端智能推理加速技术综述

章晋睿[1],龙婷婷[2],张德宇[2],许 愿[2],任 炬[1*],张尧学[1]
(1. 清华大学计算机与科学技术系,北京 100084;2. 中南大学计算机学院,湖南长沙 410083)

摘 要: 智能下沉是迈向泛在智能时代的必经之路,也推动了端智能(on-device intelligence)技术的飞速发展.通过在终端设备直接部署运行深度学习模型,端智能在实时性、安全性、个性化等方面具有天然优势,已在自动驾驶、卫星侦察、虚拟现实/增强现实(Virtual Reality/Augmented Reality,VR/AR)等众多场景广泛应用.然而,随着深度学习模型参数量不断增大,端侧受限的硬件资源已难以支撑不断增长的计算开销.为提升终端设备在模型推理的计算效率,研究人员从模型算法、编译软件、设备硬件等多个层面开展了系统性优化,有效推动了端智能的发展与演进.本文从算法、软硬件结合优化等方面对现有端侧深度学习模型推理优化工作进行了总结,涵盖模型压缩技术、模型-软件-硬件的协同设计、模型异构并行部署策略以及大模型的端侧优化技术.最后,本文梳理了当前端智能推理加速技术所面临的挑战,并对未来发展趋势进行了展望.

关键词: 端智能;模型压缩;推理加速;深度学习;软硬件结合优化
基金项目: 国家重点研发计划(No.2022YFF0604502);国家自然科学基金(No.62122095,No.62341201)
中图分类号: TP393.0 **文献标识码:** A **文章编号:** 0372-2112(2025)04-1063-40
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20240691

On-Device Intelligence Acceleration Technologies: A Survey

ZHANG Jin-rui[1], LONG Ting-ting[2], ZHANG De-yu[2], XU Yuan[2], REN Ju[1*], ZHANG Yao-xue[1]
(1. *Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084, *China*;
2. *School of Computer Science and Engineering*, *Central South University*, *Changsha*, *Hunan* 410083, *China*)

Abstract: Intelligent edge computing is an essential pathway towards the era of pervasive intelligence, and it has propelled the rapid advancement of on-device intelligence technology. By directly deploying and running deep learning models on edge devices, on-device intelligence holds natural advantages in real-time processing, security, and personalization, among other aspects, and has found extensive applications in various scenarios such as autonomous driving, satellite reconnaissance, virtual reality/augmented reality (VR/AR), and more. However, as the parameters of deep learning models continue to increase, the limited hardware resources at the edge struggle to sustain the growing computational costs. To enhance the computational efficiency of model inference on edge devices, researchers have systematically optimized from multiple perspectives including model algorithms, compilation software, and device hardware, driving the advancement and evolution of on-device intelligence. This paper summarizes existing optimization efforts for deep learning model inference at the edge, covering techniques such as model compression, collaborative design of model-software-hardware, heterogeneous model parallel deployment strategies, and optimizations for large models. Lastly, it outlines the challenges faced by current on-device intelligence inference acceleration technologies and provides insights into future development trends.

Key words: on-device intelligence; model compression; inference acceleration; deep learning; collaborative design of model-software-hardware
Foundation Item(s): National Key Research and Development Program of China (No.2022YFF0604502); National Natural Science Foundation of China (No.62122095, No.62341201)

1 引言

随着智能手机和智能可穿戴设备等移动终端的普及,其性能持续提升,计算与存储能力显著增强.与此同时,以深度学习为代表的人工智能(Artificial Intelli-

收稿日期:2024-07-22;修回日期:2025-01-15;责任编辑:宋甲英
*通讯作者:任炬

gence,AI)技术在图像识别、语音识别、自然语言处理等领域取得了显著进展. 端智能(on-device intelligence)作为一种将终端设备计算与AI相结合的新兴计算范式,通过在设备本地进行深度学习模型推理,充分利用终端设备的离线计算和存储能力,为用户提供更加智能、实时、安全和个性化的服务与体验. 近年来,端智能在自动驾驶[1,2]、智能家居、智能安防[3]以及工业互联网等领域得到了广泛应用.

然而,目前端智能的高效部署与实时推理仍面临两大主要挑战. 首先,终端设备的资源依然有限. 由于设备尺寸和便携性等限制,智能手机、物联网设备和边缘计算节点的芯片算力、内存容量、带宽和电池电量都存在不足,表1列出了部分常见终端设备的硬件性能. 与专用数据中心或云端的高性能服务器相比,移动终端设备的计算能力相差一个数量级,例如2023款的高通骁龙8 Gen3芯片,其图形处理单元(Graphics Processing Unit,GPU)的内存带宽仅为54.2 GB/s,并且这些带宽还需要和中央处理单元(Central Processing Unit,CPU)共享,这显著低于NVIDIA的Tesla V100 GPU的内存带宽900 GB/s. 因此在处理复杂的计算任务时,端智能设备的运行效率将受到极大的限制. 其次,当前的深度学习模型日趋复杂. 为了实现更高的任务精度,研究人员设计的模型具有更多层次和更复杂的结构,对计算资源提出了严苛要求. 以OpenAI提出的大语言模型GPT-3[4]为例,该模型包含96层Transformer编码器,每层具有12 288个隐藏单元,总参数量达1 750亿. 其推理单实例需要至少40 GB显存的GPU进行加速计算. 然而,终端设备的计算能力和内存容量远不足以满足这一需求,这意味着在端侧环境中部署和推理复杂的深度学习模型将面临显著性能瓶颈,影响实时性和效率.

表1 部分常见终端设备硬件性能表

端侧设备	芯片名称	峰值算力(CPU/GPU)	带宽(GPU/内存)/(GB·s^{-1})	内存/GB
Vivo X90s*	联发科天玑9200+	GPU:1 762.46 GFLOPS(FP32)/3 471.00 GFLOPS(FP16)	GPU:40.15	12
小米14*	骁龙8 Gen3	GPU:1 914.79 GFLOPS(FP32)/2 233.70 GFLOPS(FP16)	GPU:54.20	16
红米k70*	骁龙8 Gen2	GPU:1 543.79 GFLOPS(FP32)/1 777.11 GFLOPS(FP16)	GPU:47.09	16
红米Note 12 Turbo*	骁龙7+ Gen2	GPU:904.74 GFLOPS(FP32)/1 122.90 GFLOPS(FP16)	GPU:37.88	12
iQOO Neo 9*	骁龙8 Gen2	GPU:1 783.87 GFLOPS(FP32)/3 457.18 GFLOPS(FP16)	GPU:60.70	12
Vivo X50*	高通骁龙765G	GPU:280.28 GFLOPS(FP32)/546.99 GFLOPS(FP16)	GPU:13.54	8
华为Nava7 5G*	华为麒麟985	GPU:403.08GFLOPS(FP32)/790.45GFLOPS(FP16)	GPU:12.24	8
OPPO Reno4 Pro 5G*	高通SDM765G	GPU:287.72GFLOPS(FP32)/541.29GFLOPS(FP16)	GPU:15.29	8
红米k30	高通骁龙865	GPU:856.06GFLOPS(FP32)/1 660.51GFLOPS(FP16)	GPU:30.26	8
红米Note 9 5G*	联发科天玑800U	GPU:179.37GFLOPS(FP32)/351.81GFLOPS(FP16)	GPU:13.70	6
荣耀Magic 5*	骁龙8 Gen2	GPU:1 382.08GFLOPS(FP32)/2 665.18GFLOPS(FP16)	GPU:49.27	8
华为MatePad Pro 2021*	高通骁870移动平台	GPU:977.3GFLOPS (FP32)/1 900.20GFLOPS(FP16)	GPU:30.94	8
联想小新Pad Plus 2021*	高通骁龙750G	GPU:193.81 GFLOPS (FP32)/379.96 GFLOPS(FP16)	GPU:10.96	6
NVIDIA Jetson TX2	NVIDIA Tegra X2 SOC	GPU:437~750 (FP32)/874~1 500(FP16)	内存:59.70	8
NVIDIA Jetson Nano	NVIDIA Tegra X1 SOC	GPU:0.5 TFLOPS (FP16)	内存:25.60	4
Skydio 2+	NVIDIA Tegra X2 SOC	GPU:437~750 GFLOPS(FP32)/874~1 500(GFLOPSFP16)	内存:59.70	4
Tesla Full Self-Driving Computer	FSD Chip	GPU:600 GFLOPS(FP32,FP64)	内存:63.58	约8
NVIDIA Jetson Orin NX 8 GB	NVIDIA Jetson Orin SOC	70 TOPS	内存:102.40	8
NVIDIA Jetson Xavier NX 16 GB	NVIDIA Jetson Xavier NX SOC	21 TOPS	内存:59.70	16

注:*为设备的峰值算力与GPU带宽通过该GitHub项目测得:https://github.com/krrishnarraj/clpeak.

针对上述挑战,现有端智能推理加速的研究工作主要集中在算法、软件和硬件三个关键领域,旨在提升终端设备进行模型推理的计算效率.

(1)模型算法优化:通过设计高效轻量化模型,或利用模型压缩方法,减小模型体积和计算复杂度并保持模型性能.

(2)编译软件优化:在系统部署与运行时(runtime),通过融合或重排核心算子、优化数据结构、改进模型执行流程、管理内存、调度硬件资源等方式,降低模型推理负载并减少执行延迟.

(3)设备硬件优化:通过定制化硬件、优化硬件架构,或引入专用芯片(例如现场可编程门阵列(Field-Programmable Gate Array,FPGA)、神经处理单元(Neural Processing Unit, NPU)、张量处理单元(Tensor Processing Unit,TPU)),弥补移动CPU和GPU的算力不足,提高模型推理速度和效率.

在模型算法与设备硬件的单一层面优化上,现有研究已取得相对令人满意的成绩.然而,在端智能部署的实践过程中,仅关注单一层面的优化,可能会因未能综合考量多方面因素的协同或制约作用,而无法充分发挥推理加速的潜力.例如,仅通过算法轻量化减少模型大小和计算需求,虽然一定程度上优化了推理速度与能耗,但若未考虑硬件特性、数据存取与传输等方面的限制,实际部署可能会造成性能瓶颈.以模型参数量化为例,使用INT8量化技术理论上可实现4倍推理加速,但在设备上实际部署时,推理加速仅达0.8倍(值小于1表示速度变慢)至3.0倍[5].

因此,研究人员对端智能推理优化通过从全局视角,针对算法与软、硬件之间的特性进行结合优化.如图1所示,算法与软硬件结合优化通过综合考虑算法轻量化、编译软件高效实现以及设备硬件特性,深入剖析三者间的复杂依赖关系与耦合机制,以期实现端智能推理在能效比、实时性及灵活性等多方面的最佳平衡,全面提升系统整体性能.近年来,通过跨领域协同创新,算法与软硬件结合优化在应对端智能推理加速所面临的诸多挑战中展现出显著成效,极大地释放了端智能在多样化应用场景中的潜在价值,逐渐成为端智能推理加速领域的一个突出研究趋势.

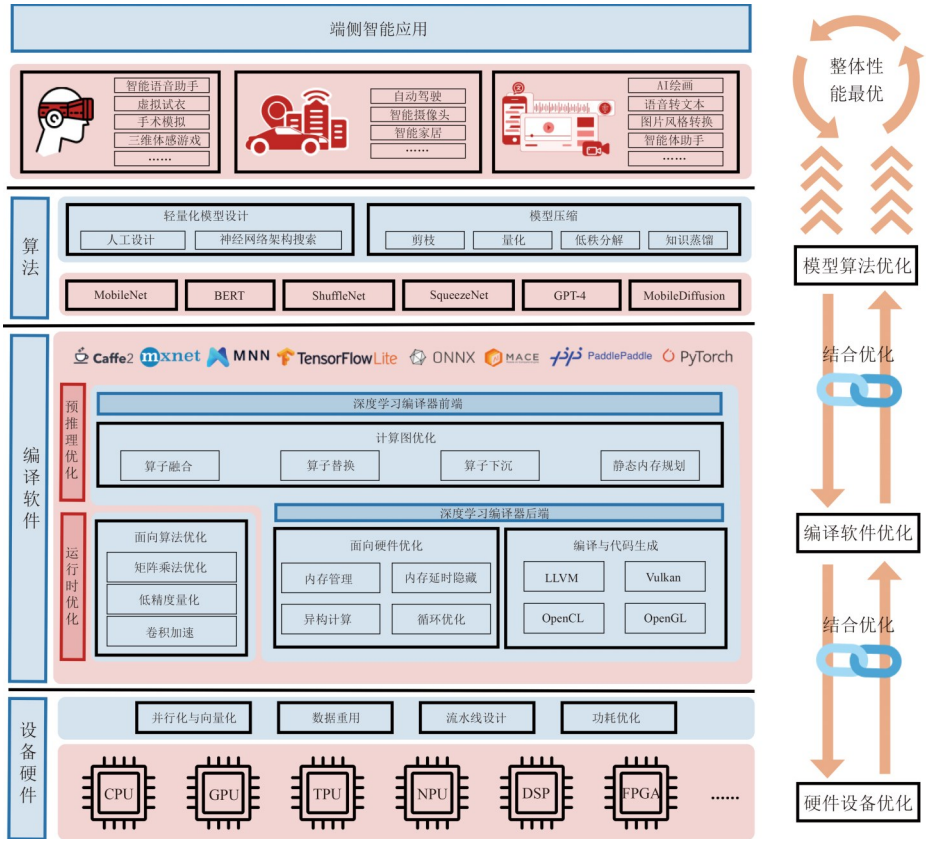


图1　端智能推理软硬件结合优化框架

目前,已有相关综述针对端智能推理加速技术进行总结[6~14],其相关内容总结如表2所示.然而,现有的综述仅从算法或硬件优化等单一层面分析和梳理了端智能推理加速工作,缺乏对算法与软硬件结合优化研究的系统性总结.端智能推理加速是一个复杂的系统性问题,单纯关注算法或硬件的优化往往无法充分发挥其潜在优势.因此,真正实现性能提升必须从整体架构出发,全面考虑各层面之间的相互影响.为此,本文从系统角度总结了端侧深度学习模型推理优化的研究进展,重点探讨以下几个关键领域:首先,总结了现有的模型压缩技术,探讨其在减少模型计算资源和提升推理速度方面的有效性;其次,梳理了模型-软件-硬件协同设计的方法,强调三者之间紧密结合对性能提升的重要性;此外,我们还归纳了模型异构并行部署策略,展示了如何在不同硬件环境中优化推理效率;最后,本文还对现有的大模型在资源受限的端侧环境中

部署存在的挑战进行了分析,并且对现有的大模型端侧部署优化技术进行了整理. 本文主要贡献在于为端侧智能推理加速提供一个更为全面的优化视角,明确未来的发展机遇与研究方向. 我们希望通过此工作,推动该领域的持续研究进展,为相关研究提供有价值的实践参考和理论指导.

表2 端智能系统推理优化相关综述介绍

相关综述	年份	领域	具体范畴与相关技术
文献[6]	2019	边缘智能	回顾了AI在网络边缘运行的背景和动机,概述了用于网络边缘训练和推理的深度学习模型的总体架构、框架和新兴关键技术,讨论了边缘智能的未来研究机会
文献[7]	2020	边缘计算与深度学习的融合	重点关注边缘计算与深度学习的应用场景、使能技术、挑战和未来趋势.详细介绍了包括模型设计与推理、边缘硬件、分布式训练、联邦学习、边缘缓存、边缘任务卸载等技术在内的边缘计算技术
文献[8]	2020	边缘计算与AI的融合	将边缘智能分为智能赋能的边缘计算和边缘AI两方面,介绍了在边缘计算领域流行的AI技术,以及如何在边缘进行模型训练和推理加速
文献[9]	2021	网络边缘的机器学习	介绍了在边缘部署机器学习系统的研究工作,包括模型压缩技术、工具、框架和硬件等方面
文献[10]	2021	资源受限设备上的机器学习	在算法和理论层面介绍了资源受限设备上机器学习系统部署技术的进展
文献[11]	2021	深度学习模型压缩与加速	分析了经典深度学习模型压缩与加速方法,包括参数剪枝、参数量化、紧凑网络、知识蒸馏、低秩分解、参数共享和混合方式7个方面;总结对比了几种主流技术的代表性方法在多个公开模型上的压缩与加速效果
文献[12]	2022	移动设备上的深度学习	介绍了流行的模型压缩方法、AutoML框架(例如神经架构搜索以及自动剪枝和量化)、设备端训练技术、特定于视频和自然语言处理等几种任务的加速方法
文献[13]	2022	边缘设备的深度学习推理加速	从边缘训练、边缘缓存、边缘卸载与边缘推理四个方向调查边缘设备的深度学习推理优化方法,讨论了边缘智能的相关技术
文献[14]	2022	移动设备上的深度学习	总结比较了移动设备上的深度学习技术,涉及视觉、语音、人类活动识别等应用领域,从算法方面介绍了移动设备上深度学习模型部署的优化技术
文献[15]	2023	AI在边缘计算中的应用	讨论了AI技术在边缘计算领域中的应用,分析了AI算法在边缘计算应用时存在的挑战,总结了现有AI算法在边缘计算中的优化工作

2 端侧模型推理算法层优化

端侧设备有限的算力、内存和能源等资源,阻碍了端智能的广泛应用部署. 为了克服这些限制,部分研究人员从深度学习模型的算法层面,通过对深度学习模型的参数进行压缩、结构进行精简,设计轻量级的模型,以提高模型的推理的效率,为端智能的高效部署提供可行解决方案.

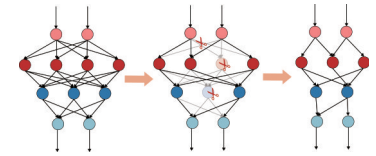
2.1 模型压缩

模型压缩通过利用或修改性能表现优异但参数量较大的基础模型,以减少内存和计算成本. 如图2所示,现有的模型压缩技术分为模型剪枝、参数量化、低秩分解与知识蒸馏四类.

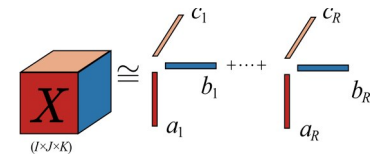
2.1.1 模型剪枝

模型剪枝通过消除模型中的冗余参数(如权重、滤波器、通道或层)构造轻量级模型,分为非结构剪枝(unstructured pruning)和结构剪枝(structured pruning). 非结构剪枝又称为权重剪枝,旨在通过移除神经网络中不重要的参数,而不考虑其所在的结构或模式,来降低神经网络模型的大小与复杂度. 剪枝的研究可以追溯到20世纪后期,文献[16]提出的最优脑损伤(Optimal Brain Damage,OBD)以及文献[17]提出的最优脑外科医生(Optimal Brain Surgeon,OBS). 这两类方法通过使用损失函数的Hessian矩阵来修剪每个非必要的权重. 此后,基于上述方法的一些改进工作被陆续提出[18~20]. 2015年,文献[21]提出权重剪枝三步法,使剪枝开始在深度学习模型推理加速领域得到广泛关注. 该方法首先训练模型以确定权重中的重要部分;然后对不重要权重剪枝;最后重新训练模型以微调权重. 后续许多剪枝工作都基于该思想展开. 文献[22]通过计算激活(activations)的熵值以确定层的重要性,优先修剪低熵层中的权重或连接,最终实现冗余参数的完全删除,有效减少了模型大小. 文献[23]基于多层权重剪枝的输出失真率,设计层自适应的剪枝方案. 在修剪每层权重时,将其他层的影响考虑在内,进一步提升了剪枝后模型的准确率. 虽然非结构剪枝能够精细地控制剪枝过程,减少对模型性能的负面影响. 但由于剪枝后得到的稀疏矩阵在现有硬件上计算效率较低,需要专门的底层硬件/库[24]和矩阵算法支持来加速,不利于在嵌入式或移动设备上部署.

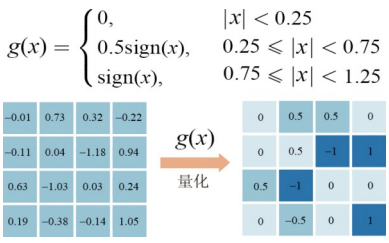
结构化剪枝(structured pruning)按照特定结构或模式,成块地移除神经网络中的某些部分,剪枝后网络中



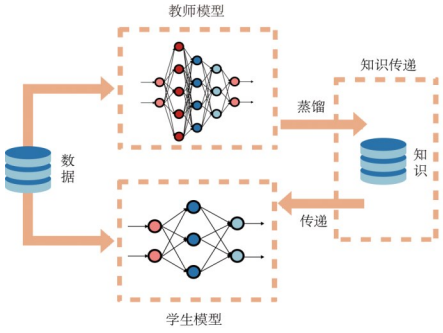
(a) 剪枝



(b) 低秩分解



(c) 量化



(d) 知识蒸馏

图2　模型压缩技术总结

不会生成稀疏矩阵,因此不需要专门的硬件支持,能够更好地适应硬件加速器. 根据剪枝参数粒度大小,结构化剪枝由粗至细可分为层级别剪枝(layer-level)、滤波器级别剪枝(filter-level)、核级别剪枝(kernel-level)等. 层级别剪枝是指对每一层的filter设置相同的稀疏模式,变成结构相同的稀疏矩阵,以稀疏化卷积核,提高运行速度[25~27]. 滤波器级别剪枝也可以看作通道级别剪枝,删去该层的某些filter,相当于删去其产生的部分feature map和原本需要与这部分feature map进行卷积运算的下一层部分filter[28~30]. 核级别剪枝是去除某个卷积核,它将丢弃对输入通道中对应计算通道的响应,以达到减少计算量的效果[31].

尽管上述结构化剪枝方法能在一定程度上提升模型的推理效率[32~34],但其仍面临三大挑战:一是难以将不同剪枝方法应用于各种模型架构;二是难以在单一框架中统一结构化剪枝方法,以便在训练的任意阶段实施剪枝;三是现有方法往往针对特定架构或训练范式设计,并被相应的深度学习框架固化. 对此,文献[35]提出一种适用于卷积神经网络(Convolutional Neural Network,CNN)、循环神经网络(Recurrent Neural Network,RNN)、图神经网络(Graph Neural Network,GNN)和Transformer等任意架构且全自动的结构剪枝方法DepGraph. 由于来自不同层的参数在网络架构中本质上相互依赖,这迫使多个层同时被修剪. DepGraph通过显式地考虑该种依赖关系,将依赖链分解和建模为一个递归过程. 这被归结为寻找图中最大连通分量问题,并通过图遍历实现了$O(N)$的复杂度. 但该方法不支持除PyTorch[36]以外的框架,并且仅支持在有微调下的训练后修剪. 对此,文献[37]利用标准化计算图和ONNX(Open Neural Network eXchange)表示修剪不同的神经网络架构,提出一种多功能的结构化剪枝框架SPA(Structurally Prune Anything),能够在任何训练阶段从任何推理框架中修剪具有任何架构的神经网络.

2.1.2　参数量化

参数量化通过压缩模型数据类型的精度来减少存储需求和加速计算. 例如,使用8位整数(INT8)代替32位高精度浮点数(FP32),内存开销减少至1/4,矩阵乘法的计算成本减少至1/16. 根据量化过程中参数的精度和表示方式,量化可以分为定点量化(fixed-point quantization)和混合精度量化(mixed-quantization). 在定点量化过程中,神经网络模型参数如权重、激活、误差等被量化为固定的位宽,例如16位[38~39]、8位[40]与2位[41~43]. 与定点量化相反,混合精度量化通常是模型参数位宽的灵活组合[44]. 2017年,文献[45]提出混合精度训练,使用FP16存储参数并计算前向和反向传播,最终结果累积到FP32中. 除了手动确定位宽组合,一些自动化方法也相继提出[46~50]. 这些方法将位宽设置视为组合搜索问题,并采用基于梯度的方法[46]、强化学习[47]、单次(one-shot)[48]或进化算法[51]确定最佳的位宽精度设置,为模型提供了更优的精度与复杂性权衡[52]. 然而,这类基于训练的方法需要大量计算资源且非常耗时. 对此,文献[53]在超网中增加位宽预测模型,能够在$O(1)$时间复杂度内完成对应于不同压缩率的帕累托最优位宽组合的搜索,效率提高了5倍. 此外,另一类无训练方法通过构建替代代理来对候选位宽配置进行排序,显著减少了计算负担[52,54]. 文献[52]提出了一种基于进化算法的无训练混合精度量化框架,能够有效搜索与量化精度密切相关的优秀代理. 同时,设计了一种多样性提示选择策略和兼容性筛选协议,以避免算法过早收敛并进一步提

高搜索效率.

2.1.3 低秩分解

低秩分解的核心思想是将高秩参数张量分解为一系列低阶张量,从而减少内存使用和计算复杂度.一个典型的神经网络可被视为一个四维矩阵(张量)运算,由权重(w)×高度(h)×通道(c)×内核(n)组成.从矩阵分析的角度出发,低秩分解一般通过以下步骤来降低模型规模和计算复杂度:(1)将神经网络中的参数矩阵视为稠密满秩矩阵;(2)使用一组低秩矩阵来近似表示该稠密满秩矩阵;(3)进一步将这些低秩矩阵分解为更小规模的矩阵组合.文献[55]使用两个连续的较低秩的卷积核来代替四维卷积核,将空间维度为$[w,h]$的滤波器分解为$[w,1]$和$[1,h]$的两个滤波器.稀疏卷积神经网络(Sparse Convolutional Neural Networks,SCNN)[56]对通道和卷积核进行两级分解,以获得稀疏核矩阵,并将卷积层的运算转换为稀疏矩阵乘法.前馈神经网络(Feed-Forward Network,FFN)[57]利用权重矩阵近似方法直接针对定点数分解.文献[58]将常规卷积层分解为滤波器组卷积和逐点卷积的线性组合.除此之外,主流的低秩分解方法还有奇异值分解(Singular Value Decomposition,SVD)[59]、典型多线性(Canonical Polyadic,CP)分解[60~62]和Tucker分解[63~65].文献[66]使用SVD进行张量近似来压缩全连接层.文献[67]应用SVD分解深度神经网络(Deep Neural Network,DNN)模型中的权重矩阵.文献[64]使用Tucker分解得到空间大小分别为[1,1]、[w,h]和[1,1]的卷积.文献[65]提出了一种基于Tucker分解和自动张量秩选择的预算感知DNN压缩方法,将秩选择过程整合到具有特定压缩预算的DNN训练过程中,从数据中学习DNN的最优张量秩.文献[68]结合SVD分解与模型剪枝提出可控矩阵分解算法.该算法将权重矩阵分解为低秩和稀疏矩阵,通过动态控制低秩分量和稀疏分量之间的分配,获取压缩比和精度之间的平衡.

2.1.4 知识蒸馏

知识蒸馏是一种专注于教师模型和学生模型的迁移学习方法.学生模型通常是一个紧凑、高效的神经网络,具有更简单和更少的参数操作.教师模型是一个具有高性能和泛化能力的大规模复杂神经网络.知识蒸馏通过将逻辑(logits)、激活(activations)或特征(feature)等"知识"从教师模型迁移到学生模型,使后者表现得与前者同样甚至更好.2015年,知识蒸馏被文献[69]首次推广用于深度神经网络.该方法将"温度"参数添加到Softmax函数中,利用教师模型的Softmax输出作为软标签,并结合Kullback-Leibler(KL)散度计算学生模型输出与软标签之间的差异,从而将大型教师模型的知识转移到较小的学生模型中,实现模型压缩和性能提升.如何提高教师与学生间的蒸馏性能是知识蒸馏研究中的一个重要课题.文献[70]通过动态且可学习的"温度"逐步提高学生模型的学习难度水平,从而提高蒸馏性能.在此基础上,文献[71]进一步优化,将温度设置为逻辑的加权标准差,并在应用Softmax和KL散度之前执行即插即用的逻辑标准化Z分数预处理,提高了现有基于逻辑的蒸馏方法的性能.然而,上述方法均在假设教师和学生模型属于同一架构类型的情况下设计的.为解决异构模型蒸馏的挑战,文献[72]提出一劳永逸知识蒸馏(One-For-All Knowledge Distillation,OFA-KD).OFA-KD将额外的出口分支合并到学生模型,使不匹配的表示转移到对齐的逻辑空间.通过匹配这些分支的输出与教师模型在逻辑空间的分类器层的输出,在中间层实现跨架构蒸馏,显著提高了异构架构之间的蒸馏性能.

2.2 轻量级模型设计

轻量级模型设计是一种专门针对计算能力、存储空间和能量消耗有限的设备进行优化的模型架构设计方法.该种设计方法考虑端侧设备的资源约束,通过减少模型的参数数量、简化计算流程和优化数据流,以实现对模型的轻量化,同时努力保持甚至提升模型性能.如图3和图4所示,端侧轻量化模型设计主要分为两类:人工设计和自动化神经网络架构设计.

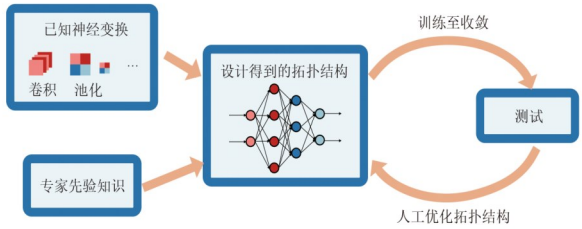


图3 人工神经网络架构设计

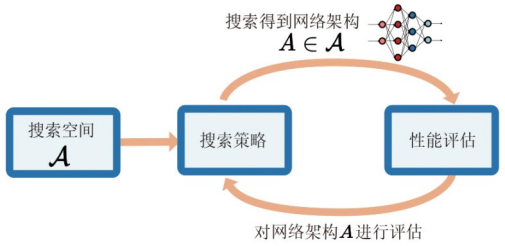


图4 自动化神经网络架构搜索

2.2.1 人工设计模型

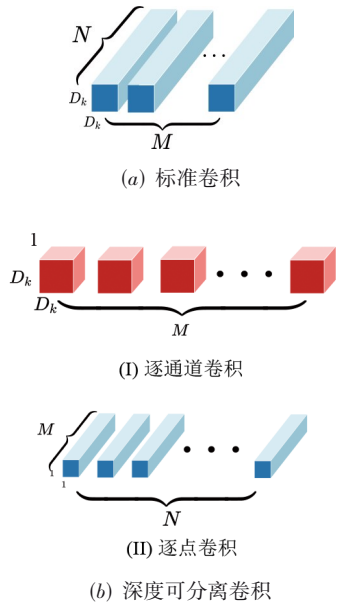
人工设计方法依赖领域专家的知识和经验来构建更高效的模型结构,例如SqueezeNet[73,74]、MobileNet[75,76]、ShuffleNet[77,78]、IGCNet[79~81]及GhostNet[82]等.

SqueezeNet系列:SqueezeNet[73]是关注轻量级模型的早期研究之一.该方法提出使用Fire模块进行参数

压缩. Fire模块由仅具有1×1卷积核的挤压层和混合了1×1和3×3卷积核的扩展层组成. 基于此,SqueezeNext[74]进一步使用低秩滤波器,将大卷积分解为多个可分离的卷积,以减少模型参数数量.

ShuffleNet系列:ShuffleNet是旷视科技提出的轻量级模型系列,包括ShuffleNet-V1[77]和ShuffleNet-V2[78]. ShuffleNet-V1使用通道洗牌操作来补偿组之间的信息交换,以减少模型计算量. 然而,ShuffleNet-V1中使用的组卷积和瓶颈结构(bottleneck)增加了内存访问成本. 对此,ShuffleNet-V2提出通道分割操作(channel split),并结合理论与实验得到了4条实用原则,以指导高效轻量级模型的设计.

MobileNet系列:MobileNet系列是Google提出的一组移动端友好的轻量级模型. 2017年,MobileNet-V1[75]引入深度可分离卷积,包括逐通道卷积和逐点卷积两步. 如图5所示,逐通道卷积在输入的每个通道上独立进行,每个通道由一个独立的卷积核处理. 接着,逐点卷积使用1×1×M的卷积核对逐通道卷积的输出进行处理(M为上一层通道数),整合各通道的信息以生成最终输出特征图. 2018年,MobileNet-V2[76]提出了一种新的高效瓶颈结构,称为倒置残差(inverted residuals). MobileNet-V2在3×3深度卷积之前使用1×1卷积来增加特征图通道,然后应用带有线性激活函数的1×1卷积,以减少特征图通道并得到输出. 2024年,Google推出针对移动设备的新一代通用高效架构MobileNet-V4[83]. MobileNet-V4引入通用倒置瓶颈和专为移动加速器量身定制的注意力模块. 同时,MobileNet-V4优化了TuNAS[84]的神经网络架构搜索技巧,将粗粒度和细粒度搜索分开,与传统的单阶段搜索相比,两阶段搜索有效提高了效率和模型质量.



(a) 标准卷积

(I) 逐通道卷积

(II) 逐点卷积

(b) 深度可分离卷积

图5　标准卷积和深度可分离卷积

IGC系列:交错组卷积(Interleaved Group Convolutions,IGC)系列的设计思路是将常规卷积分解为多个分组卷积. IGC-V1[79]提出一种新颖的交错分组卷积,由主分组卷积(primary group convolutions)和次分组卷积(secondary group convolutions)组成. 主分组卷积对输入进行分组特征提取并对分区进行采样,然后重新排列主分组卷积和次分组卷积之间的特征图. IGCNet-V1将原始卷积分解为两个分组卷积,在减少参数的同时能够保持完整的信息提取. 然而,由于主次分组卷积的分组数互补,导致次分组卷积的分组数一般较小,每个分组的维度较大,次卷积核较为稠密. 为解决该问题,文献[80]提出了IGC-V2. 该方法使用多个连续的稀疏分组卷积来代替原来的次分组卷积,使每个分组卷积的分组数足够多,保证了卷积核的稀疏性. 文献[81]则受IGC-V1和倒置残差瓶颈的启发提出了IGC-V3. 该方法对单个通道的特征图进行扩展、排列和压缩,以产生相应的输出特征图,在CIFAR和ImageNet图像分类数据集中的性能表现优于IGC-V2和MobileNet-V2.

其他高效网络:文献[85]设计了一种简单而有效的模型复合缩放方法,通过平衡宽度、深度和分辨率扩展基线模型,提升推理效率. 同时,利用神经架构搜索设计了新的基线网络,并将其扩展为一系列模型,称为EfficientNets. 随后,文献[86]针对目标检测提出加权双向特征金字塔网络(Bidirectional Feature Pyramid Network,BiFPN)与一种复合缩放方法,并将其与EfficientNet主干网相结合,开发了高效网络EfficientDet. 华为提出Ghost模块[82],通过低成本操作构建高效神经网络. Ghost模块将标准卷积分为两部分:首先使用较少的卷积生成特征图,然后使用廉价的变换操作生成更多特征图. 文献[87]提出了部分卷积(PConv). PConv仅在部分输入通道上应用标准卷积,而其余通道保持不变,从而减少冗余计算和内存访问,更有效地提取空间特征. 苹果公司针对移动设备提出高效骨干MobileOne[88]. MobileOne的核心模块基于MobileNet-V1设计,并吸收重参数化思想,引入过度参数化分支以实现精度增益. 文献[89]结合轻量级ViT(Vision Transformer)的高效架构设计,从主干、下采样层、分类器到整体等各个层面优化了MobileNet-V3的性能,并提出新的轻量级CNN模型系列RepViT,在各种视觉任务中表现出良好的延迟.

2.2.2　自动神经网络架构搜索

自动神经网络架构搜索(Neural Architecture Search,NAS)技术旨在在用户定义的约束(例如准确性、模型大小和推理时间)下,使用搜索策略测试和评

估搜索空间中的备选架构,通过最大化目标函数来选择满足给定约束的最佳模型架构.相比于基于专家经验的人工模型设计,NAS能通过自动化的方式,更全面地探索网络结构的设计空间,找到更优的解决方案.

文献[90]于2016年进行了自动化神经网络架构搜索的首次尝试.首先他们根据预定义的搜索空间标准生成多个候选网络架构.其次对每个网络架构都训练直至收敛,并根据验证集上的网络准确性进行排名.该排名可以作为反馈来调整搜索策略,并获得新的架构.再次,重复该过程直到达到终止条件.最后,使用测试集评估最佳网络架构.但这种方法需要在巨大的搜索空间中训练和评估数千个神经网络,导致巨大的计算和时间成本.为了降低搜索成本并找到性能最佳的网络架构,研究人员开始从搜索空间、策略以及性能评估策略等方面着手改进.

搜索空间:搜索空间决定了可以表征的神经网络的结构.文献[90]设计的搜索空间包括网络每层的滤波器高度和宽度、步幅高度和宽度、滤波器数量和残差点.在这样的搜索空间上进行搜索,相当于直接搜索整个神经网络结构,非常耗时.为了减少搜索时间,文献[91]进一步做出改进,通过仅搜索有限数量的单元来限制搜索空间.然后根据先验知识手动将这些单元堆叠到整个神经网络的结构中.但这类方法限制了神经网络层的多样性,这对于能否构建出符合目标限制条件的网络架构至关重要,例如高精度和低延迟.文献[92]引入一种分解分层搜索空间,将CNN模型分解为独特的块,然后分别搜索每个块的操作和连接,从而允许不同块中存在不同的层架构.此外,部分研究人员[92~94]通过使用现有的高质量神经网络结构作为主干或架构参考以优化搜索空间.文献[95]构建了一个具有固定宏架构的分层搜索空间,每一层都可以选择不同的块.其中,块结构受到MobileNet-V2和ShiftNet[96]的启发,包含一个逐点卷积,一个$k \times k$的深度可分离卷积(k为内核大小)以及再一个逐点卷积.类似的,文献[93]和文献[97]使用MobileNet-V2作为构建架构空间的骨干,在构建网络时调整不同的内核大小和扩展比以确定每个倒置残差层.

搜索策略:搜索策略用于指导算法在搜索空间中找到最佳的性能神经网络结构.常见的搜索算法包括随机搜索、进化算法、贝叶斯优化、强化学习和基于梯度的优化等.在神经网络的早期优化研究中,进化算法被广泛应用于优化网络权重[98~101].文献[90]首次使用强化学习探索搜索空间.该方法使用RNN作为强化学习代理,使用策略梯度方法优化RNN所采样的神经网络的性能.强化学习的替代方法是进化算法,文献[102]和文献[103]使用进化算法来搜索网络结构,并使用随机梯度下降来估计参数.然而,强化学习和进化算法都需要评估大量候选架构以找到最优模型,导致巨大的计算资源与时间消耗.对此,文献[104]和文献[105]使用基于梯度的方法和参数共享训练策略,大大提高了神经网络架构的搜索速度.文献[106]使用基于顺序模型的优化策略,按照复杂性增加的顺序搜索结构,同时学习代理模型来指导结构空间中的搜索.在相同搜索空间下进行直接比较,该方法比文献[90]的强化学习方法效率提高了5倍.

性能评估策略:性能评估策略旨在通过评估所探索的网络结构的性能来找到最佳的神经架构.早期的NAS[90,107]算法通过从头开始训练来评估每个采样网络的性能,成本较为昂贵.对此,研究人员提出以下三类主要方法.

(1)低保真度预测,减少训练步骤数并在部分或较低分辨率数据集上进行训练[91,108~111].由于将自动化神经网络架构搜索直接应用于大型数据集(例如ImageNet数据集)的计算成本高昂,文献[91]首先在代理数据集(例如较小的CIFAR-10数据集)上搜索架构,然后将学习到的架构转移到ImageNet.文献[109]则是直接限制每个采样网络的训练时间以降低计算成本.

(2)学习曲线插值和准确率预测,通过利用先前训练过的模型的学习曲线,预测未来模型的性能,从而加速性能评估过程[112~115].文献[113]引入学习曲线的加权概率模型,并利用该模型来加速小型CNN中的超参数搜索.基于此,文献[114]使用完全和部分观察到的学习曲线的训练集训练贝叶斯神经网络来预测未观察到的学习曲线.但这两种方法都依赖于昂贵的马尔可夫链蒙特卡罗采样程序和手工制作的学习曲线基函数.对此,文献[115]使用从模型架构、训练超参数和学习曲线的早期时间序列测量得出的简单特征来参数化学习曲线轨迹,以预测部分训练的模型配置的最终性能.

(3)One-shot方法,一些研究[116~119]将所有搜索到的结构视为一些更大的超网结构的子结构,并且所有子结构都继承超网的权重.在评估每个子结构性能时共享计算资源,从而降低评估成本.

3 端侧模型推理硬件加速

随着深度学习模型日趋复杂,移动端CPU和GPU无法满足端侧模型高效推理所需的计算能力和内存带宽,越来越多的端侧神经网络专用芯片被用于端侧深度学习模型推理加速.例如,谷歌在2018年推出的Edge TPU[120],是一款专为移动和边缘设备设计的专用集成电路(Application-Specific Integrated Circuit,ASIC),旨在加速深度学习模型的推理.本节将详细介绍端侧模型推

理的常用硬件加速器,包括移动CPU、GPU、NPU、FPGA以及ASIC.

端侧模型推理常见加速硬件包括移动CPU、GPU、NPU、FPGA以及ASIC. 移动设备与嵌入式设备通常采用这些不同硬件集成的片上系统(System on Chip,SoC)芯片,以平衡性能和能效需求. 表3中列举了几款典型的代表性移动端SoC芯片,包括高通骁龙系列、苹果A系列、联发科天玑系列以及三星猎户座系列. 由表3可知,移动端SoC的制造工艺已经进步到3 nm技术节点,这一突破显著提升了芯片的性能与能效比. 随着移动端CPU的时钟频率的持续提高以及GPU的核心数量的增加,这些芯片的处理能力得到了极大的增强. 此外,现代SoC普遍集成了专用的NPU,这些单元专门优化了机器学习和深度学习算法的执行效率. 这些技术进步不仅提升了移动设备的处理能力,也极大地促进了端智能的发展,使得设备能够在本地执行复杂的AI任务,从而减少对云计算资源的依赖,并提高了响应速度和隐私保护水平.

表3　部分最新的移动端SoC芯片及相关配置

SoC芯片	年份	制程/nm	CPU	GPU	ASIC/AI引擎
海思麒麟9000	2020	5	1×Cortex-A77 大核@3.13 GHz 3×Cortex-A77 中核@2.54 GHz 4×Cortex-A55 小核@2.05 GHz	Mali-G78 GPU	华为达芬奇架构NPU 2.0
高通骁龙8Gen2	2022	4	1×ARM Cortex-X3 超大核@3.2 GHz 2×ARM Cortex-A715 大核@2.8 GHz 2×ARM Cortex-A710 大核@2.8 GHz 3×ARM Cortex-A510 小核@2.0 GHz	Adreno-740 GPU	高通Hexagon处理器
高通骁龙8Gen3	2023	4	1×Arm Cortex-X4 超大核@3.3 GHz 5×Arm Cortex-A720 大核@3.2 GHz 2×Arm Cortex-A520 小核@2.27 GHz	Adreno-750 GPU @903 MHz	高通Hexagon NPU
三星猎户座2 400	2023	4	1×Arm Cortex-X4 超大核@3.21 GHz 2×Arm Cortex-A720 大核@2.9 GHz 3×Arm Cortex-A720 中核@2.59 GHz 4×Arm Cortex-A520 小核@1.96 GHz	三星Xclipse 940 GPU	17K MAC NPU(2-GNPU+2-SNPU)DSP
谷歌Tensor G3	2023	4	1×ARM Cortex-X3 超大核@3 GHz 4×ARM Cortex-A715 大核@2.45 GHz 4×ARM Cortex-A510 小核@2.15 GHz	Mali Immortalis-G715 GPU	Mali Immortalis-G715 GPU
联发科天玑9300	2023	4	4×Arm Cortex-X4 超大核@3.25 GHz 4×Arm Cortex-A720 大核@2.0 GHz	Mali Immortalis-G720 GPU	联发科 NPU 790
联发科天玑9300+	2024	4	1×Arm Cortex-X4 超大核@3.4 GHz 3×Arm Cortex-X4 大核@2.85 GHz 4×Arm Cortex-A720 小核@2.0 GHz	Mali Immortalis-G720 GPU	联发科 NPU 790
苹果A16 Bionic	2023	4	2×性能核心@3.46 GHz 4×效率核心@2.02 GHz	苹果5核 GPU	16核神经引擎
苹果A17pro	2023	3	2×性能核心@3.78 GHz 4×效率核心@2.11 GHz	苹果6核 GPU	16核神经引擎
苹果A18pro	2024	3	2×性能核心@4.05 GHz 4×效率核心@2.42 GHz	苹果6核 GPU	16核神经引擎

3.1　移动CPU与移动GPU

CPU是移动设备的“大脑”,负责执行指令、处理任务、管理资源. 为提高多任务处理能力,移动CPU通常采用多核心设计(如双核、四核、八核),每个核心可以同时处理不同任务. 现代移动CPU还集成了诸多节能技术,如big.LITTLE架构,其中性能更强的“大核”(如Arm Cortex-X系列)处理高负载任务,而低功耗的“小核”(如Arm Cortex-A系列)处理日常轻负载任务,从而平衡性能与能效.

GPU的设计目标是并行处理大量数据流,尤其擅长加速图像处理、3D渲染等需要并行计算的任务. 桌面GPU通常采用即时渲染(Immediate Mode Rendering,IMR)架构. 在渲染过程中,每处理一个像素都会与系统内存中的帧缓冲区(framebuffer)进行交互,导致频繁

的带宽消耗,进而带来较大的功耗和发热. 由于端侧设备带宽有限,并且需要考虑电池续航和散热,IMR 架构并不适用. 为了解决这一问题,移动 GPU 广泛采用基于图块的渲染(Tile-Based Rendering,TBR)架构. TBR 将帧缓冲区划分为多个图块(Tile)进行逐块渲染. 渲染时,结果先存储在高速的片上存储器(On-chip Memory)中,待图块渲染完成后再统一写入内存. 通过减少对系统内存的频繁访问,TBR 显著降低了带宽消耗和能量浪费,提升了能效. 目前,许多高性能的移动 GPU 都采用了这种架构,典型代表包括高通骁龙的 Adreno 系列、Arm 的 Mali 系列以及苹果自研的 GPU.

由于移动 CPU 和 GPU 通常集成在同一块 SoC 芯片上为端侧设备提供计算服务,相较于服务器,具有以下特点:(1)性能可比性,与运行速度比 CPU 快几个数量级的服务器 GPU 不同,由于芯片和功耗的限制,移动 CPU 和 GPU 具有相似的性能,尤其是在深度学习模型推理方面;(2)共享统一的内存,与服务器通常具有独立的 CPU 和 GPU 内存单元不同,移动设备的 CPU 和 GPU 共享同一内存空间,从而避免了数据复制的额外开销. 这两大特性为研究人员提供了采用异构并发计算来加速端侧模型推理的机会.

3.2 FPGA 与 ASIC

FPGA 是一种灵活的硬件加速器,可以在部署后通过编程重新配置其内部电路结构. FPGA 由大量可编程逻辑块(Configurable Logic Blocks,CLB)、存储器块(如片上存储器、块随机存取存储器(Block Random Access Memory,BRAM)和可重配置的互连网络组成. 逻辑块可以根据特定应用的需求被配置为执行不同的计算任务,而可重配置的互连网络则允许灵活地连接不同的逻辑块和存储器,确保数据在处理过程中快速传输. 这种灵活性使 FPGA 能够在不依赖固定硬件的前提下,实现高度并行的定制化计算,从而加速多种计算任务.

ASIC 是一种针对特定应用或任务优化设计的硬件加速器,通常用于加速深度学习、加密处理或通信等特定任务. ASIC 的基本构建块为处理元件(Processing Elements,PE). 每个 PE 都具备独立的本地存储和控制逻辑,并通过二维阵列的方式与其他 PE 高效连接,从而实现数据的快速传输和并行处理. 与 FPGA 不同,ASIC 在设计和制造完成后无法重新配置. 但由于其架构为特定任务量身定制,ASIC 能够充分优化硬件资源利用率,极大提升计算效率和能效比. 近年来,TPU、数据处理单元(Data Processing Unit,DPU)、NPU 等 ASIC 被广泛设计用于加速端侧设备上的 AI 计算任务,例如谷歌 TPU、高通 Hexagon NPU.

3.3 移动端 NPU

移动端 NPU 的架构是专门为优化神经网络运算而设计的,旨在实现机器学习任务的高效处理,同时保持低功耗. NPU 通常包含一组专门针对深度学习的典型操作,如卷积、池化、归一化和激活函数进行优化的硬件加速器,它们可以并行处理大量数据,以提高处理速度和效率. 此外,NPU 的设计还包括高度并行的数据通路和高效的内存管理系统,这些都有助于提高数据处理能力和吞吐量,减少数据传输延迟和能耗. 为了适应不同的应用需求和优化能耗,NPU 支持多种计算精度,如 FP32、FP16、INT8 等. 现代 NPU 还具有一定程度的可编程性,允许开发者自定义操作和优化算法,这对支持新兴的深度学习模型和算法至关重要. 在系统级别,NPU 与 CPU、GPU 以及其他处理单元紧密集成在 SoC 中,优化了数据交互和功耗管理. 为了充分利用 NPU 的硬件能力,通常需要一个丰富的软件生态系统,包括驱动程序、开发框架和工具链,这些工具可以支持高效的模型部署和优化,简化开发过程.

4 端侧模型推理算法与软硬件结合优化

算法层面的优化技术,如模型结构精简、网络剪枝、参数量化等,对于降低计算复杂度、提高运算速度具有显著效果. 同时,借助移动 GPU 和 TPU 等 AI 加速硬件,也能进一步加速端侧模型的推理. 然而,若仅从算法或硬件单方面进行优化,仍难以充分满足端侧设备的部署与推理需求. 算法与软硬件结合优化从系统级别上充分挖掘推理加速的潜力,通过综合考虑算法架构、编译软件和运行时环境以及硬件设计,进行深层次的协同优化,可以实现对计算资源的高效调度,显著减少能量消耗,并大幅降低响应延时. 本章后续部分,将从算法-硬件结合优化,软件优化中的算法与硬件考量,以及算法-硬件协同的调度策略优化三个方面对此展开详细介绍.

4.1 算法-硬件结合优化

算法-硬件结合优化通过将延迟、能耗等直接指标纳入自适应算法,或进一步在算法设计过程中考虑设备硬件的特性(例如数据存储访问模式、指令集架构),从而使端智能系统能够适应要求更为严苛的应用场景,在更低的延迟、能耗或者给定资源预算下实现系统的高效运行. 前者利用从硬件中获得的延迟或者能耗来指导模型的压缩或设计,以解决间接指标不准确的问题,并未深入探索硬件设计空间. 后者通过深入考虑算法与硬件的相关性,例如模型的内核大小、滤波器、层数和精度等参数,硬件加速器的数据存取特性、并行性等,设计相匹配的模型架构与硬件架构,提高计算阵列和片上存储器的利用率,从而最大限度地提高系统效率和性能[121]. 本节将深入讨论算法-硬件结合优化方法,包括基于延迟与能耗的算法设计,模型剪枝与稀疏

计算中的硬件考量,网络架构搜索、模型量化以及低秩分解与加速器的协同设计.

4.1.1 基于延迟与能耗的算法设计

基于延迟或能量的直接测量而不是每秒浮点运算次数(FLoating point Operations Per second,FLOPs)等间接指标的优化可以更好地探索硬件特征. 对于延迟,多数工作通过在终端设备上直接测量整个模型的延迟,或测量模型中的算子、滤波器、层等参数的延迟以构建延迟查找表[95,121~124]. 在执行算法时,根据查找表对延迟进行求和,选出不符合条件的冗余参数从而移除,最终选出符合延迟条件的模型. 然而,通过直接测量来提取模型的延迟对于NAS这种需要大规模搜索算法来说是昂贵的. 为了能够高效地搜索符合延迟条件的模型,部分研究人员基于经验或者使用模型估计延迟. 文献[124]使用TensorRT库提供的分析器来获取模型的分层延迟. 他们凭经验发现,具有特定配置的块总是消耗相同的延迟. 因此,通过构造延时查找表提供每个块配置的延迟,可以有效地估计搜索空间中架构的延迟. 同时,将所有在设定延迟范围内的架构形成子空间,能够显著缩小搜索空间,加速架构选择.

对于能耗的获取,文献[125]提出了一个可估计的CNN推理能耗框架(eyeriss). 每个CNN层的能耗由计算能耗和数据移动能耗两部分组成. 计算能耗等于层中乘累加(Multiply-ACcumulate,MAC)的数量与计算核心中运行每个MAC操作所消耗的能量的加权. 数据移动能耗则通过计算硬件中内存层次结构每个级别的内存访问次数,并将其与该内存级别的每次访问所消耗的能量进行加权获得. 对于每个MAC操作和内存访问的能量数据,则直接使用实际硬件测量值.

4.1.2 剪枝与稀疏计算中的硬件考量

硬件感知的模型剪枝:NetAdapt[123]通过使用经验测量来评估直接指标,从而消除对特定于设备信息的要求. 在每次迭代中,NetAdapt生成多个模型提案并在目标设备上进行测量,测量结果用于指导NetAdapt的下一轮迭代. 基于延迟查找表,文献[122]提出硬件感知的延迟修剪. 该算法首先根据滤波器的重要性进行排序,然后动态调整它们的延迟贡献以对滤波器进行分组,并使用增强背包求解器确定要修剪的滤波器. 文献[126]提出一种基于延迟阶梯特征的三点延迟阶梯判别方法,通过动态定位延迟阶梯的下降位置以获得网络中各层的候选可剪枝坐标. 文献[127]对流行CNN模型的能耗作出详细的分解,并提出CNN能量感知的剪枝算法,直接使用CNN的能耗来指导剪枝过程. 其中,能量估计方法基于文献[125]所提出Eyeriss框架,进一步考虑数据稀疏性和位宽减少的影响,并使用实际硬件测量的能耗推断得到计算和数据移动的能耗. 剪枝方法根据CNN模型中每一层的估计能量,从能耗最高的层开始,到能耗最低的层进行逐层剪枝,以删除对输出特征图具有最小联合影响的权重. 与原始密集模型相比,所提出的剪枝方法使AlexNet和GoogLeNet的能耗分别降低了1/3.7倍和1/1.6倍. 与仅使用模型大小或MAC作为指标的剪枝方法[128]相比,能耗分别降低了20%和10%.

硬件友好型稀疏计算:模型剪枝通过消除深度神经网络中的非关键元素,降低其高计算成本. 然而,剪枝导致了计算和内存访问的稀疏性,使得剪枝后的稀疏模型难以在通用商用硬件(如GPU、TPU、FPGA、Volta tensor core等)上实现有效加速. 因为这些硬件主要针对密集矩阵计算进行优化,而对稀疏矩阵的优化支持有限. 此外,稀疏矩阵需要额外的数据来描述其格式. 以压缩稀疏行格式(Compressed Sparse Row,CSR)为例,如图6所示,稠密的权重矩阵经过剪枝后变为稀疏矩阵,然后使用CSR格式记录. A保存所有非零值,LA记录稀疏矩阵每行中第一个非零元素的索引,JA存储非零元素的列索引. 由于索引数组JA与数据数组A的大小相同,CSR格式中一半以上的数据都用于存储稀疏矩阵格式[129].

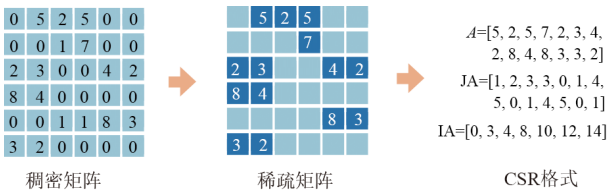


图6　压缩稀疏行格式示例[129]

为了释放稀疏模型的加速潜力,部分研究人员通过深入考虑算法与硬件的耦合关系,对剪枝算法进行优化. 其中,一部分研究人员针对专用硬件FPGA设计了相应的剪枝算法. 文献[130]将权重张量分成多个与FPGA缓冲区大小相同的块,在计算过程中,利用使能信号决定是否将相应的输入特征和权重块加载到片上存储器中. 对于无效的块使能信号,表明该权重块已被修剪,跳过一次加载和计算过程. Plochaet[131]根据脉动阵列(systolic array)的大小删除滤波器,在提高推理时间的同时进一步压缩网络大小. 文献[132]根据每一行的重要性对每个卷积核进行剪枝,每个卷积核只保留一个行权重,其余的所有权重都被剪枝. 这种剪枝粒度介于非结构化和结构化剪枝之间,在相同剪枝率下容易获得比结构化剪枝更高的准确率. 同时由于行尺度剪枝后每个卷积核中保留相同数量的权重,卷积核权重分布具有较高的规律性. 在硬件部署过程中,通过选择输入特征数据进入的行,可以直接跳过剪枝产生的所有零计算,从而避免硬件工作负载不平衡的问题.

然而,专用硬件设计的算法可能无法适用于其他硬件,例如GPU、TPU、NPU等.为了获得算法的通用性,Scalpel[129]通过深入考虑模型结构与目标硬件数据并行结构的特性,设计了一种可以在各种目标硬件平台上高效执行的剪枝方案.这个方案包括两种算法:单指令多数据(Single Instruction, Multiple Data, SIMD)感知权重剪枝和节点剪枝.SIMD感知的权重剪枝基于传统的权重剪枝方法,将连续的权重放入大小等于SIMD宽度的组中.这个操作减少了记录稀疏矩阵格式的额外数据,同时提高了SIMD单元的利用率.节点剪枝则通过删除每层中的冗余节点来压缩模型.由于该算法没有破坏DNN的常规结构,从而避免了现有剪枝技术带来的稀疏性开销.Scalpel的第一步是分析和确定硬件平台的并行级别.根据所含处理器的并行度,Scalpel将所有通用硬件平台分为高中低三类.对于微控制器等并行性较低的硬件,应用SIMD感知权重修剪.对于GPU等高并行度硬件,应用节点修剪.对于中等并行度的硬件,将SIMD感知权重剪枝应用于全连接层,节点剪枝应用于卷积层.不同于Scalpel针对硬件平台定制剪枝技术,文献[133]提出了适用于所有基于通用矩阵乘法(GEneral Matrix Multiplication, GEMM)的硬件的剪枝算法.他们将整个矩阵划分为多个图块,根据每行和列的集体重要性分数,修剪每个图块的整个行或列.然后通过结合智能的数据布局方式与并发/批处理优化方法,解决频繁的未合并内存访问,以及由不同图块导致的负载不平衡和GPU资源利用不足的问题.

此外,许多架构师提出了各种针对稀疏计算的专用加速器或加速引擎设计[134~138].文献[134]提出了Eyeriss加速器,能够支持低功耗SOTA(State Of The Art)网络推理.它通过数据重用最大限度地减少数据移动,并通过图像数据的统计稀疏性减少不必要的数据读取和计算.为了帮助更紧凑和稀疏权重的网络,文献[135]进一步介绍了Eyeriss-V2. Eyeriss-V2通过将全局缓冲区和PE放入单个集群中,并为Eyeriss添加了分层网格,以适应不同场景下的数据利用率和带宽要求.文献[138]提出了稀疏CNN加速器架构SCNN. SCNN利用训练时网络剪枝产生的零值权重,以及推理期间应用的常见线性整流单元(Rectified Linear Unit, ReLU)算子产生的零值激活来提高性能和能源效率.与同等配置的密集CNN加速器相比,SCNN的性能和能效分别提高了2.7倍和2.3倍.

4.1.3 面向硬件特性的神经网络架构搜索

硬件感知的神经网络架构搜索:早期的硬件感知神经网络架构搜索(Hardware-Aware NAS, HA-NAS)将精度、延迟、能量等指标纳入搜索算法的目标函数中指导模型的搭建,如图7所示.

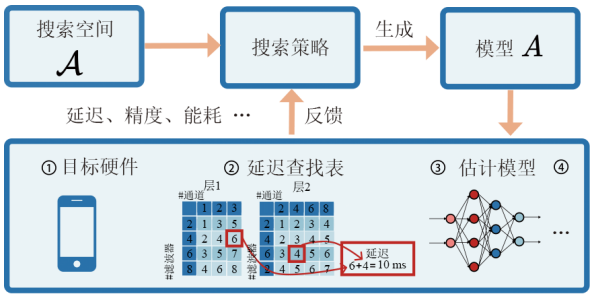


图7 硬件感知的神经网络架构搜索

文献[92]基于神经网络架构搜索和强化学习提出MnasNet.首先,每个采样模型需要在目标任务上进行训练以获得准确性;然后在设备上运行模型以获得推理延迟;接着根据准确性和推理延迟计算奖励值.在每个步骤结束时,使用近端策略优化[139]最大化定义的预期奖励值,以更新强化学习控制器的参数θ,直到达到最大步数或参数θ收敛.在ImageNet分类任务上,MnasNet在Pixel手机上实现了78 ms的延迟与75.2%的top-1准确率.相比于MobileNet-V2提速1.8倍,准确率高0.5%.类似的,文献[95]将延迟纳入损失函数,基于可微神经架构搜索框架提出FBnet.文献[140]基于遗传算法提出Chamnet,将给定的准确性、延迟和能量描述为约束,然后将约束合并为适应度函数,在不断迭代中选择适应度最高的架构.文献[141]利用主动学习测量模型的延迟、能耗及功耗,然后结合代理模型和贝叶斯优化,寻找满足性能约束的最优模型-设备组合,为边缘设备提供了高效的Transformer推理能力.近年来,以MobileNet-V3[94]为典型的HA-NAS协同人工模型设计方法获得了研究人员的关注.MobileNet-V3首先基于HA-NAS构建轻量化网络,然后手动微调优化.其中,HA-NAS算法采用MnasNet作为初始模型,应用NetAdapt[123]进行局部自动微调直至所构建的网络到达期望的延迟.手动微调引入了SE(Squeeze-and-Excitation)通道注意力结构和计算高效的h-swish(x)非线性激活函数,以进一步提高网络的准确性与计算速度.

神经网络架构搜索与加速器协同:为了使设计的模型架构能够与硬件更好地适配,进一步提高硬件计算阵列和片上存储器的利用率,部分研究人员将来自模型和硬件加速器的超参数纳入搜索空间,共同指导模型或加速器的自动化设计[142~149].搜索空间通常在模型与软硬件参数等多个维度进行配置,包括PE阵列大小、MAC电路配置、数据流、平铺策略、主存储器大小和类型以及其他特定领域的模块.

能量延迟乘积(Energy-Delay Product, EDD)[144]在搜索空间中加入FPGA的并行性和循环平铺因子,但由于不能泛化为常见的加速器参数,例如内存大小和每个内存的平铺策略,通用性受限.对此,文献[145]在

EDD基础上,提出适用于不同的加速器架构和映射方法的高效搜索引擎Auto-NBA. Auto-NBA对硬件设计空间采用统一模板,这是一种参数化的基于块的管道微架构. 模板包含多个子加速器并以管道方式执行模型. 每个块被分配有多个但不一定连续的层,这些层在块内顺序执行. 类似的,文献[143]通过创建一组模板,从而为不同类型的ASIC加速器搜索最佳配置. 由于ASIC加速器设计都有特定数据流,因此,文献[143]通过构建一组包含不同数据流样式的加速器模板,将设计空间显著缩小到模板的选择以及模板所对应到的硬件资源的分配,例如PE数量和片上网络带宽(Network on Chip bandwidth,NoC)带宽. CODEBench[147]将CNN加速器的设计空间表示为一个13维向量,包含卷积核大小、卷积核数量、处理单元数量、缓存大小、主存储器类型和大小等超参数. 通过在这些超参数的范围内进行搜索,生成不同的加速器设计并实现模拟.

除了模型与加速器架构,网络即服务(Network As A Servic,NAAS)[150]进一步考虑编译器映射策略,在一个优化循环中全面搜索神经网络架构、加速器架构和编译器映射. NAAS将加速器设计和编译器映射编码为向量,其中,加速器设计通过架构尺寸和连接参数来描述,而映射策略由循环顺序和数组每个维度对应的平铺表示. 在架构评估环节,NAAS使用度量能量延迟乘积来评估给定加速器配置的模型在延迟与能耗上的表现. 由于硬件设计空间与映射空间是巨大的,为加速搜索,文献[148]提出一种基于单循环微分模型的高效硬件设计空间探索框架拒绝服务攻击(Design Of Systems and Architectures,DOSA). DOSA通过构建可微分和可解释的性能模型,捕获DNN映射因素和性能目标之间的关系. 基于该模型使用梯度下降以找到最有效的硬件参数并映射到目标多层DNN,显著减少了加速器设计和映射的时间与成本. 文献[149]提出一种领域信息(domain infomation)引导的贝叶斯优化框架以提高搜索算法效率. 通过为搜索算法提供高级领域信息,算法能够更有效地学习与分类搜索参数之间的相关性并更快收敛.

4.1.4　模型量化与加速器协同

模型量化与硬件的结合优化旨在充分利用硬件的多位宽算术运算能力. 部分研究者利用设备上延迟、能耗或硬件架构设计参数指导量化位宽的选择,以实现高质量的混合精度量化[47,151~154]. 文献[47]以及文献[154]利用强化学习自动确定量化策略,以获得给定资源预算的模型. 以文献[47]所使用方法为例,首先,由强化学习代理决定特定层的权重或激活的量化位宽;然后,从硬件加速器中测量量化模型将使用的资源量. 如果当前策略超出了给定的资源预算(例如延迟、能量或模型大小),强化学习代理将依次减少每层的位宽,直到最终满足约束. 文献[152]利用整数线性规划求解器,对模型扰动、模型大小、延迟和总位操作等约束进行权衡,以找到最佳位精度设置. 文献[153]基于乘数交替方向方法(Alternating Direction Method of Multipliers,ADMM)对模型进行权重剪枝和量化,通过综合考虑硬件性能开销设置特定权重剪枝比率实现的高效模型压缩.

此外,量化与硬件加速器的联合设计在近年来也受到了研究者的关注. 文献[155]提出基于区域的动态量化算法(Dynamic Region-based Quantization,DRQ). DRQ首先识别输入特征图中对模型有重要影响的敏感区域. 然后根据识别结果,对输入特征图进行动态量化. 由于动态量化导致不同敏感区域的数值精度不同,因此需要对底层硬件进行细粒度的控制. 对此,作者进一步提出加速器设计,将不同的精度级别交织在单个PE数组中,以通过不同的计算和内存吞吐量进行卷积. 文献[156]提出芯片硬件感知的量化框架(On-chip Hardware-aware Quantization,OHQ). OHQ由片上量化感知管道(On-chip Quantization Awareness,OQA)和掩模引导的量化估计技术(Mask-guided Quantization Estimation,MQE)构成. OQA能够获取量化算子在硬件上的延迟、内存使用和功耗等指标. MQE利用线性规划来优化位宽配置,同时考虑芯片级别计算能力的限制,从而实现对算子准确性指标的高效估计. 最后,采用im2col算法、乘法和加法树以及子矩阵切片的传输和计算等技术,对量化模型的FPGA部署作进一步优化.

4.1.5　低秩分解与张量加速器协同

低秩分解因独特的分解方式能够获得极高的压缩比,从而显著降低神经网络的存储需求与计算复杂度. 然而,尽管在理论上降低了计算成本,但由于算法特殊的分解方式和存储机制,部署到硬件推理时可能会引入新的计算瓶颈[157]. 因此,需要考虑算法与硬件的关系,制定相应的推理方案和硬件架构,以充分利用低秩分解的压缩优势.

文献[158]提出硬件感知的自动低秩压缩框架(Hardware-aware Automatic LOw-rank Compression,HALOC)采用基于神经架构搜索的自动秩选择方案,通过构建低秩搜索空间,迭代地对不同候选秩设置进行采样和评估,以可微的方式学习最合适的秩设置. 在秩的选择策略中,HALOC通过预测每个候选秩设置在目标硬件设备上的性能表现,指导秩选择过程. 硬件性能可以通过预先测量目标硬件设备的性能,或者通过回归模型进行估计.

此外,部分研究人员通过设计张量加速器提高低秩分解算法实现的效率[157,159,160]. 他们首先通过推导、分析、分解公式等方法设计紧凑的推理方案. 然后针对推理方

案定制硬件加速器,包括硬件架构设计和存储器读取方法等,以充分利用所提出方案的优势. 文献[160]提出模型-加速器协同设计框架StreamSVD. StreamSVD首先使用低秩近似算法压缩模型,为每个卷积层选择适当的分解方案和秩. 然后通过迭代生成量化的权重参数,生成一组具有不同压缩比的压缩模型. 在将每个压缩模型部署在目标FPGA设备上后,输出加速器设计方案以及包括延迟和折叠因子在内的性能信息. 利用加速器性能信息,调整上一步压缩好的模型. 最后将调整好的模型映射到FPGA设备上,生成最终的加速器设计方案.

4.2 软件优化中的算法与硬件考量

端智能部署与推理的编译软件通常指各类深度学习编译器、推理框架、推理引擎与支持库. 它们向上提供高级接口和功能,例如模型加载与推理,以及针对模型算法的优化策略,例如卷积加速、矩阵乘法优化;向下提供针对不同硬件平台的优化支持,根据硬件的特性自动选择合适的计算方式、数据布局和并行方式等优化策略,以提高模型在编译与运行时的推理性能,如图1中编译软件部分所示.

移动端深度学习框架:为了更方便地构建、训练和部署深度学习模型到端侧(如移动设备、微控制器和边缘物联网设备),学术界和工业界提出了各种针对这些设备的高效移动深度学习框架和库,例如TFLite(TensorFlowLite)[161]、神经网络构建(Net Construction Neural Network,NCNN)[162]、移动神经网络(Mobile Neural Network,MNN)[163]、移动人工智能计算引擎(Mobile AI Compute Engine,MACE)[164]、Arm神经网络(Arm Neural Network,ArmNN)[165]、Paddle Lite[166]和PytorchMobile[167]. 这些框架通过综合使用模型压缩、网络结构改进、推理策略优化、异构计算、编译优化等技术,以实现高性能和低能耗的端侧模型推理. 表4总结了现有常用移动深度学习开发框架的特点,包括性能、灵活性、易用性和数据精度支持方面的内容.

表4 当前主流的端侧深度学习模型推理框架

项目	TFLite	NCNN	MACE	MNN	PaddleLite	ArmNN	PyTorchMobile
安卓系统	√	√	√	√	√	√	√
苹果系统	√	√	√(仅CPU)	√	√	×	√
NEON加速	√	√	√	√	√	√	√
CPU大小核调度	×	√	√	√	√	×	×
ONXX模型转换	×	√	√	√	√	√	×
DSP部署	√(NNAPI)	×	√	×	√	×	√(NNAPI)
NPU部署	√(NNAPI)	×	×	√	√	×	√(NNAPI)
半精度浮点数(FP16)	√	√	仅存储	√	√	√	√
8位整数(INT8)	√	√	√	√	√	√	√

为在端侧设备上实现高性能推理,深度学习模型推理框架通常采用了多种软硬件协同优化策略,包括模型压缩、网络结构改进、推理策略优化、异构计算、编译优化等技术等.

(1)压缩技术支持:为了减少模型的存储和计算需求,框架通常支持多种压缩技术,包括权重量化(如INT8、FP16)、剪枝、低秩分解等. 这些技术通过减少冗余参数和计算操作,有效降低内存占用和推理延迟,尤其在资源受限的设备上表现突出.

(2)网络结构改进:框架通常对经典网络结构进行改进,使其更适合在移动设备上的推理需求. 例如,TensorFlow Lite框架中集成了针对移动设备设计的轻量化网络MobileNet和EfficientNet,并提供了对这些高效网络的优化支持.

(3)推理策略优化:框架通过动态调整推理策略(如基于输入大小的自适应推理和层级推理)来提升推理效率. 此外,框架还引入了流水线并行、批处理推理等技术,进一步减少推理延迟并提高处理吞吐量.

(4)异构计算支持:为了充分利用移动设备上的不同计算单元(如CPU、GPU、NPU),这些框架实现了异构计算支持. 通过自动调度和任务分配,模型的不同计算任务可以被分配到最合适的件单元,以实现最高效的推理. 例如,TensorFlowLite的GPU Delegate、MNN的NPU支持以及NCNN对Vulkan的支持,都是异构计算优化的典型案例. 此外,许多框架支持与特定硬件加速器(如ArmNN的Mali GPU、MNN的Huawei Kirin NPU)进行无缝对接,利用硬件加速器的能力显著提升推理速度,同时降低功耗. 对于支持量化推理的加速器,框架可以充分利用低精度计算单元进一步加速推理过程.

(5)编译优化:框架通过针对具体硬件平台的编译优化来提升推理性能. 例如,内联优化、运算符融合和卷积内核优化技术,能够减少模型推理中的函数调用开销和内存带宽需求. 某些框架还提供了针对特定处理器架构(如Arm Cortex-A系列)的深度优化,通过SIMD指令加速矩阵计算、卷积操作等核心运算.

深度学习编译器:作为模型部署到硬件的最后一环,深度学习编译器通常集成在Pytorch、TensorFlow-Lite、MNN等高级机器学习框架的后端.通过结合面向模型的优化与通用编译器的成熟工具链,深度学习编译器能够实现从深度学习框架中描述的模型定义到能在硬件上高效运行的代码转换,并充当硬件加速器的执行引擎[168].工业和学术界提出了几种流行的深度学习编译器,例如TVM[169]、Glow[170]、nGraph[171]和实验性线性代数(eXperimental Linear Algebra,XLA)[172].

深度学习编译器通常采用分层设计,包括前端、中间表示(Intermediate Representation,IR)和后端.模型在深度学习编译器中被转换为多级IR,其中,高级IR表示驻留在前端,低级IR表示驻留在后端.基于高级IR,编译器前端负责与硬件无关的转换和优化,例如计算图优化与子图分裂、常量折叠、算子融合与下沉.基于低级IR,编译器后端负责特定于硬件的优化,包括代码生成与硬件指令优化、静态编译与动态运行时优化,例如指令调度、静态内存分配、布局转换.这些优化方法的共同目标是通过软硬件协同的方式,在资源有限的移动端设备上提升深度学习模型的推理效率和性能.通过深度学习编译器的IR与硬件指令的高效转换,模型能够充分利用硬件加速器的计算能力,实现高效推理.详细内容见4.2.2节与4.2.3节.

4.2.1　算法与编译器结合优化

模型设计和优化完成后,需要利用深度学习编译器生成针对目标硬件的优化代码.一种直接的方式是将模型算法设计与编译器集成优化以提升性能.然而,直接集成难以为目标设备产生最有效的模型[173].在传统的优化工作中,模型设计和编译器优化是两个相对独立的方向.模型设计侧重于在算法层面上设计和训练高性能的深度学习模型,而编译器优化则专注于利用操作系统层面的优化技术,例如指令调度、并行化、内存访问优化等,将模型映射到目标硬件上,以实现高效的执行.仅通过将这两个阶段串联起来,往往无法达到最佳性能,需要紧密结合模型设计和编译器特性进行优化.例如,编译器提供关于目标硬件特性和性能的信息,以指导模型设计的决策.

模型剪枝与编译器的结合优化在近几年来获得了广泛关注,一种方式是先设计高效的剪枝算法,然后为剪枝算法设计相应的编译器优化方案[174-178].实时三维(Real-Time 3D,RT3D)[177]为3D CNN设计了两种结构化稀疏方案和一种多目标剪枝算法.基于此,RT3D采用编译器辅助的代码生成框架,利用微调的高效SIMD执行、微调权重布局组织等方法将剪枝优势转化为性能提升.剪枝卷积(Pruned CONVolution,PCONV)[174]、PatDNN[175]和ClickTrain[176]基于模式剪枝设计了高效的压缩方法,然后采用一组基于编译器的优化进一步加速移动设备上的端到端推理.以PCONV为例,首先利用剪枝得到压缩后的模型,包括模式剪枝与连通性剪枝.模式剪枝通过在卷积核内剪枝固定数量的权重,使得在每个过滤器中产生相同的稀疏率与有限数量的模式形状.其中,模式形状的设计考虑了计算机视觉特性以提高准确性.连通性剪枝在卷积核间进行,它切断某些输入和输出通道之间的连接,以进一步提高压缩率并保持滤波器计算的平衡工作负载.为了部署PCONV模型,研究人员进一步开发了编译器辅助的推理框架,通过执行步骤(1)分层信息提取、步骤(2)过滤器内核重新排布与步骤(3)负载冗余消除,在编译器层面为剪枝后模型加速.其中,编译器通过利用在分层信息提取时得到的模型信息(例如模式分布、模式顺序以及通过内核的输入/输出通道之间的连接)来执行步骤(2)和步骤(3)的优化,实现了模型算法到编译器的紧密结合.

另一种方法是编译器感知的模型剪枝[173,179].网络剪枝与架构搜索(Network Pruning and Architecture Search,NPAS)[179]是一种编译器感知的联合网络修剪和架构搜索框架.基于强化学习和贝叶斯优化,NPAS对预处理后的模型进行搜索,以找到最佳的剪枝方案和架构,例如每层的滤波器类型与剪枝率.同时,NPAS考虑模型的延迟约束.它通过编译器生成优化代码,并在目标硬件上进行实际测量,将这些信息纳入方案搜索的奖励函数中,以确保搜索结果满足约束.CPrune[173]直接利用编译器优化期间的模型子图结构及其在目标设备上的执行时间,指导模型剪枝过程.通过这种编译器感知的模型剪枝,CPrune实现了面向目标的性能提升.

4.2.2　编译软件的算法优化策略

编译软件的算法优化策略主要体现在计算图优化与卷积算子的加速上.在模型正式部署到端侧设备以前,编译软件(通常是深度学习编译器前端部分)需要将模型从TensorFlowLite、NCNN等机器学习框架中解析出来并转换成计算图,完成对模型推理过程的高级抽象.计算图是由节点和边构成的有向无环图,节点代表了算子(operator),边代表算子之间的数据依赖关系.根据计算图,推理即从输入数据开始,按照节点间的依赖关系一步步计算,最终得到输出结果.算子可以理解为模型中的各类运算单元,通常包括代数运算(例如,+、×),神经网络运算(例如,卷积、池化),张量运算(例如,reshape,resize),控制流运算(例如,条件、循环)等.编译软件通常还支持程序员自行定义新的算子,以提高可扩展性.

常量折叠:常量折叠旨在将模型中可以在编译时计算的表达式提前求解,从而减少推理时的计算负

担. 它的核心思想是在编译阶段识别计算图中涉及常量的操作,提前执行这些操作,将结果替换为常量存储在模型中,以此减少推理阶段的计算量,降低内存和带宽需求,提升整体推理性能. 例如,TensorFlow Lite会在模型转换为推理格式(TFLite FlatBuffer格式)时执行常量折叠. 如果模型的某些层仅依赖于常量输入(例如常量张量与某个动态输入进行加法操作),TensorFlow Lite会在编译时计算常量部分的结果,并将其作为优化后的模型的一部分,从而减少运行时的计算.

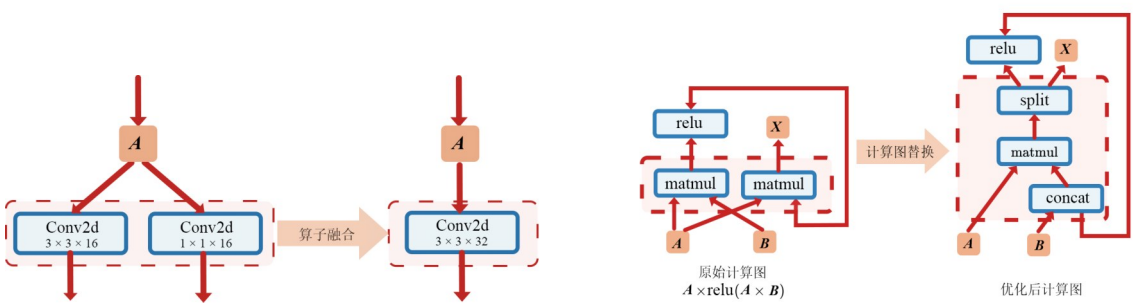
常量折叠主要用于模型中与常量相关的算子,例如加法、乘法、矩阵乘法等涉及固定参数的操作. 深度学习编译器会在生成IR时对这些操作进行检测,并执行静态求值. 常见场景包括以下三种.

(1)固定参数的算术操作:在卷积神经网络中,常常有固定权重与输入数据进行卷积. 如果某些层的输入是常量值,编译器可以提前计算这些常量的卷积运算结果,将其存储为一个新的常量张量,直接用于后续计算.

(2)常量矩阵乘法:对于常量矩阵的乘法操作,编译器可以提前计算结果,尤其是涉及大型全连接层时,这种优化能大幅降低推理的计算需求.

(3)常量函数调用:如果模型中包含涉及常量输入的非线性函数(如ReLU、Sigmoid等)的计算,编译器可以预先计算这些函数的输出结果,避免在每次推理时重新计算.

计算图优化:深度学习编译器通过对计算图进行全局优化,移除冗余节点、剪枝无效等操作,并对计算节点进行重新排序,从模型运算的整体拓扑结构层面完成预推理的优化. 其中,算子融合与计算图替换是常见的优化方式,如图8所示. 此外,针对异构计算设备(如CPU、GPU、NPU),深度学习编译器可以将计算图划分为多个子图,分别分配到不同的计算单元上进行并行计算,从而实现计算任务的最大化加速.



(a) 卷积算子融合 (b) 计算图替换

图8 计算图优化方法

算子融合将多个算子合并为一个算子,从而减少中间结果的存储和读取操作,降低内存带宽的使用,提高执行效率. 例如,卷积和激活函数可以在硬件执行中同时进行,避免不必要的数据搬运. TensorFlowLite、TVM、MNN、PytorchMobile和DNNFusion[180]基于识别所设计的融合模式进行优化. 文献[181]提出内存成本模型驱动的最优算子融合框架Optimus. 所设计的内存成本模型可以通过调度程序,有效地将融合算子组合映射到给定的加速器. 在考虑特征图和参数的情况下,找到最小片外内存开销的融合算子组合.

计算图替换是将原始的计算图替换为等效但计算量更小或更高效的计算图. 对于给定的计算图中,某些子图或算子组合可能具有多个不同的计算实现方式. 通过分析这些实现方式,编译器可以选择计算开销最低的替代方案,从而优化整个计算图的执行效率. 文献[182]提出自动生成图替换的优化器张量代数替换(Tensor Algebra SubstitutiOns,TASO). TASO将算子规范列表作为输入,并使用给定的算子作为基本构建块生成候选替代. 然后执行基于成本的回溯搜索,应用替代来找到优化的计算图.

卷积加速与矩阵乘法优化:流行的轻量级模型,如ResNet[183]、MobileNet[75]和EfficientNet[184],都大量使用了各种卷积计算方法,包括普通卷积、深度可分离卷积和分组卷积等. 与全连接层和池化层相比,卷积计算在模型推理中占据了大部分的时间和能量消耗[185]. 因此,优化卷积计算对于提高端智能推理性能至关重要. 编译软件中常见的卷积优化方法主要包括im2col,Winograd[186]和滑动窗口(slide-window). im2col算法的核心思想,是将卷积操作转换为通用矩阵乘法操作来实现计算速度的提高. 然而,im2col算法在计算时需要将影响输出像素值的输入像素块,复制到对应输出像素的矩阵行中,该转换引入了存储和带宽的非平凡开销,不利于端智能的高效推理. 对此,FaceBook的移动端量化神经网络加速库(Quantized Neural Networks

PACKage,QNNPACK)提出间接卷积算法(the indirect convolution algorithm)[187]. 通过引入一个间接缓冲区,算法将输入张量的每一行的地址存储在缓冲区中,然后将这些地址作为参数传递给修改后的通用矩阵乘法函数,从而实现卷积操作. 与im2col算法中直接复制到内存的方法相比,有效地减小了内存消耗并成功提速.

除此之外,工业界的移动端深度学习引擎广泛采用了Winograd算法以提升卷积性能,例如阿里巴巴的MNN,小米的MACE,Google的TensorFlowLite和腾讯的NCNN. 给定输入特征图的大小为$[i_w, i_h, i_c]$,输出特征图的大小为$[o_w, o_h, o_c]$,Winograd卷积可以表示为

$$\boldsymbol{Y}=\boldsymbol{A}^{\mathrm{T}}\left[\sum_{\text{channel}}\left(\boldsymbol{G}\boldsymbol{W}\boldsymbol{G}^{\mathrm{T}}\right)\otimes\left(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{B}\right)\right]\boldsymbol{A} \tag{1}$$

其中,$\boldsymbol{G}$、$\boldsymbol{B}$、$\boldsymbol{A}$分别为卷积核$\boldsymbol{W}$(空间大小为$[k,k]$)、输入$\boldsymbol{X}$(空间大小为$[i_w+k-1, i_h+k-1]$)、输出$\boldsymbol{Y}$(空间大小为$[n,n]$)的变换矩阵,这三个矩阵仅依赖于$\boldsymbol{W}$和$\boldsymbol{X}$的形状. Winograd算法通过变换卷积核和输入数据,将卷积运算转化为更简单的矩阵乘法运算,从而提高计算效率. TensorflowLite和MACE将$\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{G}$矩阵直接写入源代码中,而MNN通过设计Winograd生成器,动态生成适用于不同卷积核和输入尺寸的$\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{G}$矩阵,实现了更好的可扩展性. 同时,MNN对输入矩阵进行合理的块划分,并基于数据布局重新排序的矩阵乘法,优化了Winograd卷积中内存访问耗时较长的Hadamard点积. 这两步优化使得能够利用并行计算和流水线处理,进一步提高Winograd卷积的计算效率. Kernel大小为1的卷积操作本质上是矩阵乘法,MNN使用Strassen算法来对其进行优化. 用更计算更快的加法操作替代若干乘法操作,并递归地应用Strassen算法加速计算过程,从而提高矩阵乘法的效率.

4.2.3　面向硬件特性的编译软件优化

在算法层面,编译软件使用算子融合、替换等方式以实现整体拓扑结构的优化. 而在硬件层面,编译软件更加关注算子的具体实现,即输入、输出、内存循环方式和计算的逻辑. 为了提高算子在硬件上的运行效率,常见的优化方法包括内存分配优化、循环优化和异构计算技术.

内存分配与延时隐藏:模型推理时产生的中间计算结果(称为中间张量)通常会预先分配以减少推理延迟,但在资源受限的端侧设备中,预先分配的方式可能会导致大量内存空间的占用. 例如,MobileNet-V2的中间张量占用了26 MB的运行时内存,大约是模型大小的2倍[188]. TensorFlow Lite通过重用中间张量的内存缓冲区,以减少总内存占用. 对此,他们首先为中间张量设计了独特的数据结构,即张量使用记录(tensor usage records). 张量使用记录记录了有关张量大小,以及在模型给定执行计划中第一次和最后一次使用的信息. 借助这些记录,内存管理器能够计算模型执行过程中任何时刻的中间张量使用情况,并优化其运行时内存以尽可能减少占用空间. 同时,根据张量使用记录与不同的后端,设计了几种启发式算法优化内存分配. 文献[189]提出一种配置文件引导的内存分配方法. 他们收集模型推理过程中有关请求的内存块的信息,然后使用启发式算法分配内存以减少峰值内存占用. 基于启发式算法的还有XLA和TVM. 然而,由于基于启发式的算法无法解决更复杂的工作负载情况,一些基于求解器的方法被提出[190],但通常因速度太慢而无法在线运行. TelaMalloc[191]通过并行运行启发式算法和求解器实现两全. 在分配缓冲区的每一步,TelaMalloc从一组可能的启发式中进行选择. 然后,用所选启发式做出的决策更新求解器,并使用求解器的输出指导未来的决策,最终在Google Pixel 6手机和TPUv4上实现了两个数量级的分配时间加速.

内存延迟隐藏的目标是通过重新组织指令执行顺序,使计算单元能够在等待内存读取结果时,继续执行其他指令,从而隐藏内存访问延迟. 针对不同的硬件,延迟隐藏的策略通常不同[169]. 对于CPU,内存延迟隐藏通过同步多线程或硬件预取实现. GPU则依赖于warp的快速上下文切换和对多线程的管理调度实现. TPU、NPU等专用深度学习加速器,倾向于通过解耦访问执行架构来控制内存和计算的同时执行,并通过令牌入队/出队操作保证流程的正确执行. 对这类加速器进行编程是困难的,对此,TVM通过引入虚拟线程调度原语,以减轻对类TPU加速器进编程的负担.

循环优化:循环优化(loop optimization)的目的是改善循环结构的计算方式,提高数据局部性,以充分利用硬件缓存提高计算速度. 在端侧编译软件中,编译器后端利用循环展开(loop unrolling)、循环平铺(loop tiling)、循环重排(loop reordering)等技术来实现这一目标.

循环展开通过增加每一步循环的步长,然后在循环体内补充上所需要的操作,使其在功能不变的情况下减少循环次数,进而减少循环开销以加快计算速度. 循环展开可为具有多个功能单元的处理器提供指令级并行,有利于指令流水线的调度.

循环平铺通过将循环分“块”(Tile)执行,使得每个小块的数据能够有效地放入硬件缓存中,从而避免因为缓存未命中(cache miss)而花费大量时间在载入数据上. 许多深度学习编译器会自动进行循环平铺的优化,通过分析代码的访存模式和硬件特性,确定合适的块模式和大小. 这样可以充分利用硬件缓存,提高计算速度,同时减少了对内存带宽的需求.

循环重排则是一种改变循环嵌套顺序的技术. 通

过重新排列循环的顺序,可以改变数据的访问模式,提高数据的局部性,减少对主存储器的访问延迟.

4.3 算法-硬件协同的调度策略优化

目前,在移动和嵌入式设备上采用多核处理器的趋势日益明显. 例如,小米14上配备的骁龙8 Gen3 CPU,采用1+5+2的八核架构设计,包括一个3.3 GHz超大核心Cortex-X4,5个3.2 GHz Cortex-A720大核和2个2.3 GHz Cortex-A530小核. 除多核处理器外,移动和嵌入式设备上广泛使用了深度学习加速芯片,例如NPU、TPU和APU. 因此,如何调度与利用移动设备上的异构计算资源,实现深度学习模型的高效端侧推理,在近年来受到了学术与工业界的广泛关注.

合理划分工作负载分区是端侧异构计算优化的重点. 为充分利用CPU上大核的计算资源,文献[192]提出一种名为AsyMo的新技术,以确保深度学习模型推理能够在非对称多处理器上获得令人满意的性能. AsyMo实现了成本模型引导的分区和非对称感知任务调度,以允许在非对称CPU上窃取任务. 此外,AsyMo根据模型的数据复用率,确定最低能源成本频率,以减少能源消耗. 对于异构处理器上的并行加速,文献[193]提出了μlayer,将卷积运算划分到移动CPU和GPU上,并利用处理器友好的数据类型来提高CNN的效率推理. 文献[194,195]提出一种并行执行算法,将深度学习模型推理分配给CPU和GPU,以平衡工作负载. 文献[196]提出带宽感知的神经网络部署(Bandwidth-Aware neural Network Deployment,BAND),可协调异构处理器上的多DNN工作负载. BAND预先检查深度学习模型并将其划分为一组子图. 然后,BAND根据当前子图与各个处理器的兼容性以及运行速度,动态地将子图调度到最优的加速处理器上,如图9所示. 文献[197]设计了一种启发式模型划分算法,根据CPU和NPU之间的计算速度和数值精度两个特征,动态决定DNN层应该在CPU上运行还是在NPU上运行. NN-Stretch[198]将顺序执行的模型,转化为具有多分支结构的模型,尝试通过改变分支汇聚位置以及每个分支如何缩放以适应异构处理器. 并设计了一种基于子图的空间调度器以并行执行多分支模型,实现了在CPU+GPU+DSP的异构处理器组合下的高效推理.

在面向实际应用场景的优化中,许多工作综合考虑算法设计硬件调度方案,以实现推理加速与性能提升[194~195,199~204]. DeepMon[199]和DeepCache[200]通过利用连续视频帧的时间局部性,合并连续帧中的相似特征以消除信息冗余,并制定特殊的缓存机制加速CNN推理. DeepSense[201]分析多个经典CNN的网络架构发现,超过90%的计算发生在卷积层内. 通过考虑移动GPU

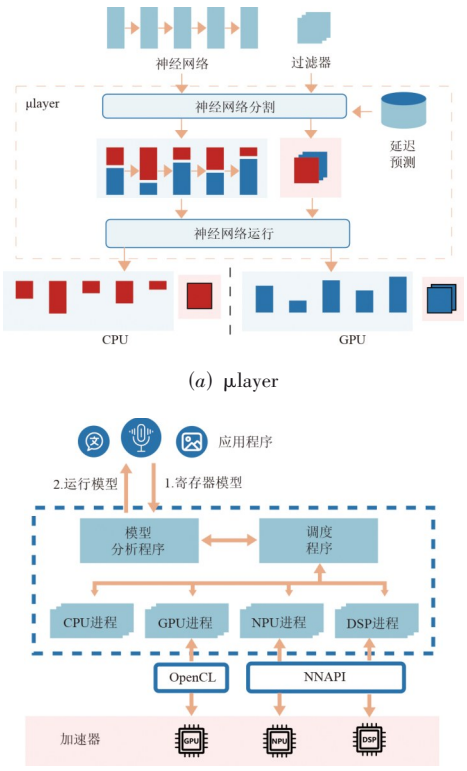


图9 两种典型异构计算优化框架的工作流程示例

的特性以及CNN结构内的数据表示,DeepSense将卷积层卸载到移动GPU以减少计算延迟. 同时,为解决移动GPU的分支分歧(branch divergence)与内存合并(memory coalescing)问题,DeepSense设计了分支分歧消除与内存向量化方法,进一步提高了模型推理效率. 文献[202]提出一种基于多路径神经网络的实时目标检测系统. 该系统利用最坏情况执行时间模型,分析GPU上的处理器和内存争用情况,以指导多路径神经网络的构建. 文献[194,195]提出移动设备友好的轻量化模型,可在移动设备上实现对多人关键点的细粒度跟踪. 同时,构建了一个并行执行引擎以利用移动设备的CPU和GPU来加速执行模型. Mobidepth[203]针对移动GPU进行立体匹配算法(Semi-Gglobal Block Matching,SGBM)优化,通过改进内存带宽利用率、减少并发线程数量等措施,降低了移动GPU上的实时深度估计的计算开销. 微控制器单元网络(MicroController Unit Network,MCUNet)[204]结合设计了高效神经架构(TinyNAS)和轻量级推理引擎(tinyengine). TinyNAS通过优化搜索空间以适应资源约束(设备、延迟、能量、内存),然后在优化的搜索空间中专门化网络架构. 根据整体网络拓扑,TinyEngine调整内存调度以扩展搜索空间并适应更大的模型.

5　端侧大模型推理加速

2022 年底，OpenAI 公司推出搭载了 GPT3.5 的 ChatGPT，凭借逼真的自然语言交互与多场景内容生成能力，迅速引爆互联网，AI 技术迎来大模型时代. 相较于传统意义上的小模型，大模型具有大规模参数和复杂网络结构，同时需要大量训练数据，并且当模型规模超过一定阈值后表现出更为复杂的特性和能力[205]，能够自主学习并发现新的、更高维度的范式和特征，这种能力被称为“涌现能力”(emergence). 涌现能力使得大模型能够在更加复杂的任务上，如理解复杂语境、生成连贯文本、理解和生成高质量图像等，显示出远超越于传统小模型的性能.

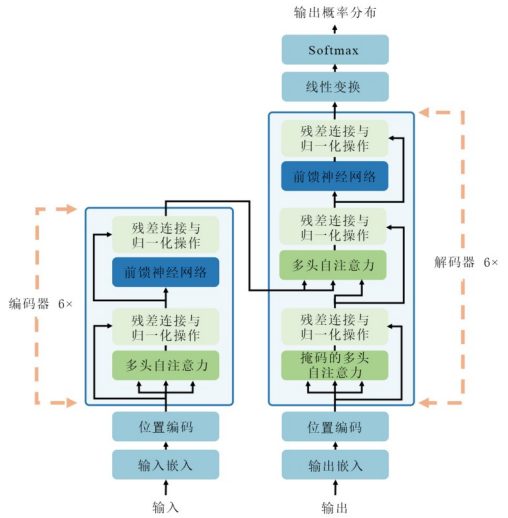
随着用户对即时高效且个性化智能服务需求的不断增长，大模型下沉到终端设备上已成为必然. 在不依赖云端数据存储与计算中心的情况下，搭载大模型的端智能系统能够直接在用户设备上提供高度个性化和即时响应的服务，同时保证隐私安全，极大提升用户体验和设备的实用价值. 例如，智能手机上的语音助手，过去主要实现信息查询、日程管理等基础功能. 嵌入大语言模型后，语音助手能够更精准地理解用户的复杂指令与语境，进行更深入的自主思考与理解，包括理解隐含意图、进行逻辑推理，甚至参与情感交流. 本节将重点探讨现有大模型在智能终端设备上部署与推理的相关优化工作，包括大模型基础架构 Transformer 及其轻量化设计，以及大模型端侧推理的优化技术路线.

5.1　Transformer 架构及其轻量化设计

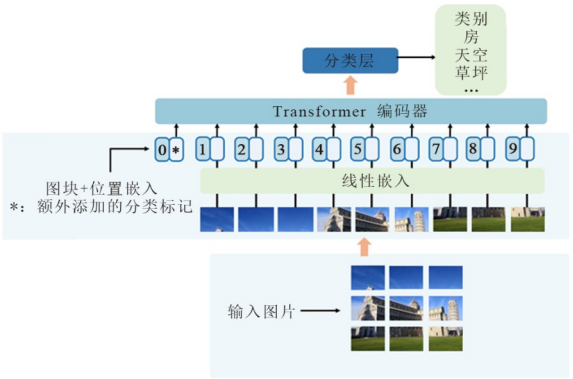
5.1.1　Transformer 基础架构

Google 于 2017 年提出 Transformer[206]，在机器翻译、文本生成等自然语言处理与图像分类、目标检测等计算机视觉任务中取得巨大成功，并随后快速应用于端智能领域，现已成为了大语言模型的核心基础架构. Transformer 是一种基于自注意力机制(self-attention)的神经网络架构. 如图 10(a)所示，Transformer 由编码器和解码器两部分组成，编码器将输入序列转换为一系列上下文向量，解码器则根据这些上下文向量生成输出序列. 其中，每个编码器和解码器又由多个相同的层叠加组成，每层包括多头自注意力机制(Multi-Head Self-Attention，MHSA)和前馈神经网络(Feed-Forward neural Network，FFN). 多头自注意力机制允许模型在计算每个 Token 的表示时，关注输入序列中所有其他 Token 的信息. 通过多个注意力头，模型可以从不同的表示子空间中获取信息，更好地捕捉全局依赖关系. 同时，由于 Transformer 不使用递归或卷积结构，无法直接获取序列位置信息. 因此需要通过位置编码(positional encoding)，向输入嵌入(input embedding)中添加位置信息，使模型能够感知序列的顺序.

2018 年，OpenAI 和 Google 分别发布了以 Transformer 为基础的 GPT-1 和基于 Transformer 的双向编码器表示(Bidirectional Encoder Representations from Transformers，BERT)[207]，奠定了 Transformer 在大模型领域的算法架构基础. 此后，各类大模型层出不穷，例如 OpenAI 的 GPT 系列、Google 的 Bard 大模型、百度的文心一言. 2020 年，Google 将标准 Transformer 架构引入图像分类任务，提出 ViTs[208](Vision Transformer)，成为了 Transformer 架构在计算机视觉领域的里程碑著作. 如图 10(b)所示，ViTs 将图像分割成固定大小的图块(patch)，对每个图块进行线性嵌入并添加位置嵌入，然后将得到的向量序列输入到标准 Transformer 编码器中. 为了分类，需要在输入序列中添加一个可学习的“分类标记”，其对应的输出即为最终的类别预测.



(a) Transfomer 结构



(b) ViT 结构

图 10　Transformer 与 ViT 结构图.

5.1.2 端侧轻量级ViTs设计

在计算机视觉领域，ViTs架构作为CNN的替代方案受到了学术界的重点关注. 首先，ViTs通过多头自注意力机制实现全局感受野，这允许模型在处理每个图像块时考虑整个图像的信息，而CNN只能处理局部信息. 其次，多头自注意力机制不仅能捕捉一阶信息，还能通过注意力热图作为动态权重学习二阶信息. 这使得ViTs能够更好地建模复杂的空间关系和特征交互，而这些特性在传统的CNN中是缺失的. 最后，ViTs具有高度的灵活性，更容易适应不同的视觉任务，如分类、分割、检测和图像合成. 其通用的Transformer架构使模型可以通过简单的调整以应用于各种任务，而CNN需要针对任务设计特定的架构. 然而，与流行的轻量CNN模型相比，由于多头自注意力机制的计算复杂度较高，致使ViTs所需的计算量更大，延迟更高. 即使压缩ViTs的大小以匹配移动设备的资源限制，其性能也明显比同等的轻量CNN模型差. 例如，参数大约在500~600万的DeIT模型[209]的准确度比同等的MobileNet-V3低3%[210].

目前，端侧ViTs的轻量化思路主要分为两个方向，包括降低注意力机制的二次复杂度(quadratic complexity)以及混合架构设计. 对于注意力机制，以$\boldsymbol{x}$作为输入，其中$\boldsymbol{x}\in\mathbb{R}^{n\times d}$，包含$n$个具有$d$维嵌入向量的Token. 使用三个矩阵$\boldsymbol{W}_Q$、$\boldsymbol{W}_K$和$\boldsymbol{W}_V$将输入$x$投影为查询(Query，$\boldsymbol{Q}$)、键(Key，$\boldsymbol{K}$)和值(Value，$\boldsymbol{V}$)，每个自注意力层包含$h$个头(head)，此时自注意力可以描述为

$$\bar{\boldsymbol{x}}=\text{Soft max}\left(\frac{\boldsymbol{Q}\cdot\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d}}\right)\cdot\boldsymbol{V} \tag{2}$$

因此由式(2)得到自注意力的复杂度是$O(n^2\cdot d)$. 随着Token数量增加，$\boldsymbol{Q}\cdot\boldsymbol{K}^{\mathrm{T}}$的计算和内存需求呈二次方增长，导致推理速度慢、内存占用高，难以在端侧进行高效计算. SwinFormer[211]及其后续工作[212,213]提出基于窗口的注意力，将感受野限制在预定义的窗口大小，这也启发了后续工作来完善注意力模式[214~217]. 他们通过预定义注意力范围，使计算复杂度降低至与图像分辨率成线性关系. 然而，由于密集的形状(shape)和索引(index)操作，这些复杂的注意力模式通常难以在移动设备上得到支持[210].

混合架构设计是指将轻量CNN与Transformer单元结合，以在保留二者架构的优点的同时，实现端侧高效计算. 常见方式是将卷积放在推理早期阶段以捕获局部信息，将多头自注意力机制放在最后几个阶段来建模全局依赖性[210]. 苹果公司于2021年提出MobileViT[218]. 该架构结合了MobileNet-V2模块和多头自注意力机制. 前者被放置在网络的早期阶段以提取低级特征，后者被放置在后期阶段以实现全局感受野. 但MobileViT的主要效率瓶颈仍是多头自注意力机制，为此，MobileViT-V2[219]提出可分离自注意力，将多头自注意力机制中计算量大的操作(例如批量矩阵乘法)替换为逐元素操作(例如求和和乘法). EfficientFormer[220]将网络分成4D分区和3D分区，其中4D分区以卷积网络的形式实现，3D分区执行线性投影和注意力机制. 网络从4D分区开始，在最后阶段应用3D分区. 然而，仅在最后阶段使用3D分区可能会导致上下文表示不一致和不足. 针对该问题，SwiftFormer[221]引入了可以合并到所有阶段的高效附加注意力模块(efficient additive attention module)，使局部和全局表示的学习更加一致. 除了卷积和Transformer单元的串联使用，MobileFormer[222]选择将MobileNet和Transformer并行化推理，并在两者之间建立双向桥进行通信，以融合本地和全局特征.

5.2 大模型端侧推理优化技术

2023年2月，高通公司通过量化、编译和硬件加速进行全栈AI优化(https://www.qualcomm.com/news/onq/2023/02/worlds-first-on-device-demonstration-of-stable-diffusion-on-android)，首次在安卓智能手机上成功部署拥有超过10亿参数的Stable Diffusion模型. 尽管大模型在端侧推理加速已经取得初步突破和进展，但相关技术尚未发展成熟，并且其在优化技术上与传统小模型存在较大不同. 本节将介绍目前大模型端侧推理加速的相关技术.

5.2.1 低精度量化

通过使用低位整数量化权重和激活，低精度量化技术可以有效降低GPU的内存需求、大小和带宽，并加速密集矩阵计算，在端侧大模型部署与推理优化领域受到了广泛关注. 然而，与传统CNN模型或BERT等常规Transformer模型不同，大型语言模型(Large Language Models，LLMs)的激活函数很难进行量化. 文献[223]的研究表明，当模型参数超过6.7亿时，激活中出现的大幅度系统性离群点会导致较大的量化误差和精度下降. 文献[224]通过两部分量化解决该问题. 该方法首先进行向量量化，对矩阵乘法中的每个内积使用单独的归一化常数，以量化大多数特征. 对于出现的离群值则采用混合精度量化，保留FP16中的离群点，并使用INT8量化来处理其他的激活函数. 文献[223]提出了一种训练后量化方案SmoothQuant，在离线期间将缩放方差从激活迁移到权重，以降低激活的量化难度. 文献[225]则参考激活的分布进行权重量化，设计了一种每通道缩放方法，自动搜索最小化全权重量化下的量化误差的最优缩放. 文献[226]结合激活引导的量化框架Agile-Quant与边缘设备(例如CPU和树莓派)中常见的SIMD单元，实现了在边缘更快地推理LLM. Agile-Quant首先

对硬件延迟和激活情况进行分析,使用基础的激活量化策略实现在任务性能和推理速度间的权衡. 然后,利用激活感知的Token剪枝技术来减少异常值(outlier)以及对注意力机制计算的不利影响. 最后,利用基于SIMD的4×4 INT4乘法器和TRIP矩阵乘法,为Agile-Quant的硬件实现设计了面向边缘的加速优化.

5.2.2　KV-Cache优化

在基于Transformer的语言大模型的向前推理中,自注意力机制(self attention)需要计算每个输入元素(称为查询,query)与所有其他输入元素(称为键,key)之间的关系,并对应每个键的值(value)进行加权求和. 在处理长序列时,每步都需要计算所有历史步骤的键和值,这会消耗大量的计算资源. KV-Cache(Key-Value Cache)技术通过缓存先前步骤的键和值到内存中,从而避免重复计算,节省计算资源并提高计算效率. 然而,随着批大小(batch size)的增加,KV-Cache所占用的显存也会迅速增加. 对此,大量研究人员通过减少缓存的键值序列的长度以压缩KV-Cache. 泛化查询注意力(Generalized Query Attentio, GQA)[227]通过使用多个注意力头共享键值矩阵,以减少需要缓存的键值矩阵数量. 文献[228]根据不同的注意力结构自适应地构建键值矩阵缓存. 文献[229~231]通过压缩Token以节省内存消耗. 文献[232]增量地将指定范围的Token压缩为紧凑的Token,以减少KV-Cache的长度. 文献[233]与文献[234]建议在KV-Cache中仅存储初始和最近Token的键值. 文献[235]提出一种动态KV-Cache驱逐策略,仅将一小部分KV-Cache保留在内存中. 文献[236]将所有层的查询与仅顶层的键值进行配对,这样就不必缓存甚至计算顶层以外的层的键值,从而节省了内存消耗和计算量. 另一方面,文献[237]从KV-Cache对GPU内存使用管理层面出发,针对现有KV-Cache系统中存储碎片化和动态键值缓存难以估计的问题,提出了PagedAttention技术. 该技术使用操作系统中的分页机制管理键值缓存,显著降低了KV-Cache的显存占用.

5.2.3　混合专家

混合专家(Mixture of Experts, MoE)的思想在于将大模型拆分成多个小模型(即专家,expert),在推理时由门控网络激活特定的专家用于计算,从而达到节省计算资源的效果. 然而,由于MoE架构需要更多的内存以容纳多个专家,其内存效率相较于计算密集型的LLM要低得多,这给基于MoE的LLM的部署带来了系统级挑战. 例如,基于MoE的Google Switch Transformer[238]的参数比FLOPs等效的密集T5模型[239]多出75倍[240]. SwapMoE[241]针对MoE的空间稀疏性和时间局部性,在内存中只保存一部分重要的动态专家,并使用重要性感知调度器维护动态专家与实际专家的映射关系. DeepSpeed-MoE[242]综合使用张量并行、数据并行、专家并行多种并行策略混合,并对系数张量和门控算子进行优化处理. EdgeMoE[243]则采用了一种新的存储策略,将非专家权重存储在设备内存中,而专家权重保存在外部存储器中,仅在被激活时才被加载到内存. 为进一步降低专家I/O交换开销,EdgeMoE实施了专家级位宽自适应技术以压缩权重大小,并预加载预测将被激活的专家到计算I/O管道,从而优化执行流程. 文献[240]结合算法与系统设计提出Pre-gated MoE. 在算法设计上,Pre-gated MoE修改了门控函数的设计. 传统的MoE架构中,第N个MoE块的门控函数选择并激活同块内的专家进行执行. Pre-gated MoE的提前门控函数(Pre-gated function)提前选择第N+1个MoE块的专家,将数据依赖从第N块的专家选择转移到第N+1块的执行上. 在系统设计上,Pre-gated MoE通过提前门控函数提前识别并迁移专家,将CPU到GPU的迁移延迟与专家执行阶段重叠,由此显著减少了迁移对性能的影响. 文献[244]对基于MoE的LLM进行了稀疏性分析,观察到专家内部的前馈神经网络仍然具有稀疏激活的潜力. 通过将MoE技术与基于dReLU的稀疏激活方法相结合,能够在保持性能的同时将模型稀疏度提高到90%,实现2~5倍的推理加速.

5.2.4　分布式计算与端边协同推理

由于单个移动设备的内存和计算限制,分布式计算是当前大模型端侧推理的一种实用解决方案. 该类方法根据各端侧设备的计算能力和内存限制,分解大模型的推理任务,将其划分为更小的部分并分配给各设备执行,从而显著降低单个设备的内存负担. 中国AI学会发表的大模型技术白皮书[245]提出了端间自动混合并行推理. 通过自动感知各个端上部署硬件的存储、带宽、算力等特性,对模型进行自适应切分,将大模型切分到多个端设备进行分布式并行推理,从而实现如百亿、千亿参数模型推理部署. 文献[246]从大语言模型的Token维度进行推理任务的划分. 大语言模型通常由多个堆叠的Transformer块组成,Transformer块中的掩码注意力机制使得输入序列S中的一个Token t_i的隐藏状态的计算只涉及先前的Token $t_0, t_1, \cdots, t_{i-1}$. 文献[246]根据可用设备的规格和传输条件,将序列S切片成子序列$s_0, s_1, \cdots, s_{n-1}$,并在设备间顺序执行. 类似的,文献[247]提出DeTransformer,将具有单个块的原始Transformer层重构为具有多个子块的解耦层,然后通过自适应放置方法在通信能力、计算能力和内存预算之间进行权衡,以自动得出子块在各个端的最佳放置策略,最后对各个端侧子块进行并行协同推理. 另一方面,文献[248]从计算图维度进行分布式推理. 当模型被加载到协调

服务器并转换为计算图后，服务器提取模型子图并将子图编译为可部署的子模块. 然后通过优化模型分配策略、网络通信和运行时负载平衡机制，进一步加速移动设备上的分布式推理.

端边协同推理通过强大的边缘服务器协作来加速密集推理. EdgeFM[249]在边侧通过个性化的知识和架构查询大模型并将其调整为特定的端侧模型，以确保低延迟和接近原始大模型的准确性. NetGPT[250]利用融合的计算和通信资源为端侧部署较小的边缘LLM，为云端部署较大的边缘LLM，最终实现协同计算以提供个性化内容生成服务. 此外，阿里达摩院等联合发布端边云协同平台“洛犀”[251]，致力于推进云侧和边侧泛化大模型和端侧个性小模型协同推理进化.

5.2.5 端到端大语言模型推理框架

大型语言模型计算机辅助设计(Large Language Model Computer-aided design，LLMCad)[252]是首个用于设备上生成式大语言模型的高效推理引擎. LLMCad采用独特的“先生成后验证”的模型协作方式，将大部分Token生成任务交由设备内存中托管的小型实时LLM(内存驻留LLM)完成，而目标LLM则扮演验证者的角色，利用其输出作为真值(ground truth)检查和校正Token生成过程中的错误. 大型语言模型服务(Large Language Model Service，LLMS)[253]是专为移动设备上LLM即服务(LLM as a System service，LLMaaS)优化而设计的框架. 通过充分利用KV-Cache的特性，LLMS提出了容差感知压缩、IO-重计算流水线加载(IO-recompute pipelined loading)以及块生命周期管理三种技术，在严格的内存约束下实现了低开销的LLM上下文切换. PowerInfer[254]利用大模型运行时的稀疏性，即只有热神经元在输入中被持续激活，来解决内存占用较多的问题. PowerInfer预先将热激活的神经元加载到GPU上以实现快速访问，而冷激活的神经元则在CPU上计算，显著减少了GPU内存需求和CPU-GPU间的数据传输. PowerInfer-2 [255]进一步引入分段神经元缓存和神经元簇级流水线技术，使一个神经元簇等待I/O时，另一个已准备好的神经元簇可即时调度到处理器进行计算，有效隐藏了I/O延迟. MobileLLM[256]通过跨Transformer层的权重共享来缩小LLM的规模，在节省参数的同时减少推理过程中的延迟，使大模型能顺利在移动设备上进行推理. 文献[257]利用设备上NPU进行推理卸载，提出了设备上LLM系统llm-NPU. llm-NPU通过算法系统协同设计，在三个层面上重构Token和模型：(1)Token层面，将可变长度的Token划分为多个固定大小的块，同时保持数据依赖性；(2)张量层面，识别并提取显著的异常值，以最小的开销在CPU/GPU上并行运行；(3)块层面，根据硬件亲和性和对准确性的敏感度，以无序方式将Transformer块调度到CPU/GPU和NPU. 基于上述优化，mllm-NPU首次在端侧设备上实现了十亿级模型(Qwen1.5-1.8 B)每秒超过1 000个Token的预填充.

6 推理加速效果对比

为进一步量化现有端智能推理加速技术的效果，我们分别从模型压缩、轻量级网络设计、软-硬件结合推理优化以及端侧大模型推理优化等几个方向中选出一些近几年具有代表性的文献，对文献中使用的模型压缩技术及其推理加速效果进行统计对比. 具体测试数据整理见表5~表8.

数据表格中的Top-1 Accuracy是指在分类任务中，排名第一的类别与实际结果相符的准确率(模型精度)，Top-1 Δaccuracy=压缩后模型精度−原始模型精度；Params代表模型参数量，ΔParams=(原始模型参数量−压缩后模型参数量)/原始模型参数量；Flops代表模型浮点计算次数，ΔFlops=(原始模型浮点计算次数−压缩后模型浮点计算次数)/原始模型浮点计算次数. 加速比=压缩后模型在设备上运行速度/原始模型在设备上运行速度；Latency是指模型设备上推理的延迟；Perplexity代表LLM模型对给定文本的预测能力；Peak Memory Usage是指模型推理期间所占用的最大内存量；Model Size是指模型占用的存储空间大小；Per-token inference latency是指每Token推理的延迟.

6.1 深度学习模型压缩效果对比

表5展示了模型剪枝、参数量化、低秩分解、知识蒸馏这4类模型压缩技术的一些代表工作在ImageNet[258~260]和CIFAR-10数据集[261]上，对ResNet-18/20/32/50/56、MobileNet-V2、WRN-40-2等模型进行压缩的效果.

从表中可以看出，各类压缩技术在减少模型参数和计算量的同时，都能够在一定程度上保持甚至提升模型的准确率. 其中，剪枝技术在去除冗余网络连接的同时，通常能减少50%以上的参数量，而准确率损失较小，甚至在某些情况下还能有所提升；参数量化大幅降低了模型的存储和计算需求，如果使用更先进的量化策略，有望进一步减少精度损失；低秩分解通过矩阵分解等方式有效地压缩了模型参数量；知识蒸馏通过小模型学习大模型的特征表达，压缩了模型规模，性能甚至超过了原始大模型的效果. 尽管这四类技术在理论上实现了显著的压缩效果，然而在实际部署过程中，硬件架构、并行度和内存带宽等因素限制了理论加速比的完全实现，导致实际的加速效果低于理论预期.

6.2 端侧轻量级网络效果对比

表6展示了人工神经网络架构设计与自动神经网络架构搜索的一些代表工作在ImageNet数据集上的效果，包括Top-1 Accuracy、Params以及Flops. 在人工神经

表5　深度学习模型压缩效果对比

优化方式	文献	年份	数据集	模型	Top-1 ΔAccuracy	ΔParams/%	ΔFlops/%	加速比/设备
剪枝	文献[34]	2022	CIFAR-10	ResNet56	−0.23	—	63.80	1.29×/Edge TPU (USB)
	文献[35]	2023	CIFAR-10	ResNet56	+0.24	—	52.61 (MACs)	—
	文献[22]	2023	CIFAR-10	ResNet18	+1.02	98.4	—	—
	文献[23]	2023	CIFAR-10	ResNet32	−0.37	58.65	65.50	—
	文献[37]	2024	CIFAR-10	MobileNet-v2	+0.21	51.70	57.10	—
量化	EMQ[52]	2023	ImageNet	ResNet18	−0.78	85.00	—	—
	AutoMPQ[50]	2024	ImageNet	MobileNet-v2	−1.32	88.04	—	—
低秩分解	FFN[57]	2017	ImageNet	ResNet50	−2.5	94.96	—	—
	ALDS[59]	2021	ImageNet	ResNet18	−0.38	—	64.50	—
	Batude[65]	2022	CIFAR-10	ResNet20	−1.78	85.29	—	—
	文献[68]	2022	ImageNet	ResNet18	0	—	—	1.2×/NVIDIA GeForce GTX 1 080 Ti
知识蒸馏	Fitnets[262]	2014	CIFAR-10	Maxout	+1.43	72.22	—	—
	Moonshine[263]	2018	CIFAR-10	WRN-40-2	−1.13	75.79	—	—

表6　端侧轻量级网络效果对比

优化方式	文献	年份	数据集	模型	Top-1Accuracy/% ↑	Params/M ↓	Flops/M ↓
人工神经网络架构设计	SqueezeNet[73]	2016	ImageNet	SqueezeNet-8bit	57.50	0.66	—
	MobileNet-V1[75]	2017	ImageNet	MobileNet	70.60	4.20	—
	SqueezeNext[74]	2018	ImageNet	2.0-SqNxt-44	69.59	4.40	—
	ShuffleNet-V2[77]	2018	ImageNet	ShuffleNet-v2-0.5x	60.30	—	41
	MobileNet-V2[76]	2018	ImageNet	MobileNetV2(1.4)	74.70	6.90	—
	IGC-V3[81]	2018	ImageNet	IGCV3-D	72.10	2.20	—
	文献[87]	2023	ImageNet	FasterNet-T0	71.90	3.90	340
	MobileOne[88]	2023	ImageNet	MobileOne-S4	79.40	14.80	2 978
	MobileNet-V4[83]	2024	ImageNet	MNv4-Conv-M	79.90	9.20	—
自动神经网络架构搜索	文献[106]	2018	ImageNet	PNASNet-5	74.20	5.10	—
	文献[105]	2019	ImageNet	MdeNAS	74.50	6.10	—
	DMaskingNAS[264]	2022	ImageNet	FBNetV2-F1	68.30	—	56
	文献[116]	2023	ImageNet	ShiftCNN-S	77.20	4.50	240
	文献[118]	2023	ImageNet	EOFGA	75.60	5.70	455

网络架构设计领域,诸如SqueezeNet、MobileNet-V1、SqueezeNext、ShuffleNet-V2、MobileNet-V2等网络模型已经被广泛研究和应用. 这些模型在设计过程中都充分考虑了模型的复杂度和性. 以MobileNet-V2和MobileNet-V4为例,这两种模型在保持较高的Top-1准确率的同时,模型参数量相对较少,这反映出其在模型设计上的优越性和高效性. 然而,这些人工设计的网络需要大量的专业知识和时间投入,这也是它的一个主要限制. 从表6可得,相比于人工设计的架构,NAS通过自动化搜索过程来生成具有更高效的模型,显著减少了人工设计的工作量. 例如,一些自动搜索的架构在参数量和FLOPs相近甚至大幅减少的情况下,仍然能够达到甚至超越手工设计架构的Top-1 Accuracy. 然而,NAS方法通常需要更大的计算资源,这是其主要的挑战.

总的来说,人工神经网络架构设计的优势在于其能够充分利用人类的专业知识和经验进行设计,通常能得到性能优秀的模型,但这需要大量的时间和努力. 相反,NAS的优势在于,它可以自动地搜索到优秀的网络架构,节省了人工设计的时间,但相应地需要较大的计算资源. 这两种策略的选择需要根据具体的应用场景和资源限制进行权衡.

6.3　软-硬件结合推理加速效果对比

表7展示了硬件感知的模型剪枝、硬件友好型稀疏计算、硬件感知的神经网络架构搜索、模型量化与加速

器协同以及低秩分解与张量加速器协同这六种软硬件结合优化方法在端侧设备上部署的实际加速效果. 对于每一篇文献,“方法”一列首行的方法即为文献中使用的对比模型.

从表7可得,相较于仅使用MobileNetV1/V2等算法级优化方法,结合软硬件协同优化的策略在保持较高精度的同时,能够显著减少参数量与计算量,并在实际硬件部署中的延迟表现出明显优势. 例如,硬件感知的剪枝和稀疏化技术不仅能够减少不必要的网络连接与计算操作,还能与硬件加速器的稀疏处理能力充分配合,从而实现更高效的计算利用率. 此外,硬件感知的神经网络架构搜索利用硬件执行性能指导网络结构优化,能够设计出更具硬件适配性和推理速度优势的架构.

尽管这些软硬件协同的优化方法在端侧设备上实现了显著的加速效果,但仍存在优化时需要考虑硬件架构细节的挑战. 不同硬件平台的特性(如存储、带宽、计算单元类型)对实际加速效果有很大的影响. 因此,在设计优化方法时,须充分考虑特定硬件的限制和特点以确保推理效率提升.

表7 算法–硬件结合优化部分方法效果

优化方式	文献	年份	数据集	实验设置	方法	Top-1Accuracy/% ↑	Latency/ms ↓	Flops(-*F*)或Params(-*P*)/M ↓
硬件感知的模型剪枝	Net-Adapt[123]	2018	Imagenet	推理引擎:TFLite 平台:谷歌Pixel 1手机 延迟测量:单个CPU大核	MobileNetV2-0.75	69.800 0(+0)	61.40 (100%)	—
					NetAdapt	70.000 0(+0.2)	53.50 (87%)	—
	文献[126]	2023	CIFAR-10	推理引擎:PyTorch 平台:NVIDIA Jetson Xavier GPU	MobileNetV1	92.590 0(+0)	38.04 (100%)	—
					文献[126]	92.020 0(−0.57)	27.10 (71%)	—
硬件友好型稀疏计算	文献[131]	2023	SUN colonoscopy video[265]	推理引擎:Easics nearbAI 平台:Intel Arria 10 660 SoM(FPGA)	Convolutional lstm[266]	0.663 3(+0)	85.25 (100%)	41.3-*P*(100%)
					文献[131]	0.617 8(−0.045 5)	45.71 (53.6%)	8.20-*P*(19.9%)
硬件感知的神经网络架构搜索	文献[92]	2019	ImageNet	平台:谷歌Pixel 1手机 延迟测量:单个CPU大核	MobileNetV2	72.000 0(+0)	75.00 (100%)	3.40-*P*(100%)
					MnasNet-A1	75.200 0(+3.2)	78.00(104%)	3.90-*P*(115%)
	MobileNet-V3[94]	2019	ImageNet	推理引擎:TFLite 平台:谷歌Pixel 1手机 延迟测量:单个CPU大核	MobileNetV2-1.0	72.000 0(+0)	64.00(100%)	3.40-*P*(100%)
					MobileNetV3-Large-1.0	75.200 0(+3.2)	51.00(79.7%)	5.40-*P*(159%)
	EdgeTran[141]	2023	GLUE[267]	平台:英伟达A100 苹果M1 GPU	Bert-base	79.600 0(+0)	10.57/seq (A100)	110.00-P(100%)
					EdgeTran	80.400 0(+0.8)	8.98/seq (M1 GPU)	39.60-*P*(36%)
神经网络架构搜索与加速器协同	CODE-Bench[147]	2023	ImageNet	平台:Xilinx XC7Z015(FPGA)	MobileNetV2	71.470 0(+0)	75.50(100%)	—
					CODEBench	72.590 0(+1.12)	70.10(92.8%)	—
模型量化与加速器协同	文献[156]	2023	ImageNet	平台:ECE-EMBD开发板 延迟测量:双核ARM Cortex-A9 量化模型:MobileNetV2	Haq[47]	71.470 0(+0)	75.50(100%)	—
					文献[156]	72.590 0(+1.12)	70.10(92.8%)	—
低秩分解与张量加速器协同	HALOC[158]	2023	CIFAR-10	平台:ASIC Eyeriss[125]	MobileNetV2	71.850 0(+0)	—	314.19-*F*(100%)
					HALOC	70.830 0(−0.57)	—	229.13-*F*(72.9%)

6.4 端侧大模型推理优化效果对比

表8和表9展示了低精度量化、KV-Cache优化、混合专家、分布式计算与端边协同推理以及端到端大语言模型推理框架共五类端侧大模型推理优化技术的一些代表工作在WikiText2[268]、C4[239]等数据集上的表现. 其中,在“LLM模型/方法”一列,第一个“/”前为所使用的LLM模型,第一个“/”后为所示一系列优化方法. 例如,对于使用分布式计算与端边协同推理方式进行优化的DeTransformer[247],原始模型(original)的精度为77.54%,利用TP方法优化后每Token推理延迟在单台

设备上约为1.5 s.

由表8可知,低精度量化方法能够极大地降低LLM模型在端侧设备上的内存占用与推理延迟,这主要得益于通过将模型权重和激活值量化为更低的位宽,减少了计算复杂度与存储需求. 然而,量化不可避免地影响模型的预测能力,体现在Perplexity指标的升高,这表明模型在处理文本时的语言理解能力有所下降,特别是在更复杂或细粒度的任务中.

端到端大语言模型推理框架专注于优化整个推理流程,从模型加载、执行到输出的每一个阶段,针对硬件平台进行特定优化. 通过充分利用硬件加速器和高效的内存管理,这些框架能够在不显著牺牲准确率的前提下,达到更好的延迟和资源占用表现.

值得注意的是,相比于其他优化方式,分布式计算与端边协同推理方法能够有效减轻单一设备的负担. 该方式通过将推理过程划分为不同部分,分布在端侧设备和边缘/云端进行协同计算,利用各个计算节点的资源最大化推理效率,显著地缩短了延迟. 然而,网络通信和负载平衡策略对实际的加速效果至关重要,尤其是在低带宽或高延迟的网络环境下,分布式推理的性能可能会受到限制.

总体来看,尽管这些优化技术能够在不同程度上减少LLM推理的资源消耗和延迟,但所部署实现的硬件平台资源仍相对富足. 未来,有望进一步探索如何在更加严格限定的硬件条件下,灵活组合这些优化技术,以获得最佳的推理效率和模型性能平衡.

表8　端侧大模型推理低精度量化方法效果对比

优化方式	量化方法	年份	数据集	测试平台	LLM模型	Perplexity ↓	Peak Memory Usage (-PMU)或Model Size (-MS)/GB ↓	Latency/s ↓
低精度量化	SmoothQuant[223]	2023	WikiText2	NVIDIA A100-80 GB GPU Token length:512	OPT-IML-30B-FP16	14.26(+0)	59.00-PMU(100%)	0.660(100%)
					SmoothQuant-O3	14.37(+1.1)	30.40-PMU(51.5%)	0.458(69.4%)
	SqueezeLLM[269]	2023	C4	NVIDIA A6000 GPU Token length:128	LLaMA-7B-FP16	7.08(+0)	12.70-PMU(100%)	3.200(100%)
					SqueezeLLM-3bit	7.75(+0.67)	2.90-PMU(22.8%)	1.500(46.9%)
	BiLLM[270]	2024	WikiText2	NVIDIA A100 GPU	LLaMA-7B-FP16	5.68(+0)	13.50-MS(100%)	—
					BiLLM	35.04 (+29.36)	1.50-MS(11.1%)	—
	AgileQ[226]	2024	WikiText2	RealmeGT 手机 (骁龙870 SoC)	LLaMA-7B-FP1	5.68(+0)	13.50-MS(100%)	10.600(100%)
					AgileQ-8	6.09(+0.41)	2.53-MS(18.7%)	5.890(55.6%)

7　技术挑战与未来优化展望

尽管端智能系统推理优化技术已经取得一定进展,但随着端侧设备算力的提升和智能化需求的日益增长,这一领域仍面临着前所未有的挑战与机遇. 未来的研究有望在算力资源受限的环境中进一步提升端侧设备的智能处理能力,缩短响应时间,提高能效比. 下面总结当前端侧深度学习模型推理所面临的挑战,并讨论几个值得关注的优化方向.

7.1　挑战

7.1.1　端侧硬件限制与能源效率

尽管端侧设备如智能手机、平板电脑、可穿戴设备、智能家居等的计算能力有了显著提升,但与数据中心强大的云服务器相比,这些设备在处理能力和存储容量上仍显不足. 因此,为了在这些资源受限的设备上部署先进的深度学习模型,特别是大模型,研究人员和工程师仍然需要采取模型压缩技术,包括但不限于网络剪枝、权重量化、知识蒸馏等,以减少模型的参数量和计算复杂度. 然而,这些模型压缩技术往往会牵涉到准确性的折中. 此外,端侧设备大都依赖于电池供电,过多的能量消耗会导致电池寿命缩短,影响用户体验. 因此如何在不牺牲设备续航能力的前提下,在模型规模和复杂度与计算效率之间寻求平衡成为了一个待解的技术难题. 这就需要开发新的能效优化算法和硬件设计,以提升能源利用效率,同时保持必要的计算性能.

7.1.2　端侧模型部署与更新

在端侧深度学习模型的部署领域,开发者面临一个关键的挑战:深度学习框架在不同模型与硬件配置组合下所表现的性能碎片化现象. 这种碎片化是通过在多样化的硬件平台上运行相同模型时所观察到的显著的性能波动来体现的. 更具体地说,即便是在单一的

表9 不同端侧大模型推理优化方法效果对比

优化方式	优化方法	年份	数据集	测试平台	LLM模型/方法	Accuracy/% ↑	Per-token inference latency ↓	其他指标
KV-Cache优化	LLMLingua[229]	2023	GSM8K[271]	Tesla V100 (32 GB)	Alpaca-7B /Selective-Context[272]	53.98(+0)	—	压缩比:1/5倍
					Alpaca-7B /LLMLingua	79.08(+25.1)	—	压缩比:1/5倍
混合专家	Pre-gated MoE[240]	2024	SQuAD[273]	AMD EPYC 7V12 64-Core CPU NVIDIA GPU A100-80G GPU	SwitchTransformer[238] /Base-8expert /MoE-OnDemand	77.40 (Base-8expert) (+0)	14 ms(100%)	—
					SwitchTransformer /Pre-gated MoE	78.20(+0.8)	9 ms(64.3%)	—
分布式计算与端边协同推理	LinguaLinked[248]	2023	Wikitext2	3×谷歌Pixel 7Pro手机 1× CUBOT X30手机	BLOOM3b-int8 /Original	—	2.526 s(100%)	—
					BLOOM3b-int8 /LinguaLinked-5threads	—	0.455 s(18%)	—
	PipeLLM[246]	2023	—	4× NVIDIA Tesla P100 GPUs 2× NVIDIA RTX3090 GPUs	LLaMA-7b/ PipeLLM	—	0.432 s	—
	DeTransformer[247]	2024	SQuAD	4× Raspberry Pi 4B 1× H3C S1850V2交换机	BERT-Base /Original/TP[274]	77.54(Original)(+0)	单设备约1.5 s (TP)(100%)	—
					BERT-Base /DeTransformer	78.37(+0.83)	单设备约0.5 s (33.3%)	—
端到端大语言模型推理框架	LLMCad[252]	2023	Truthfulqa[275]	推理引擎:llama.cpp 平台:小米10	LLaMa2 /Standard pipeline	100.00(+0)	16.2 s(100%)	—
					LLaMa2/LLMCad	100.00(+0)	3.8 s(23.5%)	—
	mllm-NPU[257]	2024	LAMBADA[276] (Accuracy)	小米14	LLaMa2-2-7B-FP16 /PowerInfer-2[255]	—	13.2 ms(100%)	—
			Longbench[277] (Lantency)		LLaMa2-2-7B-FP16 /mllm-NPU	86.30	3.5 ms(26.5%)	—

深度学习框架内,不同硬件上的性能表现可以极端不同. 此外,在固定硬件配置上,不同的模型架构也可能导致效能的显著分化,这种分化超越了简单的非线性性能缩放,而是根源于硬件属性与模型计算需求间的复杂相互作用. 目前并没有一种“一刀切”的深度学习框架能够在所有场景(模型⊗设备)中表现最佳. 此外,为了维持端智能系统的性能,模型的持续更新至关重要,但端侧设备的网络连接性可能较差,如何高效地将更新推送至所有设备,尤其是在不干扰用户体验的情况下,成为了一个技术上和逻辑上的挑战.

7.1.3 隐私与数据安全

在端侧部署深度学习模型时,隐私安全成为一个不容忽视的挑战,这主要体现在两个方面:数据安全和模型安全. 数据安全问题涉及在端侧设备上处理和存储的敏感信息. 由于端侧设备(如智能手机、物联网设备)通常位于用户的物理控制之下,这些设备上存储的数据容易受到未授权访问的风险. 尽管可以通过加密技术来保护数据的传输和存储,但是在模型的训练和推理过程中,数据需要被解密,这就为潜在的数据泄露提供了机会. 此外,即便数据不被直接泄露,通过对模型的输出或者侧信道信息(如电力使用情况、设备发射的电磁信号等)的分析,也可能导致敏感信息的间接泄露. 其次,深度学习模型本身可能成为攻击的目标. 端侧设备上的模型可能遭受到模型逆向工程、模型抽取

攻击或对抗性攻击. 这些攻击可能会揭露模型的内部结构和参数,从而威胁到算法的知识产权,并可能被用来制造对抗样本,即故意设计的输入数据,用以欺骗模型做出错误的决策. 此外,模型更新过程中的不当安全措施可能导致恶意代码的注入,从而使端侧设备成为攻击者的跳板.

7.2 未来展望

7.2.1 稀疏计算与硬件优化

随着端侧设备(如智能手机、物联网设备等)对高效能和低功耗的需求日益增长,稀疏计算作为一种降低计算复杂度和存储需求的技术,其重要性愈加凸显. 通过从模型中抽走部分“不重要”的参数,稀疏计算能够减少模型推理所需的矩阵乘法次数,在保持模型性能的同时,显著减少对算力和内存的需求. 同时,压缩稀疏矩阵还可以减少占用端侧设备宝贵的内存和带宽,在大量端智能系统部署实践中已经证明了其巨大的优化潜力. 然而,由于目前绝大部分端侧硬件是为密集矩阵运算而设计,在实际部署中无法为稀疏模型提供相应的加速支持. 未来,我们应该更多探索与硬件相适配的稀疏化算法,设计更高效的稀疏硬件架构,包括专门的稀疏计算单元,这些单元可以快速识别和处理稀疏矩阵中的非零元素. 此外,研究新的内存技术,如处理器内执行内存中处理(Process-In-Memory,PIM)或近存储计算,将会是之后学术界与工业界研究的重要议题.

7.2.2 端侧通用推理框架设计

在端侧推理框架的设计中,解决性能碎片化问题是提高模型推理效率的关键. 性能碎片化主要源于端侧设备的多样性,包括不同的处理器架构、内存大小和能效比. 为了应对这一挑战,未来的端侧推理框架需要采用一种模块化和可配置的设计,使得框架能够根据不同设备的特性动态调整推理策略. 首先,框架应当内置一个硬件感知模块,能够实时监测和识别端侧设备的计算资源,如处理器类型、运算速度和可用内存. 基于这些信息,框架能够自动选择或调整推理引擎的参数,如线程数目、批量大小和内存分配策略,以最大化设备的计算能力. 其次,为了进一步提升推理性能,框架需要集成先进的模型优化技术. 例如,实现自动化的模型压缩和加速算法,如网络剪枝、模型量化和编译器优化技术,这些技术可以减少模型的参数量和计算操作,降低推理延迟. 同时,框架应支持模型的即时编译和优化,根据当前的硬件环境生成特定的执行代码,以进一步提高运行效率. 此外,为了保证推理框架的广泛适用性和未来兼容性,设计中还应考虑到与新兴硬件的兼容性,如专用AI加速器和异构计算资源. 框架应支持这些新硬件的接口标准,以便能够无缝集成这些资源,从而在不断演进的硬件生态中保持推理性能的领先.

7.2.3 端侧智能系统能效比优化

当前,端智能系统的推理优化工作主要集中在推理加速上,而缺乏对模型推理能效优化的研究. 一些现有工作通过直接在目标硬件上测量能耗以指导模型设计,然而,这种方法得到的模型通常只适配于特定的硬件,对于其他硬件,可能需要重新对模型进行调整,这在一定程度上限制了模型的通用性和适应性. 还有的工作使用能耗估计方法,但在处理涉及内存访问等步骤的能耗估计时,仍需对硬件进行实际测量. 其次,端侧设备常常需要在有限的电池容量和散热能力的约束下,同时进行多个计算密集型的深度学习模型的推理,例如同时运行多个应用程序. 因此,除了高能效比模型的设计,研究如何在运行时调整深度学习模型的推理策略,以适应模型的资源需求与系统可用的运行时资源,也是极具潜力的方向. 例如,根据设备的电源状态和热管理策略为每个深度学习模型动态选择最优的资源精度权衡.

7.2.4 端侧大模型优化

随着深度学习进入大模型时代,将大模型部署到资源受限的端侧设备已成为热点方向. 未来的研究将致力于针对大模型的压缩技术,如知识蒸馏、权重剪枝和量化,以减小模型尺寸和计算需求. 例如,知识蒸馏可以用来训练小型模型以模仿大型模型的行为,而权重剪枝和量化可以直接减少模型的大小和运算需求. 同时,通过软硬件协同设计,可以进一步提高大模型在端侧设备上的运行效率,如开发支持模型分割和并行处理的硬件结构,可以实现大型模型的分布式执行. 此外,设计效率优先的网络架构,如MobileBERT[278]和TinyBERT[279],专为移动设备的计算和存储限制而优化. 其次,大型模型还可以被分割成多个子模块,这些子模块可以根据端侧设备的计算能力和存储容量进行优化部署. 这种分割可以是动态的,以适应不同的运行条件. 边缘计算和设备协同的策略也将成为端侧优化大模型部署的研究方向. 利用边缘计算资源,将大模型的某些计算部分外包到边缘服务器,而将核心推理任务保留在移动设备上. 在减少计算量和存储的同时,保持模型精度.

7.2.5 基于大模型的端智能体

端智能体(edge AI agent)是一种能够感知环境、进行自主理解、进行决策和执行动作的端侧智能实体,近年来广泛应用于个人助手、自动驾驶、智能家居等多领域. 未来,端智能体的发展将依赖于大模型的多模态能力,通过融合视觉、语音、文本等多种数据形式,实现更加深入和精准的感知与理解. 例如,智能手机和物联网

设备将能够同时处理图像、声音和文本信息,在复杂环境中提供更加丰富的用户体验和服务[280]. 同时,检索增强技术[281]将显著提升端智能体的信息获取和处理效率,使其能够快速响应复杂查询和任务,包括实时数据检索、上下文理解和基于大数据的快速推理,从而确保智能体在各种应用场景中表现出色. 在协同决策方面[282,283],端智能体将能够与其他设备和用户高效协作,共同完成复杂任务. 例如,在智能交通系统中,车辆、交通信号和基础设施之间的协同工作将显著提升交通效率和安全性. 总体而言,基于大模型的端智能体将在多模态感知、检索增强、工具学习和协同决策等方面实现突破,推动智能生态系统的全面进化,满足未来更加多样化和复杂的应用需求.

8 结论

本文从算法与软硬件结合优化的视角,对端智能推理加速领域进行了全面而系统的综述. 首先,本文介绍了端侧智能推理加速的背景及其现实意义,阐述了在资源受限的端侧设备上进行高效推理的需求与挑战. 其次,本文系统性地回顾了端侧推理算法优化的经典策略及最新研究进展,包括模型压缩和结构设计等核心方法. 再次,本文详细介绍了端侧推理常用的硬件加速器. 进而,本文聚焦于算法与软硬件协同优化的核心议题,围绕三个关键方面展开讨论,包括算法-硬件结合优化策略,软件优化层面中的算法与硬件考量,以及算法-硬件协同调度策略的优化. 然后,在此基础上,本文全面梳理了当前端侧大模型部署与优化的技术路径及前沿实践. 为更直观地展示现有端侧智能推理加速技术的效果,本文选取了近年来一些具有代表性的文献,对其在推理性能和效率方面的加速效果进行量化和对比分析. 最后,本文探讨了端智能推理加速的技术挑战与未来优化方向. 希望本文能够为端智能推理加速领域的研究者提供一个更为全面的优化视角,推动该领域的持续创新与研究发展.

参考文献

[1] 廉筱峪. 复杂噪声环境下基于轻量化模型的车内交互语音增强和识别方法[J]. 电子学报, 2024, 52(4): 1282-1287.
LIAN X Y. In-vehicle interactive speech enhancement and recognition method based on lightweight model in complex noise environments[J]. Acta Electronica Sinica, 2024, 52(4): 1282-1287. (in Chinese)

[2] 周治国, 马文浩. 一种多层多模态融合3D目标检测方法[J]. 电子学报, 2024, 52(3): 696-708.
ZHOU Z G, MA W H. 3D object detection based on multi-layer multimodal fusion[J]. Acta Electronica Sinica, 2024, 52(3): 696-708. (in Chinese)

[3] 惠康华, 闫建青, 高思华, 等. 基于特征融合的轻量级新残差人脸识别方法[J]. 电子学报, 2024, 52(3): 937-944.
HUI K H, YAN J Q, GAO S H, et al. Lightweight new fesidual face recognition method based on feature fusion[J]. Acta Electronica Sinica, 2024, 52(3): 937-944. (in Chinese)

[4] BROWN T, MANN B, RYDER N, et al. Language models are few-shot learners[J]. Advances in Neural Information Processing Systems, 2020, 33: 1877-1901.

[5] ZHANG Q Y, LI X, CHE X Y, et al. A comprehensive benchmark of deep learning libraries on mobile devices[C]// Proceedings of the ACM Web Conference 2022. New York: ACM, 2022: 3298-3307.

[6] ZHOU Z, CHEN X, LI E, et al. Edge intelligence: Paving the last mile of artificial intelligence with edge computing[J]. Proceedings of the IEEE, 2019, 107(8): 1738-1762.

[7] WANG X F, HAN Y W, LEUNG V C M, et al. Convergence of edge computing and deep learning: A comprehensive survey[J]. IEEE Communications Surveys & Tutorials, 2020, 22(2): 869-904.

[8] DENG S G, ZHAO H L, FANG W J, et al. Edge intelligence: The confluence of edge computing and artificial intelligence[J]. IEEE Internet of Things Journal, 2020, 7(8): 7457-7469.

[9] SARWAR MURSHED M G, MURPHY C, HOU D Q, et al. Machine learning at the network edge: A survey[J]. ACM Computing Surveys, 2022, 54(8): 1-37.

[10] DHAR S, GUO J Y, LIU J J, et al. A survey of on-device machine learning[J]. ACM Transactions on Internet of Things, 2021, 2(3): 1-49.

[11] 高晗, 田育龙, 许封元, 等. 深度学习模型压缩与加速综述[J]. 软件学报, 2021, 32(1): 68-92.
GAO H, TIAN Y L, XU F Y, et al. Survey of deep learning model compression and acceleration[J]. Journal of Software, 2021, 32(1): 68-92. (in Chinese)

[12] CAI H, LIN J, LIN Y J, et al. Enable deep learning on mobile devices: Methods, systems, and applications[J]. ACM Transactions on Design Automation of Electronic Systems, 2022, 27(3): 1-50.

[13] SHUVO M M H, ISLAM S K, CHENG J L, et al. Efficient acceleration of deep learning inference on resource-constrained edge devices: A review[J]. Proceedings of the IEEE, 2023, 111(1): 42-91.

[14] ZHAO T M, XIE Y C, WANG Y, et al. A survey of deep learning on mobile devices: Applications, optimizations, challenges, and research opportunities[J]. Proceedings of

the IEEE, 2022, 110(3): 334-354.

[15] HUA H C, LI Y T, WANG T H, et al. Edge computing with artificial intelligence: A machine learning perspective[J]. ACM Computing Surveys, 2023, 55(9): 1-35.

[16] LECUN Y, DENKER J, SOLLA S. Optimal brain damage[J]. Advances in Neural Information Processing Systems, 1989, 2: 1-1990.

[17] HASSIBI B, STORK D. Second order derivatives for network pruning: Optimal brain surgeon[J]. Advances in Neural Information Processing Systems, 1992, 5: 1.

[18] DONG X, CHEN S Y, PAN S J. Learning to prune deep neural networks via layer-wise optimal brain surgeon[EB/OL]. (2017-11-09)[2024-07-22]. https://arxiv.org/abs/1705.07565v2.

[19] SINGH S P, ALISTARH D. WoodFisher: Efficient second-order approximation for neural network compression [EB/OL]. (2020-01-04)[2024-07-22]. https://arxiv.org/abs/2004.14340v5.

[20] YU X, SERRA T, RAMALINGAM S, et al. The combinatorial brain surgeon: Pruning weights that cancel one another in neural networks[C]//International Conference on Machine Learning. New York: PMLR, 2022: 25668-25683.

[21] 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: 1.

[22] LIAO Z, QUÉTU V, NGUYEN V T, et al. Can unstructured pruning reduce the depth in deep neural networks[C]// 2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW). Piscataway: IEEE, 2023: 1394-1398.

[23] XU K X, WANG Z, GENG X, et al. Efficient joint optimization of layer-adaptive weight pruning in deep neural networks[C]//2023 IEEE/CVF International Conference on Computer Vision (ICCV). Piscataway: IEEE, 2023: 17401-17411.

[24] SHARIFY S, LASCORZ A D, MAHMOUD M, et al. Laconic deep learning inference acceleration[C]//Proceedings of the 46th International Symposium on Computer Architecture. New York: ACM, 2019: 304-317.

[25] ZHANG K, LIU G Z. Layer pruning for obtaining shallower ResNets[J]. IEEE Signal Processing Letters, 2022, 29: 1172-1176.

[26] XIA M Z, ZHONG Z X, CHEN D Q. Structured pruning learns compact and accurate models[EB/OL]. (2022-05-02)[2024-07-22]. https://arxiv.org/abs/2204.00408.

[27] YU L, XIANG W. X-pruner: Explainable pruning for vision transformers[C]//2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2023: 24355-24363.

[28] HE Z Q, QIAN Y G, WANG Y Q, et al. Filter pruning via feature discrimination in deep neural networks[M]// Computer Vision - ECCV 2022. Cham: Springer Nature Switzerland, 2022: 245-261.

[29] MONDAL M, DAS B, ROY S D, et al. Adaptive CNN filter pruning using global importance metric[J]. Computer Vision and Image Understanding, 2022, 222: 103511.

[30] LIU Y J, FAN K F, ZHOU W J. FPWT: Filter pruning via wavelet transform for CNNs[J]. Neural Networks, 2024, 179: 106577.

[31] YANG W, XIAO Y C. Structured pruning via feature channels similarity and mutual learning for convolutional neural network compression[J]. Applied Intelligence, 2022, 52(12): 14560-14570.

[32] ELKERDAWY S, ELHOUSHI M, SINGH A, et al. To filter prune, or to layer prune, that is the question[C]// Computer Vision - ACCV 2020. Cham: Springer International Publishing, 2021, 1: 737-753.

[33] WANG Z, LI C C, WANG X Y. Convolutional neural network pruning with structural redundancy reduction[C]//2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2021: 14913-14922.

[34] LI G L, MA X, WANG X Y, et al. Optimizing deep neural networks on intelligent edge accelerators via flexible-rate filter pruning[J]. Journal of Systems Architecture, 2022, 124: 102431.

[35] FANG G F, MA X Y, SONG M L, et al. DepGraph: Towards any structural pruning[C]//2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2023: 16091-16101.

[36] PASZKE A, GROSS S, MASSA F, et al. PyTorch: An imperative style, high-performance deep learning library [EB/OL]. (2019-12-03)[2024-07-22]. https://arxiv.org/abs/1912.01703v1.

[37] WANG X, RACHWAN J, GÜNNEMANN S, et al. Structurally prune anything: Any architecture, any framework, any time[EB/OL]. (2024-03-03) [2024-07-22]. https://arxiv.org/abs/2403.18955v1.

[38] GUPTA S, AGRAWAL A, GOPALAKRISHNAN K, et al. Deep learning with limited numerical precision[C]//International conference on machine learning. New York: PMLR, 2015: 1737-1746.

[39] KÖSTER U, WEBB T, WANG X, et al. Flexpoint: An adaptive numerical format for efficient training of deep

neural networks[J]. Advances in Neural Information Processing Systems, 2017, 30: 1.

[40] JACOB B, KLIGYS S, CHEN B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 2704-2713.

[41] COURBARIAUX M, BENGIO Y, DAVID J P. BinaryConnect: Training deep neural networks with binary weights during propagations[EB/OL]. (2016-04-18)[2024-07-22]. https://arxiv.org/abs/1511.00363v3.

[42] RASTEGARI M, ORDONEZ V, REDMON J, et al. XNOR-Net: ImageNet classification using binary convolutional neural networks[M]//Computer Vision-ECCV 2016. Cham: Springer International Publishing, 2016: 525-542.

[43] JUEFEI-XU F, BODDETI V N, SAVVIDES M. Local binary convolutional neural networks[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2017: 4284-4293.

[44] KIM H B, LEE J H, YOO S, et al. MetaMix: Meta-state precision searcher for mixed-precision activation quantization[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2024, 38(12): 13132-13141.

[45] MICIKEVICIUS P, NARANG S R, ALBEN J, et al. Mixed precision training[EB/OL]. (2018-02-15)[2024-07-22]. https://arxiv.org/abs/1710.03740v3.

[46] WU B C, WANG Y H, ZHANG P Z, et al. Mixed precision quantization of ConvNets via differentiable neural architecture search[EB/OL]. (2018-11-30)[2024-07-22]. https://arxiv.org/abs/1812.00090v1.

[47] WANG K, LIU Z J, LIN Y J, et al. HAQ: Hardware-aware automated quantization with mixed precision[C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2019: 8604-8612.

[48] HU Y M, WANG X G, LI L J, et al. Improving one-shot NAS with shrinking-and-expanding supernet[J]. Pattern Recognition, 2021, 118: 108025.

[49] JIN Q, YANG L J, LIAO Z Y. AdaBits: Neural network quantization with adaptive bit-widths[C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2020: 2146-2156.

[50] XU K, SHAO X Y, TIAN Y, et al. AutoMPQ: Automatic mixed-precision neural network search via few-shot quantization adapter[J/OL]. IEEE Transactions on Emerging Topics in Computational Intelligence. (2024-05-08)[2024-07-22]. https://ieeexplore.ieee.org/document/10523945.

[51] WANG T Z, WANG K, CAI H, et al. APQ: Joint search for network architecture, pruning and quantization policy[C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2020: 2075-2084.

[52] DONG P J, LI L J, WEI Z M, et al. EMQ: Evolving training-free proxies for automated mixed precision quantization[C]//2023 IEEE/CVF International Conference on Computer Vision (ICCV). Piscataway: IEEE, 2023: 17030-17040.

[53] KORYAKOVSKIY I, YAKOVLEVA A, BUCHNEV V, et al. One-shot model for mixed-precision quantization[C]//2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2023: 7939-7949.

[54] SUN Z, GE C, WANG J, et al. Entropy-driven mixed-precision quantization for deep network design[J]. Advances in Neural Information Processing Systems, 2022, 35: 21508-21520.

[55] JADERBERG M, VEDALDI A, ZISSERMAN A. Speeding up convolutional neural networks with low rank expansions[EB/OL]. (2014-05-15)[2024-07-22]. https://arxiv.org/abs/1405.3866v1.

[56] LIU B Y, WANG M, FOROOSH H, et al. Sparse convolutional neural networks[C]//2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2015: 806-814.

[57] WANG P S, CHENG J. Fixed-point factorized networks[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2017: 3966-3974.

[58] PENG B, TAN W M, LI Z Y, et al. Extreme network compression via filter group approximation[M]//Computer Vision - ECCV 2018. Cham: Springer International Publishing, 2018: 307-323.

[59] LIEBENWEIN L, MAALOUF A, GAL O, et al. Compressing neural networks: Towards determining the optimal layer-wise decomposition[EB/OL]. (2021-11-18)[2024-07-22]. https://arxiv.org/abs/2107.11442v2.

[60] HITCHCOCK F L. The expression of a tensor or a polyadic as a sum of products[J]. Journal of Mathematics and Physics, 1927, 6(1): 164-189.

[61] HITCHCOCK F L. Multiple invariants and generalized rank of a P-way matrix or tensor[J]. Journal of Mathematics and Physics, 1928, 7(1/2/3/4): 39-79.

[62] KIERS H A L. Towards a standardized notation and terminology in multiway analysis[J]. Journal of Chemometrics, 2000, 14(3): 105-122.

[63] TUCKER L R. Some mathematical notes on three-mode factor analysis[J]. Psychometrika, 1966, 31(3): 279-311.

[64] KIM Y D, PARK E, YOO S, et al. Compression of deep

convolutional neural networks for fast and low power mobile applications[EB/OL]. (2016-02-24) [2024-07-22]. https://arxiv.org/abs/1511.06530v2.

[65] YIN M, PHAN H, ZANG X, et al. BATUDE: Budget-aware neural network compression based on tucker decomposition[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2022, 36(8): 8874-8882.

[66] DENTON E, ZAREMBA W, BRUNA J, et al. Exploiting linear structure within convolutional networks for efficient evaluation[J]. Advances in Neural Information Processing Systems, 2014, 2(January): 1269-1277.

[67] XUE J, LI J Y, GONG Y F. Restructuring of deep neural network acoustic models with singular value decomposition[C]//Interspeech 2013. Singapore: ISCA, 2013: 2365-2369.

[68] ZHANG H N, LIU L J, ZHOU H Y, et al. CMD: Controllable matrix decomposition with global optimization for deep neural network compression[J]. Machine Learning, 2022, 111(3): 831-851.

[69] HINTON G, VINYALS O, DEAN J. Distilling the knowledge in a neural network[EB/OL]. (2015-03-09)[2024-07-22]. https://arxiv.org/abs/1503.02531v1.

[70] LI Z, LI X, YANG L F, et al. Curriculum temperature for knowledge distillation[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2023, 37(2): 1504-1512.

[71] SUN S Q, REN W Q, LI J Z, et al. Logit standardization in knowledge distillation[C]//2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2024: 15731-15740.

[72] HAO Z W, GUO J Y, HAN K, et al. One-for-all: Bridge the gap between heterogeneous architectures in knowledge distillation[EB/OL]. (2023-10-30)[2024-07-22]. https://arxiv.org/abs/2310.19444v1.

[73] IANDOLA F N, HAN S, MOSKEWICZ M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5MB model size[EB/OL]. (2016-11-04)[2024-07-22]. https://arxiv.org/abs/1602.07360v4.

[74] GHOLAMI A, KWON K, WU B C, et al. SqueezeNext: Hardware-aware neural network design[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Piscataway: IEEE, 2018: 1751-1760.

[75] HOWARD A G, ZHU M L, CHEN B, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications[EB/OL]. (2017-04-17) [2024-07-22]. https://arxiv.org/abs/1704.04861v1.

[76] SANDLER M, HOWARD A, ZHU M L, et al. MobileNetV2: Inverted residuals and linear bottlenecks[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 4510-4520.

[77] ZHANG X Y, ZHOU X Y, LIN M X, et al. ShuffleNet: An extremely efficient convolutional neural network for mobile devices[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 6848-6856.

[78] MA N N, ZHANG X Y, ZHENG H T, et al. ShuffleNet V2: Practical guidelines for efficient CNN architecture design[C]//Computer Vision-ECCV 2018. Cham: Springer International Publishing, 2018: 122-138.

[79] ZHANG T, QI G J, XIAO B, et al. Interleaved group convolutions[C]//2017 IEEE International Conference on Computer Vision (ICCV). Piscataway: IEEE, 2017: 4383-4392.

[80] XIE G T, WANG J D, ZHANG T, et al. Interleaved structured sparse convolutional neural networks[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 8847-8856.

[81] SUN K, LI M J, LIU D, et al. IGCV3: Interleaved low-rank group convolutions for efficient deep neural networks[EB/OL]. (2018-07-20)[2024-07-22]. https://arxiv.org/abs/1806.00178v2.

[82] HAN K, WANG Y H, TIAN Q, et al. GhostNet: More features from cheap operations[C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2020: 1577-1586.

[83] QIN D F, LEICHNER C, DELAKIS M, et al. MobileNetV4: Universal models for the mobile ecosystem[EB/OL]. (2024-09-29)[2025-05-06]. https://arxiv.org/abs/2404.10518v2.

[84] BENDER G, LIU H X, CHEN B, et al. Can weight sharing outperform random architecture search? An investigation with TuNAS[C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2020: 14323-14332.

[85] TAN M, LE Q. Efficientnet: Rethinking model scaling for convolutional neural networks[C]//International Conference on Machine Learning. New York: PMLR, 2019: 6105-6114.

[86] TAN M X, PANG R M, LE Q V. EfficientDet: Scalable and efficient object detection[C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2020: 10781-10790.

[87] CHEN J R, KAO S H, HE H, et al. Run, don't walk: Chasing higher FLOPS for faster neural networks[C]//2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2023: 12021-12031.

[88] VASU P K A, GABRIEL J, ZHU J, et al. MobileOne: An

improved one millisecond mobile backbone[C]//2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2023: 7907-7917.

[89] WANG A, CHEN H, LIN Z J, et al. Rep ViT: Revisiting mobile CNN from ViT perspective[C]//2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2024: 15909-15920.

[90] ZOPH B, LE Q V, MATHUR V, et al. Neural architecture search with reinforcement learning[EB/OL]. (2017-02-15)[2024-07-22]. https://arxiv.org/abs/1611.01578v2.

[91] ZOPH B, VASUDEVAN V, SHLENS J, et al. Learning transferable architectures for scalable image recognition [C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 8697-8710.

[92] TAN M X, CHEN B, PANG R M, et al. MnasNet: Platform-aware neural architecture search for mobile[C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2019: 2820-2828.

[93] CAI H, ZHU L, HAN S. Proxylessnas: Direct neural architecture search on target task and hardware[EB/OL]. (2019-02-23)[2024-07-22]. https://arxiv.org/abs/1812.00332.

[94] HOWARD A, SANDLER M, CHEN B, et al. Searching for MobileNetV3[C]//2019 IEEE/CVF International Conference on Computer Vision (ICCV). Piscataway: IEEE, 2019: 1314-1324.

[95] WU B C, KEUTZER K, DAI X L, et al. FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search[C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2019: 10734-10742.

[96] WU B C, WAN A, YUE X Y, et al. Shift: A zero FLOP, zero parameter alternative to spatial convolutions[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 9127-9135.

[97] STAMOULIS D, DING R Z, WANG D, et al. Single-path NAS: Designing hardware-efficient ConvNets in less than 4 hours[M]//Machine Learning and Knowledge Discovery in Databases. Cham: Springer International Publishing, 2020: 481-497.

[98] ANGELINE P J, SAUNDERS G M, POLLACK J B. An evolutionary algorithm that constructs recurrent neural networks[J]. IEEE Transactions on Neural Networks, 1994, 5(1): 54-65.

[99] STANLEY K O, MIIKKULAINEN R. Efficient evolution of neural network topologies[C]//Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02). Piscataway: IEEE, 2002: 1757-1762.

[100] FLOREANO D, DÜRR P, MATTIUSSI C. Neuroevolution: From architectures to learning[J]. Evolutionary Intelligence, 2008, 1(1): 47-62.

[101] JÓZEFOWICZ R, ZAREMBA W, SUTSKEVER I. An empirical exploration of recurrent network architectures[C]//International Conference on Machine Learning. New York: PMLR, 2015: 2342-2350.

[102] MIIKKULAINEN R, LIANG J, MEYERSON E, et al. Evolving deep neural networks[M]//Artificial Intelligence in the Age of Neural Networks and Brain Computing. Pittsburg: Academic Press, 2019: 293-312.

[103] LIU H X, SIMONYAN K, VINYALS O, et al. Hierarchical representations for efficient architecture search[EB/OL]. (2018-02-22)[2024-07-22]. https://arxiv.org/abs/1711.00436v2.

[104] LIU H X, SIMONYAN K, YANG Y M, et al. DARTS: Differentiable architecture search[EB/OL]. (2019-04-23)[2024-07-22]. https://arxiv.org/abs/1806.09055v2.

[105] ZHENG X W, JI R R, TANG L, et al. Multinomial distribution learning for effective neural architecture search[C]//2019 IEEE/CVF International Conference on Computer Vision (ICCV). Piscataway: IEEE, 2019: 1304-1313.

[106] LIU C X, ZOPH B, NEUMANN M, et al. Progressive neural architecture search[C]//Computer Vision - ECCV 2018. Cham: Springer International Publishing, 2018: 19-35.

[107] REAL E, AGGARWAL A, HUANG Y P, et al. Regularized evolution for image classifier architecture search[EB/OL]. (2019-02-16)[2024-07-22]. https://arxiv.org/abs/1802.01548v7.

[108] LI L S, JAMIESON K G, DESALVO G, et al. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization[C]//Computer Science. Washington DC: ICLR, 2017: 53.

[109] ZELA A, KLEIN A, FALKNER S, et al. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search[EB/OL]. (2018-07-18)[2024-07-22]. https://arxiv.org/abs/1807.06906v1.

[110] CHRABĄSZCZ P, LOSHCHILOV I, HUTTER F. Back to basics: Benchmarking canonical evolution strategies for playing atari[C]//Proceedings of the 27th International Joint Conference on Artificial Intelligence. Freiburg: International Joint Conferences on Artificial Intelligence Organization, 2018: 1419-1426.

[111] RUNGE F, STOLL D, FALKNER S, et al. Learning to design RNA[EB/OL]. (2019-04-12)[2024-07-22]. https://arxiv.org/abs/1812.11951v2.

[112] SWERSKY K, SNOEK J, ADAMS R P. Freeze-thaw

Bayesian optimization[EB/OL]. (2014-06-16) [2024-07-22]. https://arxiv.org/abs/1406.3896v1.

[113] DOMHAN T, SPRINGENBERG J T, HUTTER F. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves[J]. IJCAI International Joint Conference on Artificial Intelligence, 2015, 1: 3460-3468.

[114] KLEIN A, FALKNER S, SPRINGENBERG J T, et al. Learning curve prediction with bayesian neural networks[J]. International Conference on Learning Representations, 2017, 1: 1.

[115] BAKER B, GUPTA O, RASKAR R, et al. Accelerating neural architecture search using performance prediction[EB/OL]. (2017-11-08) [2024-07-22]. https://arxiv.org/abs/1705.10823v2.

[116] ZHANG M Y, YU X Y, ZHAO H D, et al. ShiftNAS: Improving one-shot NAS via probability shift[C]//2023 IEEE/CVF International Conference on Computer Vision (ICCV). Piscataway: IEEE, 2023: 5896-5905.

[117] WANG H B, GE C, CHEN H S, et al. PreNAS: Preferred one-shot learning towards efficient neural architecture search[EB/OL]. (2023-06-16)[2024-07-22]. https://arxiv.org/abs/2304.14636v3.

[118] YUAN G L, XUE B, ZHANG M J. An effective one-shot neural architecture search method with supernet fine-tuning for image classification[C]//Proceedings of the Genetic and Evolutionary Computation Conference. New York: ACM, 2023: 615-623.

[119] ZHENG H, LIU K H, FEDOROV I, et al. SiGeo: Sub-one-shot NAS via geometry of loss landscape[C]//Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. New York: ACM, 2024: 4536-4547.

[120] SESHADRI K, AKIN B, LAUDON J, et al. An evaluation of edge TPU accelerators for convolutional neural networks[C]//IEEE International Symposium on Workload Characterization (IISWC). Piscataway: IEEE, 2022: 79-91.

[121] LIN Y, HAFDI D, WANG K, et al. Neural-hardware architecture search[J]. NeurIPS WS, 2019, 1: 1-36.

[122] SHEN M Y, YIN H X, MOLCHANOV P, et al. HALP: Hardware-aware latency pruning[EB/OL]. (2021-10-20) [2024-07-22]. https://arxiv.org/abs/2110.10811v1.

[123] YANG T J, HOWARD A, CHEN B, et al. NetAdapt: Platform-aware neural network adaptation for mobile applications[C]//Computer Vision - ECCV 2018. Cham: Springer International Publishing, 2018: 289-304.

[124] LI X, ZHOU Y M, PAN Z, et al. Partial order pruning: For best speed/accuracy trade-off in neural architecture search[C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2019: 9145-9153.42.

[125] CHEN Y H, EMER J, SZE V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks[C]//2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). Piscataway: IEEE, 2016: 367-379.

[126] ZHANG Y H, JIANG H X, ZHU Y T, et al. LOCP: Latency-optimized channel pruning for CNN inference acceleration on GPUs[J]. The Journal of Supercomputing, 2023, 79(13): 14313-14341.

[127] YANG T J, CHEN Y H, SZE V. Designing energy-efficient convolutional neural networks using energy-aware pruning[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2017: 6071-6079.

[128] HAN S, MAO H Z, DALLY W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[EB/OL]. (2016-02-15) [2024-07-22]. https://arxiv.org/abs/1510.00149v5.

[129] YU J C, LUKEFAHR A, PALFRAMAN D, et al. Scalpel: Customizing DNN pruning to the underlying hardware parallelism[C]//2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA). Piscataway: IEEE, 2017: 548-560.

[130] SUN M S, ZHAO P, GUNGOR M, et al. 3D CNN acceleration on FPGA using hardware-aware pruning[C]//Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC). Piscataway: IEEE, 2020: 1-6.

[131] PLOCHAET J, GOEDEMÉ T. Hardware-aware pruning for FPGA deep learning accelerators[C]//2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Piscataway: IEEE, 2023: 4482-4490.

[132] SUI X F, LV Q B, ZHI L J, et al. A hardware-friendly high-precision CNN pruning method and its FPGA implementation[J]. Sensors, 2023, 23(2): 824.

[133] GUO C, HSUEH B Y, LENG J W, et al. Accelerating sparse DNN models without hardware-support via tile-wise sparsity[C]//SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. Piscataway: IEEE, 2020: 1-15.

[134] CHEN Y H, KRISHNA T, EMER J S, et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep con-

volutional neural networks[J]. IEEE Journal of Solid-State Circuits, 2017, 52(1): 127-138.

[135] 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.

[136] CAVIGELLI L, BENINI L. Origami: A 803-GOp/s/W convolutional network accelerator[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2017, 27(11): 2461-2475.

[137] WEI X C, LIANG Y, CONG J. Overcoming data transfer bottlenecks in FPGA-based DNN accelerators via layer conscious memory management[C]//Proceedings of the 56th ACM/IEEE Design Automation Conference (DAC). Piscataway: IEEE, 2019: 1-6.

[138] 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 (ISCA). Piscataway: IEEE, 2017: 27-40.

[139] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal policy optimization algorithms[EB/OL]. (2017-08-28)[2024-07-22]. https://arxiv.org/abs/1707.06347v2.

[140] DAI X L, JIA Y Q, VAJDA P, et al. ChamNet: Towards efficient network design through platform-aware model adaptation[C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2019: 11398-11407.

[141] TULI S, JHA N K. EdgeTran: Device-aware co-search of transformers for efficient inference on mobile edge platforms[J]. IEEE Transactions on Mobile Computing, 2024, 23(6): 7012-7029.

[142] ABDELFATTAH M S, DUDZIAK L, CHAU T, et al. Best of both worlds: AutoML codesign of a CNN and its hardware accelerator[C]//Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC). Piscataway: IEEE, 2020: 1-6.

[143] YANG L, YAN Z Y, LI M, et al. Co-exploration of neural architectures and heterogeneous ASIC accelerator designs targeting multiple tasks[C]//Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC). Piscataway: IEEE, 2020: 1-6.

[144] LI Y H, HAO C, ZHANG X F, et al. EDD: Efficient differentiable DNN architecture and implementation co-search for embedded AI solutions[C]//Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC). Piscataway: IEEE, 2020: 1-6.

[145] FU Y G, ZHANG Y A, ZHANG Y, et al. Auto-NBA: Efficient and effective search over the joint space of networks, bitwidths, and accelerators[EB/OL]. (2025-01-04)[2024-07-22]. https://arxiv.org/abs/2106.06575v3.

[146] SEKANINA L. Neural architecture search and hardware accelerator co-search: A survey[J]. IEEE Access, 2021, 9: 151337-151362.

[147] TULI S, LI C H, SHARMA R, et al. CODEBench: A neural architecture and hardware accelerator co-design framework[J]. ACM Transactions on Embedded Computing Systems, 2023, 22(3): 1-30.

[148] HONG C, HUANG Q J, DINH G, et al. DOSA: Differentiable model-based one-loop search for DNN accelerators[C]//Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture. New York: ACM, 2023: 209-224.

[149] SAKHUJA C, SHI Z, LIN C. Leveraging domain information for the efficient automated design of deep learning accelerators[C]//2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA). Piscataway: IEEE, 2023: 287-301.

[150] LIN Y J, YANG M T, HAN S. NAAS: Neural accelerator architecture search[C]//2021 58th ACM/IEEE Design Automation Conference (DAC). Piscataway: IEEE, 2021: 1051-1056.

[151] DONG Z, YAO Z W, CAI Y H, et al. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks [EB/OL]. (2019-11-10) [2024-07-22]. https://arxiv. org/abs/1911.03852v1.

[152] YAO Z, DONG Z, ZHENG Z, et al. Hawq-v3: Dyadic neural network quantization[C]//International Conference on Machine Learning. New York: PMLR, 2021: 11875-11886.

[153] REN A, ZHANG T Y, YE S K, et al. ADMM-NN: An algorithm-hardware co-design framework of DNNs using alternating direction methods of multipliers[C]//Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2019: 925-938.

[154] BALASKAS K, KARATZAS A, SAD C, et al. Hardware-aware DNN compression via diverse pruning and mixed-precision quantization[J]. IEEE Transactions on Emerging Topics in Computing, 2024, 12(4): 1079-1092.

[155] SONG Z R, FU B Q, WU F Y, et al. DRQ: Dynamic Region-based quantization for deep neural network acceler-

ation[C]//2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). Piscataway: IEEE, 2020: 1010-1021.

[156] HUANG W, QIN H T, LIU Y D, et al. On-chip hardware-aware quantization for mixed precision neural networks [EB/OL]. (2024-05-23) [2024-07-22]. https://arxiv.org/abs/2309.01945v5.

[157] DENG C H, SUN F X, QIAN X H, et al. TIE: Energy-efficient tensor train-based inference engine for deep neural network[C]//2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA). Piscataway: IEEE, 2019: 264-277.

[158] XIAO J Q, ZHANG C M, GONG Y, et al. HALOC: Hardware-aware automatic low-rank compression for compact neural networks[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2023, 37(9): 10464-10472.

[159] LI S, HANSON E, LI H, et al. PENNI: Pruned kernel sharing for efficient CNN inference[C]//International Conference on Machine Learning. New York: PMLR, 2020: 5863-5873.

[160] YU Z W, BOUGANIS C S. StreamSVD: Low-rank approximation and streaming accelerator co-design[C]// 2021 International Conference on Field-Programmable Technology (ICFPT). Piscataway: IEEE, 2021: 1-9.

[161] TensorFlow lite[EB/OL]. (2017-11-15)[2024-07-22]. https://www.tensorflow.org/lite/guide?hl=zh-cn.

[162] Tencent, ncnn[CP/OL]. (2022-11-13)[2024-07-22]. https://github.com/Tencent/ncnn.

[163] JIANG X T, WANG H, CHEN Y L, et al. MNN: A universal and efficient inference engine[J]. Proceedings of Machine Learning and Systems, 2020, 2: 1-13.

[164] Xiaomi. XiaoMi mace[CP/OL]. (2022-11-12) [2024-07-22]. https://github.com/XiaoMi/mace.

[165] Software Arm. Arm NN[CP/OL]. (2022-11-11) [2024-07-22]. https://github.com/ARM-software/armnn.

[166] PaddlePaddle. Paddlelite[CP/OL]. (2022-11-12)[2024-07-22]. https://github.com/PaddlePaddle/Paddle-Lite.

[167] PyTorch mobile[EB/OL]. (2021-06-15)[2024-07-22]. https://pytorch.org/mobile/home.

[168] LI M Z, LIU Y, LIU X Y, et al. The deep learning compiler: A comprehensive survey[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(3): 708-727.

[169] CHEN T Q, MOREAU T, JIANG Z H, et al. TVM: An automated end-to-end optimizing compiler for deep learning[EB/OL]. (2018-10-05)[2024-07-22]. https://arxiv.org/abs/1802.04799v3.

[170] ROTEM N, FIX J, ABDULRASOOL S, et al. Glow: Graph lowering compiler techniques for neural networks[EB/OL]. (2019-04-03)[2024-07-22]. https://arxiv.org/abs/1805.00907v3.

[171] CYPHERS S, BANSAL A K, BHIWANDIWALLA A, et al. Intel nGraph: An intermediate representation, compiler, and executor for deep learning[EB/OL]. (2018-01-30)[2024-07-22]. https://arxiv.org/abs/1801.08058v2.

[172] Xla-tensorflow, compiled[EB/OL]. (2017-03-01) [2024-07-22]. https://developers.googleblog.com/2017/03/xla-tensorflow-compiled.html.

[173] KIM T, KWON Y, LEE J, et al. CPrune: Compiler-informed model pruning for efficient target-aware DNN execution[M]//Computer Vision - ECCV 2022. Cham: Springer Nature Switzerland, 2022: 651-667.

[174] MA X L, GUO F M, NIU W, et al. PCONV: The missing but desirable sparsity in DNN weight pruning for real-time execution on mobile devices[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2020, 34(4): 5117-5124.

[175] NIU W, MA X L, LIN S, et al. PatDNN: Achieving real-time DNN execution on mobile devices with pattern-based weight pruning[C]//Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2020: 907-922.

[176] ZHANG C M, YUAN G, NIU W, et al. ClickTrain: Efficient and accurate end-to-end deep learning training via fine-grained architecture-preserving pruning[C]//Proceedings of the ACM International Conference on Supercomputing. New York: ACM, 2021: 266-278.

[177] NIU W, SUN M S, LI Z G, et al. RT3D: Achieving real-time execution of 3D convolutional neural networks on mobile devices[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2021, 35(10): 9179-9187.

[178] GONG Y F, YUAN G, ZHAN Z, et al. Automatic mapping of the best-suited DNN pruning schemes for real-time mobile acceleration[J]. ACM Transactions on Design Automation of Electronic Systems, 2022, 27(5): 1-26.

[179] LI Z G, YUAN G, NIU W, et al. NPAS: A compiler-aware framework of unified network pruning and architecture search for beyond real-time mobile acceleration[C]//2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2021: 14255-14266.

[180] NIU W, GUAN J X, WANG Y Z, et al. DNNFusion: Accelerating deep neural networks execution with advanced operator fusion[C]//Proceedings of the 42nd ACM SIGPLAN International Conference on Programming

Language Design and Implementation. New York: ACM, 2021: 883-898.

[181] CAI X Y, WANG Y, ZHANG L. Optimus: An operator fusion framework for deep neural networks[J]. ACM Transactions on Embedded Computing Systems, 2023, 22(1): 1-26.

[182] JIA Z H, PADON O, THOMAS J, et al. TASO: Optimizing deep learning computation with automatic generation of graph substitutions[C]//Proceedings of the 27th ACM Symposium on Operating Systems Principles. New York: ACM, 2019: 47-62.

[183] TARG S, ALMEIDA D, LYMAN K. Resnet in resnet: Generalizing residual architectures[EB/OL]. (2016-03-25)[2024-07-22]. https://arxiv.org/abs/1603.08029v1.

[184] CAI H, CHEN T Y, ZHANG W N, et al. Efficient architecture search by network transformation[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2018, 32(1): 1.

[185] JIA Y. Learning Semantic Image Representations at a Large Scale[M]. Berkeley: University of California, 2014.

[186] 童敢, 黄立波, 吕雅帅 . 面向现代 GPU 的 Winograd 卷积加速研究[J]. 电子学报, 2024, 52(1): 244-257.
TONG G, HUANG L B, LYU Y S. Research on winograd convolution acceleration for modern GPU[J]. Acta Electronica Sinica, 2024, 52(1): 244-257. (in Chinese)

[187] DUKHAN M. The indirect convolution algorithm[EB/OL]. (2019-07-03) [2024-07-22]. https://arxiv. org/abs/1907.02129v1.

[188] UHYUN LEE Y P. Optimizing tensorflow lite runtime memory[EB/OL]. (2020-10-02)[2024-07-22]. https://blog.tensorflow.org/2020/10/optimizing-tensorflow-lite-runtime.html.

[189] SEKIYAMA T, IMAMICHI T, IMAI H, et al. Profile-guided memory optimization for deep neural networks [EB/OL]. (2018-04-26)[2024-07-22]. https://arxiv.org/abs/1804.10001v1.

[190] JAIN P, JAIN A, NRUSIMHA A, et al. Checkmate: Breaking the memory wall with optimal tensor rematerialization[J]. Proceedings of Machine Learning and Systems, 2020, 2: 497-511.

[191] MAAS M, BEAUGNON U, CHAUHAN A, et al. Tela-Malloc: Efficient on-chip memory allocation for production machine learning accelerators[C]//Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1. New York: ACM, 2022: 123-137.

[192] WANG M N, DING S H, CAO T, et al. AsyMo: Scalable and efficient deep-learning inference on asymmetric mobile CPUs[C]//Proceedings of the 27th Annual International Conference on Mobile Computing and Networking. New York: ACM, 2021: 215-228.

[193] KIM Y, KIM J, CHAE D J, et al. μLayer: Low latency on-device inference using cooperative single-layer acceleration and processor-friendly quantization[C]//Proceedings of the 14th EuroSys Conference 2019. New York: ACM, 2019: 1-15.

[194] ZHANG J R, ZHANG D Y, XU X H, et al. MobiPose: Real-time multi-person pose estimation on mobile devices[C]//Proceedings of the 18th Conference on Embedded Networked Sensor Systems. New York: ACM, 2020: 136-149.

[195] ZHANG J R, ZHANG D Y, YANG H, et al. MVPose: Realtime multi-person pose estimation using motion vector on mobile devices[J]. IEEE Transactions on Mobile Computing, 2023, 22(6): 3508-3524.

[196] JEONG J S, LEE J Y, KIM D, et al. Band: Coordinated multi-DNN inference on heterogeneous mobile processors[C]//Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services. New York: ACM, 2022: 235-247.

[197] TAN T X, CAO G H. Efficient execution of deep neural networks on mobile devices with NPU[C]//Proceedings of the 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021). New York: ACM, 2021: 283-298.

[198] WEI J Y, CAO T, CAO S J, et al. NN-stretch: Automatic neural network branching for parallel inference on heterogeneous multi-processors[C]//Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services. New York: ACM, 2023: 70-83.

[199] HUYNH L N, LEE Y, BALAN R K. DeepMon: Mobile GPU-based deep learning framework for continuous vision applications[C]//Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services. New York: ACM, 2017: 82-95.

[200] XU M W, ZHU M Z, LIU Y X, et al. DeepCache: Principled cache for mobile deep vision[C]//Proceedings of the 24th Annual International Conference on Mobile Computing and Networking. New York: ACM, 2018: 129-144.

[201] HUYNH L N, BALAN R K, LEE Y. DeepSense: A GPU-based deep convolutional neural network framework on commodity mobile devices[C]//Proceedings of the 2016 Workshop on Wearable Systems and Applications. New York: ACM, 2016: 25-30.

[202] HEO S, CHO S, KIM Y, et al. Real-time object detection

system with multi-path neural networks[C]//2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). Piscataway: IEEE, 2020: 174-187.

[203] ZHANG J R, YANG H, REN J, et al. MobiDepth: Real-time depth estimation using on-device dual cameras[C]//Proceedings of the 28th Annual International Conference on Mobile Computing and Networking. New York: ACM, 2022: 528-541.

[204] LIN J, CHEN W M, LIN Y, et al. Mcunet: Tiny deep learning on iot devices[J]. Advances in Neural Information Processing Systems, 2020, 33: 11711-11722.

[205] WEI J, TAY Y, BOMMASANI R, et al. Emergent abilities of large language models[EB/OL]. (2022-10-26)[2024-07-22]. https://arxiv.org/abs/2206.07682v2.

[206] Vaswani A. Attention is all you need[J]. Advances in Neural Information Processing Systems, 2017, 1: 1.

[207] DEVLIN J, CHANG M W, LEE K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding[EB/OL]. (2019-05-24)[2024-07-22]. https://arxiv.org/abs/1810.04805v2.

[208] Dosovitskiy A. An image is worth 16x16 words: Transformers for image recognition at scale[EB/OL]. (2021-06-03)[2024-07-22]. https://arxiv.org/abs/2010.11929.

[209] Touvron H, Cord M, Douze M, et al. Training data-efficient image transformers & distillation through attention[C]//International Conference on Machine Learning. New York: PMLR, 2021: 10347-10357.

[210] LI Y Y, HU J, WEN Y, et al. Rethinking vision transformers for MobileNet size and speed[C]//2023 IEEE/CVF International Conference on Computer Vision (ICCV). Piscataway: IEEE, 2023: 16843-16854.

[211] LIU Z, LIN Y T, CAO Y, et al. Swin transformer: Hierarchical vision transformer using shifted windows[C]//2021 IEEE/CVF International Conference on Computer Vision (ICCV). Piscataway: IEEE, 2021: 9992-10002.

[212] LIU Z, HU H, LIN Y T, et al. Swin transformer V2: Scaling up capacity and resolution[C]//2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2022: 11999-12009.

[213] DONG X Y, BAO J M, CHEN D D, et al. CSWin transformer: A general vision transformer backbone with cross-shaped windows[C]//2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2022: 12114-12124.

[214] CHEN C R, FAN Q F, PANDA R. CrossViT: Cross-attention multi-scale vision transformer for image classification[C]//2021 IEEE/CVF International Conference on Computer Vision (ICCV). Piscataway: IEEE, 2021: 347-356.

[215] WU S T, WU T Y, TAN H R, et al. Pale transformer: A general vision transformer backbone with pale-shaped attention[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2022, 36(3): 2731-2739.

[216] PAN Z, CAI J, ZHUANG B. Fast vision transformers with hilo attention[J]. Advances in Neural Information Processing Systems, 2022, 35: 14541-14554.

[217] WANG W, CHEN W, QIU Q, et al. CrossFormer++: A versatile vision transformer hinging on cross-scale attention[J]. IEEE Trans Pattern Anal Mach Intell, 2024, 46(5): 3123-3136.

[218] MEHTA S, RASTEGARI M. MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer[EB/OL]. (2022-03-04) [2024-07-22]. https://arxiv. org/abs/2110.02178v2.

[219] MEHTA S, RASTEGARI M. Separable self-attention for mobile vision transformers[EB/OL]. (2022-06-06) [2024-07-22]. https://arxiv.org/abs/2206.02680v1.

[220] LI Y, YUAN G, WEN Y, et al. Efficientformer: Vision transformers at mobilenet speed[J]. Advances in Neural Information Processing Systems, 2022, 35: 12934-12949.

[221] SHAKER A, MAAZ M, RASHEED H, et al. SwiftFormer: Efficient additive attention for transformer-based real-time mobile vision applications[C]//2023 IEEE/CVF International Conference on Computer Vision (ICCV). Piscataway: IEEE, 2023: 17379-17390.

[222] CHEN Y P, DAI X Y, CHEN D D, et al. Mobile-former: Bridging MobileNet and transformer[C]//2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2022: 5260-5269.

[223] 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. New York: PMLR, 2023: 38087-38099.

[224] Dettmers T, Lewis M, Belkada Y, et al. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale[J]. Advances in Neural Information Processing Systems, 2022, 35: 30318-30332.

[225] LIN J, TANG J M, TANG H T, et al. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration[J]. Proceedings of Machine Learning and Systems, 2024, 6: 87-100.

[226] SHEN X, DONG P Y, LU L, et al. Agile-quant: Activation-guided quantization for faster inference of LLMs on the edge[J]. Proceedings of the AAAI Conference on Ar-

tificial Intelligence, 2024, 38(17): 18944-18951.

[227] AINSLIE J, LEE-THORP J, DE JONG M, et al. GQA: Training generalized multi-query transformer models from multi-head checkpoints[EB/OL]. (2023-12-23)[2024-07-22]. https://arxiv.org/abs/2305.13245v3.

[228] GE S Y, ZHANG Y N, LIU L Y, et al. Model tells you what to discard: Adaptive KV cache compression for LLMs[EB/OL]. (2024-10-29) [2024-07-22]. https://arxiv. org/abs/2310.01801v4.

[229] JIANG H Q, WU Q H, LIN C Y, et al. LLMLingua: Compressing prompts for accelerated inference of large language models[EB/OL]. (2023-12-06)[2024-07-22]. https://arxiv.org/abs/2310.05736v2.

[230] LI Y C, DONG B, LIN C H, et al. Compressing context to enhance inference efficiency of large language models[EB/OL]. (2023-10-09)[2024-07-22]. https://arxiv.org/abs/2310.06201v1.

[231] MU J, LI X, GOODMAN N. Learning to compress prompts with gist tokens[J]. Advances in Neural Information Processing Systems, 2024, 36: 1.

[232] REN S Y, JIA Q, ZHU K Q. Context compression for auto-regressive transformers with sentinel tokens[EB/OL]. (2023-10-15)[2024-07-22]. https://arxiv.org/abs/2310.08152v2.

[233] XIAO G X, TIAN Y D, CHEN B D, et al. Efficient streaming language models with attention sinks[EB/OL]. (2024-04-07) [2024-07-22]. https://arxiv.org/abs/2309.17453v4.

[234] HAN C, WANG Q F, PENG H, et al. LM-infinite: Zero-shot extreme length generalization for large language models[EB/OL]. (2024-06-24) [2024-07-22]. https://arxiv.org/abs/2308.16137v7.

[235] ZHANG Z, SHENG Y, ZHOU T, et al. H2O: Heavy-hitter oracle for efficient generative inference of large language models[J]. Advances in Neural Information Processing Systems, 2024, 36: 1.

[236] WU H Y, TU K W. Layer-condensed KV cache for efficient inference of large language models[EB/OL]. (2024-06-04)[2024-07-22]. https://arxiv.org/abs/2405.10637v2.

[237] KWON W, LI Z H, ZHUANG S Y, et al. Efficient memory management for large language model serving with PagedAttention[C]//Proceedings of the 29th Symposium on Operating Systems Principles. New York: ACM, 2023: 611-626.

[238] FEDUS W, ZOPH B, SHAZEER N M. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity[J]. Journal of Machine Learning Research, 2022, 23(120): 1-39.

[239] COLIN R, NOAM S, ADAM R, et al. Exploring the limits of transfer learning with a unified text-to-text transformer[J]. Journal of Machine Learning Research, 2020, 21: 1-67.

[240] HWANG R, WEI J Y, CAO S J, et al. Pre-gated MoE: An algorithm-system co-design for fast and scalable mixture-of-expert inference[C]//2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA). Piscataway: IEEE, 2024: 1018-1031.

[241] KONG R, LI Y C, FENG Q T, et al. SwapMoE: Serving off-the-shelf MoE-based large language models with tunable memory budget[C]//Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Stroudsburg: USAACL, 2024: 6710-6720.

[242] RAJBHANDARI S, LI C, YAO Z, et al. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale[C]//International Conference on Machine Learning. New York: PMLR, 2022: 18332-18346.

[243] YI R J, GUO L W, WEI S Y, et al. EdgeMoE: Empowering sparse large language models on mobile devices[EB/OL]. (2025-03-07)[2025-05-06]. https://arxiv.org/abs/2308.14352v2.

[244] SONG Y X, XIE H T, ZHANG Z Y, et al. Turbo sparse: Achieving LLM SOTA performance with minimal activated parameters[EB/OL]. (2024-06-11) [2024-07-22]. https://arxiv.org/abs/2406.05955v2.

[245] 陶建华, 吴飞, 黄民烈, 等 . 中国人工智能系列白皮书——大模型技术[R]. 北京: 中国人工智能学会, 2023.
TAO J H, WU F, HUANG M L, et al. China AI Series White Paper Large Model Technology[R]. Beijing: Chinese Association for Artificial Intelligence, 2023. (in Chinese)

[246] MA R L, WANG J Y, QI Q, et al. Poster: PipeLLM: Pipeline LLM inference on heterogeneous devices with sequence slicing[C]//Proceedings of the ACM SIGCOMM 2023 Conference. New York: ACM, 2023: 1126-1128.

[247] WEI Y X, YE S Y, JIANG J Z, et al. Communication-efficient model parallelism for distributed in situ transformer inference[C]//2024 Design, Automation & Test in Europe Conference & Exhibition (DATE). Piscataway: IEEE, 2024: 1-6.

[248] ZHAO J C, SONG Y R, LIU S M, et al. LinguaLinked: A distributed large language model inference system for mobile devices[EB/OL]. (2023-12-01)[2024-07-22]. https://arxiv.org/abs/2312.00388v1.

[249] YANG B F, HE L X, LING N W, et al. EdgeFM: Leveraging foundation model for open-set learning on the edge[C]// Proceedings of the 21st ACM Conference on Embedded

Networked Sensor Systems. New York: ACM, 2023: 111-124.

[250] CHEN Y X, LI R P, ZHAO Z F, et al. NetGPT: A native-AI network architecture beyond provisioning personalized generative services[EB/OL]. (2024-03-09)[2024-07-22]. https://arxiv.org/abs/2307.06148v4.

[251] YAO J C, ZHANG S Y, YAO Y, et al. Edge-cloud polarization and collaboration: A comprehensive survey for AI[J]. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(7): 6866-6886.

[252] XU D L, YIN W S, JIN X, et al. LLMCad: Fast and scalable on-device large language model inference[EB/OL]. (2023-09-08)[2024-07-22]. https://arxiv.org/abs/2309.04255v1.

[253] YIN W S, XU M W, LI Y C, et al. LLM as a system service on mobile devices[EB/OL]. (2024-03-18)[2024-07-22]. https://arxiv.org/abs/2403.11805v1.

[254] SONG Y X, MI Z Y, XIE H T, et al. PowerInfer: Fast large language model serving with a consumer-grade GPU[EB/OL]. (2024-12-12)[2024-07-22]. https://arxiv.org/abs/2312.12456v2.

[255] XUE Z L, SONG Y X, MI Z Y, et al. PowerInfer-2: Fast large language model inference on a smartphone[EB/OL]. (2024-12-12)[2024-07-22]. https://arxiv.org/abs/2406.06282v3.

[256] LIU Z C, ZHAO C S, IANDOLA F, et al. MobileLLM: Optimizing sub-billion parameter language models for on-device use cases[EB/OL]. (2024-06-27)[2024-07-22]. https://arxiv.org/abs/2402.14905v2.

[257] XU D, ZHANG H, YANG L, et al. Empowering 1000 tokens/second on-device llm prefilling with mllm-npu[EB/OL]. (2024-12-15)[2024-07-22]. https://arxiv.org/pdf/2407.05858.

[258] DENG J, DONG W, SOCHER R, et al. ImageNet: A large-scale hierarchical image database[C]//2009 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2009: 248-255.

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

[260] RUSSAKOVSKY O, DENG J, SU H, et al. ImageNet large scale visual recognition challenge[J]. International Journal of Computer Vision, 2015, 115(3): 211-252.

[261] KRIZHEVSKY A, HINTON G. Learning multiple layers of features from tiny images[J]. Handbook of Systemic Autoimmune Diseases, 2009, 1(4): 1-60.

[262] ROMERO A, BALLAS N, KAHOU S E, et al. FitNets: Hints for thin deep nets[EB/OL]. (2015-03-27)[2024-07-22]. https://arxiv.org/abs/1412.6550v4.

[263] CROWLEY E J, GRAY G, STORKEY A. Moonshine: Distilling with cheap convolutions[EB/OL]. (2019-11-07)[2024-07-22]. https://arxiv.org/abs/1711.02613v4.

[264] WAN A, DAI X L, ZHANG P Z, et al. FBNetV2: Differentiable neural architecture search for spatial and channel dimensions[C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2020: 12965-12974.

[265] MISAWA M, KUDO S E, MORI Y, et al. Development of a computer-aided detection system for colonoscopy and a publicly accessible large colonoscopy video database (with video)[J]. Gastrointestinal Endoscopy, 2021, 93(4): 960-967.

[266] EELBODE T, SINONQUEL P, BISSCHOPS R, et al. Convolutional LSTM[M]//Computer-Aided Analysis of Gastrointestinal Videos. Cham: Springer International Publishing, 2021: 121-126.

[267] WANG A, SINGH A, MICHAEL J, et al. GLUE: A multi-task benchmark and analysis platform for natural language understanding[EB/OL]. (2019-02-22)[2024-07-22]. https://arxiv.org/abs/1804.07461v3.

[268] MERITY S, XIONG C M, BRADBURY J, et al. Pointer sentinel mixture models[EB/OL]. (2016-09-26)[2024-07-22]. https://arxiv.org/abs/1609.07843v1.

[269] KIM S, HOOPER C, GHOLAMI A, et al. SqueezeLLM: Dense-and-sparse quantization[EB/OL]. (2023-06-13)[2024-07-22]. https://arxiv.org/abs/2306.07629v4.

[270] HUANG W, LIU Y D, QIN H T, et al. BiLLM: Pushing the limit of post-training quantization for LLMs[EB/OL]. (2024-05-15)[2024-07-22]. https://arxiv.org/abs/2402.04291v2.

[271] COBBE K, KOSARAJU V, BAVARIAN M, et al. Training verifiers to solve math word problems[EB/OL]. (2021-11-18)[2024-07-22]. https://arxiv.org/abs/2110.14168v2.

[272] LI Y C, LI Y C, DONG B, et al. Unlocking context constraints of LLMs: Enhancing context efficiency of LLMs with self-information-based content filtering[EB/OL]. (2023-04-24)[2024-07-22]. https://arxiv.org/abs/2304.12102v1.

[273] RAJPURKAR P, ZHANG J, LOPYREV K, et al. SQuAD: 100, 000+ questions for machine comprehension of text[EB/OL]. (2016-10-11)[2024-07-22]. https://arxiv.org/abs/1606.05250v3.

[274] SHOEYBI M, PATWARY M, PURI R, et al. Megatron-LM: Training multi-billion parameter language models using model parallelism[EB/OL]. (2020-03-13)[2024-07-22]. https://arxiv.org/abs/1909.08053v4.

[275] LIN S, HILTON J, EVANS O. TruthfulQA: Measuring how models mimic human falsehoods[EB/OL]. (2022-05-08)[2024-07-22]. https://arxiv.org/abs/2109.07958v2.
[276] PAPERNO D, KRUSZEWSKI G, LAZARIDOU A, et al. The LAMBADA dataset: Word prediction requiring a broad discourse context[EB/OL]. (2016-06-20)[2024-07-22]. https://arxiv.org/abs/1606.06031v1.
[277] BAI Y S, LV X, ZHANG J J, et al. LongBench: A bilingual, multitask benchmark for long context understanding[EB/OL]. (2024-06-19)[2024-07-22]. https://arxiv.org/abs/2308.14508v2.
[278] SUN Z Q, YU H K, SONG X D, et al. MobileBERT: A compact task-agnostic BERT for resource-limited devices[EB/OL]. (2020-04-14)[2024-07-22]. https://arxiv.org/abs/2004.02984v2.
[279] JIAO X Q, YIN Y C, SHANG L F, et al. TinyBERT: Distilling BERT for natural language understanding[EB/OL]. (2020-10-16) [2024-07-22]. https://arxiv. org/abs/1909.10351v5.
[280] YOU K E, ZHANG H T, SCHOOP E, et al. Ferret-UI: Grounded mobile UI understanding with multimodal LLMs[EB/OL]. (2024-04-08)[2024-07-22]. https://arxiv.org/abs/2404.05719v1.
[281] HU Z N, ISCEN A, SUN C, et al. Reveal: Retrieval-augmented visual-language pre-training with multi-source multimodal knowledge memory[C]//2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2023: 23369-23379.
[282] HU T M, LUO B, YANG C H, et al. MO-MIX: Multi-objective multi-agent cooperative decision-making with deep reinforcement learning[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023, 45(10): 12098-12112.
[283] ZHANG J J, HOU Y P, XIE R B, et al. AgentCF: Collaborative learning with autonomous language agents for recommender systems[C]//Proceedings of the ACM Web Conference 2024. New York: ACM, 2024: 3679-3689.

作者简介

章晋睿 男,1992年12月出生于湖南省湘潭市,博士. 现为清华大学计算机与技术系博士后. 主要研究领域为边缘智能、移动计算加速、端侧异构计算、计算机视觉.
E-mail: zhangjinrui@tsinghua.edu.cn

许 愿 女,2003年2月出生于浙江省杭州市. 现为中南大学计算机学院本科生. 主要研究领域为边缘智能、人机交互.
E-mail: _xuan_@csu.edu.cn

龙婷婷 女,2001年6月出生于湖北省枣阳市. 现为中南大学计算机学院硕士研究生. 主要研究领域为边缘智能、模型压缩、文本视频检索优化.
E-mail: TingtingLong@csu.edu.cn

任 炬 男,1987年12月出生于湖南省汨罗市,博士. 现为清华大学计算机与技术系长聘副教授. 国家级人才项目获得者. 主要研究领域为边缘智能计算与智能协作、边缘智能安全与隐私保护. 中国电子学会会员编号:E190018924.
E-mail: renju@tsinghua.edu.cn

张德宇 男,1987年6月出生于河南省新乡市,博士. 现为中南大学计算机学院副教授. 主要研究领域为边缘计算、物联网、移动端深度学习加速. 中国电子学会会员编号:E190085074M.
E-mail: zdy876@csu.edu.cn

张尧学 男,1956年1月出生于湖南省常德市,博士. 现为清华大学计算机与技术系长聘教授. 中国工程院院士. 主要研究领域为计算机网络、操作系统、普适计算. 中国电子学会会员编号:E190004903F.
E-mail: zhangyx@tsinghua.edu.cn