XIE J, HUANG R, et al. An efficient hardware accelerator for sparse convolutional neural networks on FPGAs[C]. 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, 2019: 17-25.

[123] LU J, LIU D, CHENG X, et al. An efficient unstructured sparse convolutional neural network accelerator for wearable ECG classification device[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(11): 4572-4582.

[124] ZHANG J-F, LEE C-E, LIU C, et al. Snap: An efficient sparse neural acceleration processor for unstructured sparse deep neural network inference[J]. IEEE Journal of Solid-State Circuits, 2020, 56(2): 636-647.

[125] DENG C, SUI Y, LIAO S, et al. Gospa: An energy-efficient high-performance globally optimized sparse convolutional neural network accelerator[C]. 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2021: 1110-1123.

[126] ZHANG Z, WANG H, HAN S, et al. Sparch: Efficient architecture for sparse matrix multiplication[C]. 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2020: 261-274.

[127] AMROUCH H, DU N, GEBREGIORGIS A, et al. Towards reliable in-memory computing: From emerging devices to post-von-neumann architectures[C]. 2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC). IEEE, 2021: 1-6.

[128] SINGH G, CHELINI L, CORDA S, et al. Near-memory computing: Past, present, and future[J]. Microprocessors and Microsystems, 2019, 71: 102868.

[129] GUPTA S, IMANI M, KAUR H, et al. Nnpim: A processing in-memory architecture for neural network acceleration[J]. IEEE Transactions on Computers,

2019, 68(9): 1325-1337.

[130] FARMAHINI-FARAHANI A, AHN J H, MORROW K, et al. Nda: Near-DRAM acceleration architecture leveraging commodity DRAM devices and standard memory modules[C]. 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2015: 283-295.

[131] KABIR M A, KABIR E, HOLLIS J, et al. FPGA processor in memory architectures (PIMs): Overlay or overhaul? [C]. 2023 33rd International Conference on Field-Programmable Logic and Applications (FPL). IEEE, 2023: 109-115.

[132] WANG Y, ZOU Z, ZHENG L. Design framework for SRAM-based computing-in-memory edge CNN accelerators[C]. 2021 International Symposium on Circuits and Systems (ISCAS). IEEE, 2021: 1-5.

[133] HAMDIOUI S, XIE L, DU NGUYEN H A, et al. Memristor based computation-in-memory architecture for data-intensive applications [C]. 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2015: 1718-1725.

[134] TIANYANG L, FAN Z, WEI G, et al. A survey: FPGA-based dynamic scheduling of hardware tasks[J]. Chinese Journal of Electronics, 2021, 30(6): 991-1007.

[135] MOSS D J M, NURVITADHI E, SIM J, et al. High performance binary neural networks on the Xeon+FPGA platform[C]. 2017 27th International Conference on Field Programmable Logic and Applications (FPL). IEEE, 2017: 1-4.

[136] GAO Y, ZHANG B, QI X, et al. Dpacs: Hardware accelerated dynamic neural network pruning through algorithm-architecture co-design[C]. Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2023, 2: 237-251.

[137] 周彦臻, 吴瑞东, 于潇, 等. 面向 FPGA 部署的 CNN-SVM 算法研究与实现[J]. 电子测量与仪器学报, 2021, 35(4): 90-98.
ZHOU Y ZH, WU R D, YU X, et al. Research and implementation of CNN-SVM algorithm for FPGA deployment[J]. Journal of Electronic Measurement and Instrumentation, 2021, 35(4): 90-98.

[138] YU Y, ZHAO T, WANG K, et al. Light-opu: An FPGA-based overlay processor for lightweight convolutional neural networks[C]. Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), 2020: 122-132.

[139] LIU L, ZHU J, LI Z, et al. A survey of coarse-grained reconfigurable architecture and design: Taxonomy, challenges, and applications [J]. ACM Computing Surveys (CSUR), 2019, 52(6): 1-39.

[140] AHMED M R, KOIKE-AKINO T, PARSONS K, et al. AutoHLS: Learning to accelerate design space exploration for HLS designs [C]. 2023 IEEE 66th International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE, 2023: 491-495.

[141] LAI Y-H, CHI Y, HU Y, et al. HeteroCL: A multi-paradigm programming infrastructure for software-defined reconfigurable computing[C]. Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), 2019: 242-251.

[142] SOHRABIZADEH A, YU C H, GAO M, et al. AutoDSE: Enabling software programmers to design efficient FPGA accelerators[J]. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2022, 27(4): 1-27.

[143] SHARDA J, MANLEY M, KWAK J, et al. 3D stacked HBM and compute accelerators for LLM: Optimizing thermal management and power delivery efficiency[J]. IEEE Journal on Exploratory Solid-State Computational Devices and Circuits, 2025, 11: 116-122.

[144] CHEN S, ZHANG H, LING Z, et al. The survey of 2.5D integrated architecture: An EDA perspective[C]. Proceedings of the 30th Asia and South Pacific Design Automation Conference, 2025: 285-293.

[145] XU Y, HUANG G, BALEWSKI J, et al. QubiC: An open-source FPGA-based control and measurement system for superconducting quantum information processors [J]. IEEE Transactions on Quantum Engineering, 2021, 2: 1-11.

[146] BALDIN I, GOODRICH M, GYURJYAN V, et al. Low latency, high bandwidth streaming of experimental data with ejfat[J]. ArXiv preprint arXiv:251012597, 2025.

[147] WANG Z, HUANG H, ZHANG J, et al. FPGANIC: An FPGA-based versatile 100gb smartNIC for GPUs[C]. 2022 USENIX Annual Technical Conference (USENIX ATC), 2022: 967-986.

[148] GUPTA A, SAVARESE S, GANGULI S, et al. Embodied intelligence via learning and evolution[J]. Nature Communications, 2021, 12(1): 5721.

[149] 彭宇, 季拓, 郭楚亮. 故障预测与健康管理领域大语言模型:应用与展望[J]. 仪器仪表学报, 2025, 46(10): 2-21.
PENG Y, JI T, GUO CH L. Prognostics and health management domain-specific large language models:

Applications and prospects [J]. Chinese Journal of Scientific Instrument, 2025,46(10):2-21.

[150] TANG W, CHO S G, HOANG T T, et al. Arvon: A heterogeneous system-in-package integrating FPGA and DSP chiplets for versatile workload acceleration [J]. IEEE Journal of Solid-State Circuits, 2024, 59(4): 1235-1245.

[151] LIU Y, LI X, YIN S. Review of chiplet-based design: System architecture and interconnection [J]. Science China Information Sciences, 2024, 67(10): 200401.

[152] CHEN A, ZHANG Y, JIA J, et al. Deepzero: Scaling up zeroth-order optimization for deep model training [C]. International Conference on Learning Representations (ICLR), 2024.

[153] HINTON G. The forward-forward algorithm: Some preliminary investigations [J]. ArXiv preprint arXiv: 221213345, 2022, 2(3):5.

[154] WANG R, GAO Z, ZHANG L, et al. Empowering large language models to edge intelligence: A survey of edge efficient LLMs and techniques [J]. Computer Science Review, 2025, 57: 100755.

作者简介

郭楚亮,2019 年于哈尔滨工业大学获得学士学位,2024 年于浙江大学获得工学博士学位,现为哈尔滨工业大学副研究员,主要研究方向为 FPGA 加速、边缘 AI 软硬协同设计。

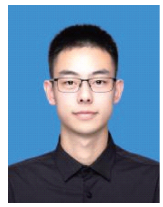
E-mail: chuliang007@hit.edu.cn

Guo Chuliang received his B. Sc. degree from Harbin Institute of Technology in 2019, and Ph. D. degree from Zhejiang University in 2024. He is currently a research associate professor in Harbin Institute of Technology. His main research interests include FPGA-based acceleration and co-design for edge AI applications.

娄越,2024 年于哈尔滨工业大学获得学士学位,现为哈尔滨工业大学硕士研究生,主要研究方向为 FPGA 硬件设计、高精度时间同步技术、异构系统软硬协同加速。

E-mail: 24S105153@stu.hit.edu.cn

Lou Yue received her B. Sc. degree from Harbin Institute of Technology in 2024. She is currently a M. Sc. candidate in Harbin Institute of Technology. Her main research interests include FPGA hardware design, high-precision time synchronization technology, and hardware-software co-acceleration of heterogeneous systems.



岳豪帅,2024 年于中国矿业大学获得学士学位,现为哈尔滨工业大学硕士研究生,主要研究方向为面向异构计算平台的深度学习推理优化、神经网络轻量化。

E-mail: 24S105183@stu.hit.edu.cn

Yue Haoshuai received his B. Sc. degree from China University of Mining and Technology in 2024. He is currently a M. Sc. candidate in Harbin Institute of Technology. His main research interests include deep learning inference optimization on heterogeneous computing platforms, as well as neural network lightweighting.

季拓,2023 年于哈尔滨工业大学获得学士学位,现为哈尔滨工业大学硕士研究生,主要研究方向为神经网络与深度学习、故障预测与健康管理、大语言模型。

E-mail: 24S105138@stu.hit.edu.cn

Ji Tuo received his B. Sc. degree from Harbin Institute of Technology in 2023. He is currently a M. Sc. candidate in Harbin Institute of Technology. His main research interests include neural networks and deep learning, prognostics and health management, and large language models.

程谓哲,2024 年于东北大学获得学士学位,现为哈尔滨工业大学硕士研究生,主要研究方向为神经网络与深度学习、计算机视觉、无人机智能感知。

E-mail: 24S105121@stu.hit.edu.cn

Cheng Xuzhe received his B. Sc. degree from Northeastern University in 2024. He is currently a M. Sc. candidate in Harbin Institute of Technology. His main research interests include neural networks and deep learning, computer vision, and UAV intelligent perception.

彭宇(通信作者),2004 年于哈尔滨工业大学获得博士学位,现为哈尔滨工业大学教授、博士生导师,主要研究方向为虚拟仪器和自动测试、故障预测与健康管理、可重构计算等。

E-mail: pengyu@hit.edu.cn

Peng Yu(Corresponding author) received his Ph. D. degree from Harbin Institute of Technology in 2004. He is currently a professor and Ph. D. advisor at Harbin Institute of Technology. His main research interests include virtual instruments and automatic test technologies, prognostics and health management, and reconfigurable computing, etc.