

面向片上自动化机器学习的 高性能算法研究

(申请清华大学工学硕士学位论文)

培养单位：集成电路学院

学 科：电子科学与技术

研 究 生：陈 洪 江

指 导 教 师：魏 少 军 教 授

二〇二三年四月

面向片上自动化机器学习的高性能算法研究

陈
洪
江

Research on the High-performance On-chip Automated Machine Learning Algorithms

Thesis submitted to
Tsinghua University
in partial fulfillment of the requirement
for the degree of

Master of Science

in

Electronic Science and Technology

by

Chen Hongjiang

Thesis Supervisor: Professor Wei Shaojun

April, 2023

学位论文公开评阅人和答辩委员会名单

公开评阅人名单

刘雷波	教授	清华大学
陈虹	研究员	清华大学

答辩委员会名单

主席	刘雷波	教授	清华大学
委员	何虎	副教授	清华大学
	李树国	研究员	清华大学
	吴行军	副研究员	清华大学
	张贺	企业专家	
秘书	简锦明	助理研究员	清华大学

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：(1) 已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；(2) 为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；(3) 按照上级教育主管部门督导、抽查等要求，报送相应的学位论文。

本人保证遵守上述规定。

作者签名:

导师签名:

日 期:

日 期:

摘要

近年来，深度学习领域核心技术的突破使得人们在计算机视觉，语音识别，自然语言处理等任务中获得了显著的效果提升。但是随着任务规模与复杂度逐年上升，神经网络模型的设计、调优、部署难度也在逐渐增加。自动化机器学习（AutoML）应运而生，旨在数据驱动下，不依赖于专家经验自动生成符合要求的模型，从而达到降低人工智能使用门槛的目的。

本文研究的对象为高性能片上自动化机器学习算法，研究的重点在于缓解庞大的算法计算需求与有限的片上资源之间的矛盾。区别于传统云端训练加终端部署的两阶段的方法，片上自动化机器学习可以获得真实的终端性能反馈，从而大大提升自动化模型生成的质量。

本工作从自动网络结构搜索与自动模型量化这两个子领域切入，从三个角度具体介绍了高性能片上自动化机器学习中的核心技术难点：首先本文提出了一种高效的多目标搜索方法，通过有效利用片上的延时、功耗等直接反馈信号并结合基于值迭代的强化学习方法，相比于云端的代理搜索，本方法在 Evolver 处理芯片上以 78% 的搜索成本获得了 1.23 倍最终模型的性能提升；其次本文将上述方法扩展至同时优化不同的自动化机器学习子领域的场景，提出了一种轻量级卷积网络设计和量化的联合搜索策略，通过结合可微分的结构搜索与基于策略梯度的强化学习量化搜索，使得最终搜出的模型以 0.9% 的精度损失为代价获得了 1.8 倍推理速度的提升；最后本文将协同优化的理念延伸至多节点的联邦学习场景，通过优化主机的数据聚合过程、节点端的搜索过程，同时结合模型的节点端量化，将搜索的通信成本下降至原先的 23%。

关键词：自动化机器学习、模型量化、轻量级网络结构搜索、强化学习、联邦学习

Abstract

In recent years, breakthroughs in core technologies in the field of deep learning have led to significant improvements in computer vision, speech recognition and natural language processing. However, as the scale and complexity of tasks continue to increase year by year, the difficulty of designing, tuning, and deploying neural network models is also gradually increasing. To address this issue, Automated Machine Learning (AutoML) has emerged, aiming to automatically generate models driven by data that meet requirements without relying on expert experience. The goal is to reduce the barriers to using artificial intelligence.

The focus of this study is on efficient on-chip automated machine learning algorithms, with the aim of alleviating the contradiction between the massive computational demands of the algorithms and the limited on-chip resources. Unlike the traditional two-stage approach of cloud-based training and terminal deployment, on-chip automated machine learning can obtain real device performance feedback, greatly improving the quality of automatically generated models.

This work focuses on two sub-fields of automated machine learning, namely Automatic Neural Architecture Search and Automatic Model Quantization. It presents three core technical challenges in on-chip automated machine learning from three perspectives. Firstly, an efficient multi-objective search method is proposed, which effectively utilizes on-chip feedback signals such as latency and power consumption and combines with a value iteration-based reinforcement learning method. This method achieved a 1.23x performance improvement with only 78% of the search cost on the Evolver processing chip. Secondly, the above method is extended to scenarios where different sub-fields of automated machine learning are optimized simultaneously. A joint search strategy for efficient convolutional network design and quantization is proposed, which combines differentiable architecture search with policy gradient-based reinforcement learning quantization. The final model achieved a 1.8x inference speed improvement at the cost of only 0.9% accuracy loss. Finally, the idea of collaborative optimization is extended to the federated learning scenario with multiple nodes. By optimizing the data aggregation process at the host side, the search process at the node

Abstract

side, and incorporating node-level quantization of the model, the communication cost of the search is reduced to only 23% of its original cost.

Keywords: Automated Machine Learning; Quantization; Neural Architecture Search; Reinforcement Learning; Federated Learning

目 录

摘要	I
ABSTRACT	II
目录	IV
插图和附表清单	VI
符号和缩略语说明	VIII
第一章 绪 论	1
1.1 研究背景和意义	1
1.2 国内外研究现状	3
1.2.1 学术界研究热点	3
1.2.2 工业界产品	5
1.3 论文研究的范围	7
1.4 论文研究的难点	8
1.5 论文内容的组织结构	8
第二章 面向片上量化-电压-频率的多目标优化	10
2.1 本章引言	10
2.2 基于 Q-LEARNING 的量化-电压-频率联合搜索算法	13
2.2.1 算法顶层结构设计	14
2.2.2 迭代优化技术	16
2.3 RLU 模块设计	17
2.4 算法评估	19
2.4.1 验证系统搭建	19
2.4.2 三种搜索方法设定	20
2.4.3 消融对比分析	21
2.4.4 多优化目标之间的取舍	24
2.4.5 可扩展性分析	24
2.4.6 在线 QVF 调优的长久效益分析	25
2.5 本章小结	26
第三章 面向网络结构与量化策略的片上联合搜索	27

目 录

3.1 本章引言	27
3.2 基于完全可微分的网络结构量化策略搜索	28
3.2.1 可微分的网络结构搜索	29
3.2.2 量化感知训练 QAT	29
3.2.3 算法框架与原理	31
3.3 算法评估	33
3.3.1 CIFAR10 数据集	33
3.3.2 Imagenet 数据集	35
3.3.3 终身学习模拟	36
3.4 本章小结	37
第四章 多节点片上联邦网络结构与量化策略搜索	38
4.1 本章引言	38
4.2 高通信效率的联邦网络结构与量化策略搜索	39
4.2.1 问题定义	39
4.2.2 通信量缩减的三个维度	41
4.2.3 整体算法流程	44
4.3 算法评估	46
4.3.1 实验设置	46
4.3.2 训练以及通信效率分析	46
4.3.3 个性化硬件感知的网络结构分析	47
4.4 本章小结	49
第五章 总结与展望	50
5.1 论文工作总结	50
5.2 未来工作展望	51
参考文献	53
致 谢	57
声 明	58
个人简历、在学期间完成的相关学术成果	59
指导教师评语	60
答辩委员会决议书	61

插图和附表清单

图 1-1 AutoML 基本流程.....	1
图 1-2 AutoML 学术界热点.....	4
图 1-3 AutoML 工业界产品.....	6
图 2-1 离线 QVF 调优框架.....	11
图 2-2 片上 QVF 调优框架.....	12
图 2-3 基于 Q-Learning 的 QVF 调优原理.....	14
图 2-4 异常点过滤原理.....	17
图 2-6 RLU 模块工作流.....	18
图 2-5 RLU 架构设计.....	18
图 2-7 QVF 调优验证系统.....	19
图 2-8 Evolver 电压频率曲线	20
图 2-9 VGG8 在本地数据集上不同方法的 QVF 调优曲线	22
图 2-10 ResNet34 在本地数据集上不同方法的 QVF 调优曲线	24
图 2-11 片上 QVF 调优的长久收益	26
图 3-1 网络结构设计与量化工作中模型的大小与精度分布	28
图 3-2 可微分网络结构搜索原理.....	29
图 3-3 QAT 训练过程原理图	30
图 3-4 端到端网络结构与量化策略结构原理	31
图 3-5 模型的基本组成结构	33
图 3-6 精度、延迟的训练曲线	34
图 3-7 在 (a) GPU (b) CPU 上最后生成的模型结构与量化方式	35
图 3-8 分阶段训练与从头训练的 (a) 精度曲线 (b) 权值差变化	36
图 4-1 联邦网络结构搜索的基本框架	38
图 4-2 通信量缩减的三个维度原理	41
图 4-3 (a) 权值共享 (b) 比特共享	41
图 4-4 三种算法的 (a) 平均精度曲线 (b) 每轮的平均通信量	46
图 4-5 三种方法的通信量以及精度对比	47
图 4-6 (a) 硬件感知型模型精度曲线 (b) 延迟与模型大小散点图	47

表 2-1 Evolver 上 VGG8 的搜索空间示例.....	14
表 2-2 每个 ϵ 阶段的搜索迭代次数	21
表 2-3 Evolver 上 VGG8 的四种不同部署策略对比.....	22
表 3-1 本方法与现有其他方法的对比	34
表 4-1 通行量关键参数对比.....	40
表 4-2 id 函数与块类型对应关系.....	42
表 4-3 id 函数与量化位宽的对应关系.....	44
表 4-4 不同节点模型的指标对比.....	48
算法 2-1 多目标量化、电压、频率联调算法.....	15
算法 3-1 端到端网络结构与量化策略联合搜索	32
算法 4-1 联邦网络结构搜索与量化策略搜索	45

符号和缩略语说明

AutoML	自动化机器学习 (Automated Machine Learning)
RL	强化学习 (Reinforcement Learning)
FL	联邦学习 (Federated Learning)
NAS	神经网络结构搜索 (Neural Architecture Search)
MONAS	多目标神经网络结构搜索 (Multi-Object Neural Architecture Search)
RLU	强化学习单元 (Reinforcement Learning Unit)
RC	奖励计算器 (Reward Calculator)
QTU	Q 表更新器 (Q-Table Updater)
PS	策略采集器 (Policy Sampler)
LSFR	线性反馈移位寄存器 (Linear Feedback Shift Register)
DNN	深度神经网络 (Deep Neural Network)
RNN	循环神经网络 (Recurrent Neural Network)
EA	进化算法 (Evolutionary Algorithm)
DARTS	可微分网络结构搜索 (Differential Architecture Search)
PTQ	训练后量化 (Post Training Quantization)
QAT	量化感知训练 (Quantization Aware Training)
SRAM	静态随机存储器 (Static Random Access Memory)
GPU	图形处理器 (Graphics Processing Unit)
CPU	中央处理器 (Central Processing Unit)
FPGA	现场可编程门阵列 (Field-Programmable Gate Array)
PLL	锁相环 (Phase-Locked Loop)
RNG	随机数发生器 (Random Number Generator)
FMC	嵌入式模块连接器 (FPGA Mezzanine Card)
LDO	线性稳压器 (Low Dropout Regulator)

第一章 绪 论

随着机器学习技术的快速发展，越来越多的人开始关注如何将其应用到各种现实世界的问题中。然而，机器学习需要大量的数据和计算资源，并且还需要熟练的数据科学家和机器学习专家进行模型开发和调优。这些因素限制了机器学习技术的推广和应用范围。传统的机器学习方法需要对数据进行预处理、特征工程、模型选择和超参数优化等繁琐的任务。这些任务需要专业知识和大量的实践经验，对于非专业用户来说往往是一道难题。因此，自动化机器学习技术（Automated Machine Learning 即 AutoML）应运而生，旨在通过自动执行机器学习中的一些繁琐任务来降低机器学习的门槛。其技术主要包括多个方面的任务：自动模型选择、超参数自动优化、模型结构搜索和自动模型压缩等。模型选择的目标是选择最适合特定数据集和任务的机器学习模型。超参数优化的目标是通过搜索超参数空间来优化模型性能，以达到最佳的性能和泛化能力。模型结构搜索的目标是自动搜索最佳的神经网络架构。自动模型压缩指不依赖人为经验，通过量化、剪枝、蒸馏等手段自动压缩网络模型从而进行进一步的部署工作。这些任务可以通过使用不同的算法来实现，如基于元学习的方法、贝叶斯优化、遗传算法、进化算法等。

然而目前该领域用到的方法以云端训练加端侧部署的两阶段方式进行，其局限性主要体现在无法有效获端侧的真实反馈指标导致生成的模型质量较低，同时由于数据隐私等问题导致某些场景无法使用云端训练。因此片上自动化机器学习成为了替代目前云端场景任务的较优选择，然而由于片上有有限的存储和计算资源，设计高效的片上自动化机器学习算法变得非常有挑战性。

本章作为绪论章节，主要介绍本论文的研究背景和意义、研究现状、研究范围与目标。具体而言 1.1 节就片上自动化机器学习进行研究背景和意义的展开，1.2 节从学术界和工业界两个角度介绍目前自动机器学习领域的研究现状，1.3 节将会介绍本课题整个工作的研究范围以及最核心的三个技术难点，1.4 节将介绍后续章节的内容组织形式。

1.1 研究背景和意义

近年来，深度学习已成为人工智能领域的重要技术之一，其应用已经覆盖了各种领域，包括计算机视觉^{[10][11][12]}、自然语言处理^{[13][14][15]}、语音识别^{[16][17][18]}、推荐系统^[19]、医疗诊断^[20]等。在深度学习的发展历程中，不断有新的网络结构和算法

被提出，从最早的卷积神经网络^[21]、循环神经网络^[22]到后来的生成对抗网络^[23]、注意力机制^[24]等，这些新的技术极大地推动了深度学习在各个领域的应用和发展。然而随着任务规模越来越大，模型复杂度越来越高，人工设计对应的机器学习模型的难度也在逐年提升。在深度学习领域，设计一个神经网络模型的确是一项难度很大的任务，这是因为神经网络模型的复杂性和规模已经超越了本工作的想象。如今的神经网络模型往往包含数以百万计的参数，需要进行大量的计算和优化才能得到准确的结果，设计神经网络模型需要掌握多种数学知识、计算机科学和统计学的技能，以及对不同领域的具体应用有一定的理解。另外，设计神经网络模型的难度也与数据集的规模和复杂性相关，如今的大型数据集往往包含数百万到数十亿条数据，这些数据具有多种特征和属性，需要进行有效的预处理和特征提取才能得到高质量的神经网络模型。所有这些工作都极其依赖于较强的专家经验，阻碍了深度学习在各行业的大规模落地和发展。

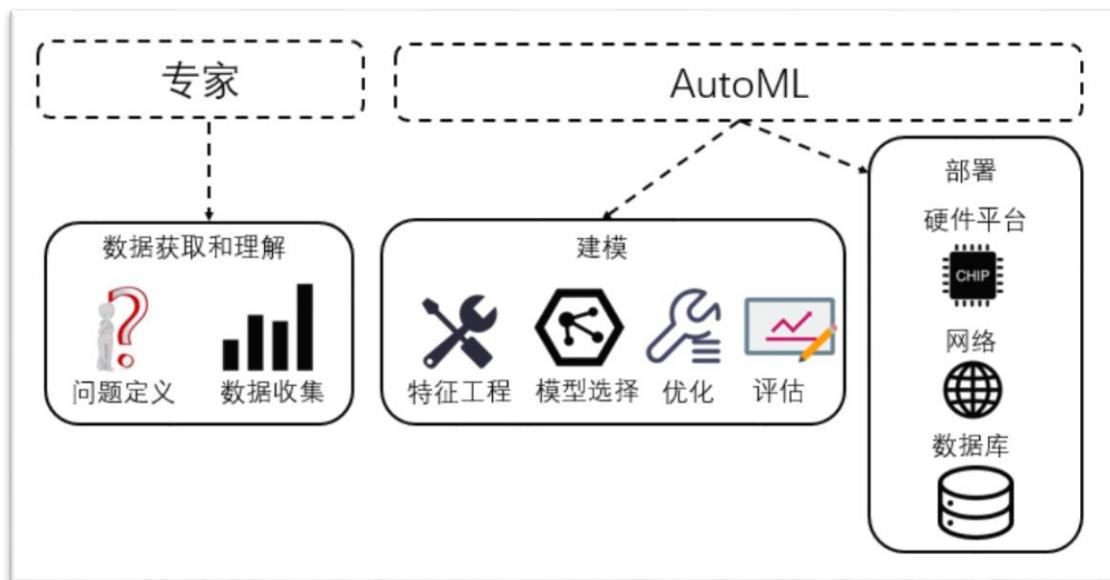


图 1-1 AutoML 基本流程

为应对这个问题，诞生了自动机器学习（AutoML）这个研究领域。如图 1-1 所示，AutoML 是一种通过自动化机器学习流程的各个阶段来加快和简化机器学习模型的开发和部署过程的方法。虽然其概念已经存在了一段时间，但直到近年来，由于计算能力的提高，AutoML 才开始受到越来越多的关注和研究。其研究动机主要是基于两个方面的考虑。第一方面是机器学习的高门槛，需要专业的知识和技能，才能够进行高质量的机器学习模型的构建和调节，而这些专业的知识和技能对于一般人来说是不容易掌握的。因此，开发自动化的机器学习方法可以帮助更多的人

能够利用机器学习技术，快速地构建高质量的模型。第二方面是机器学习在实践中的挑战，例如如何选择合适的模型和参数，如何避免过拟合和欠拟合等。这些问题需要进行反复的实验和调试，需要耗费大量的时间和人力资源。开发自动化的机器学习方法可以帮助解决这些问题，减少人工干预的需求，提高机器学习的效率和准确率。

当前的 AutoML 产品以云端训练加终端部署的形式为主，云端通常涉及使用大规模的云计算资源，通过在云端进行大规模的模型训练和优化，来自动化生成高质量的机器学习模型。云端计算优势在于，它可以快速地进行大规模模型训练和搜索，使用先进的算法和技术，高效率地进行模型迭代。当然其劣势也很明显。首先，由于数据需要上传到云端进行训练，存在数据隐私和安全的问题。其次，云端训练无法实时获得真实硬件的物理反馈信息，导致在进行硬件感知型的模型搜索时只能采用某种代理信号作为评价指标，影响了在这类任务中模型的生成质量。

与之相对的，片上自动化机器学习是一种在边缘设备上进行自动化机器学习的技术，通常涉及在设备上使用轻量级的模型，通过使用硬件感知的算法和技术，不断优化模型，以适应不同的环境和场景中对模型精度、延时、能耗的需求。在实现低延迟的响应和高效的资源利用的同时，还可以保护数据隐私和安全。然而，片上自动化机器学习最大的挑战便是有限的计算资源与庞大的算法复杂度之间的矛盾。因此从算法的角度来说，设计一套高效的片上自动化机器学习系统变得关键。

1.2 国内外研究现状

在过去几年中，自动化机器学习得到了广泛的关注和研究。研究人员和工业界企业纷纷投入了大量的精力和资源来推动这项技术的发展和应用。学术界自从 2016 年以来，每年发表的 AutoML 领域的论文数量呈指数增长。以 2019 年为例，谷歌学术搜索结果显示，仅 2019 年一年就发表了超过 200 篇关于该领域的论文，这些论文覆盖了该领域的各个方面，包括超参数优化、神经结构搜索、模型选择和集成等等。工业界例如，Google、Microsoft、IBM 等知名企业都推出了自己的自动化机器学习平台，国内的如第四范式、探知立方等初创公司也在这个赛道吸引了各方领域的关注。

1.2.1 学术界研究热点

AutoML 的研究历史可以追溯到上个世纪九十年代，当时研究人员开始探索如何自动化地训练机器学习模型。由于当时计算能力和数据量的限制，机器学习模型的训练非常困难，需要大量的人力和物力投入。因此，研究人员开始考虑如何自动化地训练机器学习模型，从而减少人力和物力的投入。早期的自动化机器学习方法主要基于符号推理技术^[39]，如决策树、规则库等。这些方法虽然可以自动地构建模型，但由于其基于人工定义的特征和规则，因此在处理复杂问题时表现不佳。随着深度学习的兴起，AutoML 的发展也得到了极大的促进。深度学习的主要优点是可以自动地从数据中学习特征和规律，因此可以处理更加复杂的问题。在这个背景下，研究人员开始探索如何自动化地构建深度学习模型。

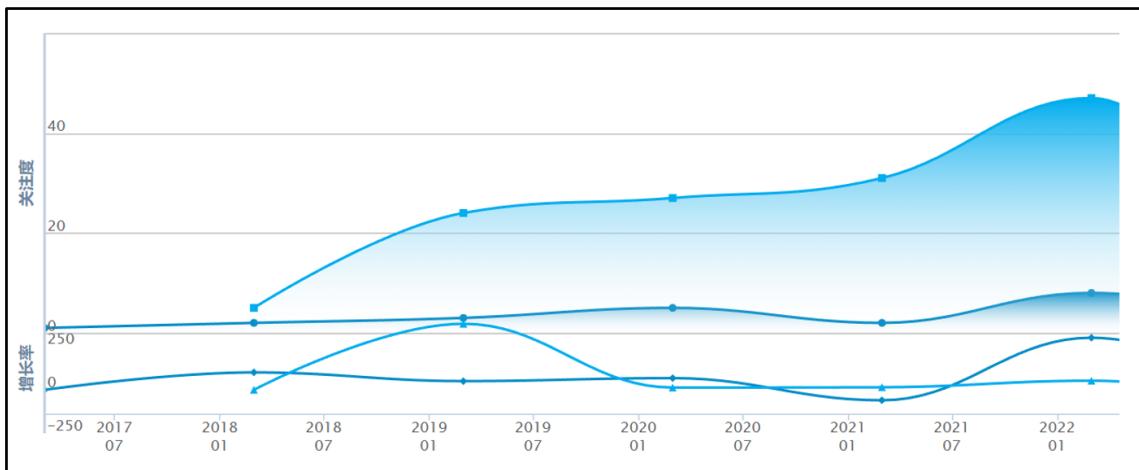


图 1-2 AutoML 学术界热点指数^[54]

如图 1-2 所示，AutoML 在学术界的研究热度逐年上升，在其发展浪潮中诞生了一批重要的里程碑式的工作。2006 年，Bergstra^[1]等人提出了一种基于随机搜索的自动机器学习方法，称为 Random Search。该方法通过随机采样超参数空间中的点，并对其进行评估，从而寻找最优的超参数组合。这一方法是 AutoML 领域中最简单的方法之一，但在实践中表现非常优秀。2015 年，Baker^[2]等人提出了一种自动化机器学习框架，称为 BAM（Bayesian Optimization and Meta-learning）。该框架将贝叶斯优化和元学习相结合，通过学习数据集的特征和模型之间的关系，自动选择最优的模型和超参数组合，在多个数据集上取得了优秀的性能。2016 年，Zoph^[3]等人提出了一种基于神经网络的自动化机器学习方法，称为 NAS（Neural Architecture Search）。该方法通过自动搜索神经网络结构来提高模型的性能，从而实现自动化机器学习。该方法在图像分类、目标检测等任务中取得了非常出色的结果。2017 年，Liu^[4]等人提出了一种基于进化算法的自动化机器学习方法，称为 ENAS（Efficient Neural Architecture Search）。该方法通过将搜索过程嵌入到神经

网络中，从而实现更加高效的神经网络搜索。该方法在多个数据集上取得了非常出色的结果。2019 年，Noy^[5]等人提出了一种基于自适应搜索的自动化机器学习方法，称为 ASAP（Adaptive Structure and Algorithm Searching for Neural Architecture and Hyperparameter），该方法通过自适应搜索算法，同时搜索神经网络结构和超参数组合，从而实现更加高效和准确的模型搜索。2020 年，Liu^[6]等人提出了一种基于强化学习的自动化机器学习方法，称为 DARTS(Differentiable Architecture Search)，该方法通过使用可微分的神经网络来搜索神经网络结构，从而实现更加高效和准确的神经网络搜索。

在 AutoML 的发展历史中，研究人员提出了许多不同的自动化机器学习方法，这些方法包括随机搜索、贝叶斯优化、元学习、神经网络搜索等。随着计算能力的提高和数据量的增加，这些方法的效果也不断得到改进和提高。目前 AutoML 的研究热点非常广泛，包括（1）自动神经架构搜索：作为该领域的重要研究方向之一，通过搜索算法在候选空间中找到最优的神经网络结构来实现自动化神经网络的设计和优化，以获得更好的性能和更高的效率。（2）硬件感知的 AutoML：随着人工智能在嵌入式系统和物联网设备中的应用越来越广泛，很多工作将研究重点放在把 AutoML 方法与低功耗嵌入式硬件结合使用上，以实现更高的效率和更好的性能，其中用到的主要方法以多目标优化为主。（3）AutoML 系统的可扩展性：在实际应用中，AutoML 系统需要能够扩展到大规模的数据集和多个任务。因此，较多的科研工作在探索如何开发可扩展的算法系统，以支持大规模的数据集和任务，其中主要涉及到如何设计高效等价的代理任务与代理模型。（4）片上自动化机器学习：片上自动化机器学习是一种新兴的研究方向，其目标是将机器学习模型和算法集成到芯片上，以实现更高的效率和更低的功耗。与云端的自动化机器学习模式相比，该方法不会受网络与用户隐私问题的影响。同时由于能实时获得硬件上的真实反馈，所以能极大提升模型的生成质量。本论文主要就是围绕设计高效的片上自动化机器学习算法展开。

1.2.2 工业界产品

在工业界，AutoML 的发展也非常迅速。越来越多的公司开始关注 AutoML，将其应用于各种实际场景中，从而提高业务效率和竞争力。一方面，很多互联网公司和科技公司已经开始在自己的业务中使用 AutoML，例如谷歌、微软、亚马逊、Facebook、腾讯等公司。这些公司都拥有自己的 AutoML 平台，能够帮助内部团队



图 1-3 AutoML 工业界产品

快速构建和优化机器学习模型。另一方面，越来越多的 AutoML 创业公司也在不断涌现。这些公司专注于开发 AutoML 产品和解决方案，提供自动化特征工程、模型选择、自动调参和模型部署等功能，同时也提供了可解释性机器学习、模型管理和数据隐私等解决方案。这些 AutoML 创业公司大多数都得到了资本市场的认可和投资，推动了其业务的快速发展。具有代表性的公司及产品有如下几个：

1. **Google Cloud AutoML:** 是谷歌云推出的一款自动化机器学习服务，旨在让开发者和数据科学家能够更轻松地构建和部署高质量的机器学习模型。它提供了多种功能，包括自动化特征工程、模型选择、自动调参和模型部署等。同时支持多种类型的机器学习任务，包括图像分类、自然语言处理和表格数据预测等。用户只需上传数据，选择相应的任务类型，AutoML 就可以自动化地完成模型训练、评估和优化等过程。同时，Google Cloud AutoML 还支持多种模型解释和模型监控的功能，帮助用户更好地理解和

管理机器学习模型。这款工具的优势在于其强大的自动化能力和云端部署的灵活性。用户可以通过简单的界面上上传数据和设置参数，就能够轻松地构建和部署高质量的机器学习模型。

2. Microsoft Azure AutoML：是微软云平台 Azure 推出的一款自动化机器学习服务，与 Google Cloud AutoML 类似，旨在帮助用户更轻松地构建和部署高质量的机器学习模型。相比之下，微软的这套工具支持更多种类型的机器学习任务，包括图像分类、文本分类、回归分析、时间序列预测和机器翻译等。其次，它也支持更多的机器学习框架和算法，如 TensorFlow、PyTorch、Scikit-Learn 等，使用户可以根据自己的需求选择合适的算法进行模型训练和优化。另外该产品还提供了可解释性机器学习和模型管理的功能，使用户可以更好地理解和管理机器学习模型，帮助用户快速进行数据清洗、特征工程和模型评估等过程。
3. H2O.ai：作为一家致力于推动自动化机器学习技术发展的公司，其产品 H2O 提供了自动特征工程、模型选择、自动调参等 AutoML 功能，并支持 Python、R、Java 等多种编程语言，可在本地或云端部署。
4. 第四范式：第四范式的 AutoML 产品主要包括两个方面，一个是 Auto-Model，另一个是 Auto-Deploy。其中，Auto-Model 是一个自动化机器学习平台，支持多种机器学习任务。Auto-Model 通过自动特征工程、模型选择和自动调参等技术，帮助用户快速构建和部署高质量的机器学习模型。另外，Auto-Deploy 是一个自动化模型部署平台，能够快速将训练好的模型部署到用户的应用中。

可以发现，工业界的平台大多以提供云端训练加本地部署的方式进行模型的落地。云端的 Automl 平台的优势在于较高算力的资源，但是在考虑硬件感知型的自动化机器学习任务中，由于其无法获得真实的硬件信号反馈，云端平台通常会考虑用模型的大小作为衡量部分硬件指标如延时、功耗等的代理信号，从而进行模型的自动化生成，导致最终模型的生成质量较低。同时在某些比较注重模型隐私或者无法联网的场景中，这种云端提供服务的模式根本无法使用。与之相对的，片上自动化机器学习能实时获得片上反馈的真实信号，提升模型生成质量。同时也因为无需联网，不会受到网络波动或者是隐私问题的影响。

1.3 论文研究的范围

本文的研究目标是实现一套高效的片上自动化机器学习算法，具体而言是同时支持网络结构搜索与模型量化的高效片上优化算法，针对的目标硬件包括自研

Evolver 芯片^[7]、GPU、CPU。考虑到片上自动化机器学习面向的场景通常是需要做硬件感知类的多目标优化任务，同时为提升算法的可扩展性，本文提出的算法需要支持以下三个方面特征：（1）支持高效的多目标优化任务，即能在各种优化条件的取舍中选择符合当前任务的最优策略，具体来说能在模型的预测精度、延时、功耗、存储空间等维度进行同时优化（2）针对网络结构搜索与模型量化这两个分离的任务，提供一套支持联合优化的方法从而提升模型整体的优化效果，更重要的是降低两阶段优化带来的较大的时间开销（3）在更复杂的离线任务场景中，支持多节点的联合优化，在联邦网络结构搜索的任务场景中有着较高的通信效率。

1.4 论文研究的难点

从上面的研究背景与当前学术以及工业界的发展情况可以看出，当前这种云端自动化机器学习主要面临隐私问题以及无实时硬件反馈导致的硬件感知类模型生成质量低下。而片上自动化机器学习能很好解决这两个问题，其最大的挑战是如何在有限的计算资源下设计一种高计算效率的算法。以本文涉及到的网络结构搜索和模型量化这两个子领域出发，从多目标优化、多领域联合优化、多节点联邦优化三个角度去实现一套高效的片上自动化机器学习算法，主要的难点包括：（1）多目标优化过程中不同优化目标的组织形式：对于多目标优化问题，奖励函数是引导搜索算法方向的关键，通过设计合理的损失函数对整个 Automl 算法的稳定性起着重要的作用。（2）多领域联合优化问题的高效迭代：由于多领域优化涉及的优化维度较多，采用常规的搜索策略会导致整个算法复杂度过高，需要较多的训练轮数才能达到收敛的地步。因此设计一种高效的联合搜索框架对于整体的算法的效率提升十分关键（3）多节点优化的通信量缩减：在多节点联邦优化的场景中，节点之间通信开销（非云端通信）成本较高，尤其在本地需要运行局部 AutoML 时，单次的迭代需要大量的通信成本。因此设计合理的权重聚合策略，减少迭代总次数以及单词迭代通信开销对降低通信成本非常重要。

1.5 论文内容的组织结构

本论文主要研究片上自动化机器学习算法，从网络结构搜索、自动模型量化两个角度切入实现三个维度的优化工作，包括：多目标优化、多领域联合优化、多节点优化，针对的目标硬件包括自研 Evolver 芯片、CPU、GPU。首先针对同时优化预测精度、延时、功耗的硬件感知型任务场景，本文提出了一套高效的基于值迭代的强化学习搜索策略，为进一步缩减迭代次数采用了各类优化技巧减少计算量。其

次，针对多领域联合优化任务，本文以网络结合与混合精度量化策略为例，提出了一种高效的一阶段优化方法，最终在轻量级卷积神经网络的设计上比目前传统的模型提升了近一半的推理速度。最后针对节点同时优化的联邦学习场景，本文提出了一种利用选择性权重聚合、超核搜索、混合精度比特共享三个维度对原先的联邦网络结构搜索框架进行了升级，大幅减少了通信量。研究内容具体描述如下：

1. 多目标优化：本文以混合精度量化与电压频域联调为任务场景，提出了一种基于值迭代的高效 Q-Learning 搜索算法，利用奖励塑造与异常点跳出等优化手段，将原始的搜索算法的复杂度进一步降低。针对自研的 Evolver 可重构 AI 训练芯片，该算法能在有限的时间内快速搜索出符合条件的 Q-V-F（量化-电压-频率）策略，从而适应不同任务场景中对模型量化位宽以及芯片工作状态的需求，达到软硬件协同演化的目的。相关内容在第二章节。
2. 多领域联合优化：本文以轻量级卷积网络的网络结构搜索与混合精度量化为基本的任务场景，提出了一种基于可微分网络结构搜索与基于量化感知训练的量化模式。该方法能同时有效搜出符合硬件特点的网络结构与量化位宽策略，并利用帕累托奖励函数在多个优化目标中做出合理的取舍，从而使模型能够满足不同场景对精度与延时的需求。最后本文将算法运用在终身学习的场景中，通过对多阶段 ImageNet 任务的训练权值最终的数值对比，本方法可以逼近原始单阶段训练任务中模型的权值，进一步说明了方法的可扩展性。相关内容在第三章节。
3. 多节点联合优化：本文以联邦学习中最著名的 FedAvg^[25]为参考的基本联邦学习框架，探讨在搜索节点异构的模型任务中通信量缩减的问题。本文提出了一套利用选择性权重聚合、超核搜索、混合比特参数共享为基础的通信量优化手段。最终与传统的搜索算法相比，本文的方法整体获得了 77% 的通信量缩减，并且通过设置不同节点上的帕累托奖励函数，能生成符合要求的节点异构的网络模型。相关内容在第四章节。

第二章 面向片上量化-电压-频率的多目标优化

第一章对自动化机器学习算法的背景、必要性以及国内外研究现状做了总结性的介绍。从本章开始，论文将详细介绍作者在片上自动化机器学习的多目标优化任务中做的创新研究型的内容。本章介绍的内容，主要以作者在集成电路领域顶刊 2021 JSSC (Journal of Solid-State Circuits) 上已发表的工作^[7]为基础进行展开。

2.1 本章引言

随着时代的发展，深度学习处理芯片在智能设备中已变得无处不在。当前 AI 模型部署到终端上的常用方法是，先在云端对神经网络模型进行训练，然后将训练好的模型部署到终端 AI 芯片上进行推断。终端 AI 芯片的能力包括神经网络的准确度、延迟、能耗以及模型所能执行的智能任务等，其主要受限于其部署的神经网络模型。由于在部署后模型的性能就已经固定不变了，很难根据不断变化的终端环境进行调整。这导致了模型准确度的下降，延迟和能耗无法满足应用需求等问题。如果数据上传到云端进行新网络的训练再进行部署，就会带来用户隐私泄露的风险。因此，如何高效地将深度神经网络 (Deep Neural Networks 即 DNN) 模型部署到不同的深度学习处理器上已经成为一个重要的研究课题。对于一个 DNN 模型，人们通常在准确度、延迟和能耗之间进行权衡，通过逐层的混合精度量化和硬件的电压-频率调整来实现各方指标的取舍。在算法方面，混合精度量化可以通过降低数据精度以获得较低的延迟和能耗，并在可接受的准确度损失范围内获得正常的推理效果^{[26][27][28][29]}。在硬件方面，通过增加处理器的工作电压和频率来实现较低的延迟，或者降低电压和频率以实现较低的能耗^{[30][31][32]}。本章节将确定量化位宽、电压和频率 (Quantization、Voltage、Frequency 即 QVF 策略) 以进行硬件部署的过程定义为 QVF 调优，最终实现在不同任务场景中精度、延迟、能耗的多目标模型优化，问题可以表述为：

$$\max_{Q,V,F} \quad f(A, L, E) \quad (2-1)$$

目标函数是准确度 (Accuracy)、延迟 (Latency) 和能量 (Energy) (即 ALE) 的函数 $f(A, L, E)$ ，其中 A 、 L 、 E 是由 Q 、 V 、 F 共同决定的。

传统的方法通常采用离线的方式来分别调整量化位宽 (Q) 和电压频率 (VF)，如图 2-1 所示：(1) 首先根据模型大小或计算量作为约束的代理信号，来进行混

合精度量化^[33]。（2）然后基于专家经验以及仿真结果，为某一特定的目标（最小延迟或能耗）设置适当的电压和频率^{[30][31][32]}。

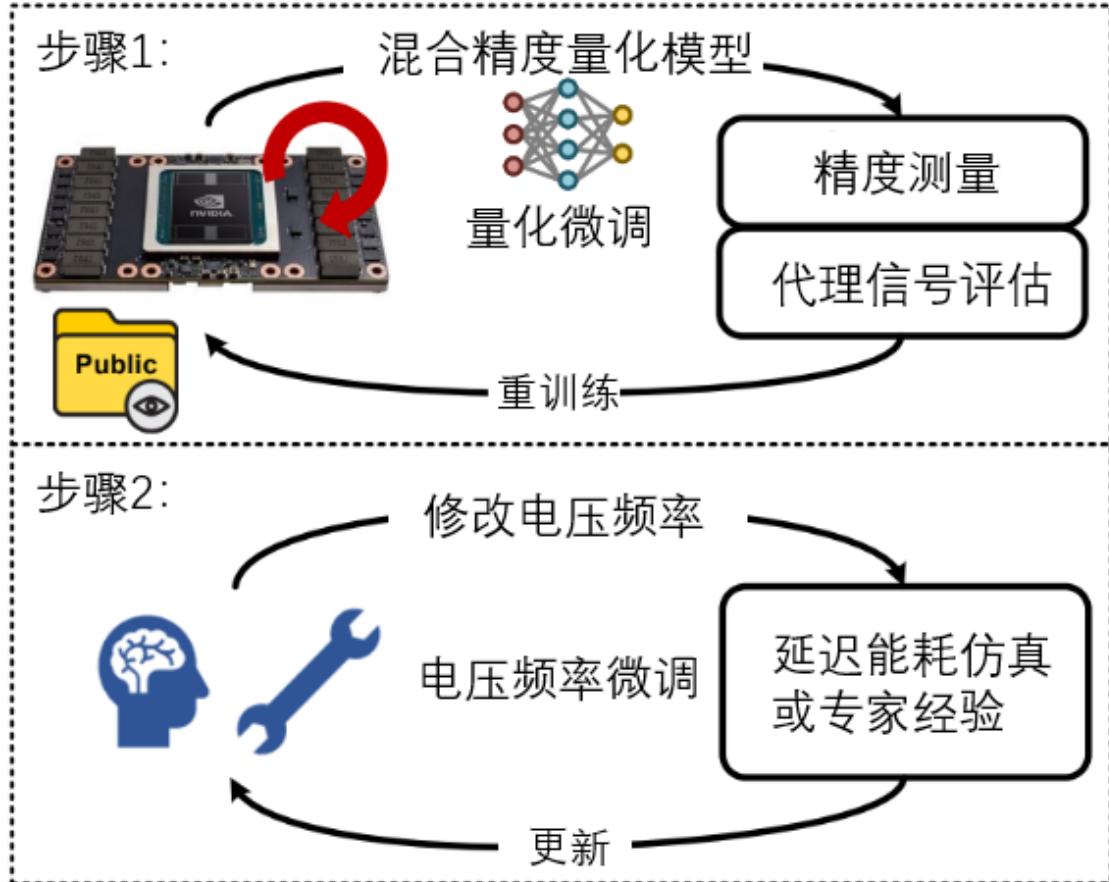


图 2-1 离线 QVF 调优框架

然而，离线的 QVF 调优存在三个限制。（1）首先是次优性：代理信号无法准确反映硬件行为。比如将模型大小作为衡量模型推理延迟的代理信号时，较小的模型并不能保证较低的延迟或能耗^[33]。这是因为实际的模型推理性能取决于真实的硬件行为，例如内存访问和计算数据流，这些行为过于复杂，无法通过代理信号准确表达。同时，由于 QVF 都对延迟和能量产生影响，独立调整 Q 或者 VF 会导致整个搜索空间中的局部最优解。所以从整体的角度来看，这种调优策略最终找到的结果是次优的。为获得最优的 QVF 策略，需要通过实际的硬件 ALE 反馈和耦合的 QVF 调优来引导最优搜索。（2）其次是定制化的限制：离线 QVF 调优存在定制化问题，因为生活中存在数十亿个边缘设备（如智能手机和物联网设备），其硬件架构各不相同，在模型部署之前为每个设备调整 QVF 是不现实的。此外，部署目标 $f(A, L, E)$ 在不同的应用场景甚至用户偏好中存在显著差异，进一步增加了定制

化的需求。（3）最后是传输成本及隐私问题：如果深度神经网络（DNN）模型需要重新量化以适应本地情况，个人数据需要上传至云端进行重新训练。这将导致高昂的传输成本以及隐私风险^[34]。最近有一些研究在模型设计^{[35][36]}或量化^[33]中引入了硬件反馈来解决优化问题。然而，如果训练数据来自云端的公共数据集，这些方法将在本地环境中失去准确性，无法适应环境变化。如果训练数据是由本地设备收集并发送到云端，就无法避免传输成本和隐私问题。

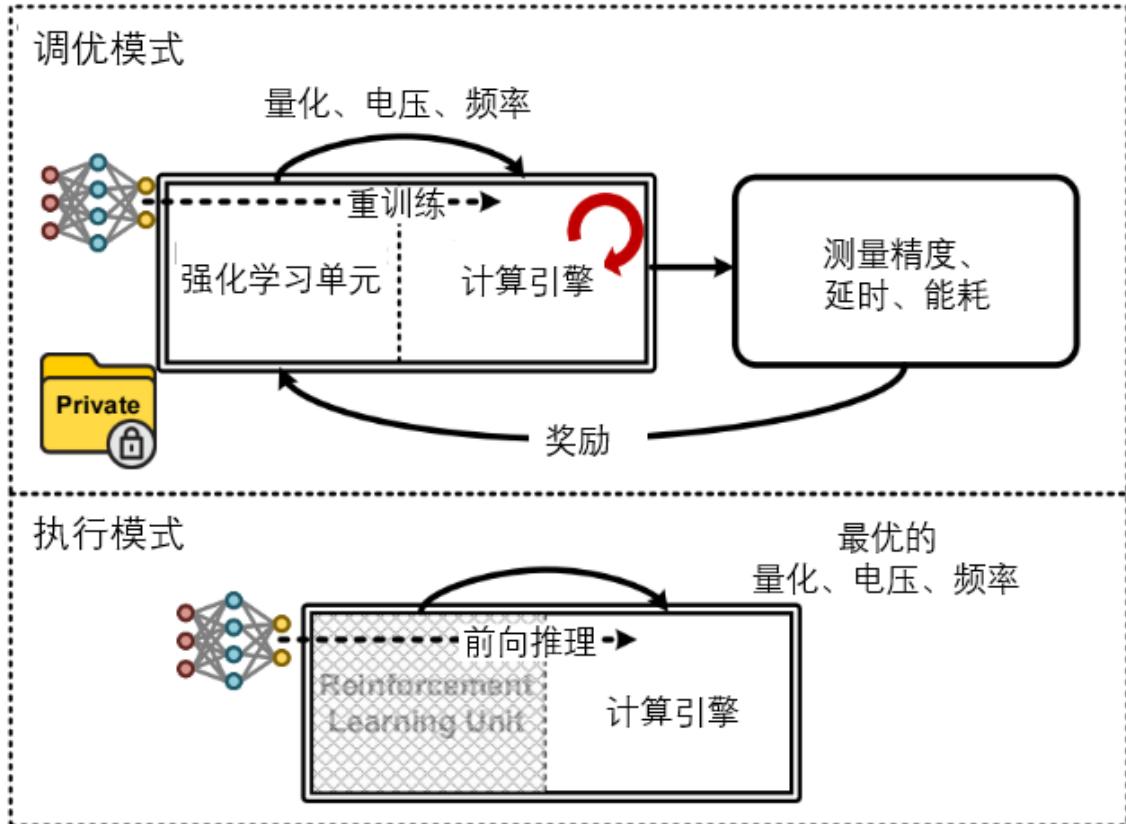


图 2-2 片上 QVF 调优框架

基于上述考虑，本文针对课题组自研的深度学习处理器架构 Evolver 架构，实现了一套片上的 QVF 多目标调优算法。该算法在 Evolver 中的具体实现逻辑如下：向芯片添加了一个 QVF 调优模式，在执行之前为 DNN 模型提供本地定制；基于芯片的直接 ALE 反馈，该算法使用强化学习（Reinforcement Learning 即 RL）来搜索最优的 QVF 策略。如图 2-2 所示，Evolver 有两种工作模式：（1）在调优模式下，原始的 DNN 模型被加载到芯片中。强化学习单元根据 Evolver 的反馈搜索最优的 QVF 策略。强化学习的奖励是根据本地设备、应用场景和用户偏好中提取的 ALE 约束进行构建的，因此搜索方向会被引导到定制的目标上。在每轮搜索迭代中，Evolver 的计算引擎使用本地数据集重新训练混合精度模型。根据采样的策略

调整 Evolver 的工作电压和频率，并计算 ALE 结果以得出奖励值。（2）在执行模式下，根据获得的最优 QVF 策略，在对应的量化策略下进行感知训练并在所选的电压和频率下进行前向推理。本章节将着重围绕该多目标优化的算法进行展开。

2.2 基于Q-Learning的量化-电压-频率联合搜索算法

已经有许多关于 DNN 模型量化方面的研究工作^{[26][27][37]}，激活值和权重可以在几乎没有精度损失的条件下被量化为低精度，如 8 比特或 4 比特。对于一些准确性要求较低的使用场景，甚至有工作将 DNN 模型量化为 2 比特甚至 1 比特。通过降低量化位宽，可以有效降低数据传输成本，并以有限的精度损失为代价，使得能在片上计算和内存访问方面实现更低的延迟和能耗。同时，也可以通过调整电压和频率来在延迟和能耗之间进行权衡。量化、电压和频率（QVF）以不同的方式影响准确性、延迟和能耗（ALE），因此调整 QVF 以实现最佳的部署非常重要，并且实际的部署策略应根据具体的硬件设备、场景和用户偏好进行定制。在本章算法中，QVF 调整被表示为：

$$\begin{aligned} \max_{Q,V,F} \quad & f(A, L, E) = a \cdot A - b \cdot L - c \cdot E \\ \text{s.t.} \quad & A > A_{TH}, L < L_{TH}, E < E_{TH} \end{aligned} \quad (2-2)$$

其中 $(a \cdot A - b \cdot L - c \cdot E)$ 为衡量模型生成质量的综合目标，通过加权系数 a 、 b 、 c 对 ALE 三个评价指标在硬约束条件（阈值 A_{TH} 、 L_{TH} 、 E_{TH} ）下进行不同程度的取舍，这种方法也被谷歌在其多目标神经架构搜索（MONAS）^[38]的框架中采用。例如，一个能耗优先的场景，要求 top-1 准确度不低于 68.5%，可以设置 $a=0.25$, $b=0.1$, $c=0.65$ ，并将 A_{TH} 设置为 68.5%。在公式 (2-2) 中，ALE 的值被归一化到 [0, 1] 范围内，使这三个值具有相似的尺度。所有探索出来的 QVF 策略的 ALE 结果都是在实际芯片上直接测量的，利用这些真实硬件的反馈信号能使得算法朝着最优的方向进行。同时，加权系数 a 、 b 、 c 以及阈值 A_{TH} 、 L_{TH} 、 E_{TH} 可以根据特定的应用程序或用户需求进行定义。

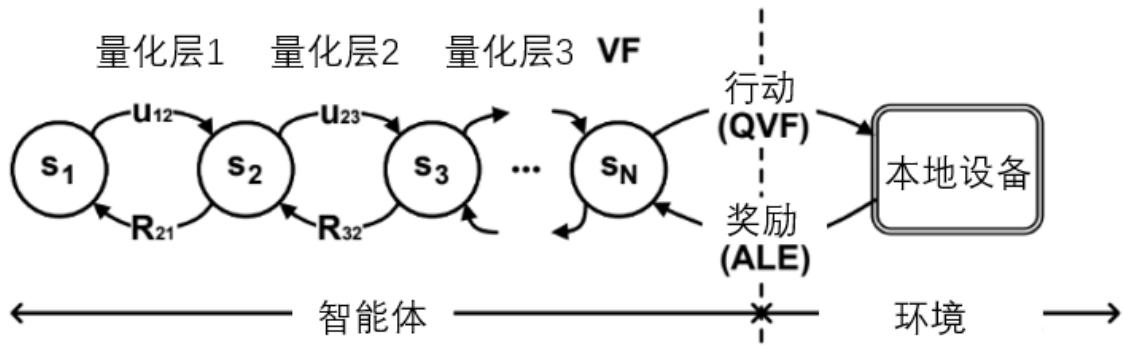


图 2-3 基于 Q-Learning 的 QVF 调优原理

2.2.1 算法顶层结构设计

图 2-3 展示了强化学习单元(RLU)基于 Q-learning 的 QVF 调优的基本原理，将 QVF 调优视为学习一个最优决策路径，选择的流程可近似建模为一个马尔可夫决策过程。类似于 MetaQNN^[40]中利用 Q-learning 探索 DNN 架构的方法，依次选择每个层的量化策略、工作电压和频率，整个 QVF 策略由 s_1 到 s_T 的状态转移路径决定。其中前 $T-1$ 个状态被定义为每个 DNN 层的激活值-权重值对的元组(状态编号=层编号)，最后一个状态被定义为工作电压-频率对的元组。表 2-1 是模型 VGG8 调优的一个示例状态空间，其中每个层的激活值与权重都可以量化为 2/4/8 比特，状态编号 $T-1$ 等于总层数 8，电压和频率则从自研的 Evolver 芯片上测量到的 9 个离散的典型工作点中选择。可以看出在本算法中，每个阶段 s_i 的行动空间等同于其下一个状态 s_{i+1} 的状态空间。

状态	状态参数	参数值
1 ~ 8	激活值位宽 权值位宽	$\in \{2, 4, 8\}$ $\in \{2, 4, 8\}$
9	电压、频率 (V, MHz)	$\in \{\{0.75, 120\}, \{0.77, 160\}, \{0.80, 232\}, \{0.85, 248\}, \{0.90, 260\}, \{0.95, 261\}, \{1.00, 262\}, \{1.05, 264\}, \{1.10, 268\}\}$

表 2-1 Evolver 上 VGG8 的搜索空间示例

如图 2-3 所示，代理程序依次根据表 2-1 中 Q 值的大小来选择每一步的决策动作 u ，直到达到最终状态 s_T 。芯片本身充当了强化学习的环境，评估采样的 QVF 策略的并得出 ALE 结果作为奖励函数来指导迭代的方向，其中奖励由 ALE 的联合函数式(2-2)中定义，主题流程如算法 2-1 所示。对于状态 s_i ，以及当前选择的动作 u ，通过标准的 Q-Learning 公式进行更新：

$$Q_{i,u} = Q_{i,u} + \alpha \cdot [Reward + \gamma \cdot MAX(Q_{i+1}) - Q_{i,u}] \quad (2-3)$$

其中， $MAX(Q_{i+1})$ 表示当前时间步骤下，状态 s_{i+1} 的所有动作中的最大 Q 值。其中重要的两个更新参数：（1） α 是表示新信息相对于旧信息的学习率权重； γ 是折扣因子，确定对短期奖励相对于未来影响^[41]。

Algorithm 1: 多目标量化、电压、频率联调 (QVF) 优化

输入：最大搜索轮数 N_{max} ，每轮训练的 $epoch$ 数量 E_{max} ；本地训练集 D^{train} ，测试集 D^{test} ，学习率 η 以及定制化加权系数 (a, b, c) ； Q 表更新学习率 α 与折扣因子 γ
输出：模型 P ，包含权重 W ，每层量化位宽 Q_i 和芯片工作电压与频率 V, F

```

预训练原模型得到初始权重  $W_0$ ，初始化  $Q$  表  $T_Q^0$ 
for  $n = 1, 2..N_{max}$  do
    根据  $\epsilon-greedy$  从当前  $Q$  表中采样出一条 QVF 策略包含每层的量化位宽
     $Q_i^n$  和芯片工作电压与频率  $V^n, F^n$ ；
    根据量化位宽与随机输入  $s$ ，测量推理延迟  $L$  与能耗  $E$ ；
    if  $E, L$  非异常点 then
        for  $e = 1, 2..E_{max}$  do
            根据量化位宽  $Q$  进行量化感知训练对  $W$  进行梯度下降：
             $w \leftarrow w - \eta \nabla_w L(batch(x, y), Q)$ 
        end
        根据  $w$  在测试集  $D_{test}$  上获得精度信息  $A$ 
        根据公式 2-2 由  $A, L, E, a, b, c$  计算出 Reward
    end
    else
         $| Reward \leftarrow 0$ 
    end
    根据公式 2-3、2-4 由  $Reward, \alpha, \gamma, T_Q^{n-1}$  计算  $T_Q^n$ 
end
由贪心策略根据最新  $Q$  表  $T_Q^{N_{max}}$  选出对应的每层量化位宽 ( $Q_i$ ) 与工作电压、
频率 ( $V, F$ )

```

算法 2-1 多目标量化、电压、频率联调算法

在本任务场景中， Q 值的更新按照时间上的逆序进行如下所示：

$$\begin{aligned} Q(s_T, a) &= 0 \\ Q(s_{T-1}, a_T) &= (1 - \alpha)Q(s_{T-1}, a_T) + \alpha r_T \\ Q(s_t, a) &= (1 - \alpha)Q(s_t, a) \\ &+ \alpha[r_t + \gamma \max_{a'} Q(s_{t+1}, a')], t \in \{1, 2, \dots, T-2\} \end{aligned} \quad (2-4)$$

即从 s_T 到 s_1 ，这已经被证明可以加快 Q 值的收敛速度^[42]，具体的更新方式如式(2-4)所示包括了首尾边界，其中 a_t 对应式(2-3)中的 u ， s_t 对应第 i 个时间片段的序列状态。

2.2.2 迭代优化技术

鉴于有限的计算资源，提高模型的搜索效率尤为关键，本工作结合了多目标优化的特点并借鉴了强化学习任务有效的提速手段，使用了下面两种优化技巧。

1. 奖励塑造：作为在强化学习中使用的技术，旨在通过改变奖励信号的方式来加速智能体的学习过程。在有时候环境的奖励信号可能很稀疏或者不明确，这会导致学习过程变得缓慢或者困难。奖励塑造的目的是通过引入附加的奖励信号，使得智能体能够更快地学习到有效的策略。这些附加的奖励信号可以基于先验的知识或者问题的结构来设计，以提供更多的指导和反馈。在本问题中，由于单个的动作不存在瞬时的奖励，只有在完成一轮策略选择后，通过对应的 ALE 计算才能得到一条序列的最终奖励。如此系数的奖励数值不适合强化学习的收敛，本算法将式(2-4)中序列的最终奖励 r_T 分摊至每个时间片上：

$$r_t = \frac{r_T}{T} \quad (2-5)$$

其中 r_t 为最终通过 ALE 计算出的最终奖励信号，通过改变奖励信号的形状，使得将原始的复杂任务分解为更容易解决的子任务，从而加速学习过程。

2. 异常点过滤：如图 2-4 所示，所有 QVF 策略形成了一个大的三维搜索空间，三个维度分别对应精度、延迟、能耗（ALE）。在这么大的空间中对所有点进行探索将会非常昂贵，本算法根据具体的应用场景和用户偏好，在式（2-2）中预先设置阈值 L_{TH} 和 E_{TH} 作为延迟和能耗的约束，对于不能满足 L_{TH} 和 E_{TH} 约束条件的策略所形成的非合理空间（图 2-4 右上中的灰色区域）采取直接跳过的策略，以避免对异常点进行不必要的训练。对于每个策略，经过重新量化的 DNN 模型将直接在 Evolver 上运行，若不能满足 L_{TH} 和 E_{TH} 约束条件，不需要训练来测量其在测试集上的精度，直接将奖励设为 0。在第 2.4 节中以能量为优先的 VGG8 调整实验中，这项技术可以跳过 37% 的异常点。

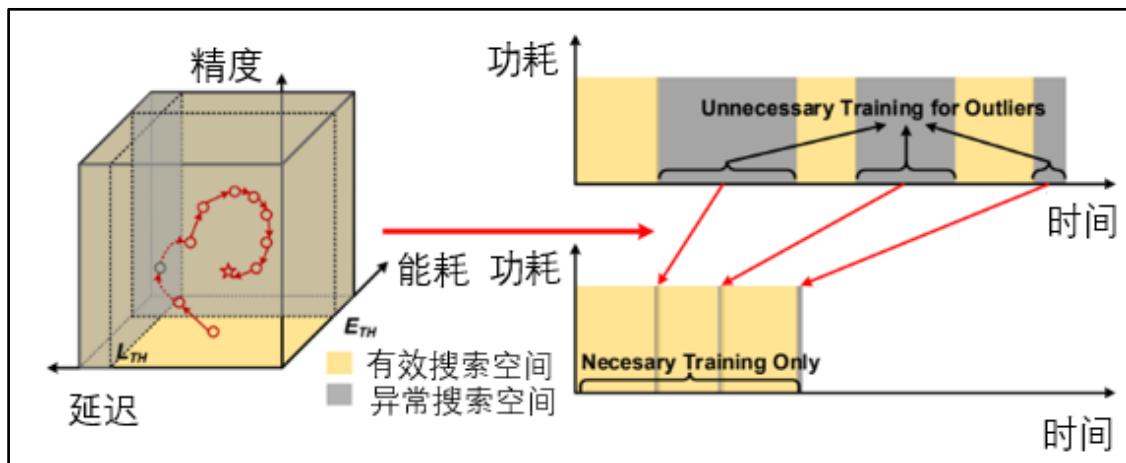


图 2-4 异常点过滤原理

2.3 RLU模块设计

针对以上基于 Q-Learning 的 QVF 调优模块，本小节提供了对应的详细的架构设计。如图 2-5 所示，芯片中采用了 8KB 大小的 SRAM 来存储 Q 表，具有 16 个 bank，每个 bank 有 256 行，单数据位宽为 16 比特，用于放置逐行存储的 Q 值。即最大状态数为 256，每个状态最多可以有 16 个动作。这样的设计具有很高的灵活性，可以通过重写 Q 表来支持不同的 DNN 结构和 QVF 任务调整。例如，表 2-1 中的示例总共有 9 个状态，其中前 8 个状态用于层级量化，每个状态有 9 个动作，而最后一个状态用于 VF 选择，也有 9 个动作。相应的 Q 表如图 2-6 所示。每个 QVF 策略形成了从第 1 行到第 9 行的选择路径。主要的模块由以下三个组成：

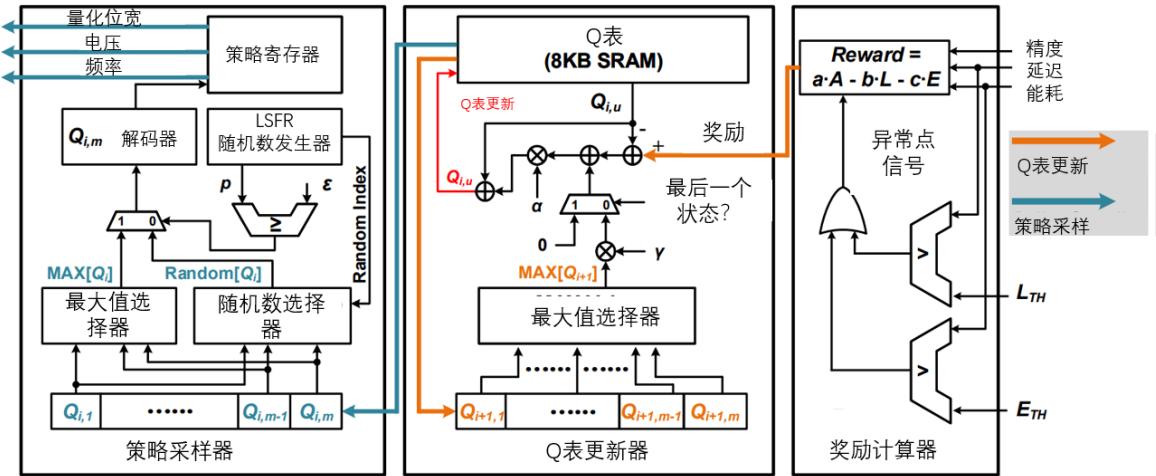


图 2-5 RLU 架构设计

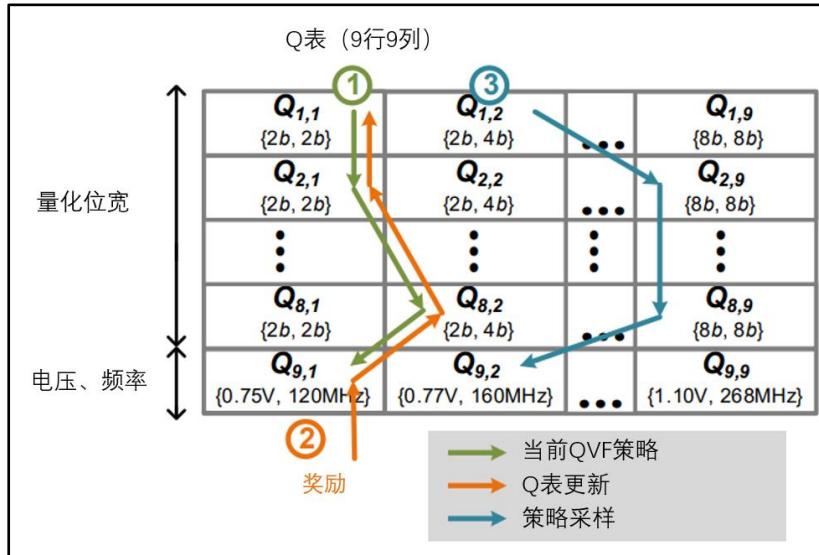


图 2-6 RLU 模块工作流

1. 奖励计算器 (Reward Calculator 即 RC)：对于一个抽样的 QVF 策略，Evolver 将重新训练 DNN，并将测得的 ALE 结果传递给奖励计算器 (RC)。如 2.2 小节所述，加权系数 a 、 b 、 c 和阈值 A_{TH} 、 L_{TH} 、 E_{TH} 是由具体的的应用程序或用户事先定义并存储在 RC 中的。ALE 值都被归一化到 $[0, 1]$ 的范围，并以 16 位格式表示。
2. Q 表更新器 (Q-Table Updater 即 QTU)：如图 2-5 所示，一旦计算得到一个奖励，当前策略选择路径上的 Q 值将按照相反的顺序 (从 s_N 到 s_1) 进行迭代更新。在 QTU 中，当前策略的 Q 值通过式 (2-4) 将当前 Q 值、奖励和下一行的最大 Q 值结合进行更新。为确保 Q 值的收敛性，在 Q 表更新期间，所有的数据和操作都以 16 位精度进行处理。

3. 策略采样器 (Policy Sampler 即 PS)：在 Q 表更新之后，策略采样器 (PS) 根据 ϵ -greedy 策略^[42]从 Q 表的每一行中选择一个动作，形成一个新的策略。如图 5(a)所示，这是通过比较参数 ϵ 和基于 8 位线性反馈移位寄存器 (LSFR) 的随机数生成器 (RNG) 的输出来实现的。更直观的，PS 以概率 ϵ 选择一个随机动作，并以概率 $1-\epsilon$ 选择一个贪婪动作（具有最大的 Q 值）。按照上述顺序选择的 Q 值将作为新的 QVF 策略存储在对应位置的 Q 表中。在整个 QVF 调优过程中，变量 ϵ 从 1 慢慢变为 0，使强化学习逐渐从探索过程转为对已有数据的利用过程^[40]。

2.4 算法评估

这一小节将首先描述基于 28 纳米芯片 Evolver 的调优实验的验证系统搭建，并将本方法与两种基准搜索策略进行了对比，以证明本方法在最优性、可扩展性、长久收益方面的优势。

2.4.1 验证系统搭建

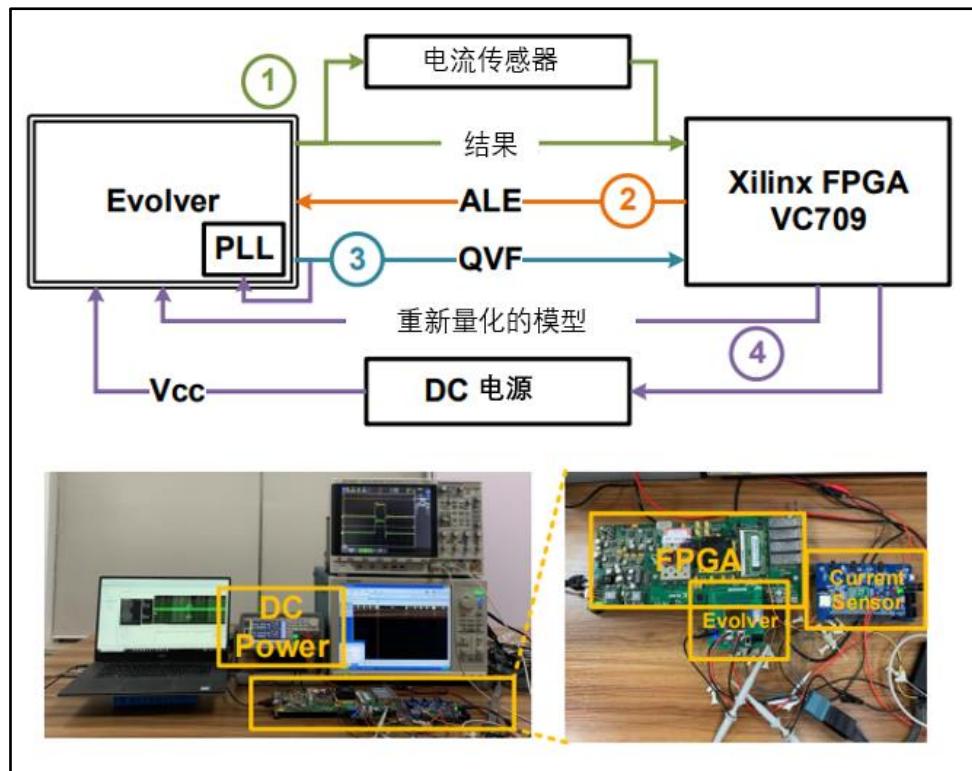


图 2-7 QVF 调优验证系统

图 2-7 展示了本节的 QVF 调优实验的验证系统。该系统由 Evolver 芯片、电流传感器、直流电源和 Xilinx VC709 FPGA 主板作为主机处理器组成。作者对 VGG8

的 QVF 调优实验进行了深入研究，VGG8 模型先在 CIFAR100 数据集上进行了均匀的 INT8 精度的量化感知训练。为了模拟 Evolver 的本地私有数据集，作者随机选择了 6000 个 CIFAR100 图像，并对它们添加高斯噪声，其中 5000 个用于本地训练，1000 个用于测试。这个本地私有数据集和 VGG8 模型存储在 FPGA 板的 DDR3 内存中。Evolver 通过 FPGA 中介连接器（FMC）与 FPGA 通信。中间涉及四个主要步骤。（1）对于每个采样的 QVF 策略，FPGA 收集在测试集上的推理结果并监测通过电流传感器的电流。同时在 FPGA 中实现了一个计时器，用于测量推理延迟。能量的消耗则通过对延迟、感测到的电流和供电电压在 FPGA 上计算得出。（2）然后将 ALE（准确率、平均延迟和每次推理的能耗）结果发送回 Evolver 进行奖励计算。（3）Evolver 采样一个新的 QVF 策略并将其发送到 FPGA。（4）FPGA 重新量化模型并将其发送到 Evolver 进行重新训练。在重新训练期间，Evolver 的工作电压和频率设置为最大值（1.10V, 268MHz）以加快训练速度。在推理过程中，根据频率策略重置 Evolver 的 PLL，并使用 FPGA 通过 RS232 接口控制直流电源。因此，直流电源可以模拟可编程 LDO 来根据电压策略改变 Evolver 的电压 V_{cc} 。

2.4.2 三种搜索方法设定

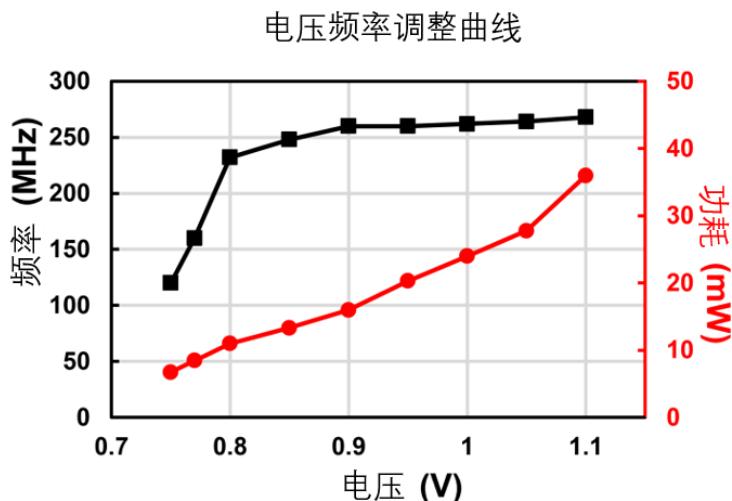


图 2-8 Evolver 电压频率曲线

为简化实验复杂度，作者通过将激活值的位宽固定为均匀的 8 比特来简化搜索空间，以帮助搜索在较短的时间内收敛，验证算法可靠性。对于权重量化，每个层有 3 个选项（2/4/8 比特），而电压和频率则是从图 2-8 的电压-频率缩放曲线中选择的 9 个操作点中选择的。因此，整个搜索空间包含 $3^8 \times 9 = 59049$ 个候选的 QVF

策略。在这个实验中，作者在验证系统上比较了三种不同的搜索方法，以评估本章节提出的 QVF 调优的有效性。

ϵ	1	7/8	6/8	5/8	4/8	3/8	2/8	1/8	0
迭代次数	90	+30	+30	+30	+30	+30	+30	+30	+30

表 2-2 每个 ϵ 阶段的搜索迭代次数

1. 基于硬件反馈信号的能量优先的强化学习搜索（本章节的方法的特例）：在此场景中，任务为寻找一个优先降低能耗的策略。在本算法中，仅需要通过设置式 (2-2) 中的奖励函数中的参数 $a=0.25$, $b=0.1$, $c=0.65$ ，并添加 ALE 约束 ($A_{TH}=68.5\%$, $L_{TH}=20$ 毫秒, $E_{TH}=300$ 微焦耳)。这些参数被加载到 Evolver 的 RLU 中，以指导 QVF 调优。RLU 构建了一个由 9 个状态族组成的空间，如表格 2-1 所描述，其中前 8 个状态用于逐层量化，最后一个状态用于电压和频率。本实验中，公式 2-4 中的 Q 学习率 α 设置为 0.25，折扣因子 γ 设置为 0.99，以避免过度优先考虑短期奖励。至于在 2.3 小节中介绍的 ϵ -greedy 策略，初始值为 1，前 90 次迭代保持不变，然后每 30 次迭代减小 1/8（见表 2-2）。与 MetaQNN^[40]类似，本工作的 QVF 调优从随机探索开始，最后以最贪婪的利用策略结束（在最后 30 次迭代中 $\epsilon=0$ ），因此最大迭代次数为 330 次。
2. 网格搜索（参照一）：这组参照实验中使用典型的穷举搜索方法即网格搜索来寻找在 59049 个候选选项中的全局最优 QVF 策略。对于参照实验一和实验二，为快速获取量化模型的准确性，训练是在 NVIDIA GPU 服务器 DGX-2 上进行的，GPU 型号 V100，而推理的延迟和能量消耗是在 Evolver 上进行测试的。
3. 采用模型大小作为代理信号的强化学习搜索（参照二）：这个参照实验代表了依赖代理信号而不是直接硬件反馈的搜索方法。由于 DNN 模型大小经常被用作混合精度量化压缩程度的指标^{[26][27]}，作者将其作为奖励而不是 Evolver 的 ALE 反馈，以指导最佳量化策略的搜索，从而来探索代理信号对生成模型质量的影响。因为电压和频率对模型大小没有影响，所以它们在量化策略确定后通过手动进行设置。

2.4.3 消融对比分析

为进一步分析本章节方法在搜索策略上的优越性，分别对两组参照搜索算法

进行了消融对比分析：

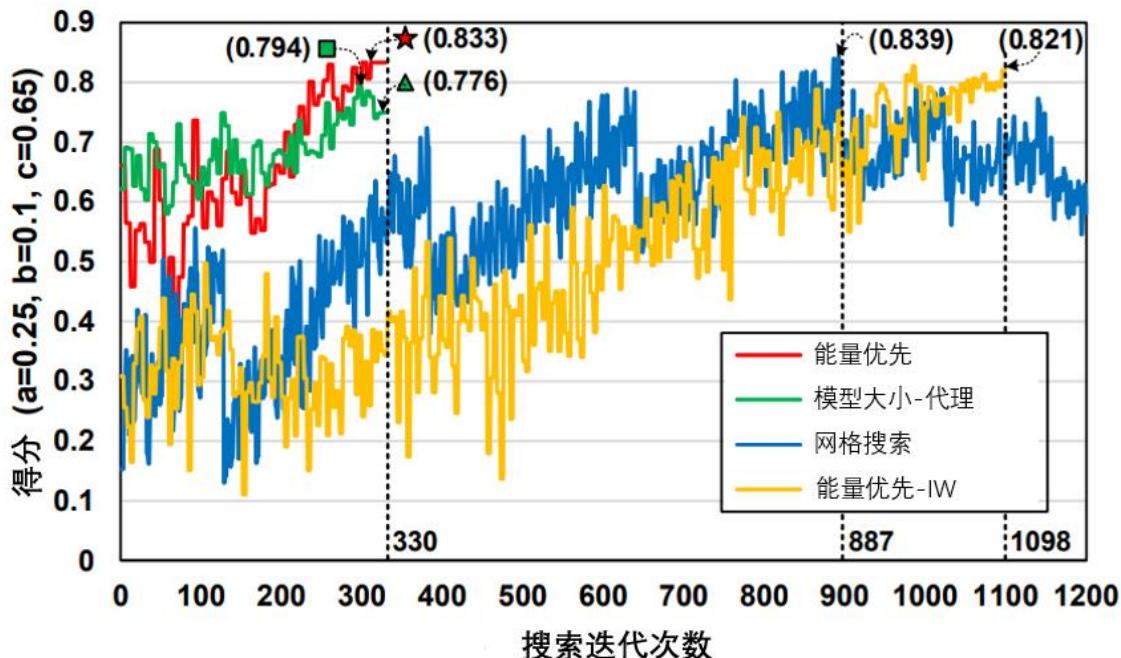


图 2-9 VGG8 在本地数据集上不同方法的 QVF 调优曲线

	模型大小-代理		能耗优先 ★	延迟优先
	▲	■		
第一层	2	8	2	4
第二层	8	4	4	8
第三层	8	4	8	8
第四层	8	8	8	8
第五层	8	8	4	4
第六层	4	8	8	4
第七层	4	4	4	8
第八层	4	8	8	8
V, F (V, MHz)	0.8, 232		0.9, 260	1.05, 264
大小 (MB)	67.28	132.25	67.83	130.77
准确度	68.50%	69.29%	68.61%	69.12%
延迟 (推理)	10.55ms	9.45ms	7.33ms	5.46ms
能耗 (推理)	59.56 μ J	58.86 μ J	55.68 μ J	111.81 μ J
a, b, c	0.25, 0.1, 0.65		0.25, 0.65, 0.1	
得分	0.776	0.794	0.833	0.744

表 2-3 Evolver 上 VGG8 的四种不同部署策略对比

- 与网格搜索（参照一）的对比：图 2-9 展示了 VGG8 的 QVF 调优结果。横轴是搜索迭代次数，纵轴是将测得的 ALE 与先前定义的目标因子 $a=0.25$ 、

$b=0.1$ 、 $c=0.65$ 相结合的奖励得分。虽然网格搜索不需要任何奖励，但仍然使用相同的公式 2-2 计算得分进行比较。如红色曲线所示，本章的算法在收敛到一定程度时，最后 30 个周期内得分几乎不变。在最后 19 次迭代中，最高得分保持为 0.833(见点☆)，此时 $A=68.61\%$, $L=7.33\text{ms}$, $E=55.68\mu\text{J}$ 。最终的 QVF 策略在表 2-2 中呈现。然而对于参照一的网格搜索（蓝色曲线）需要 887 次迭代才能达到类似的得分 0.839。在此如此庞大的搜索空间中，本方法通过使用 RL 来指导搜索方向，节省了 2.69 倍的搜索迭代次数。此外，在 2.2 小节提出的异常值跳过方案在这 330 次迭代中进一步减少了 122 个策略的不必要训练。

2. 与代理任务（参照二）的对比：由于仅使用模型大小的 RL 仅探索量化位宽，根据图 2-8 中能效最高点，本实验经验性地选择 (0.8V, 232MHz) 作为操作电压和频率。作者收集了 330 个量化策略，并在固定的 VF 策略设置的前提下，在 Evolver 上运行相应的量化模型。根据测得的 ALE 结果，将计算得分绘制为图 2-9 中的绿色曲线。最小模型大小 (67.28MB) 在第 313 次迭代（见点△）获得，但最高得分 (0.794) 出现在第 295 次迭代（见点□）。尽管的模型大小比△更大，但在 Layer2 和 Layer3 中具有较低的权重量化位宽。这是由于这两个层在实际硬件执行中对延迟和能耗有很大贡献，实现了较低的延迟和能耗。这个结果表明，代理信号无法反映诸如复杂的内存访问或时变的工作负载之类的硬件行为，因此更小的模型大小不一定意味着更高的得分。

本章的方法优于仅考虑模型大小的方法有两个原因。首先，本方法使用硬件反馈作为奖励来捕捉真实的硬件执行行为。其次，QVF 作为一个整体被同时调整，直接朝向定制的目标进行搜索，具体而言就是本方法将调整 QVF 制定为一个能量优先的多目标问题 ($a=0.25$, $b=0.1$, $c=0.65$)，能耗具有最高优先级。以 VF 选择为例，最终模型的表现在点 (0.9V, 260MHz) 高于 (0.8V, 232MHz)。本方法发现，略微把电压从 0.8V 增加到 0.9V 几乎对能耗没有改善，但对减少延迟有好处。通过结合量化和 VF，本方法的最终策略☆的能耗比基于代理任务的策略△降低了 1.07 倍，延迟降低了 1.44 倍。从而说明由于 Q、V 和 F 都对延迟和能耗产生了影响，因此将它们一起调整可以得到更好的优化策略。

2.4.4 多优化目标之间的取舍

在另外一组实验中，通过将加权系数设置为 $a = 0.25$, $b = 0.65$, $c = 0.1$ ，定义了一个以延迟为优先的部署任务场景，其他实验设置与之前的以能耗为优先的任务相同，最后 Evolver 选择的策略也在表 2-3 中呈现。能量优先策略和延迟优先策略之间的差异说明了在不同目标本方法是如何权衡 QVF 的：由于早期的卷积层具有较少的通道数（Layer1 和 Layer2），表现出较低的工作负载并行性，因此延迟优先策略采用更高的精度。而在后面的全连接层（Layer7）中，由于它们是访存密集型的，更高的位宽会导致更高的内存访问能耗，因此延迟优先实验中采用较高的位宽 8 比特而能耗优先实验则采用较低的 4 比特。至于 VF 点的选择，延迟优先策略选择的（1.05V, 264MHz）也高于能耗优先策略中的（0.9V, 260MHz），以实现较低的延迟。然而，延迟优先也不是采用的最高 VF 点（1.1V, 268MHz），原因是最高的 VF 点消耗的能量过大，在目标因素 $a=0.25$, $b=0.65$, $c=0.1$ 下得分较差。

2.4.5 可扩展性分析

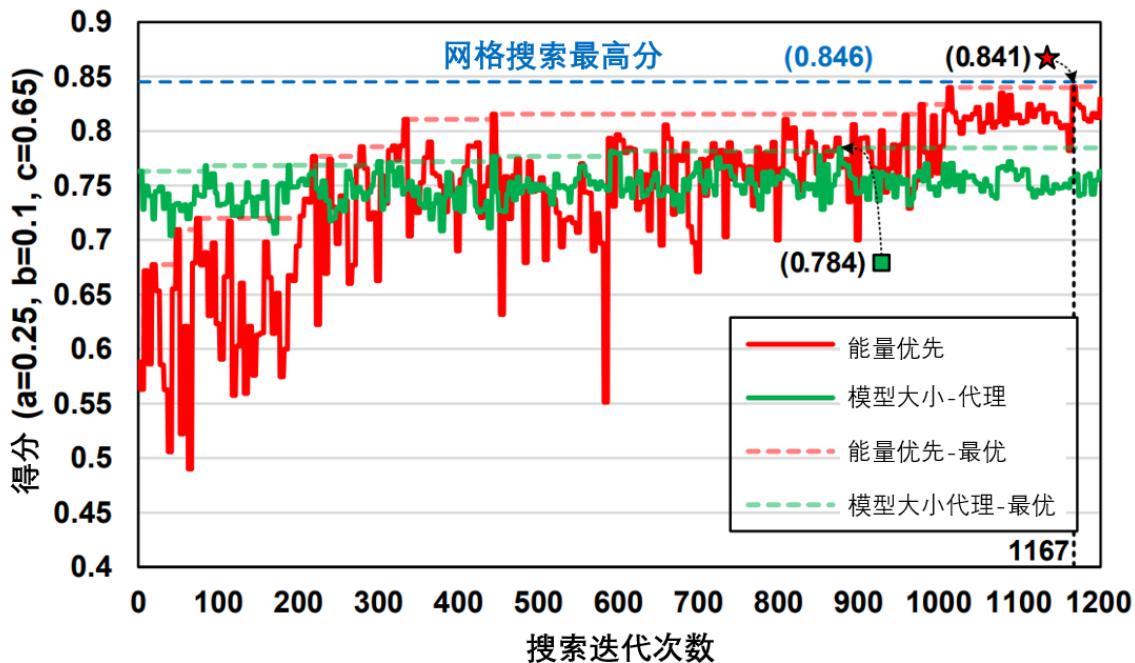


图 2-10 ResNet34 在本地数据集上不同方法的 QVF 调优曲线

本工作通过探索激活值量化位宽 2/4/8 比特选项，来进一步分析能耗优先实验中的本方法的可扩展性。图 2-9 中的曲线“能耗优先-IW”显示了曲线收敛过程，其 s 中最高得分为 0.821 ($A=68.90\%$, $L=7.96\text{ms}$, $E=57.70\mu\text{J}$)，出现在第 1098 次迭

代。最终的 QVF 策略是：权重为 8-4-8-8-4-4-4-8 比特量化，激活值为 8-8-8-8-8-4-8-8 比特量化，工作电压为 0.9V，频率为 260MHz。整个 QVF 调优过程耗时 32.36 小时，能耗为 1282.05J。在调整过程中，作者观察到，如果激活值中经常出现 2 或 4 比特量化，模型的准确性会显著下降，导致奖励非常低。因此，最终的策略只会出现至多一层的输入精度为 INT4，从而保持高准确性。另一个发现是，较大的搜索空间并不保证更好的结果，这是因为低于 8 量比特的激活值实际上对准确性有非常负面影响。更多不恰当的候选策略（具有更多的 2/4 比特的激活值位宽）使得在更大的空间中找到最佳策略变得更加困难。依赖于强化学习的强大搜索能力，尽管理论上需要更多的迭代来次数收敛，但最终的最高模型奖励（0.821）仍然与之前只探索权重量化位宽的结果（0.833）相似。这也意味着，如果可以在事先具有先验知识的情况下减小搜索空间，本算法可能会在较低的调整开销下获得更好的结果，展现了较大的可优化空间。

作者还将 QVF 调优实验扩展到了 ResNet34^[12]模型。图 2-10 展示了在本地 CIFAR100 数据集上对 ResNet34 进行的 QVF 调整曲线。同样的，激活值的量化位宽被固定为 8 比特，而其他设置与 VGG8 调整相同。为了节省调整开销，本组实验单独搜索每个残差块的权重量化策略，而块中的两个层共享相同的量化位宽。搜索运行了 1200 次迭代，耗时 47.35 小时，能耗为 1950.28J。在第 1167 次迭代时，能耗优先的实验组达到了最高得分（0.841），非常接近通过网格搜索获得的最高得分（0.846）。图 2-10 中的虚线曲线表示能耗优先实现和模型大小作为代理任务的实验的最优得分在搜索过程中的变化情况。本方法在第 277 次迭代时就超过了最优的代理任务实验中的模型。另外为了减少初始部署的开销，可以在奖励超过某个阈值之前就停止，而这个阈值可以根据代理任务的策略得分或其他可以离线获得的策略得分来预先设置。在之后的长期使用中，可以在设备不繁忙时继续进行，通过这样的方式减少用户等待时间，以获得更好的用户体验。

2.4.6 在线QVF调优的长久效益分析

本章节以能耗优先部署为例，讨论片上 QVF 多目标调优的长期益处。实验室团队构建了一个新的纯 8 比特基准处理器，没有使用本章节使用的技术。在 6000 张本地数据集上，整个调整过程耗时 9.33 小时，能耗为 369.44J。如图 2-11 所示，与在最高能效点（0.8V, 232MHz）运行的基础 8 比特 VGG8 模型相比，本章的方法只需运行一天的推断，即可弥补调整模式的能耗开销。随着执行时间的增加，能耗节约逐渐增长。当执行时间达到 10 天时，能耗节约率变得稳定（约为 1.3 倍）。在 20 天之后，本方法在相同时间段内比没有 QVF 调整的基准方法多运行了 1.12

亿次推断，总共节约了 4046.04J 的能耗。与仅考虑模型大小的策略△相比，本方法在 20 天内多运行了 7200 万次推断，并节约了 545.25J 的能耗。

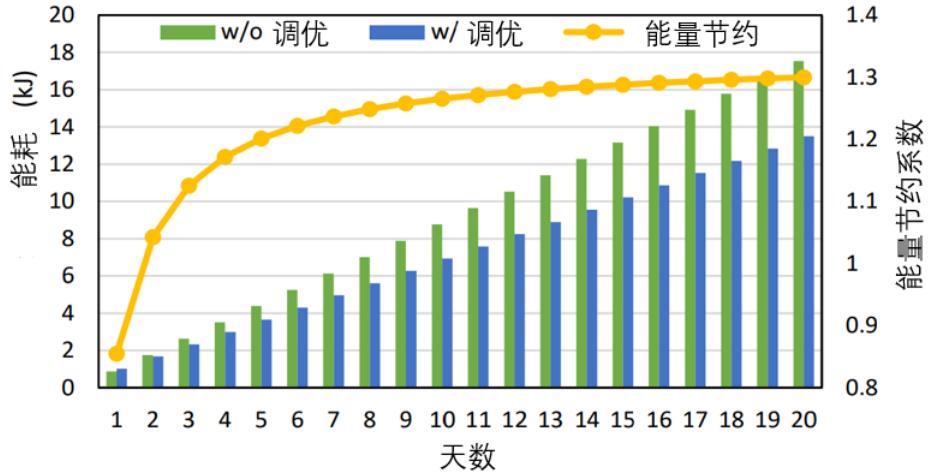


图 2-11 片上 QVF 调优的长久收益

2.5 本章小结

本章主要介绍了一种基于强化学习 Q-Learning 的面向量化策略、电压、频率的多目标优化算法，通过使用奖励塑造和异常点省略等技巧进一步提升了算法的搜索效率。对硬件结构中关键的 RLU 模块采用了基于 LSFU 的设计，并最终与基于网格搜索和基于代理任务搜索的算法相比，本章节介绍的算法无论在模型性能、搜索速度等方面都有更加卓越的表现。同时，本方法具有高度可扩展性，在各类网络结构中都能以较快的速度获得满足要求的模型。最后从长远角度来看，本算法基于 QVF 调整由自我演化带来的芯片配置调整带来的能耗节约的收益，在很短的时间内就能弥补上配置调整带来的损失，具有较高的长期价值。

第三章 面向网络结构与量化策略的片上联合搜索

上一章主要围绕离散搜索空间的量化-电压-频率多目标优化问题提出了一种基于 Q-Learning 的搜索方法。本章将在此基础上把搜索空间扩展至包括网络结构，提出一种完全可微的网络结构与量化策略的联合搜索算法。本章的内容主要以作者在 2021 DAC WIP poster 上展示的工作^[8]为主题展开。

3.1 本章引言

近年来，随着嵌入式设备的高速发展，深度学习应用正逐渐从云端转移至边缘设备。然而把神经网络部署到嵌入式设备上依然是一件非常有挑战性的工作，通常来说需要经历两个阶段的耗时步骤。首先，针对某个特定的硬件平台，算法人员需要对网络结构进行十分严格的设计。其次，在网络结构确定好之后模型需要通过一定手段进行进一步的压缩。这两个阶段的任务都是非常耗时且十分依赖于专家经验。对于网络结构的设计，近几年已经有很多的研究着力于开发相关的 AutoML 工具中的网络结构搜索（Neural Architecture Search 即 NAS）功能，来代替专家进行模型的高效设计工作。在这些 NAS 的方法中，采取最多的主要是 3 种，基于强化学习（Reinforce Learning 即 RL）、基于进化策略（Evolution Algorithm 即 EA）以及基于可微分的搜索策略。一个典型的基于强化学习的 NAS 方法通常是建立一个基于循环神经网络（Recurrent Neural Network 即 RNN）的控制器来发现一组网络算子和连接关系的选择策略。而进化算法则是通过模拟自然选择、基因遗传等过程，每次随机选择和保留部分的网络结构，通过多次变异得到最终的网络结构。可微分网络结构搜索是最近提出的一种十分新颖的搜索策略，其本质思想是把网络的选择过程模拟为对一个超网络的剪枝过程，其优点在于迭代速率更快，且不需要同时存储过多的模型网络结构参数。对于模型压缩，量化、剪枝、蒸馏是三个最主要步骤。

然而，采用传统的两部分离的神经网络部署方式非常耗时，且最终会导致非最优的部署策略。除此以外，一旦所有的部署工作完成，网络结构就不能再发生变化了（即使有新的需求出现时）。基于上述观察，本章节提出了一种新颖的完全可微的网络结构与量化策略的联合搜索方法，即结合了可微分的网络结构搜索和基于量化感知训练的量化方式。同时，本工作利用帕累托边界来修正多目标中的权重系数，从而使得模型能够满足不同场景对精度与延时的需求。如图 3-1 所示，本章提

出的算法由于结合了网络结构搜索与模型量化，与其他的只搜结构或只做量化的工作相比，无论在模型精度还是模型的大小上本方法都有极大的优势。

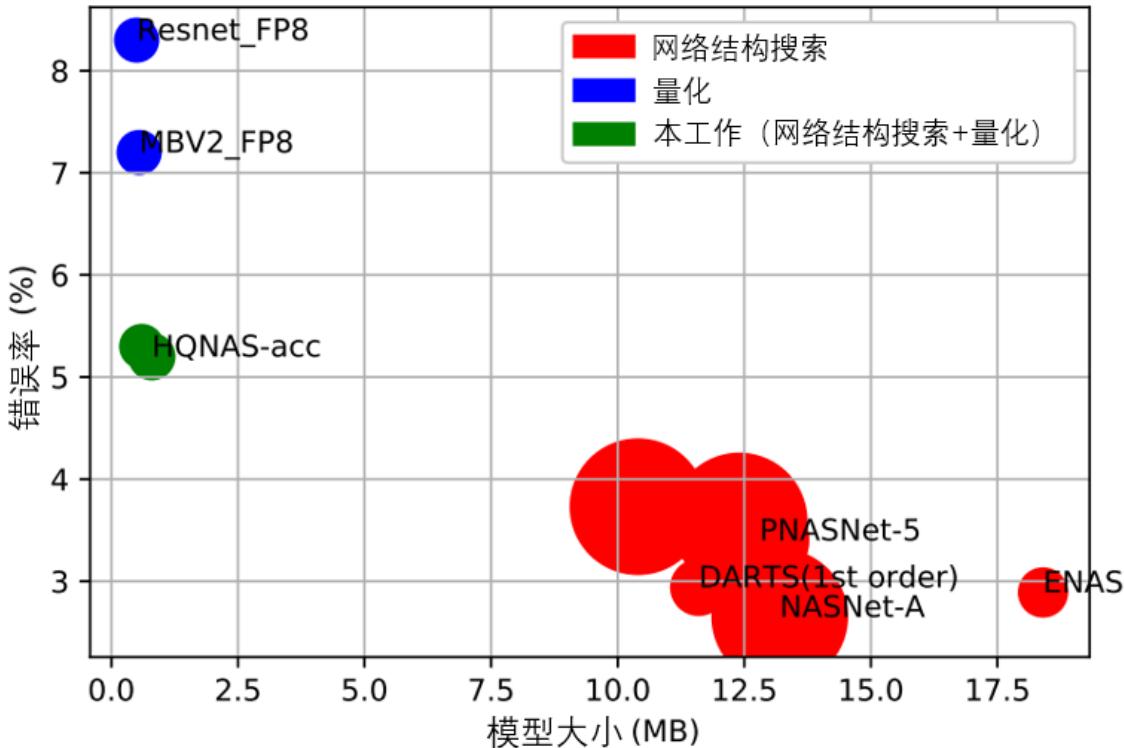


图 3-1 网络结构设计与量化工作中模型的大小与精度分布

本章节的组织形式如下，3.2 小结介绍最主要的可微分网络结构搜索和 QAT 两个概念，从而引出本工作的算法原理与整体流程。接着针对两类数据集 CIFAR10 和 ImageNet 分别验证了本方法在大小尺度数据集上的可扩展性，同时通过一组 500 小类别的 ImageNet 定制化数据集验证了本章方法在终身学习上具有很强的适应能力。

3.2 基于完全可微分的网络结构量化策略搜索

本节将具体展开介绍算法的框架，此方法结合了基于可微分的网络结构搜索（Differentiable Architecture Search 即 DARTS）以及量化感知训练（Quantization Awarre Training 即 QAT）来同时实现结构搜索与模型量化。通过设定一个多目标的损失函数来定制化不同权重下模型对网络精度以及延迟的需求，同时采用^[35]针中的硬件感知的损失函数，使得其对结构与量化参数完全可导，方便端到端的训练。

3.2.1 可微分的网络结构搜索

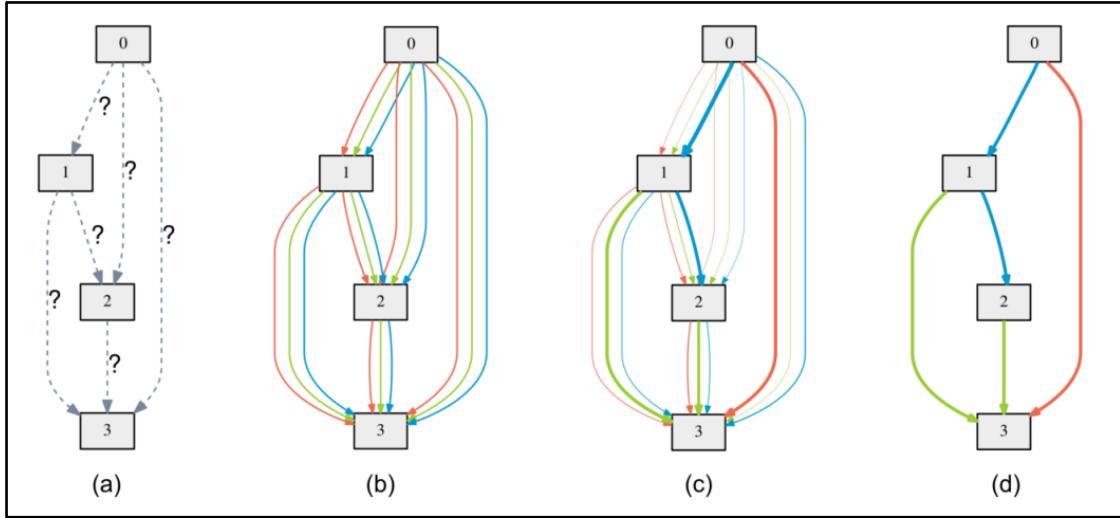


图 3-2 可微分网络结构搜索原理^[6]

DARTS (Differentiable Architecture Search) 是一种神经架构搜索 (NAS) 算法，旨在通过自动搜索神经网络的最佳架构来提高模型的性能。传统的神经网络架构设计需要人工进行尝试和调整，非常耗时和困难。而 DARTS 的出现解决了这个问题。它引入了可微分的搜索空间，允许通过梯度优化来自动搜索最佳的网络架构。如图 3-2 所示，它使用一个可微分的超图来表示搜索空间，其中节点表示网络的操作（如卷积、池化等），边表示操作之间的连接，每个节点的输出都是由若干可选网络结构进行加权求和而得到，由公式 3-1^[53]列出：

$$m_{\mathcal{O}}^{\text{DARTS}}(x) = \sum_{i=1}^N p_i o_i(x) = \sum_{i=1}^N \frac{\exp(\alpha_i)}{\sum_j \exp(\alpha_j)} o_i(x) \quad (3-1)$$

其中 α_i 为某个备选结构的结构参数，则当前层的输出等于按某个概率分布加权求和当前层所有可能的网络结构的输出。通过对超图进行优化，DARTS 可以自动搜索最优的操作组合和连接方式。与传统的架构搜索方法相比，DARTS 具有以下优势。首先，它的搜索过程是可微分的，可以使用梯度下降等优化算法来高效地搜索架构。其次，DARTS 不需要昂贵的计算资源，搜索过程非常快速。最重要的是，DARTS 可以直接在端到端的训练过程中进行架构搜索，减少了人工干预的需求。通过 DARTS，可以更轻松地设计出性能更好的神经网络架构，而无需进行繁琐的手动调整。

3.2.2 量化感知训练QAT

量化是将浮点数参数和激活值转换为较低精度的固定点数表示的过程，通过将神经网络中的浮点数转换为定点数表示，可以减少模型的内存占用和计算量，从而在较低的硬件资源上进行部署。

量化感知训练（Quantization Aware Training 即 QAT）是一种在训练过程中考虑量化的方法。区别于训练后量化（Post Training Quantization 即 PTQ）在模型训练完之后用无标签数据进行中间值的标定。QAT 使用少量有标签的数据在训练过程中模拟带来的精度损失，并且在反向传播过程中使用梯度估计方法来近似量化操作的梯度。依赖于伪量化模块，QAT 能获得更高的量化精度。

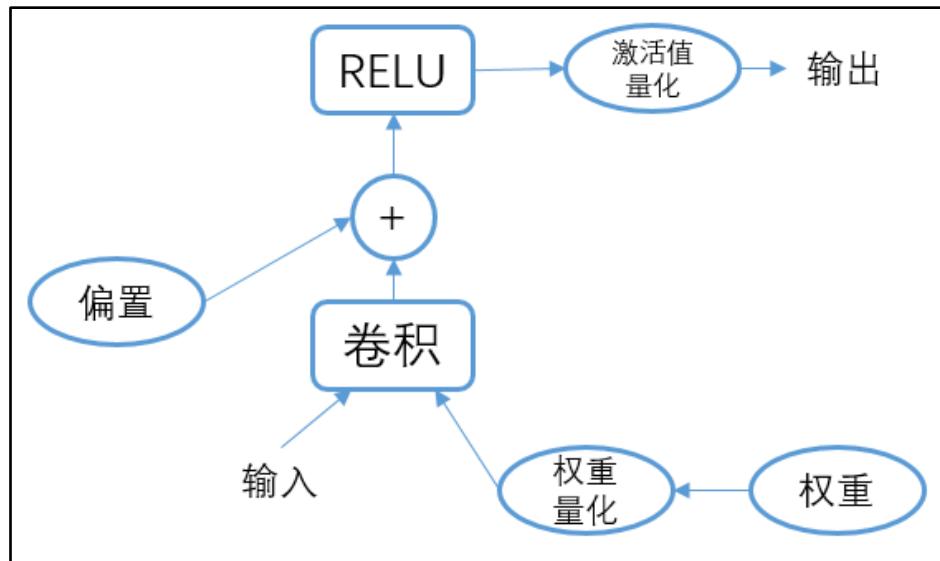


图 3-3 QAT 训练过程原理图

如图 3-3 所示，量化感知训练的核心就是在权值与激活值输入进运算层（卷积、全连接等）前插入伪量化模块。伪量化模块有两部分组成，首先将原始的浮点输入量化为整形数值（在实际训练时依然用浮点格式存储），然后将量化过后整形数值反量化回浮点域数值。由于量化本身会有信息损失，反量化没有信息损失，所以整理来看，伪量化模块是存在信息丢失的。其目的也在于此，通过模拟量化带来的精度损失来使得模型朝着更加符合量化的方向进行训练，从而降低量化带来的精度损失影响。本章算法中采用的是均匀非对称量化，用公式 3-2 表示：

$$\begin{aligned}
 r &= S(q - Z) \\
 q &= \text{round}\left(\frac{r}{S} + Z\right)
 \end{aligned} \tag{3-2}$$

其中， r 、 q 为浮点域以及量化域对应的数值， S 为放缩系数， Z 为浮点数 0 在量化

域的具体数值， S 、 Z 表达形式如 (3-3) 所示：

$$S = \frac{r_{max} - r_{min}}{q_{max} - q_{min}} \quad (3-3)$$

$$Z = round(q_{max} - \frac{r_{max}}{S})$$

其中， r_{max} 、 r_{min} 分别为浮点数中的最大最小值， q_{max} 为确定量化位宽之后的最大整型值（以 8 比特为例，对应的 q_{max} 就是 255）。之所以采用 QAT 作为本工作的量化方式，除了 QAT 本身具有的高精度的优点外，还有一个原因是 QAT 能与现有的 DARTS 框架完美组合，形成一套完全可微分的网络结构与量化策略的端到端搜索方式，与分离的先搜结构再搜量化的方式相比，大大提升了算法搜索的效率。

3.2.3 算法框架与原理

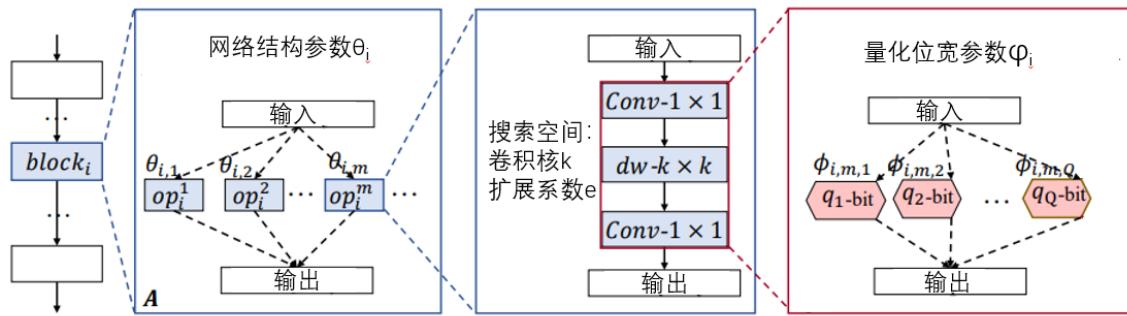


图 3-4 端到端网络结构与量化策略结构原理

整个框架结合了 DARTS 和 QAT 来形成一套完全可微分的网络结构与量化的搜索策略。如图 3-4 所示，整个算法框架由外向内可以做多级展开。首先采用 DARTS，每一层所有可选择的操作（例：3*3 卷积、5*5 卷积）构成了一组多路径的输出，如图 3-4（左侧）所示。在本章实验中，采用业界经常使用的基于 MBConv 的基础搜索空间^{[49][53]}，对应的搜索维度为内部卷积核的大小 k 以及扩展系数 e ，如图 3-4（中）所示。在本实验中假设每个 MBConv 块内部的采用相同的量化位宽，量化策略的选择依然可以仿照 DARTS 中的多路径并联的形式来构成，如图 3-4（右侧）。在此基础上，本工作的总优化目标如公式 3-4 所示：

$$\min Loss = \mu * Acc + v * Lat \quad (3-4)$$

与第二章中的目标函数类似，该优化目标的本质是在精度和延迟之间取舍（ μ ， v 表示取舍的权重系数，其和为 1），其中精度（Acc），延迟（Lat）的损失函数如公

式 3-5 所示：

$$Acc = \frac{1}{n} \sum_{x_i, y_i} CE(x_i, y_i, w) \quad (3-5)$$

$$Lat = \sum_{layer} \sum_l \theta_l \sum_q \phi_q Lat_q^l$$

其中 Acc 为标准的交叉熵损失函数，而 Lat 参考了工作^[35]中可微分硬件反馈函数的方法，其中 Lat_q^l 表示配置形状为 1 量化位宽为 q 的单一 MBCConv 块在对应硬件上的实现测量出来的实际延时（事先记录好的一组数据）。可以看出无论是 Acc 损失还是 Lat 损失，都对权值 w、网络结构参数 θ_l 、量化位宽参数 ϕ_q 完全可导，因此整个损失函数可以实现端到端的训练，整体流程如算法 3-1 所示。

Algorithm 1: 端到端网络结构与量化策略的联合搜索

输入：最大迭代的 epoch 数量 E_{max} ；本地训练集 D^{train} ，学习率 η 以及节点定制化的帕累托系数 (μ, v) ；
输出：定制化模型 P ，包含每层结构 A_i ，量化位宽 Q_i 和权重 W

```

初始化权重  $w$ ，每层的多路径结构参数  $\theta_l^i$  与多路径量化参数  $\phi_q^i$ 
for  $e = 1, 2..E_{max}$  do
    for  $batch(x, y) \in D^{train}$  do
        根据公式 3-4、3-5 由  $\mu, v, w, \theta_l, \phi_q$  与数据集  $(x, y)$  得损失函数  $L$ 
        梯度下降：
         $w \leftarrow w - \eta_e \nabla_w L(batch(x, y), \mu, v, w, \theta_l, \phi_q)$ 
         $\theta_l \leftarrow \theta_l - \eta_e \nabla_{\theta_l} L(batch(x, y), \mu, v, w, \theta_l, \phi_q)$ 
         $\phi_q \leftarrow \phi_q - \eta_e \nabla_{\phi_q} L(batch(x, y), \mu, v, w, \theta_l, \phi_q)$ 
    end
end
for  $layer i = 1, 2..N$  do
    由贪心策略根据最大权重选出对应的网络结构与量化位宽：
     $A_i \leftarrow \arg \max_l \theta_l^i$ 
     $Q_i \leftarrow \arg \max_q \phi_q^i$ 
end
根据每层  $A_i, Q_i$  进行训练微调得出权值  $W$ ，即构成最终模型  $P$ 

```

算法 3-1 端到端网络结构与量化策略联合搜索

3.3 算法评估

本工作基于 DARTS 和 QAT 方法在 CIFAR10 和 Imagenet 数据集上搜索最优的神经架构和量化策略，以方便与现有的部分工作进行对比。作者事先在 CPU 和 GPU 上进行目标硬件平台的测试得到 3.2 小节中对应各个配置下 Lat_q^l 延迟。值得区分的是，GPU 上的延迟是在 Tesla V100 上以批量大小为 8 进行测量，而 CPU 延迟是在 Intel(R) Xeon(R) Platinum 8168 上以批量大小为 1 进行测量。其中，CPU 延迟用于 CIFAR10 实验，以证明联合优化的有效性，而 GPU 延迟则作为与在 CPU 上得到的结果进行比较的对象的硬件感知搜索。本工作的主干架构基于 MobileNetV2 中的 MBCConv 块。对于 CIFAR10 数据集，整个网络由 5 个大块组成，对于 ImageNet 数据集则变为 7 个，如图 3-5 所示，每个块由 4 个 MBCConv 组成（第一个和最后一个块只有 1 个）。网络结构的选择 $l=\{k,e\}$ ，其中 k 为卷积核大，区间为 {3, 5}， e 为扩展系数大小，区间 {3, 6}。量化位宽在本实验中只对权值 w 进行量化，可选位宽为 {4, 8, 16} 比特，激活值位宽统一为 16 比特。

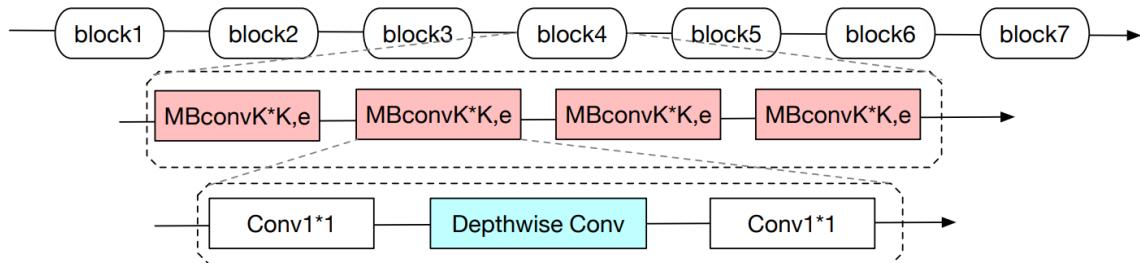


图 3-5 模型的基本组成结构

3.3.1 CIFAR10 数据集

CIFAR-10 (Canadian Institute for Advanced Research 10) 是一个常用的计算机视觉数据集，由加拿大高级研究院创建。它由 10 个类别的彩色图像组成，每个类别包含 6000 张图像，总共有 60000 张图像。本工作随机选择训练集的一半作为验证集。最大迭代次数设置为 100，训练集和验证集的批量大小均为 256。本工作使用批次特定的统计数据进行批次归一化，而不是全局移动平均，以避免搜索过程中由于架构变化引起的问题。这个数值比 DARTS 中的要大得多，因为单路径可以减少内存爆炸问题。本工作在实验中采用动量 SGD 来优化权重，初始学习率 α 为 0.05，动量为 0.9，权重衰减为 3×10^{-4} 。作者分别进行了两组实验，使用不同的权重因子来衡量延迟和准确性的奖励：(1) $\mu = 0.2, \nu = 0.8$ ，即以准确性为优先的搜索；(2) $\mu = 0.8, \nu = 0.2$ ，以延迟为优先的搜索。

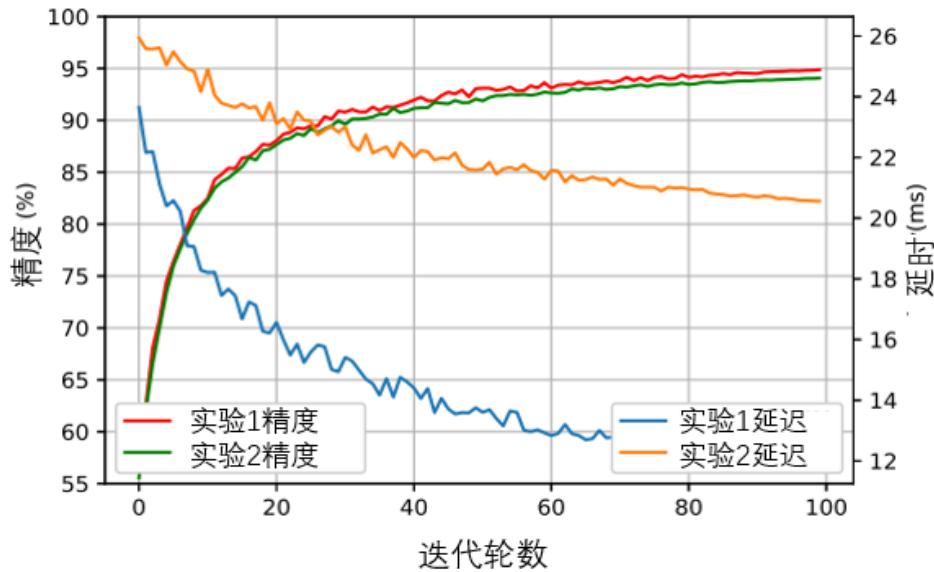


图 3-6 精度、延迟的训练曲线

搜索方法	GPU	时间/天	大小/MB	误差/%	方法分类	量化
PNASNet-5 ^[4]	100	1.5	12.8	3.41 ± 0.09	贝叶斯优化	否
NASNet-A ^[55]	500	4	13.2	2.65	强化学习	否
NASNet-B ^[55]	500	4	10.4	3.73	强化学习	否
NASNet-C ^[55]	500	4	12.4	3.59	强化学习	否
ENAS ^[43]	1	0.5	18.4	2.89	强化学习	否
DARTS (一阶) ^[6]	1	1.5	11.6	2.94	梯度	否
DARTS (二阶) ^[6]	1	4	13.6	2.83 ± 0.06	梯度	否
JASQNet ^[56]	1	3	2.5	2.9	进化	是
本文 (精度优先)	1	0.5	2.62	2.93	梯度	是
本文 (延时优先)	1	0.5	2.39	3.13	梯度	是

表 3-1 本章方法与现有其他方法的对比

如图 3-6 所示，本工作记录了在搜索过程中在验证集上准确性和延迟的曲线。这两个实验都在 4 小时内使用单个 GPU 完成。在前 20 个迭代周期中，使用 dropout 技术来防止过拟合。从最终结果来看，以准确度为优先的实验(1)在最终模型生成

的进度上略高于以延迟为优先的实验(2)，然而实验(2)的最终推理延迟远低于实验(1)，因此证明了本方法在硬件感知任务场景中拥有不错的搜索结果。本工作的特点与其他经典方法的对比详细列在了表3-1中。本方法不仅支持网络结构与量化策略的联合搜索，同时在模型的生成质量上(包括推理进度、延迟等方面)都有全方位的提升。

3.3.2 Imagenet数据集

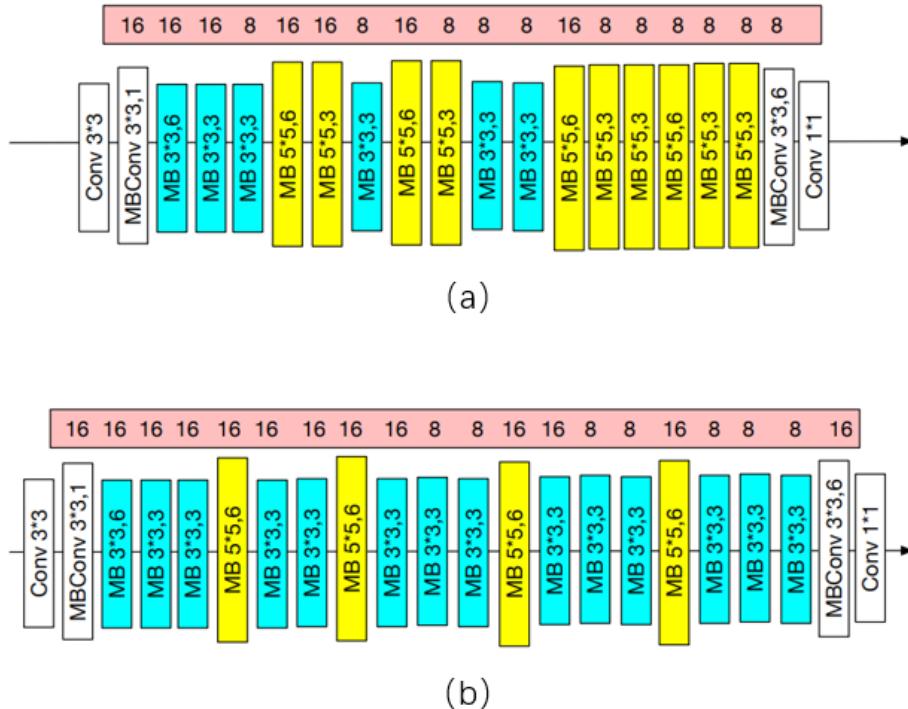


图 3-7 在 (a) GPU (b) CPU 上最后生成的模型结构与量化方式

对于 Imagenet, , 本工作将之前 CIFAR10 实验中基础块的数量从 5 增加到 7。因此，总的搜索空间数量为 $N = 4^{1+4*5+1} * 3^{1+4*5+1} \approx 5.5 * 10^{23}$ 。本工作使用 GPU 和 CPU 作为延迟反馈平台分别进行了两次完整的搜索，损失函数的加权系数为 $\mu = \nu = 0.5$ 。

图 3-7 展示了在 GPU 与 CPU 上搜出的最终的网络架构和量化方式。可以发现，两个硬件平台上的架构和量化策略是相当不同的。与在 GPU 上搜出的模型相比 CPU 模型更加浅而宽，因为 GPU 具有更高的并行性。而 GPU 上的模型尤其偏好较短的位宽，尤其是在最后几层。两组实验在最早的几层都采用相当浅的层来降低延迟，同时采用较高的量化位宽来保证进度。这是因为最开始几层的特征相对

较小，需要较高位宽来支持分辨率。此外，GPU 模型和 CPU 模型都更喜欢在降采样层使用较大的 MBConv 和更宽的量化位数来减少这类网络层上信息的损失，这与工作^[53]中的结论十分相似。

3.3.3 终身学习模拟

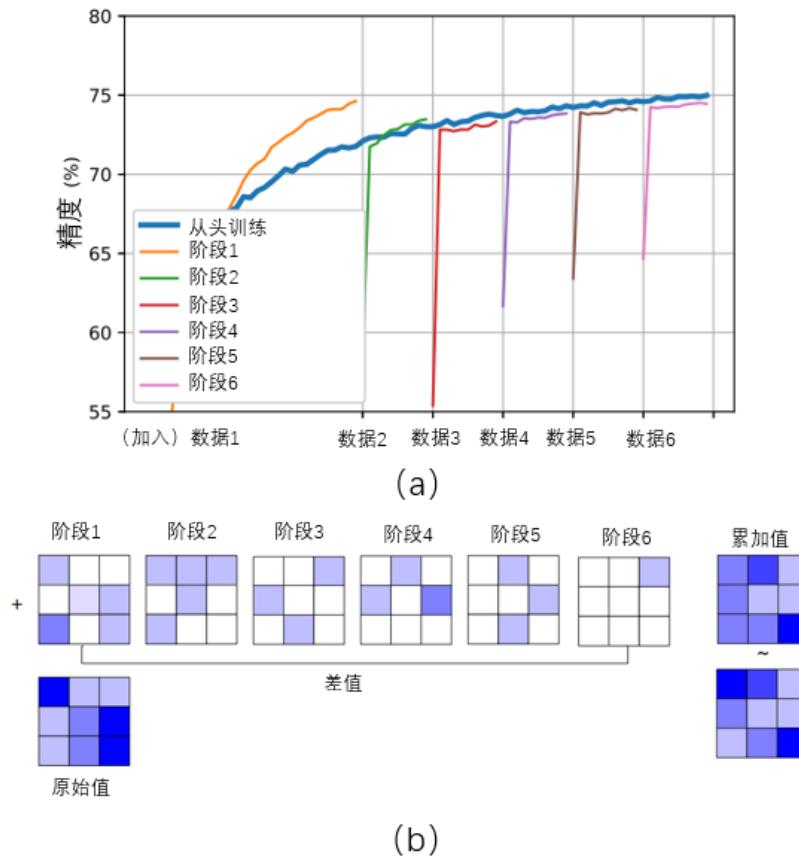


图 3-8 分阶段训练与从头训练的 (a) 精度曲线 (b) 权值差变化

在这部分实验中展示了本算法如何适应终身学习的情况。典型的神经网络部署框架通常在模型部署到终端设备后就保持不变了，但这并不合理，因为本地数据集可能会增加（例如，新的标签）。因此，神经网络需要从增量的知识中学习并进化自身。然而在每次遇到新类别的数据时，从头开始设计网络结构来重新训练部署是非常低效的。本实验通过从 Imagenet 中随机选择 500 个类别作为原始数据集，并使用本章节中的网络架构和量化策略的联合搜索方法，每训练 50 个 epoch 就添加 100 个新的标签的数据到原来的数据集中。随着时间的增加，类别数量越来越多直到 1000 个类别完全使用，观察每个 epoch 之后的验证集精度。同时本工作也记录了从头开始完整学习这 1000 个类别的学习曲线来对比差异。如图 3-8 (a) 所

示，尽管本章节方法所花费的总时间略多于从头开始，但每次更新所使用的平均时间非常少，这在现实生活中有较高的应用价值，每次只需较小的等待就能获得较为不错的增量模型。此外，通过观察一个固定层 3×3 的权重值变化可以发现，多轮累加后的权值是非常逼近与从头开始训练的对应位置权值的。足以说明本工作在面对缓慢增量知识的终身学习场景中具有很大的潜力。

3.4 本章小结

本章主要介绍了一种同时搜索网络结构与量化策略的优化算法，通过结合基于可微分的网络结构搜索与基于量化感知训练的量化方法，本方法相比于传统先设计网络结构后确定量化策略的两阶段方法，在搜索效率和模型生成质量上都有了显著的提升。通过调整基本组成模块的数量，本方法在各类数据集中都有着较好的可扩展性。同时在增量学习的场景中，采用同样的算法框架，本章介绍的方法经过多轮权重的更新再数值上可以十分逼近完整的从头开始训练，充分说明了本算法在终身学习任务中的潜力与价值。

第四章 多节点片上联邦网络结构与量化策略搜索

第二章、第三章分别从多目标优化、多领域联合优化的角度对片上自动化机器学习算法进行了优化，然而都只针对单节点场景。从本章开始，本文将在之前工作的基础将场景拓展至离线的多节点任务中，从减少通信量的角度去探讨在联邦学习范式下片上自动化机器学习的优化方法。本章介绍的研究内容，以作者将要在2023 DAC WIP poster 上展示的工作^[19]为主题展开。

4.1 本章引言

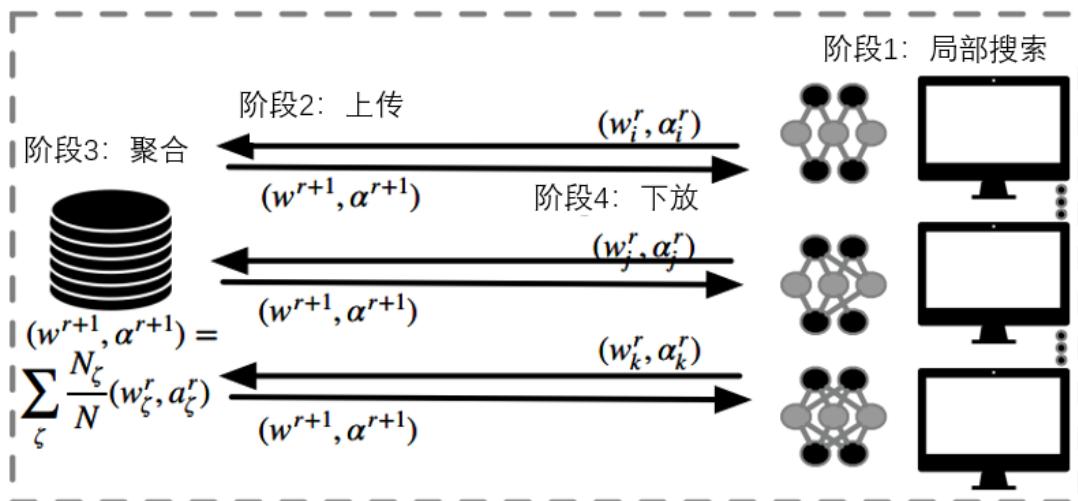


图 4-1 联邦网络结构搜索的基本框架

联邦学习（Federated Learning 即 FL）是一种有前景的分布式训练方法，用于处理分散数据时的深度神经网络（DNN）训练，以满足用户隐私和监管限制需求^[25]。它允许多个数据客户端通过交换中间参数（例如权重和偏置）来共同训练一个全局模型，同时保持原始本地数据样本的私密性。因此，FL 已广泛应用于各种具有挑战性的机器学习任务，例如数据挖掘、计算机视觉和自然语言处理。尽管 FL 具有广泛的优势，但其中一个主要挑战是数据的异构性，即每个设备端的数据分布是非独立同分布的。因此，一个统一的预定义的模型架构无法在所有节点上都拥有较好的泛化能力。为了缓解这个问题，最新的一些研究将神经架构搜索（Neural Architecture Search 即 NAS）算法与联邦学习结合，尝试探索节点异构的模型结构。

^{[43][45]}。如图 4-1 所示，常见的基于 NAS 的 FL 框架可以分为 4 个步骤。以第 r 轮循环为例，首先，在本地进行若干次迭代进行局部的 NAS，获得更新过的权值 w_r^r 与结构参数 α_r^r 。接着，每个设备节点将 w_r^r 和 α_r^r 上传到中央主机节点（非云端）。第三，主机节点聚合所有参数，得到全局版本的 w^{r+1} 和 α^{r+1} 。最后，设备节点获取更新后的参数并进行下一轮的搜索。然而，当前框架面临着一个巨大的挑战。设备节点的 NAS 算法需要大量的计算和庞大的模型存储空间，这使得参数传输非常昂贵。这是因为先前的研究通常都基于可微分的多路径 NAS 算法^[6]，该算法在一个包含所有候选架构路径的“超网络”中进行搜索（相应介绍在章节三中）。尽管它将总的搜索轮数减少到了可接受的程度，但随着超参数类型的增加整个可选路径的数量会呈指数级增长，使整个超网络变得过于庞大。因此，设备节点在每一轮中需要向中央节点传输巨大的本地模型的权重和结构参数，从而导致巨大的通信成本。

基于前面的观察，本章节提出了 FAQS 算法，一个高效的结合个性化联邦（Federated）、网络结构搜索（Architecture）、量化（Quantization）的搜索算法框架，在三个维度上大幅降低原先的通信成本。首先，本章工作基于单路径的 NAS 算法^[46]作为的搜索引擎，利用超核共享，可以将冗余的超网络减小到与普通网络相似的大小。其次，网络参数进一步通过比特共享的混合精度量化进行压缩，这与神经架构搜索同时进行，以节省额外的搜索时间开销。第三，设备节点与主机节点进行数据传输时，只传输部分参数以节省大量的通信空间。所有这些因素都有助于以高通信效率的方式生成个性化的本地模型。此外，FAQS 可以通过在不同的设备节点上设置各种个性化帕累托函数来搜索出不同的硬件偏好的模型。

4.2 高通信效率的联邦网络结构与量化策略搜索

在本小节中，作者将详细介绍 FAQS 各个实现细节。首先将介绍想要解决的问题，包括 FAQS 问题的形式化定义以及中间涉及到的通信成本定义。接下来，本文将详细介绍和解释框架中用于在不同维度上降低通信成本的三个关键技术。最后，对算法整个流程进行梳理和总结。

4.2.1 问题定义

首先是优化目标，FAQS 的优化问题的数学表示可以通过以下公式来说明：

$$\begin{aligned} \min_{\theta_1, \dots, \theta_k} G(\theta_1, \dots, \theta_k) &= \frac{1}{k} \sum_{i=1}^k L_i(\theta_i, x_i, y_i) \\ L_i(\theta_i, x_i, y_i) &= \alpha_i CE(\theta_i, x_i, y_i) + \beta_i Lat(\theta_i) + \gamma_i MS(\theta_i) \\ \alpha_i + \beta_i + \gamma_i &= 1 \end{aligned} \quad (4-1)$$

目标是为每个设备节点 i 找到一个个性化模型，该模型在所在节点的本地数据上有良好的性能（这里为了简化，假设每个设备节点都拥有相同数量的样本），其中 k 是设备节点的数量， θ_i 是可训练的参数，包括权重 w 和网络结构参数 t 。 (x_i, y_i) 表示每个设备节点的本地数据。 L_i 是设备 i 的个性化损失函数，其中包含帕累托加权系数^[47] ($\alpha_i, \beta_i, \gamma_i$)（与第二章节中类似），用于衡量准确性、延迟和模型大小。本章方法使用了^[48]中的可微分的延迟和模型大小的损失函数，以用于端到端训练。

通信开销：在 FAQS 中，在所有轮次中，主机节点和设备节点之间传输的总通信成本（以比特为单位）可以表示为：

$$Comm = 2 \sum_{r=1}^{nRnd} \sum_{i=1}^{nClient} \sum_{j=1}^{nLayer_i} \gamma_{i,r}^j N_{i,r}^j(w, t) * Q_{i,r}^j(w, t) \quad (4-2)$$

其中， $nRnd$ 表示 FL 的总轮次， $nClient$ 是参与协作的设备节点数量， $nLayer_i$ 是第 i 个设备节点的总层数， w 和 t 分别是可学习的权重和网络结构参数。FAQS 基于阈值 t 生成具体的架构，而不是多路径^[43]中使用的路径加权系数 α 。 $\gamma_{i,r}$ 表示实际传输的数据量的比例。 $N_{i,r}^j(w, t)$ 是第 i 个设备节点第 j 网络层，包括权重（或梯度）和网络结构参数在内的总参数数量，而 $Q_{i,r}^j(w, t)$ 表示第 i 个设备节点第 j 网络层的参数平均比特数。由于每个轮次包含参数的拉取（从主机节点到设备节点）和上传（从设备节点到主机节点），通信成本在最终结果的基础上乘以 2。为了更好地理解前面的参数，本文将 FAQS 现有的几种方法在每个变量上进行对比。如表 4-1 所示，关注单个基于 MobileNetV2 的搜索块^[49]，通过使用超核搜索和比特共享量化，FAQS 大大减少了传输的比特数。 $\gamma_{i,r}^j$ 表示实际传输的参数百分比，FedAvg^[25]和 FedNAS-EDD^[47]都传输整个网络，因此该值为 1，而 FAQS 只传输了部分重要的参数，因此可以减小 $\gamma_{i,r}^j$ 。

	$\gamma_{i,r}^j$	$N_{i,r}^j(w, t)$	$Q_{i,r}^j(w, t)$	$Comm_{i,r}^j(\text{KB})$	是否支持定制化
FedAvg(MBV2)	1	9k	32(浮点)	72	否
FedNAS+EDD	1	25k	28(整型)	175	是
本工作	0.51	9k	16(整型)	18.4	是

表 4-1 通行量关键参数对比

4.2.2 通信量缩减的三个维度

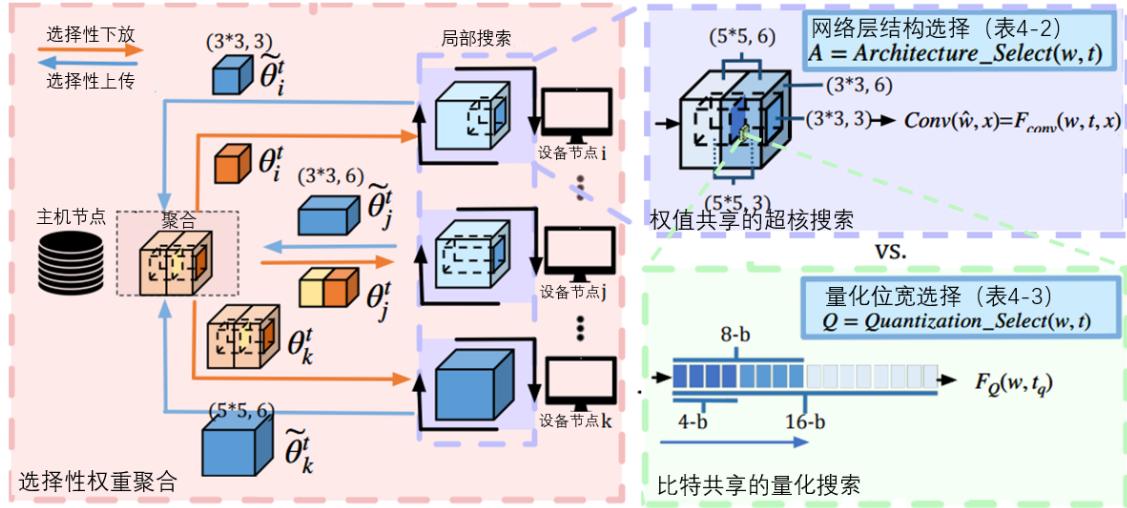


图 4-2 通信量缩减的三个维度

如图 4-2 所示，FAQS 的框架采用了三种技术来减少公式 4-2 中的不同参数，以实现较高的通信效率，包括权重共享的超核搜索（右上）、比特共享的量化（右下）和选择性权重聚合（左侧）。本工作构建的网络结构与章节三中相同，都是基于 MobileNetV2 的基础块。

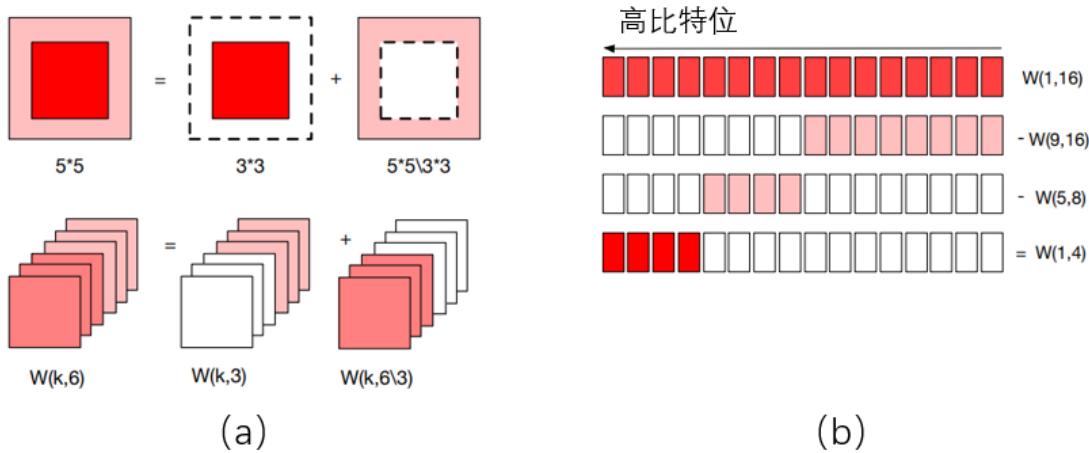


图 4-3 (a) 权值共享 (b) 比特共享

- 权值共享的核心思路是用一个超大核代替传统可微分搜索中的多指路结构。在这个维度上，通过减少公式 4-2 中的总参数数量 $N_{i,r}(w, t)$ 来进行优化。与著名的多路径可微分 NAS^[6]相比，该方法构建并直接训练一个名为 SuperNet 的过参数化网络，其中包含所有可能的候选路径。单路径 NAS 的关键思想是通过称为超核^[46]的权重共享来确定对应的网络结构。一个超

大核指的是在所有的搜索空间中，最大形状的核大小，通过对不同的核部分的筛选从而达到结构选择的目的。在本工作的搜索样例中，最大的形状为一个 $5*5$ 大小并且扩张系数为 6 的卷积核。为了更好的说明权值共享的原理，将这个超核按照如图 4-2 所示分别沿着核特征图方向以及沿着通道方向进行分割，分割后如图 4-3 (a) 所示。选择 $3*3$ 或者 $5*5$ 的卷积核等价于是否在原始的卷积核中保留 $w_{55\backslash 33}$ 的部分，如果不保留则为 $3*3$ 的卷积核大小，如果保留则为 $5*5$ 卷积核的大小。而选择扩张系数或是跳线操作等价于是否选择保留整个通道长度或是一般的通道长度或者是不保留任何通道。进而可以提炼出以下公式：

$$w_k = w_{33} + \sigma(id(w_{55\backslash 33}, t_{k,5})) * w_{55\backslash 33}$$

$$\hat{w} = \sigma(id(w_{k,3}, t_{e,3})) * (w_{k,6\backslash 3} + \sigma(id(w_{k,3}, t_{e,6})) * w_{k,6\backslash 3}) \quad (4-3)$$

其中， w_{33} 和 $w_{55\backslash 33}$ 表示原始超核的 3×3 部分和 $5\times 5\backslash 3\times 3$ 部分的权重。而 $w_{k,3}$ 和 $w_{k,6\backslash 3}$ 分别表示扩展超内核的前半部分和后半部分。 $t_{k,5}$ 、 $t_{e,3}$ 和 $t_{e,6}$ 是三个阈值用于对应的三个 $\sigma(\cdot)$ 函数的计算，最后用这三个 $\sigma(\cdot)$ 的输出来决定是否选择 $w_{55\backslash 33}$ 的子集、跳线操作和扩展系数。例如，对于以下公式：

$$id(w_{55\backslash 33}, t_{k,5}) = \|w_{55\backslash 33}\|^2 - t_{k,5} \quad (4-4)$$

可以从 $id(\cdot)$ 的结果推断出，如果 $id(\cdot)$ 远大于零，那么 $w_{55\backslash 33}$ 中的大多数变量可能具有重要的值，因此 $\sigma(\cdot)$ 几乎等于 1。因此，根据公式 4-3， w_k 会产生一个完整的 w_{55} 内核大小。通过使用基于权重共享超内核的单路径 NAS，多路径 NAS 中的原始冗余超网络可以减小到与普通网络类似的大小，使得整个搜索代理轻巧高效。核大小以及扩展系数与 $id(\cdot)$ 结果之间的关系列举在表 4-2 中：

$id(w_{k,3}, t_{e,3})$	$id(w_{k,3}, t_{e,6})$	$id(w_{55\backslash 33}, t_{k,5})$	$block$ (卷积核、扩展系数)
+	+	+	$5 * 5, 6$
+	+	-	$3 * 3, 6$
+	-	+	$5 * 5, 3$
+	-	-	$3 * 3, 3$
-	*	*	跳过

表 4-2 id 函数与块类型对应关系

这一层的最终输出可以通过对转换后的权重 w 和输入 x 进行卷积计算，或

者通过原始权重 w 、架构阈值 t 和输入 x 的通用函数 F_{conv} 进行计算。

$$output = Conv(\hat{w}, x) = F_{conv}(w, t, x) \quad (4-5)$$

2. 比特共享的量化：在这个维度上，本工作专注于在主机节点和设备节点之间，减少公式 4-2 中所有量化变量 $Q_{i,r}^j(w, t)$ 的平均比特数。量化是在部署深度神经网络时使用的模型压缩方法，本工作遵循广泛使用的量化感知训练方法^[50]，该方法根据最大值和最小值差值，将浮点数值均匀地映射为最接近的整数点值。本工作将量化过程表示如下：

$$w^* = Norm^{-1}(R(Norm(w), b)) = Q(w, b) \quad (4-6)$$

其中 (w, w^*) 是实际值和近似量化值的权重向量， b 是比特数。 $Norm(w) = (w - w_{min}) / (w_{max} - w_{min})$ 是一个线性缩放函数，将向量的值归一化到[0,1]的范围内，而 $Norm^{-1}$ 是其逆函数。将其化作一个整体便是量化函数 $Q(\cdot)$ ，它接受归一化的真实权值 w 和比特数 b ，并输出相应权重的最接近的量化后的值。与^[47]中的多路径量化策略不同，FAQS 将所有量化策略存储在共享空间中，而^[47]为不同的量化策略都分配了独立的存储空间。例如，在一个 {4, 8, 16} 比特的搜索空间中，^[47]独立存储每个潜在量化选择需要 $28(4 + 8 + 16)$ 比特，而 FAQS 只需要 16 个固定比特 ($b=16$)，可以节省 42% 的模型大小。如图 4-2 所示（右下），与从独立的多路径量化策略中选择量化策略不同，比特共享量化的基本思想类似于权重共享的超核网络结构搜索。如图 4-3 (b) 所示，对于每一层，量化位宽的策略取决于是否选择在一个固定的 16 比特空间中保留 5~8 比特和 9~16 比特。对应的建模公式为：

$$\begin{aligned} w_q &= w_{1 \sim 4}^* + \sigma(id(w_{5 \sim 8}^*, t_{q,5 \sim 8})) \\ &\quad * (w_{5 \sim 8}^* + \sigma(id(w_{9 \sim 16}^*, t_{q,9 \sim 16})) * w_{9 \sim 16}^*) \\ &= f(w^*, t_q) = f(Q(w, 16), t_q) = F_Q(w, t_q) \end{aligned} \quad (4-7)$$

其中 $w_{i \sim j}^*$ 是公式 4-5 中近似的量化值的第 i 到第 j 位（从左到右依次为高位到低位）。类似地， $t_{q,5 \sim 8}$ 和 $t_{q,9 \sim 16}$ 是两个阈值，用于相应 $\sigma(\cdot)$ 函数的计算，从而来确定是否保留 5~8 位和 9~16 位。与前面的权重共享超核类似，对于以下的 $id(\cdot)$ 函数：

$$id(w_{5 \sim 8}^*, t_{q,5 \sim 8}) = \|w_{5 \sim 8}^*\|^2 - t_{q,5 \sim 8} \quad (4-8)$$

如果 $\|w^*_{5 \sim 8}\|^2$ 远大于阈值 $t^q_{5 \sim 8}$, 则大部分权重的 5~8 位为非零值。因此, 需要一个至少 8 位的量化位宽。相反, 当大部分 5~8 位为零时, 则可以近似将原始的 16 位权重视为 4 位 (1~4 位) 方式。量化策略与 $\text{id}(\cdot)$ 结果之间的关系列在表 4-3 中。

$\text{id}(w^*_{5 \sim 8}, t_{q,5 \sim 8})$	$\text{id}(w^*_{9 \sim 16}, t_{q,9 \sim 16})$	量化位宽 (比特)
+	+	16
+	-	8
-	*	4

表 4-3 id 函数与量化位宽的对应关系

3. 选择性权值聚合: 在这部分中, 作者寻求减少公式 4-2 中网络参数传输的有效比例 $\gamma_{i,r}^j$ 的机会。受到 FedMask^[51] 的启发, 该方法忽略了不重要的通道, 并仅对应重要的权重, 本文此处用到的方法与之基本类似: 只传输本地训练中的卷积核的重要部分。例如, 如图 4-2 所示 (左侧): 对于第 r 轮的特定层, 选择性传输意味着不下载或上传由公式 4-3 和公式 4-4 计算的完整 MBV2 块中的权值。相反, 根据表 4-2 中的映射关系选择对应零散的部分。在这种情况下, client_i 、 client_j 和 client_k 分别选择 $(3 * 3, 3)$ 、 $(3 * 3, 6)$ 和 $(5 * 5, 6)$ 的 MBV2 块。其中 $(k * k, e)$ 表示具有卷积核大小为 k 、扩张比例为 e 的 MBV2 块。然后, FAQS 仅传输具有相应形状的块的部分, 并忽略掉其他不重要的值。通过这种方式, 可以排除大量的无关紧要的权重, 是的比例系数 $\gamma_{i,r}^j$ 可以减小到较小的数值。在权重聚合方面, 仅计算相应重叠形状的卷积核进行聚合。因此, $(3 * 3, 3)$ 部分在所有设备节点之间共享, $(3 * 3, 6 \setminus 3)$ 部分由 client_j 和 client_k 下载, 而 $(5 * 5 \setminus 3 * 3, 6)$ 部分仅由 client_k 下载使用。因为不需要在每一轮中为所有设备节点传输完整的 MBV2 块, 所以 $\gamma_{i,r}^j$ 大大减少, 从而提高了通信效率。

4.2.3 整体算法流程

如算法 4-1 所示, FAQS 整体是一个两阶段算法。首先, 采用一个两层嵌套循环以较高的通信效率生成个性化的本地模型。其次, 在最后一轮局部搜索与权重聚合之后, 每个本地模型被以贪心的策略采样出来, 即含有网络结构参数 A_i 、量化参数 Q_i 和权重 W_i 的最终模型 P_i 。具体来说, FAQS 首先使用可训练参数 θ_i 来初始化每个设备节点 i , 包括权重 w_i^0 和结构参数阈值 t_i^0 。对于每一轮迭代, 每个设

Algorithm 1: 联邦网络结构与量化策略搜索 (FAQS)

输入: 局部搜索的 epoch 数量 E ; 联邦训练的总轮数 T ; 设备节点数量 K ; 节点本地训练集 D_i^{train} 以及节点定制化的帕累托系数 $(\alpha_i, \beta_i, \gamma_i)$;

输出: 每个节点生成的定制化模型 P_i , 包含结构 A_i 、量化位宽 Q_i 和权重 W_i

主函数 $Main(E, T, K, D_i^{train}, (\alpha_i, \beta_i, \gamma_i))$:

初始化设备节点参数 θ_i^0 ;

for $t = 1, 2..T$ **do**

for $i = 1, 2..K$ **in parallel do**

| 经过 E 轮局部搜索 $LocalSearch(\theta_i^{t-1}, D_i^{train}, \alpha_i, \beta_i, \gamma_i, E)$ 得到 $\tilde{\theta}_i^t$

end

| 将 $\tilde{\theta}_1^t, \dots, \tilde{\theta}_k^t$ 选择性权重聚合与下放得到新一轮的 $\theta_1^t, \dots, \theta_k^t$

end

for $i = 1, 2..K$ **in parallel do**

| 设备节点局部微调获得模型参数 P_i

end

局部搜索 $LocalSearch(\theta, D^{train}, \alpha, \beta, \gamma, E)$:

从 θ 中提取权值参数 w 与阈值参数 t

for $e = 1, 2..E$ **do**

for $batch(x, y) \in D^{train}$ **do**

比特共享量化: 根据公式 4-6、4-7、4-8 由 w 、 t_q 计算得出 w_q

权值共享: 根据公式 4-3、4-4、4-5 由 w_q 、 $t_{k,e}$ 计算得出 w

梯度下降: $w \leftarrow w - \eta_w \nabla_w L(batch(x, y), \alpha, \beta, \gamma, \hat{w})$

$t \leftarrow t - \eta_t \nabla_t L(batch(x, y), \alpha, \beta, \gamma, \hat{w})$

end

根据表 4-2 由 w 选择上传的权值部分 \hat{w} , 并将 $\hat{\theta} = (\hat{w}, t)$ 上传至服务器节点

end

算法 4-1 联邦网络结构搜索与量化策略搜索

设备节点先在本地进行网络结构和量化联合搜索。对应的局部搜索过程为, 首先使用公式 4-5 和公式 4-7 将原始权重参数以固定的 16 位格式进行量化。然后, 将输出的量化权重使用公式 4-3 和公式 4-4 转换为原始超核中不同部分的组合(表示为 $w-hat$)。将 $w-hat$ 视为内核权重的直接形式, 并计算输出 $output=Conv(\hat{w}, x)$ 。每个节点上的损失函数是一个由 $w-hat$ 和加权系数($\alpha_i, \beta_i, \gamma_i$)参数化的函数, 其中 $w-hat$ 是 w 和 t 函数, 意味着节点损失函数对 w 和 t 完全可微。因此, 可以直接在 w 和 t 上使用小批量梯度下降算法。随后, 根据表 4-2 选择每层的实际传输的 w , 并将它们传输到主机节点进行聚合。当完成最后一轮本地搜索时, FAQS 根据当前的 w 和 t

依照表 4-2 和表 4-3 进行网络结构与量化策略的选择，最终获得对应的 A_i 、 Q_i 和 W_i 即最终本地模型 P_i 。

4.3 算法评估

本工作参照的实验为最基本的图片分类任务，在分布式计算环境中使用九个节点进行 FAQS 的实现，每个节点都配备了一个 GPU（NVIDIA Tesla V100），其中一个节点代表主机节点，八个节点代表设备节点。

4.3.1 实验设置

为了保证公平的对比，本算法参照 FL-NAS^[43]中最常用的数据集 CIFAR10 来对图片分类任务进行对比分析。为了生成符合要求的非独立同分布的数据，采用工作^[43]中的 LDA 分布来制作数据，并设置参数 $\alpha=0.2$ 。

对于神经架构搜索空间，本搜索框架基于在^{[2][52]}中使用的类 MBV2 基础块组成的搜索空间。具体来说，MBCConv 层的类型主要由卷积核大小 k 和扩展系数 e 决定。在本实验中设定卷积核大小为(3,5)和扩展系数为(3,6)。基础网络由 4 个大块组成，每个大块由 4 个 MBCConv 块组成。为了降低问题的复杂性，只对权重进行量化，并假设 MBCConv 块内每层共享相同的量化位宽。量化位宽的选择仅限于(4, 8, 16)比特。

4.3.2 训练以及通信效率分析

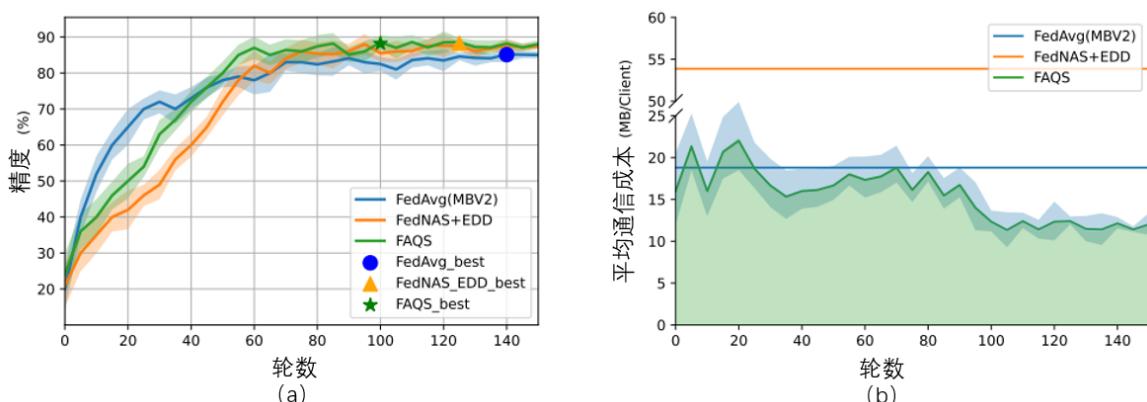


图 4-4 三种算法的 (a) 平均精度曲线 (b) 每轮的平均通信量

为了与 FedAvg 和 FedNAS+EDD 进行公平比较，首先在没有考虑硬件感知偏好的条件下 ($\alpha = 1$, $\beta = \gamma = 0$) 进行了简单的实验（为 FedAvg 不使用延迟或模型

大小分析)。如图 4-4 (a) 所示, 由于没有额外的网络结构参数需要学习, FedAvg 的收敛速度明显比 FedNAS+EDD 和 FAQS 快。然而, 在足够的训练轮次之后(约 100 轮), FedNAS+EDD 和 FAQS 在最终的 Top 1 准确率方面(88.3、88.2)表现优于 FedAvg(85.1)。与 FedNAS+EDD 相比, 得益于权值共享与比特共享, FAQS 能以更快的速度收敛到最优值。在图 4-4 (b) 中绘制了 FAQS 每轮的平均通信量曲线, 可以通过曲线下的面积计算总通信成本。如图 4-5 所示, 与普通的 FedAvg 相比, FAQS 每轮平均减少了 1.58 倍的通信成本, 在总通信方面与 FedNAS+EDD 框架相比减少了约 4.51 倍

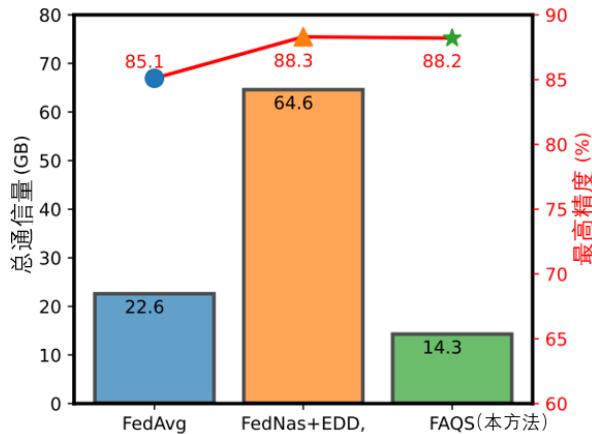


图 4-5 三种方法的通信量以及精度对比

4.3.3 个性化硬件感知的网络结构分析

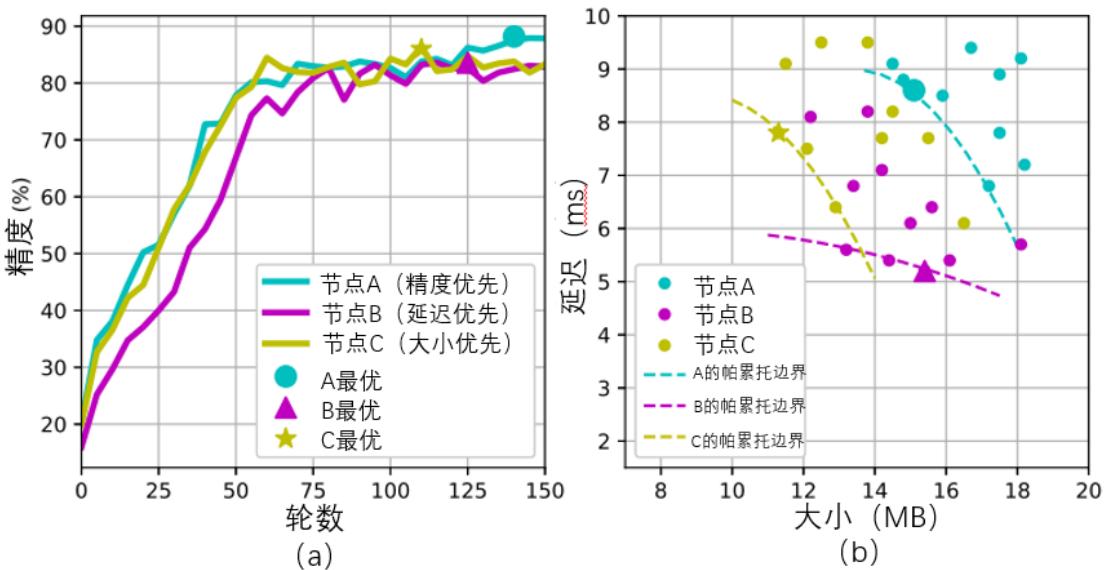


图 4-6 三个节点的 (a) 硬件感知型模型精度曲线 (b) 延迟与模型大小散点图

为了展示能生成个性化硬件感知型模型的能力, 本工作为每个节点使用了 8 组

不同的帕累托损失函数（Pareto loss），对应于 8 组不同的(α ， β ， γ)值。通过运行 150 轮 FAQS，记录了在三个特殊的设备节点上准确率曲线以及延迟和模型大小构成的散点图。节点 A 的偏好参数为(0.8, 0.1, 0.1)，因此是更看重准确率而不是延迟和模型大小的任务节点；节点 B 的偏好参数为(0.4, 0.5, 0.1)，可看出是更看重延迟的；节点 C 的偏好参数为(0.4, 0.1, 0.5)，偏好小模型。如图 4-4 (a) 所示，节点 A 搜索到的模型在准确率方面明显优于节点 B 和节点 C。同时，图 4-4 (b) 显示了节点 B 和节点 C 所对应的 帕累托边界附近的高性能模型，节点 B 的边界更贴近横轴（延迟优先）而节点 C 的帕累托边界更靠近纵轴（更小的模型）。

模型	FedAvg (MBV2)	FAQS _A (0.8,0.1,0.1)		FAQS _B (0.4,0.5,0.1)		FAQS _C (0.4,0.1,0.5)	
参数	(k,e)	(k,e)	q	(k,e)	q	(k,e)	q
L ₁ (R)	(3,6)	(3,6)	16	(5,6)	16	(3,6)	16
L ₂	(3,6)	(3,3)	16	(5,3)	16	(3,3)	16
L ₃	(3,6)	(3,3)	16	跳过	\	(3,3)	16
L ₄	(3,6)	(3,3)	8	跳过	\	跳过	\
L ₅ (R)	(3,6)	(5,6)	16	(5,6)	16	(3,6)	16
L ₆	(3,6)	(5,3)	16	(5,6)	16	(3,6)	8
L ₇	(3,6)	(3,3)	8	跳过	\	跳过	\
L ₈	(3,6)	(3,3)	8	跳过	\	(3,3)	8
L ₉ (R)	(3,6)	(5,6)	16	(5,6)	16	(3,6)	16
L ₁₀	(3,6)	(5,3)	16	(5,6)	16	(3,3)	8
L ₁₁	(3,6)	(3,3)	16	跳过	\	(3,3)	8
L ₁₂	(3,6)	(3,3)	8	跳过	\	(3,3)	8
L ₁₃ (R)	(3,6)	(5,6)	16	(5,6)	16	(3,6)	16
L ₁₄	(3,6)	(5,6)	8	(5,6)	16	(3,6)	8
L ₁₅	(3,6)	(3,6)	16	(5,6)	16	(3,6)	16
L ₁₆	(3,6)	(3,6)	16	(5,6)	16	(3,6)	16
精度(%)	85.1	88.2		83.6		86.1	
延时(ms)	8.9	8.6		5.2		7.8	
大小(MB)	19.2	15.1		15.4		11.3	

表 4-4 不同节点模型的结构、量化位宽、最终指标对比
(其中 R 为下采样层，k、e、q 分别为卷积核大小、扩展系数、量化位宽)

搜索到的结构和量化策略详见表 4-5，其中详细列出了它们的各项硬件指标。简要分析来看，由于节点 A 更看重准确率，它倾向于搜索深度和宽度较大的结构，并倾向于使用长位宽（16 比特）来作为自己各层的量化策略。节点 B 主要优化目标是延迟，因此它更倾向于宽度较大但较浅的架构，因为 GPU 设备在并行计算方面表现出色，使用宽而浅的模型能在保证一定精度的同时充分利用 GPU 的并行计算，提升推理速度，降低延迟。对于优先选择小模型的节点 C，许多层采用窄而浅的块类型来减小模型大小。在[6]中也有类似的发现，即模型大小优先和延迟优先的模型都具有相当浅的层，但在最开始的几层都会采用较长的量化位宽，因为网络最开始几层的特征输入相对较小且需要更高的分辨率来保真精度。此外，对于每个下采样的层，三个节点都倾向于使用更大的 MBCConv 和更长的量化位宽，可见下采样层对保证模型精度十分关键。

4.4 本章小结

本章将原先的单节点任务中的算法扩展至了多节点的场景中，提出了一种高通信效率的联邦网络结构与量化策略搜索。为尽可能地减少每轮搜索迭代中通信量的开销，本章提出的算法从选择性权重聚合、超核搜索、混合精度比特共享三个维度极致地减少了上传和下载的数据量。通过调整每个节点上帕累托函数内的权重，本算法可以有效搜索出节点异构的网络结构与量化策略，同时满足所在节点各优化目标的优先级顺序。本算法与传统的基于 FedAvg 加可微分搜索的算法相比，节省了大约 77% 的理论数据传输开销。

第五章 总结与展望

5.1 论文工作总结

随着机器学习技术的不断发展和普及，越来越多的企业和个人开始关注如何将其应用到各种现实世界的问题中，以提高效率、减少成本、优化决策等。然而，机器学习需要大量的数据和计算资源，并且还需要熟练的数据科学家和机器学习专家进行模型开发和调优，这些因素限制了机器学习技术的推广和应用范围。为了解决这些问题，诞生了 AutoML 这个领域，其主要目的是通过自动执行机器学习中的一些繁琐任务来降低机器学习的门槛，使更多的人可以使用和受益于机器学习技术。相较于之前基于云端训练加上终端部署的方式进行的自动化机器学习模式，片上自动化机器学习能够充分利用片上实时反馈的信号从而提升模型的生成质量。通过在自动机器学习过程中考虑这些硬件特定因素，片上自动机器学习可以帮助优化训练出的模型在目标硬件上的性能和效率，从而提高整个系统的性能。另一个推动片上自动化机器学习发展的动机是数据隐私。在传统的云端训练中，数据需要上传到云端进行训练，由于数据隐私的问题，某些敏感数据无法上传到云端进行训练。而采用片上自动化机器学习算法可以在不泄露数据隐私的前提下实现模型的训练和推理，因此在数据隐私保护方面具有一定的优势。片上自动化机器学习已经被应用于各种领域，如智能音箱、智能摄像头、自动驾驶汽车和医疗设备等，使得这些设备能够实时进行识别、检测和决策，提高了设备的智能化程度。

然而将自动化机器学习算法集成到芯片或硬件加速器中，需要面临设计复杂度的挑战。这涉及到硬件架构、芯片设计、算法优化等多个方面，需要综合考虑多种因素。边缘设备和嵌入式系统通常具有受限的计算能力和存储容量，这意味着需要开发高效的算法和技术来满足资源限制。同时为了在边缘设备和嵌入式系统上实现实时推断，需要具有高准确度的模型。此外，在边缘设备和嵌入式系统上运行机器学习模型通常需要大量的能源，这可能会影响设备的电池寿命和使用体验。最后，片上自动化机器学习需要考虑到长期的可维护性。随着技术和算法的发展，需要对系统进行升级和更新，因此需要设计可扩展的系统架构和算法框架。

本文主要从算法的角度对片上自动化机器学习技术进行了深入研究，从神经网络结构搜索与模型量化这两个子领域切入，从多目标优化、多领域联合优化、多节点同时优化这三个角度提出了一套高计算、通信效率的联合优化算法。本论文的主要内容与创新点如下：

1. 本文首先从自动混合精度量化这个单独领域出发，设计了一种高效的基于 Q-Learning 的离散点多目标优化方法。基于自研的 Evolver 可重构芯片架构，主要优化目标除了模型的预测精度外，还有模型在芯片上的推理延时以及推理能耗。相对于云端采用模型大小作为代理信号搜索方法，本方法在最终的模型生成质量上综合指标拥有近 27% 的评分提升。同时为了缓解片上有限的计算资源的与较高计算负载的冲突，本方法引入了多种优化技巧包括奖励函数设计、异常点提前判别等。最终在可接受时间开销下，仅仅只需要百轮的模型迭代就能够搜出满足要求的混合精度量化策略。本工作相关内容发表在了 2020 年集成电路领域顶刊 JSSC 上，对应本论文中的第二章节。
2. 其次，本论文在第二章单领域多目标优化的算法基础上，将优化的领域扩展到了网络结构的层次，并提出了一种针对高效的神经网络结构与混合精度量化同时搜索的联合优化方法。通过结合基于可微分的网络结构搜索与基于量化感知训练的量化模式，通过采用帕累托奖励函数使得搜索出的模型能适应多目标优化的任务。同时该方法能在终身学习的场景中使用，在对应的分阶段 ImageNet 数据集中，采用本方法的多步模型训练相比于整体的单步模型训练，在最终模型权重分布上几乎没有过大的变化。该工作相关内容展示在了 2021 年 DAC WIP poster 上，对应本论文的第三章节。
3. 最后，在前面多领域优化的基础上，本论文又进一步的将 Automl 扩展至了多节点联合优化的场景，即联邦学习的任务场景中。为解决多节点网络结构搜索时，不同终端节点之间通信量过高的问题，本论文提出了一种高效的多节点网络结构与量化的算法。从三个角度对传统方法进行了进一步的优化，包括搜索策略的选择、量化方法的选择、权重聚合模式的选择。通过选择性权重聚合、超核搜索、混合比特共享三个维度使整个模型的通信量与传统的 FLNAS^[44]相比获得了近 77% 的通信量缩减。在最终模型的产出上，通过设置不同的节点帕累托函数，可以生成出形式不同且符合节点要求的异构网络模型。该工作相关内容将要展示在 2023 年 DAC WIP poster 上，对应本论文的第四章节。

5.2 未来工作展望

本论文研究的重点是缓解在片上自动化机器学习领域中较高的计算负载与有限的计算资源之间的矛盾。以网络结构搜索与模型量化为切入点，由浅入深分别在多目标优化、多领域优化、多节点优化三个任务层次上对片上自动化机器学习算法

进行了一定程度的探索。然而网络结构搜索与模型量化只是自动化机器学习中有限的两个分支，不同的任务场景与算力数据规模都拥有不同的最佳自动化的策略，如何将不同的自动机器学习算法结合至一个系统中依然非常有挑战性的事情。本文对未来的工作展望主要从以下两个方面展开：

1. 首先是研究范围的进一步扩展。对于已经涉及到的网络结构搜索与量化，将现有的基于 MobileNetV2^[49]的基础骨骼模型扩展至更加自由与灵活的基于微观搜索空间的网络模型搜索，将涉及到的基于 Min-Max 的混合精度量化感知训练均匀 QAT 的量化模式扩展至更低比特或者是更加复杂的非均匀量化模式中。同时将涉及的联合优化领域扩展至更多的维度，包含自动特征提取，自动网络剪枝等等。
2. 其次是研究范式的探索，目前主流的自动化机器学习还是带有较强的经验偏置，以网络结构搜索为例，搜索空间的设计通常是需要较强的人工专家经验的。所以从更广义的自动化机器学习的角度来说，当前领域中的做法实际是将调参的维度从原始的权重，转换到了更高层搜索空间的参数设计，并没有完全跳脱原本的这种调参的模式。最近有一些工作从模型生长的模式去进行网络模型结构的自动化搜索，相比于之前在优先的搜索空间内进行探索的方法而言更具有普适性，是一个非常有价值的研究方向。

参考文献

- [1] Bergstra J, Bengio Y. Random search for hyper-parameter optimization[J]. Journal of machine learning research, 2012, 13(2).
- [2] Baker B, Gupta O, Naik N, et al. Designing neural network architectures using reinforcement learning[J]. arXiv preprint arXiv:1611.02167, 2016.
- [3] Zoph B, Le Q V. Neural architecture search with reinforcement learning[J]. arXiv preprint arXiv:1611.01578, 2016.
- [4] Liu C, Zoph B, Neumann M, et al. Progressive neural architecture search[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 19-34.
- [5] Noy A, Nayman N, Ridnik T, et al. Asap: Architecture search, anneal and prune[C]//International Conference on Artificial Intelligence and Statistics. PMLR, 2020: 493-503.
- [6] Liu H, Simonyan K, Yang Y. Darts: Differentiable architecture search[J]. arXiv preprint arXiv:1806.09055, 2018.
- [7] Tu F, Wu W, Wang Y, et al. Evolver: A deep learning processor with on-device quantization–voltage–frequency tuning[J]. IEEE Journal of Solid-State Circuits, 2020, 56(2): 658-673.
- [8] Chen H, Wang Y, Liu L, et al. HQNAS: Auto CNN deployment framework for joint quantization and architecture search[J]. arXiv preprint arXiv:2210.08485, 2022.
- [9] Chen H, Wang Y, Liu L, et al. FAQS: Communication-efficient Federate DNN Architecture and Quantization Co-Search for personalized Hardware-aware Preferences[J]. arXiv preprint arXiv:2210.08450, 2022.
- [10] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84-90.
- [11] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
- [12] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
- [13] Collobert R, Weston J. A unified architecture for natural language processing: Deep neural networks with multitask learning[C]//Proceedings of the 25th international conference on Machine learning. 2008: 160-167.
- [14] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks[J]. Advances in neural information processing systems, 2014, 27.
- [15] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.
- [16] Carlini N, Wagner D. Audio adversarial examples: Targeted attacks on speech-to-

- text[C]//2018 IEEE security and privacy workshops (SPW). IEEE, 2018: 1-7.
- [17] Yuan X, Chen Y, Zhao Y, et al. Commandersong: A systematic approach for practical adversarial voice recognition[C]//27th {USENIX} Security Symposium ({USENIX} Security 18). 2018: 49-64.
- [18] Tong S, Garner P N, Bourlard H. Multilingual training and cross-lingual adaptation on CTC-based acoustic model[J]. arXiv preprint arXiv:1711.10025, 2017.
- [19] Cheng H T, Koc L, Harmsen J, et al. Wide & deep learning for recommender systems[C]//Proceedings of the 1st workshop on deep learning for recommender systems. 2016: 7-10.
- [20] Zhang J, Xie Y, Wu Q, et al. Medical image classification using synergic deep learning[J]. Medical image analysis, 2019, 54: 10-19.
- [21] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.
- [22] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
- [23] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial networks[J]. Communications of the ACM, 2020, 63(11): 139-144.
- [24] Creswell A, White T, Dumoulin V, et al. Generative adversarial networks: An overview[J]. IEEE signal processing magazine, 2018, 35(1): 53-65.
- [25] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial intelligence and statistics. PMLR, 2017: 1273-1282.
- [26] Zhou S, Wu Y, Ni Z, et al. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients[J]. arXiv preprint arXiv:1606.06160, 2016.
- [27] Gysel P M. Ristretto: Hardware-oriented approximation of convolutional neural networks[M]. University of California, Davis, 2016.
- [28] Judd P, Albericio J, Hetherington T, et al. Stripes: Bit-serial deep neural network computing[C]//2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2016: 1-12.
- [29] Sharma H, Park J, Suda N, et al. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network[C]//2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2018: 764-775.
- [30] Moons B, Uytterhoeven R, Dehaene W, et al. 14.5 envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi[C]//2017 IEEE International Solid-State Circuits Conference (ISSCC). IEEE, 2017: 246-247.
- [31] Chen Y H, Krishna T, Emer J S, et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks[J]. IEEE journal of solid-state circuits, 2016, 52(1): 127-138.

-
- [32] Lee J, Kim C, Kang S, et al. UNPU: A 50.6 TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision[C]//2018 IEEE International Solid-State Circuits Conference-(ISSCC). IEEE, 2018: 218-220.
 - [33] Wang K, Liu Z, Lin Y, et al. Haq: Hardware-aware automated quantization with mixed precision[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 8612-8620.
 - [34] Choi S, Shin J, Choi Y, et al. An optimized design technique of low-bit neural network training for personalization on IoT devices[C]//Proceedings of the 56th Annual Design Automation Conference 2019. 2019: 1-6.
 - [35] Wu B, Dai X, Zhang P, et al. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 10734-10742.
 - [36] Dai X, Zhang P, Wu B, et al. Chamnet: Towards efficient network design through platform-aware model adaptation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 11398-11407.
 - [37] Hubara I, Courbariaux M, Soudry D, et al. Quantized neural networks: Training neural networks with low precision weights and activations[J]. The Journal of Machine Learning Research, 2017, 18(1): 6869-6898.
 - [38] Hsu C H, Chang S H, Liang J H, et al. Monas: Multi-objective neural architecture search using reinforcement learning[J]. arXiv preprint arXiv:1806.10332, 2018.
 - [39] Lindsay R K, Buchanan B G, Feigenbaum E A, et al. DENDRAL: a case study of the first expert system for scientific hypothesis formation[J]. Artificial intelligence, 1993, 61(2): 209-261.
 - [40] Baker B, Gupta O, Naik N, et al. Designing neural network architectures using reinforcement learning[J]. arXiv preprint arXiv:1611.02167, 2016.
 - [41] Watkins C J C H. Learning from delayed rewards[J]. 1989.
 - [42] Lin L J. Reinforcement learning for robots using neural networks[M]. Carnegie Mellon University, 1992.
 - [43] Pham H, Guan M, Zoph B, et al. Efficient neural architecture search via parameters sharing[C]//International conference on machine learning. PMLR, 2018: 4095-4104.
 - [44] He C, Mushtaq E, Ding J, et al. Fednas: Federated deep learning via neural architecture search[J]. 2020.
 - [45] Garg A, Saha A K, Dutta D. Direct federated neural architecture search[J]. arXiv preprint arXiv:2010.06223, 2020.
 - [46] Stamoulis D, Ding R, Wang D, et al. Single-Path NAS: Designing Hardware-E cient ConvNets in less than 4 Hours[C]//European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. 2019.
 - [47] Li Y, Hao C, Zhang X, et al. Edd: Efficient differentiable dnn architecture and implementation co-search for embedded ai solutions[C]//2020 57th ACM/IEEE Design Automation

- Conference (DAC). IEEE, 2020: 1-6.
- [48] Wu B, Dai X, Zhang P, et al. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 10734-10742.
- [49] Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4510-4520.
- [50] Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 2704-2713.
- [51] Li A, Sun J, Zeng X, et al. Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking[C]//Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems. 2021: 42-55.
- [52] Zhu H, Jin Y. Real-time federated evolutionary neural architecture search[J]. IEEE Transactions on Evolutionary Computation, 2021, 26(2): 364-378.
- [53] Cai H, Zhu L, Han S. Proxylessnas: Direct neural architecture search on target task and hardware[J]. arXiv preprint arXiv:1812.00332, 2018.
- [54] 中国知网“AutoML”热点关键词搜索
- [55] Zoph B, Vasudevan V, Shlens J, et al. Learning transferable architectures for scalable image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 8697-8710.
- [56] Chen Y, Meng G, Zhang Q, et al. Joint neural architecture search and quantization[J]. arXiv preprint arXiv:1811.09426, 2018.