



上海交通大学学位论文

面向软硬件协同的视觉模型搜索 与优化

姓 名：包清泉
学 号：519030910402
导 师：严骏驰
学 院：电子信息与电气工程学院
学科/专业名称：人工智能
申请学位层次：学士

2023 年 6 月

**A Dissertation Submitted to
Shanghai Jiao Tong University for Bachelor Degree**

**VISUAL MODEL SEARCH AND
OPTIMIZATION FOR SOFTWARE AND
HARDWARE CO-DESIGN**

Author: Qingquan Bao
Supervisor: Junchi Yan

School of Electric Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, P.R.China
June 6th, 2023

上海交通大学 学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全知晓本声明的法律后果由本人承担。

学位论文作者签名：包清泉
日期：2023年6月6日

上海交通大学 学位论文使用授权书

本人同意学校保留并向国家有关部门或机构递交论文的复印件和电子版，允许论文被查阅和借阅。

本学位论文属于：

- 公开论文
 - 内部论文，保密 1 年/2 年/3 年，过保密期后适用本授权书。
 - 秘密论文，保密 ____ 年（不超过 10 年），过保密期后适用本授权书。
 - 机密论文，保密 ____ 年（不超过 20 年），过保密期后适用本授权书。
- （请在以上方框内选择打“√”）

学位论文作者签名：包清泉 指导教师签名：郭殿池
日期：2023 年 6 月 6 日 日期：2023 年 6 月 6 日

摘要

本文介绍了一种软硬件协同设计框架，用于高效的神经架构搜索和优化，以解决传统神经网络架构设计难以满足各种设备和应用场景需求的问题。为了克服传统神经架构搜索方法中忽略硬件平台限制和先验知识的缺点，本文提出了一种自动化神经架构搜索（NAS）方法，结合权重共享技术和 NSGA-II 遗传算法，以快速、精准地找到适合特定硬件的高效神经架构。

本文采用了一种在单次预训练超网络上采样子网络进行多目标优化的方法，该方法使用权重共享的技术加速了搜索过程，同时将模型精度和硬件性能进行了解耦，在多目标搜索阶段才引入硬件性能。本文采用 NSGA-II 遗传算法进行帕累托最优搜索，以实现快速且精准地找到适合特定硬件的高效神经架构。

本文的贡献在于提出了一种非硬件专家友好的、硬件扩展性好、多目标输出从而用户可自行根据需求选择的方法，同时还能作为在特定硬件下搜索空间的分析工具。本文在多个硬件平台上进行了评估，并在一国产 AI 加速芯片上进行了实验，演示了如何使用该方法不断改进搜索空间，以获得软硬件下的最优模型。实验结果表明，该方法在保持高准确性的同时，实现了高效的推理性能。此外，该方法在不同硬件上具有良好的泛化性能，可以为其他领域的研究和应用提供参考和借鉴。

关键词：深度学习，视觉模型，神经架构搜索，软硬件协同

ABSTRACT

This paper presents a software-hardware co-design framework for efficient neural architecture search and optimization, aiming to address the challenge of traditional neural network architecture design that struggles to meet the demands of various devices and application scenarios. To overcome the limitations of conventional neural architecture search methods that neglect hardware platform constraints and prior knowledge, we propose an automated neural architecture search (NAS) approach that incorporates weight sharing techniques and NSGA-II genetic algorithm to quickly and accurately identify efficient neural architectures that are suitable for specific hardware.

Specifically, we adopt a method that samples sub-networks on a one-shot pre-trained supernet for multi-objective optimization, utilizing weight sharing techniques to accelerate the search process while decoupling model accuracy and hardware performance until the multi-objective search stage. We apply the NSGA-II genetic algorithm to achieve Pareto optimal search and find efficient neural architectures that are compatible with specific hardware platforms.

The contribution of this paper lies in the proposal of a hardware-agnostic, highly extensible, multi-objective output approach that enables users to select models according to their needs while serving as an analysis tool for search space under specific hardware. We evaluate the proposed approach on multiple hardware platforms, including EdgeGPU, Eyeriss, and FPGA, and conduct experiments on a domestically-produced AI acceleration chip, demonstrating how this approach can continuously improve the search space to obtain the optimal model under software and hardware constraints. Experimental results indicate that the proposed approach achieves high accuracy and efficient inference performance while demonstrating good generalization performance across different hardware platforms, providing references and insights for research and application in other fields.

Key words: Deep Learning, Computer Vision, Neural Architecture Search

目 录

| | |
|---------------------------------|-----------|
| 摘 要 | I |
| ABSTRACT | II |
| 第一章 绪论 | 1 |
| 1.1 引言 | 1 |
| 1.2 本文主要研究内容 | 3 |
| 1.3 本文研究意义 | 5 |
| 1.4 本章小结 | 6 |
| 第二章 相关工作 | 7 |
| 2.1 视觉模型的发展历史 | 7 |
| 2.2 神经架构搜索 | 8 |
| 2.3 基于单次超网络训练的神经架构搜索 | 10 |
| 2.4 硬件约束下的神经架构搜索 | 11 |
| 2.5 本章小结 | 13 |
| 第三章 预备知识 | 14 |
| 3.1 神经网络基本运算和模块 | 14 |
| 3.2 搜索空间构建 | 16 |
| 3.3 多目标搜索 | 17 |
| 3.4 本章小结 | 20 |
| 第四章 单次超网络下的多目标优化搜索 | 21 |
| 4.1 问题假设和形式化 | 21 |
| 4.2 单次超网络建模 | 22 |
| 4.3 单次超网络训练 | 24 |
| 4.4 多目标遗传算法 | 25 |
| 4.5 本章小结 | 28 |
| 第五章 实验 | 29 |
| 5.1 搜索空间介绍 | 29 |
| 5.2 实验数据集 | 29 |

| | | |
|------------------------------------|---------------------------|-----------|
| 5.3 | 实验基线及训练细节 | 30 |
| 5.4 | 实验结果分析 | 33 |
| 5.4.1 | 不同数据集与硬件下算法分析 | 34 |
| 5.4.2 | NSGA 搜索过程分析 | 35 |
| 5.5 | 本章小结 | 39 |
| 第六章 | 国产芯片应用示例 | 42 |
| 6.1 | 应用芯片介绍和目标设定 | 42 |
| 6.2 | 搜索空间和大模型训练方式 | 42 |
| 6.3 | 实验结果 | 43 |
| 6.4 | 本章小结 | 44 |
| 第七章 | 全文总结 | 45 |
| 7.1 | 主要结论 | 45 |
| 7.2 | 研究展望 | 45 |
| 参 考 文 献 | 47 | |
| 攻读学位期间学术论文和科研成果目录 | 51 | |
| 致 谢 | 52 | |

第一章 绪论

1.1 引言

随着卷积神经网络（Convolutional Neural Network）的广泛使用，越来越多的人们开始需要快速推理速度和高精确度的神经网络。最初，神经网络架构是人工设计的，例如 VGGNet 和 ResNet，并且针对强大的 GPU 进行了优化，因为它们是深度 CNN 的主要计算平台。然而，这些架构及其变体被认为是标准，直到需要部署在边缘设备和标准 CPU 上成为优先事项。然而，人工为各种设备设计最佳的神经架构是昂贵的。因此，越来越多的研究人员开始关注神经架构设计的自动化。

神经架构搜索（Neural Architecture Search）是一种自动化的深度学习模型设计和训练方法，可以在最小的人工干预下实现最佳性能。它使用各种搜索算法，例如进化算法或强化学习，来探索可能的架构的大空间，并确定在给定任务中表现良好的架构。与手工设计架构相比，这可以节省大量时间和精力，并且可以产生比手工设计的模型更好的性能。

在早期的工作中，NAS 从预先定义的搜索空间开始，通常是一组操作集，并使用控制器来生成神经架构候选者。候选架构在训练集上接受训练后，再在验证集上进行评估和排名。排名将作为反馈，指导控制器生成新的候选集。当搜索过程结束时，选择最佳的神经结构并在测试集上进行测试。

根据上述流程中的关键组件，NAS 的工作可以从搜索空间、搜索策略和评估策略方面进行分类。早期的 NAS 用离散的搜索策略，如随机搜索、强化学习（Reinforcement Learning）^[1]、进化算法（Evolutionary Algorithm）^[2] 和贝叶斯优化（Bayesian Optimization）^[3]，搜索全局空间中所有可能的组件。然而，在全局空间搜索需要过多的计算成本。例如，基于强化学习的神经架构搜索需要 2000 个 GPU 天来获得 CIFAR-10 和 ImageNet 的最先进架构，而进化算法需要 3150 个 GPU 天。同时，他们忽略了现有优秀神经架构设计中的人类先验因素。

为了克服这一点，一种广泛使用的方法是使用模块化搜索空间而不是全局空间。这得到了人工设计的启发，并堆积了一定的结构（这被称为细胞 cell）来构建而架构。NASNet^[4]是最早引入基于单元搜索空间的作品之

一。它将搜索空间分类为两种类型的单元，即正常单元和缩减单元。前者保持输入特征的空间分辨率，而后者用于降低空间分辨率。一般来说，多个重复的正常单元尾随一个缩减单元构成一个可重复的结构，被堆叠起来形成最终的架构。

除了采用模块化搜索空间，研究人员在使用连续搜索策略方面取得了很大进展，因为离散搜索策略将 NAS 视为黑箱优化问题，并带来更高性能成本。DARTS^[5]通过使用一个有向无环图来表示细胞结构，以潜在的表征为节点，所有可能的操作为边，来放松这个问题。DARTS 用基于梯度的方法联合优化可能操作的权重和架构的权重，最后选择具有最大边缘权重的操作。DARTS 显示了与离散方法相比具有竞争力的性能，而且搜索效率更高。然后，很多后续工作试图在超网络设计 (I-DARTS^[6])、内存成本 (PC-DARTS^[7]) 以及搜索阶段和评估阶段的性能差距 (P-DARTS^[8]) 等方面改进 DARTS。

尽管像 NAS-RL^[1] 和 DARTS^[5] 这样的 NAS 在标准数据集（如 CIFAR-10 和 ImageNet）的准确性和内存成本方面成功地找到了比人类设计的网络架构更好的模型，但在各种硬件设备上部署这些模型可能是困难的。这是由几个原因造成的。

- 资源限制。训练深度神经网络需要大量的计算资源，如内存、延迟和处理能力。这些资源通常被限制在硬件设备上，特别是在边缘设备上，如智能手机和无人机。NAS 设计的架构可能是资源密集型的，在这些设备上部署可能不实用。
- 特定硬件的体系结构。为通用 CPU 搜索到的架构可能不是特定类型硬件的最佳架构。例如，为 CPU 设计的架构可能不适合在 GPU 或 TPU 上部署。另一个例子是，在 FBNet 中，为 iPhone X 搜索的 CNN 架构 (19.84 毫秒的运行时间) 在三星 S8 上显示出额外的 3.49 毫秒的执行；为三星 S8 搜索的架构 (22.12 毫秒的运行时间) 在 iPhone X 上显示出额外的 5.12 毫秒的执行^[9]。所以在不同设备上的次优性能可能是 NAS 在实际应用的一个重要障碍。

为了进一步分析这个挑战，即硬件是如何限制了 NAS 的搜索空间，以下是一些可能的原因，

- 少数硬件平台对运算器有先决条件或特殊要求，例如，Hexagon v62 DSP 喜欢运算器中 32 的倍数的过滤器尺寸；
- 不同设备上的不同运算子可能有非常不同的延迟；
- 少数运算器的某些过滤器尺寸或架构的某一部分可能超过资源约束设备的内存容量，例如，MobileNetv2 的峰值内存成本不满足一般微控制器（Micro Controller Unit）中仅仅 320 kB 的要求^[10]。

为了克服不同的资源限制，满足不同的性能要求，研究人员将硬件感知的 NAS 放在各种设备上，如 MCU、移动/边缘/高端 CPU/GPUS 和 AI 加速器。MCUNet^[10]编码了 TinyNAS，这是一个两阶段的 NAS，首先根据 FLOPs 在 MCU 的内存约束下缩小搜索空间，然后执行一次 NAS 以找到好的模型；TinyEngine 是一个内存高效的推理库，既减少内存又加快推理时间。HardCoRe-NAS^[11]在严格的约束条件下解决搜索问题，而不是在损失函数上应用软惩罚，并在英特尔至强 CPU 上实现了各种延迟水平的巨大性能。MONAS 使用奖励函数混合硬约束，如功率、能量和乘加运算数（MAC），其中 MAC 被视为衡量功率的替代指标。

1.2 本文主要研究内容

本文旨在研究如何在软硬件协同设计的框架下实现高效的神经架构搜索和优化。具体而言，我们将关注以下问题：

- 如何在搜索过程中高效评估架构的性能，以便更快地收敛并避免不必要的计算成本？
- 如何将硬件性能纳入搜索目标，以生成适合特定硬件的高效架构，并兼顾可扩展性和易用性？

在本文中，我们将介绍一种软硬件协同设计框架，该框架可以实现自动化的神经架构搜索和优化，并在特定硬件上实现高效的推理性能。具体而言，我们将采用一种在单次预训练超网络（One Shot Pretrained Supernet）上采样子网络进行多目标优化的方法，该方法使用权重共享的技术加速了搜索过程，同时将模型精度和硬件性能首先进行了解耦，在多目标搜索阶

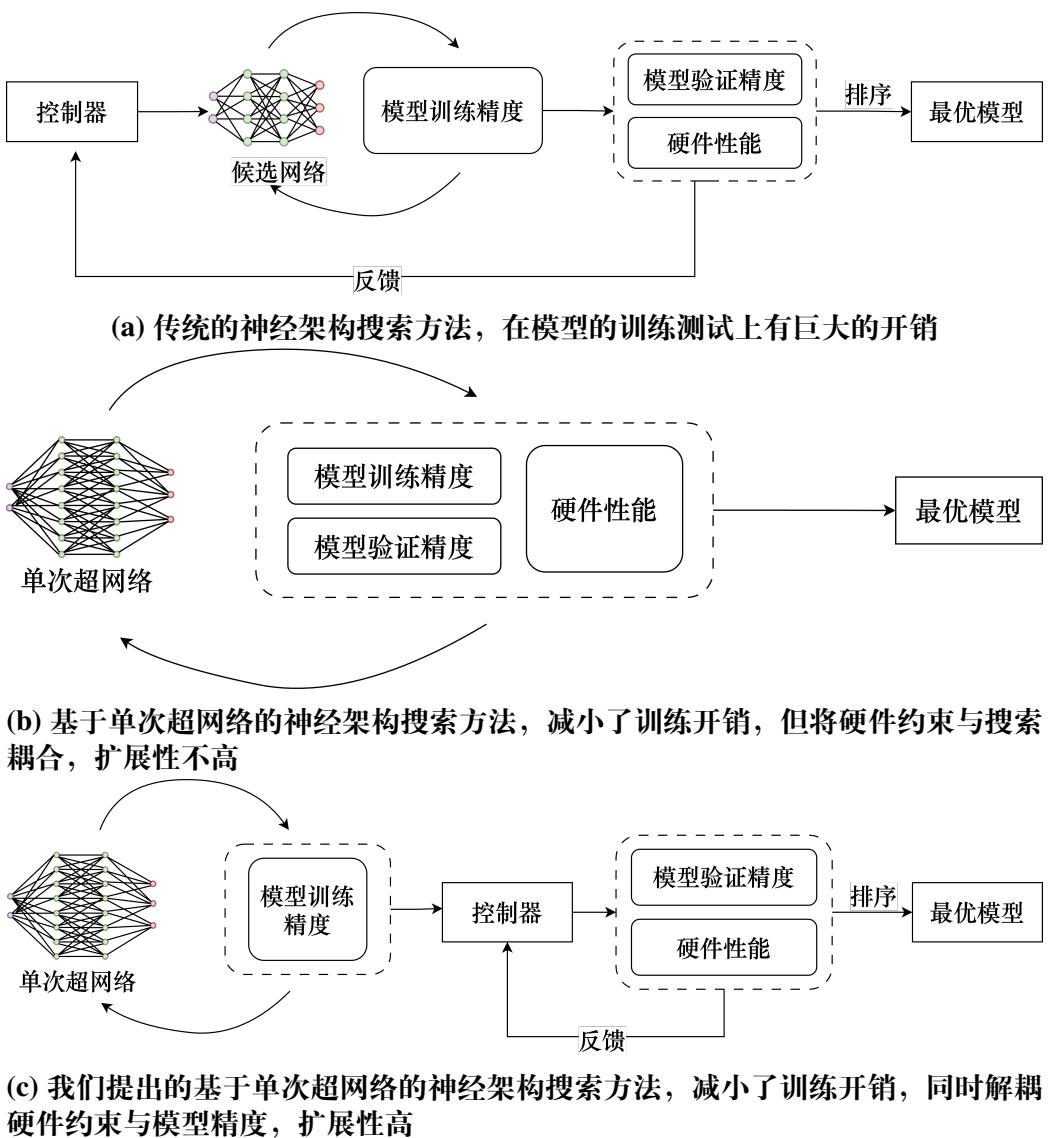


图 1-1 各类方法流程比较

段才引入硬件性能，同时方便了硬件小白对于特定硬件生成适合特定硬件的高效架构的需求。

我们使用了一些公共数据集，如 CIFAR-10 和 ImageNet，在多个硬件平台上对我们提出的方法进行评估，包括 EdgeGPU、Eyeriss、FGPA 等。实验结果表明，我们的方法可以在保持良好准确性的同时，实现高效的推理性能。同时，我们的方法在不同硬件上具有良好的泛化性能，可以生成适合各种硬件的高效架构。最后，我们在国产 AI 加速芯片上进行了实验，演示了如何使用我们的方法不断改进搜索空间，以获得软硬件下的最优模型。我们的实验结果表明，我们的方法可以实现在国产 AI 加速芯片上实现高效的推理性能，同时为用户提供了一种可靠的方法来生成适合的架构。

在本文的主体，我们将详细介绍我们提出的软硬件协同设计框架，包括搜索空间的定义、硬件约束、多目标优化。接下来，我们将讨论我们的实验设置和实验结果。最后，我们将对我们的工作进行总结并提出未来的研究方向。

1.3 本文研究意义

本文旨在研究如何在软硬件协同设计的框架下实现高效的神经架构搜索和优化。随着人工智能技术的不断发展和应用场景的不断拓展，如何快速地设计出高效的神经网络架构，已经成为一个备受关注的问题。在实际应用中，不同的硬件平台和应用场景需要不同的网络架构，因此如何将硬件约束纳入搜索空间，以生成适合特定硬件的高效架构，也是一个亟待解决的问题。本文的研究意义在于：

首先，我们提出的软硬件协同设计框架可以大大提高神经架构搜索的效率和准确性，可以兼容现有的各种基于单次大模型训练的、未考虑硬件性能的 NAS 方法。具体而言，我们的方法使用权重共享技术，可以减少不必要的计算成本，同时结合多目标遗传算法 NSGA-II 进行帕累托最优搜索，可以在满足特定硬件要求的前提下获得更优秀的架构。具有很高的实际应用价值。

其次，我们提出的方法在多个硬件平台上进行了评估，证明了其具有较好的泛化性能，可以生成适合各种硬件的高效架构。我们还在国产 AI 加速芯片上进行了实验，演示了如何使用我们的方法不断改进搜索空间，以

获得软硬件下的最优模型。这些实验结果表明，我们的方法为实际应用提供了一种可靠的解决方案。

最后，本文提出的方法不仅可以在计算机视觉领域得到广泛应用，也可以为其他领域的研究和应用提供参考和借鉴。例如，在无人驾驶、医疗影像分析等领域，高效的神经网络模型也是一个关键问题，而本文提出的软硬件协同设计框架可以为这些领域的研究和应用提供一个新的思路和方向。因此，本文的研究成果对我国在人工智能领域的发展提供了重要的支持和帮助，具有重要的理论和实际应用价值。

1.4 本章小结

本章介绍了如何在软硬件协同设计的框架下实现高效的神经架构搜索和优化。我们首先介绍了神经网络架构搜索的背景和意义，并指出了不同硬件平台和应用场景需要不同的网络架构这一问题。接着，我们提出了一种软硬件协同设计框架，该框架可以在搜索过程中将硬件约束纳入搜索空间，以生成适合特定硬件的高效架构。具体而言，我们采用了一种在单次预训练大网络上采样子网络进行多目标优化的方法，使用权重共享技术加速了搜索过程，同时结合多目标遗传算法 NSGA-II 进行帕累托最优搜索，可以在满足特定硬件要求的前提下获得更优秀的架构。

我们还在多个硬件平台上对我们提出的方法进行了评估，实验结果表明，我们的方法可以在保持良好准确性的同时，实现高效的推理性能，具有很好的泛化性能。我们在国产 AI 加速芯片上进行了实验，演示了如何使用我们的方法不断改进搜索空间，以获得软硬件下的最优模型。最后，我们指出了本文提出的方法可以为其他领域的研究和应用提供参考和借鉴，并为我国在人工智能领域的发展提供了重要的支持和帮助。

第二章 相关工作

2.1 视觉模型的发展历史

视觉模型是计算机视觉领域中的重要组成部分，被广泛应用于处理图像或视频数据。在视觉模型的发展历程中，学者们不断提出一系列手工设计的卷积网络模型（Convolutional Neural Network）来提升性能。其中，Alexnet^[12]是广为人知最早的运用于图像分类的模型，它采用了卷积、池化、非线性激活函数等基本操作。然而，这种模型存在着一些问题，如模型深度有限、准确率低、模型参数过多、延时高、内存占用大等。为了解决这些问题，学者们进一步手工设计提出了一些新的模型结构，例如 ResNet^[13]、MobileNet^[14-15]、ShuffleNet^[16] 等。

ResNet 是由何凯明等人在 2015 年提出的一种残差网络结构，它通过增加跨层连接来解决深度模型训练中的梯度消失和梯度爆炸问题。这种结构能够训练出非常深的模型，在图像分类、物体检测等任务上取得了显著的性能提升。在 ImageNet 上进行预训练的 ResNet 常作为各类下游图像任务的基线和骨架。

MobileNet 是 Google 在 2017 年提出的一种轻量级卷积神经网络，它采用了深度可分离卷积（Depthwise Separable Convolution）来减少模型的参数数量和计算量，以适应移动设备等轻量级场景。深度可分离卷积是将普通卷积分成深度卷积和逐点卷积两部分，以降低计算复杂度。这种结构在准确率和模型大小之间取得了很好的平衡。

ShuffleNet 是一种适用于轻量级设备的网络结构，它主要采用了分组卷积（Group Convolution）和通道重排（Channel Shuffling）等技术来降低计算量。其中，分组卷积是将卷积核分成多组进行卷积操作，以降低计算复杂度；通道重排则是通过通道的重新排列来增加信息的交互性。这种结构在达到更好性能的同时，具有更少的计算复杂度。

然而，这些结构仍然需要人工设计，这种设计过程需要大量的经验和时间，并且设计出的模型不一定是最优的。为了自动化地设计视觉模型架构，学者们提出了神经架构搜索。

2.2 神经架构搜索

神经架构搜索（Neural Architecture Search，简称 NAS）是自动机器学习（Auto Machine Learning）的一个分支，旨在自动化模型架构的选择。早期的视觉模型发展历史中，学者们通过手工设计各种卷积神经网络模型来提升性能。然而，这种设计过程需要大量的经验和时间，并且设计出的模型不一定是最优的。为了自动化地设计视觉模型架构，神经架构搜索应运而生。

神经架构搜索通常从一组预定义的操作集开始，基于这些操作集构成的搜索空间，控制器（Controller）生成大量候选神经结构。然后，在训练集上对这些候选结构进行训练，并根据它们在验证集上的准确性进行排名。这些候选神经结构的排名信息将被用作反馈信息来调整搜索策略，从而获得一组新的候选神经结构。当达到终止条件时，搜索过程终止以选择最佳神经结构。最终被选定的神经结构会在测试集上进行性能评估。

早期的神经架构搜索方法基本遵循以上的框架进行。在 NAS-RL^[1] 中，神经网络的架构被视作一个可变长度的字符串，从而，我们可以使用序列生成模型，如循环神经网络（Recurrent Neural Network），作为控制器来生成这样的字符串，再使用强化学习（Reinforcement Learning）的方法来优化控制器，最终获得令人满意的神经网络结构。类似地，MnasNet 也使用了强化学习进行搜索。

与 NAS-RL 中编码模型架构的方法类似，GeNet 将神经结构表示视为一系列固定长度二进制编码的字符串，这些编码可以被称为神经结构的 DNA。种群（population）被随机初始化，然后使用进化算法（Evolution Algorithm）来繁殖（crossover）、变异（mutation）和选择种群，最终迭代选择出最佳个体。同样使用进化算法，Large-scale Evolution^[2] 仅使用单层模型而不使用卷积作为个体进化的起点，在种群繁衍变异最后选择种群中最具竞争力的个体。从上述分析中可以看出，这些方法并不使用现有的优秀人工设计的神经结构，而是在各自的方法中从头开始搜索神经结构。他们的最初实践展现了神经架构搜索的巨大潜力，但也同时反映出了神经架构搜索中的计算量巨大的难点，这是由多种原因造成的：

- 搜索空间上，由于各类神经网络算子极多，还有多变的网络深度、宽度、通道数，这造成了极大的搜索空间以及随之而来的超大的计算量；

- 从零开始搜索。这类方法在构建神经网络时完全不依赖现有的神经网络结构，直到最终生成所需的神经网络结构。这些方法忽视了现有的神经网络结构设计经验，无法利用现有的优秀神经网络结构。换言之，从零开始搜索进一步加剧了庞大的搜索空间。
- 完全训练。这类方法要求从头开始训练每个候选神经网络结构，直至收敛，由此，对每一个候选网络搜索本身就需要消耗巨大的计算量。然而，相邻网络的结构和前一代的神经网络结构相似，同一阶段的神经网络结构也类似。因此，如果每个候选神经网络结构都从头开始训练，则无法充分利用这种关系。此外，我们只需要获得候选架构的相对性能排名。是否有必要让每个候选架构训练到收敛也是一个值得考虑的问题。

基于以上，我们可以用一句话来概括神经架构的难点，即搜索空间大，验证（即训练并测试）时间长。为此，神经架构搜索引入以下的技术来减轻搜索开销：

- 模块化搜索空间。与全局搜索空间不同，模块化搜索空间将神经网络视为多个不同类型模块的堆栈。因此，搜索任务从原来的全局搜索简化为对不同类型的一个或多个模块的搜索。另外，模块化搜索空间还可以参考人工设计的宏观、微观模型结构，大幅降低搜索空间。
- 参数共享。参数共享指在搜索过程中在不同的神经结构之间共享相同的权重集合。通过这种方式，可以大大减少重复训练的计算成本和内存开销，提高搜索效率。这种方法广泛用于单次 NAS 和渐进式 NAS 中。
- 连续搜索策略。相对于离散搜索策略，连续搜索策略不断松弛结构参数以构建可微的神经结构，从而使用梯度下降等连续优化方法进行搜索。通过优化连续参数，可以在搜索空间中自适应地发现潜在的最优结构。这种方法在模型设计的优化中具有广泛的应用前景，因为它使得搜索过程更加高效、灵活，并能够在优化目标和计算复杂度之间取得平衡。

综上所述，神经架构搜索是一项旨在自动化模型架构选择的任务。尽管神经架构搜索面临着搜索空间巨大和验证时间长的难题，但通过模块化搜索空间、参数共享和连续搜索策略等技术，我们可以减轻搜索的计算开销并提高搜索效率。在下一个节中，我们将介绍一种基于单次超网络训练的神经架构搜索方法，该方法采用参数共享来加速搜索过程，并能够在预训练超网络参数上快速评估候选神经结构的性能。

2.3 基于单次超网络训练的神经架构搜索

单次神经架构搜索（One-shot NAS）采用权重共享（weight sharing）的技术，这避免了每次验证候选网络所需要的大量训练计算开销。本小节我们将介绍几篇经典的单次神经架构搜索作品来作为参考。

ENAS^[17]采用了权重共享的技术进行神经架构搜索。ENAS 强制所有子模型共享权重，交替训练共享模型权重和强化学习控制器，其中控制器用于在超级计算图中采样子图。具有相同信息流的子图在搜索阶段共享参数，因此每次迭代只需要优化采样的子图。如此可避免从头开始训练每个子模型以达到收敛，从而提高了 NAS 的效率，比标准神经网络架构搜索便宜了 1000 倍，且能达到与 NASNet 相当的实验结果。

SMASH^[18]绕过完全训练候选模型的昂贵过程的方法，而是训练一个辅助大网络模型（HyperNet）来动态生成具有可变架构的主模型的权重。尽管这些生成的权重比针对固定架构自由学习的权重要差，但它们利用了这样一种观察结果：在训练的早期，不同网络的相对性能通常能够指引最优性能的排序，以此高效搜索到了与手工设计网络相当的性能。

不同于以上工作，Gabriel Bender 等人的论文^[19]不再使用控制器，而是训练了一个包含搜索空间中所有可能操作的大型一次性模型（后人往往称作 supernet），其中包含搜索空间中的所有可能操作。随后，他们发现，置零那些对模型推断不太重要的子结构只会轻微影响模型的精度。由此他们展现了另一种有效地从复杂的空间中搜索出有前途的架构的方法，而无需依靠大网络模型或强化学习控制器。

SPOS (Single Path One-Shot)^[20]指出前人的单次网络训练中，联合优化造成了各个子网络间的耦合。于是，SPOS 提出将架构搜索问题与超级网络训练问题分离开来，通过构建单路（Single Path）的大型网络（supernet），通

过均匀采样各个路径进行训练，从而缓解了权重耦合的问题，而且还能大幅减少大型网络显存的开销。

可微神经架构搜索（Differentiable NAS）也采用了类似的参数共享思想。传统上，研究者将神经架构搜索视为一种离散搜索策略中的黑盒优化问题，从而使得搜索开销极大。可微神经架构搜索探索了将离散神经架构空间转化为连续可微形式的可能性，并进一步使用梯度优化技术来搜索神经架构。比如，DARTS^[5]将候选操作的选择放松为所有可能操作的 softmax 来使离散搜索策略变得连续，通过交替训练大型网络和操作权重，同时进行子网络优化和模型搜索，最后选择具有最大边缘权重的操作。

总之，单次神经架构搜索在有效减少搜索时间和计算开销方面取得了巨大的成功。本小节介绍的几篇经典单次神经架构搜索方法在不同的方向上进行了探索，其中部分方法使用了强化学习和进化算法，而另一些方法则采用了可微分的方式进行搜索，以实现对搜索空间的更加高效的探索。然而，在实践中，这些方法往往无法直接应用于硬件约束下的神经架构搜索，因为硬件资源限制了搜索空间大小和搜索时间。在接下来的小节中，我们将重点介绍硬件约束下的神经架构搜索方法。

2.4 硬件约束下的神经架构搜索

初始的深度卷积神经网络架构，例如 VGG^[21] 和 ResNet^[13]，是为 GPU 等高性能计算平台设计的。然而，在部署到边缘设备和标准 CPU 等资源有限的平台上时，这些架构的许多变体面临着挑战。因此，出现了一系列研究旨在找到具有高性能和有限资源需求的架构的方法。

硬件约束下的神经架构搜索是为了解决这个问题，它在神经架构搜索的基础上添加了对各种平台（例如 TPU、CPU、EdgeGPU 和 FPGA 等）目标延迟的约束条件。和神经架构搜索类似，硬件约束下的神经架构搜索可以分为两类方法：

- 基于奖励的方法，如强化学习或进化算法，其中搜索通过对网络进行采样并通过在某些验证集上评估其最终准确性和资源需求来进行。这种方法的预测器成本高昂，往往不准确，因此我们不再赘述。
- 基于资源感知（Resource-aware）梯度的方法制定了一个可微的损失

函数，由准确性损失函数加权正则惩罚项的组成。因此，可以直接使用随机梯度下降（Stochastic Gradient Descent）来优化架构。我们将在下文介绍几个主要的工作。

FBNet^[9]使用可微神经架构搜索来加速搜索过程，并使用了考虑时延（Latency-Aware）的损失函数，来加速网络在特定设备的性能。具体来说，FBNet 事先建立了每一层网络时延的查找表（Lookup Table），假设每一层的网络的时延之和即为最终模型的时延，使用时延的对数作为最终交叉熵损失函数的系数，从而指导梯度下降算法前往时延更少的子网络选项。

ProxylessNas^[22]是第一篇研究不同硬件平台（CPU, GPU, 移动设备）专用神经网络架构的工作，提出了一种新颖的梯度优化方法来减少传统可微分神经架构搜索中过大的内存占用和计算消耗问题。同时，他们类似于 FBNet，使用每层网络的查找表，将每个选项的网络延时的数学期望加入到最终的损失函数中，来实现基于梯度的架构搜索。

当然，这些方法在硬件约束下进行的神经架构搜索仍然具有一些缺点。例如，它们往往采取的是加入软约束，即可能会违反对资源的硬性约束。另外，加入正则的办法很难调整准确性和资源之间的权衡，需要人工对相关超参数进行大量调试，这会降低网络的准确性并未能完全满足资源要求。最后，由于最终的离散化步骤将架构从可微搜索空间投影到离散的架构空间中，因此对资源的硬性约束进一步被违反。为了解决这些问题，近年来出现了一些新的方法。如 HardCoRe-NAS^[11]使用 Frank-Wolfe (FW) 算法，在整个搜索过程中严格满足硬性约束条件，并且超过了其他 NAS 方法。

上述算法都假设了能够获取到每一层网络在特定硬件平台上的信息，然而，面对海量的非常见的边缘器件，获取此类信息需要特定硬件的专业知识，因而此类方法对于非硬件专家带来了挑战；另外，如能耗相关的目标，在关于单层网络的测量未必能直接加总为总模型的能耗值；再次，他们将问题建模为在硬件约束下的网络准确度的优化问题，并没有在事实上搜索拥有更好硬件性能的模型。最后，将架构选择和模型网络参数共同训练，将模型参数的更新和其他硬件性能指标耦合在一起，很可能会影响子网络模型本身关于精度的优化。

同时，这些方法往往假设能够获取到每一层网络在特定硬件平台上的信息，这需要具有特定硬件的专业知识，因而此类方法对于非硬件专家带

来了挑战。此外，如能耗相关的目标，在关于单层网络的测量未必能直接加总为总模型的能耗值；再次，这些方法将问题建模为在硬件约束下的网络准确度的优化问题，并没有在事实上搜索拥有更好硬件性能的模型。最后，将架构选择和模型网络参数共同训练，将模型参数的更新和其他硬件性能指标耦合在一起，很可能会影响子网络模型本身关于精度的优化。

相比之下，我们提出的方法能够将网络的权重训练和硬件性能进行解耦，并且能够基于同一个预训练超网络参数，搜索部署到不同的硬件设备的模型。我们提出的方法是直接使用最终的硬件性能指标，能够更好地搜索到用户需要的模型架构，同时避免了网络 block 粒度的硬件信息获取和昂贵计算的困扰。此外，我们的方法能够在多个目标上协同搜索，提供帕累托最优解集，且不需要过多的人工调整超参数。

2.5 本章小结

本章介绍了视觉模型的发展历程和神经架构搜索的演化历史。我们重点介绍了基于一次性搜索的 NAS 方法和在硬件约束下的神经架构搜索方法。本章还介绍了在硬件约束下的神经架构搜索方法的局限性，例如软约束难以满足、硬件专业知识要求高等问题，并讨论了本文提出方法如何在软硬件多目标搜索上改进现有方法的局限性。

第三章 预备知识

3.1 神经网络基本运算和模块

在现代基于深度学习的视觉模型中，卷积神经网络（Convolutional Neural Network）是关键的组件。本节，我们将介绍常用的卷积神经网络及其变体，他们的子集经过排列组合，构成了神经架构搜索的常用空间。

标准空间卷积（Standard Spatial Convolution）。卷积操作在特征提取中起着关键作用 [70]。标准空间卷积层是一组三维滤波器，其中每个二维滤波器被分解为不同的 $k \times k$ 的卷积核。每个二维卷积核与输入特征图中的一个通道中的区域卷积以产生部分和。所有输入通道上的所有部分和累加以产生一个输出像素。

扩张卷积（Dilation Convolution）。扩张卷积在卷积层的卷积核中插入零，既保持参数量不会上升，同时扩展了卷积核的感受野 [144]。扩张率为 1 的卷积便退化为标准卷积，而 k -扩张卷积是一个具有 k 个间隔的卷积核的标准卷积。

深度卷积（Depthwise Convolution）。标准空间卷积需要 $k \times k \times C_I$ 次乘法运算才能生成一个输出像素，其中 C 是输入特征的通道数，因为它将一个三维滤波器应用于输入特征图中的所有通道。而深度卷积 [51] 仅仅将一个 $k \times k$ 滤波器应用于输入特征图中的一个通道来产生一个输出像素。每个 $k \times k \times C$ 滤波器首先被分成 C_I 个切片，每个切片的尺寸为 $k \times k \times 1$ 。卷积操作在每个通道中分别与相应的滤波器切片进行。因此，每层中的乘法次数减少了一个 C_I 的数量级。经过深度卷积层后，输出特征的通道数与输入特征一致。

点卷积（Pointwise Convolution）。点卷积是一个卷积核大小为 1×1 的标准卷积。通过与 $1 \times 1 \times C_I \times C_O$ 滤波器卷积，将输出特征图的通道数由 C_I 转变为 C_O 。

深度可分离卷积（Depthwise-separable Convolution）。深度可分离卷积将每个通道的深度卷积和点卷积组合在一起，由此，它能实现比标准空间卷积更少的乘加数，往往运用于内存受限的使用场景。

分组卷积（Group Convolution）。分组卷积的概念最早在 AlexNet^[12]中使用，其中滤波器和输入激活被分成 g 组。每个组中的一个滤波器仅与相应组中的激活进行卷积，从而相对于标准卷积减少了产生输出激活所需的乘法次数。

压缩激活网络（Squeeze and Excitation Network (SENet)）。SENet^[23]建模特征图的相互依赖关系，从而提高特征图的质量。它可以作为辅助并行连接添加到任何标准卷积中。具体来说，在辅助路径的第一个压缩（squeeze）步骤中，采用全局平均池化方法来收集输入特征图中每个通道的表示形式，以生成一个 $1 \times 1 \times k$ 的向量。在第二个激活（excitation）步骤中，经全连接层处理池化向量，以产生一组 $1 \times 1 \times k$ 的通道权重，其通道大小等于输入特征图中的通道数。通道权重和输入特征图按通道相乘，生成最终的 SE 块。

移动倒置瓶颈网络（Mobile Inverted Bottleneck Layer，简称 MBConv）。MBConv 是 MobileNetV2^[15] 网络的基本单元，由三个操作组成： 1×1 的点卷积作用于 $H \times W \times C$ 大小的输入特征映射，产生 $H \times W \times C \cdot e$ 大小的输出特征映射；再对 $H \times W \times C \cdot e$ 进行 $k \times k$ 的深度卷积，步长为 s ，产生 $H/s \times W/s \times C \cdot e$ 的特征图；随后对 $H/s \times W/s \times C \cdot e$ 进行点卷积，产生 $H/s \times W/s \times C'$ 的输出特征。扩张比 e 控制特征映射的加宽， $C \cdot e$ 称为扩张通道。由 MBconv，研究者们还发展出许多变体：

- 融合移动倒置瓶颈网络（Fused-MBConv）是由 MBConv 派生出来的，它使用一个 $k \times k$ 的标准卷积替换原来 MBConv 中第一个 1×1 的点卷积和 $k \times k$ 的深度卷积的组合。虽然这种融合比标准的 MBConv 具有更多的参数，但它们在某些输入和输出特征映射形状上可以在一些设备上运行得更快。

- FBNet^[9]块使用 1×1 的组卷积替换标准的 MBConv 中的 1×1 点卷积，可以更有效地减少计算量。
- MobileNetV3 中，MBConv 模块中加入 SE 连接，以此进一步提升模型的特征提取能力。

通道重排网络模块（ShuffleNet Block）。ShuffleNet^[16]在组卷积模块中加入了通道重排（Channel Shuffle）的操作，以此促进不同组间特征图的信息交换，从而加强模型的表征能力，进而达到更高的模型精度和更小的计算花销。

这些卷积模块是构建神经架构搜索空间的基本单元。在接下来的章节中，我们将介绍如何使用这些模块来构建搜索空间。

3.2 搜索空间构建

搜索空间是一组可行的架构，我们希望从中找到性能高的架构。通常，它定义了一组基本的网络运算符，以及如何连接这些运算符来构建模型的计算图。不同于固定架构的超参数优化（仅优化如通道数量、步幅、内核大小等架构超参数），架构搜索空间允许优化器选择操作之间的连接并更改操作类型。此处，我们介绍两类常见的搜索空间。

基于单元或微结构的神经架构搜索（Cell-based/Micro Search） 在搜索网络的建立块时，可以搜索一个单元或模块，而不是直接搜索整个网络。研究者往往使用有向无环图（DAG）或单元拓扑来表示被搜索网络的构建块。单元中的每个边缘（edge）都是一个操作（比如卷积、池化或跳过连接）。NAS 的任务则是在每个边缘上搜索单个操作。搜索到的单元可以多次堆叠以形成最终的卷积神经网络，从而在整个网络中重用相同的单元体系结构。基于输入和输出特征映射分辨率的不同，存在两种类型的单元：一种是“正常单元”（Normal Cell），不改变特征映射的分辨率；而另一种是“缩减单元”（Reduction Cell），将输入特征映射的尺寸减半。基于单元的 NASNet 网络^[4]的参数和 FLOPs 比 MobileNet^[14]更低。然而，单元中的碎片化结构对许多硬件平台不友好，导致运行时显著增加。

层次搜索空间（Layer-wise Search） 事先固定好宏观的模型层次，搜索每层的可选模块集合。例如，FBNet^[9] 搜索空间基于 MobileNetV2^[15] 的结构，采用了一种层次搜索空间，其中包含一个固定的宏结构，确定每层的层数和维度，其中第一个和最后三个层具有固定的运算符，其余层需要进行优化。与基于单元的搜索空间相比，这种方法更灵活。这使得他们能够减小总搜索空间的大小，从而提高搜索效率。

总的来说，搜索空间是建立在一组可行的网络运算符之上的，不同的搜索空间定义了不同的操作类型和连接方式。基于单元或微结构的神经架构搜索允许优化器搜索单元或模块，然后重复堆叠以形成最终的卷积神经网络，而层次搜索空间则事先固定了宏观的模型层次，搜索每层的可选模块集合。无论使用哪种搜索空间，我们都希望从中找到性能高的架构。本文的方法也将同时兼容上述不同类型的空间。

3.3 多目标搜索

在硬件感知的神经架构搜索(Hardware Aware Neural Architecture Search, 简称 HANAS) 领域中，我们需要同时对模型精确度、时延、能耗等多个指标进行优化。因而，我们引入多目标优化(Multi-Objective Optimization)对该问题进行建模，其主要目的是在多个目标之间找到一组最优解。在本节中，我们将介绍多目标优化的基本数学形式，通过支配关系(Dominance)定义帕累托最优解集合和帕累托最优解前沿，并讨论两种朴素方法：加权求和法(Weighted Sum Method)和 ϵ 约束法(Epsilon Constraint Method)，以及它们各自的局限性。

多目标优化的数学形式 假设有 m 个目标函数需要优化，每个目标函数的取值范围为 $f_i(x), i = 1, 2, \dots, m$ ，其中 x 是自变量向量。则 MOO 问题的数学形式可以表示为：

$$\begin{aligned} \min \quad & F(x) = (f_1(x), f_2(x), \dots, f_m(x)), \\ \text{s.t.} \quad & x \in X \end{aligned} \tag{3.1}$$

其中 X 表示可行解的搜索空间。由于多目标优化问题中存在多个目标函数，因此解空间中可能存在多个最优解，而这些最优解可能并不是互相可比较

的。因此，我们需要使用定义支配关系来解决这个问题。

支配关系 是一种用于比较多个解之间关系的方法。假设解 x 和解 y 属于搜索空间 X ，则有以下三种可能的情况：

- x 支配 y ，当且仅当对于所有的目标函数 $f_i(x) \geq f_i(y)$ ，且至少存在一个目标函数 $f_j(x) < f_j(y)$ ；
- y 支配 x ，当且仅当对于所有的目标函数 $f_i(y) \leq f_i(x)$ ，且至少存在一个目标函数 $f_j(y) < f_j(x)$ ；
- x 和 y 互不支配。

在使用支配关系比较不同解的优劣性之后，我们需要定义一个新的概念，即帕累托最优解集合。

帕累托最优解集合（Pareto-Optimal Set） 是指所有非支配解构成的集合。具体来说，给定一个解的集合，其帕累托最优解集合是指该解集合中的所有非支配解构成的集合。非支配解是指没有其他解能够在所有目标函数上优于该解的解。因此，帕累托最优解集合是指在所有可能的解集合中，所有非支配解构成的集合。

在解决一个多目标优化问题时，我们通常需要找到其帕累托最优解集合，而帕累托最优解集合是一个关于所有目标函数的最优解集合。通过寻找帕累托最优解集合，我们可以了解所有可能的最优解，并且可以从这些解中选择一个最适合我们的解。

帕累托最优前沿（Pareto-Optimal Front） 是帕累托最优集合所对应的解集在决策空间上的投影，其是一个边界集合，其中包含了所有帕累托最优解的投影。具体来说，对于任意一个帕累托最优解集合中的解，它的决策变量对应的点都在帕累托前沿上。而帕累托前沿则是指所有投影点都在帕累托最优集合上的点集合。

因此，在多目标优化问题中，我们需要寻找帕累托最优解集合和其对应的帕累托前沿，以便我们可以了解所有可能的最优解，并且可以从这些解中选择一个最适合我们的解。

多目标优化常用求解方法 在上述定义的基础上，加权求和法（Weighted Sum Method）是一种常用的求解多目标优化问题的方法。基于梯度反向传播来优化多目标的方法正是使用了加权求和法，如 ProxylessNAS 中在最终的损失函数里加上每层网络估计延时的期望作为加权项^{II}，FBNet。该方法将多个目标函数综合为一个单一的目标函数，通过对目标函数进行加权求和，从而将多个目标函数转化为一个目标函数。具体来说，加权求和法通过将每个目标函数 $f_i(x)$ 乘以一个权重系数 w_i ，然后将它们相加，从而得到一个加权求和的目标函数，公式如下

$$\min F(x) = \sum_i^m w_i f_i(x)$$

这种方法的优点是简单易用，但缺点是在非凸问题中无法找到帕累托最优集，且对于加权参数结果较为敏感，无法保证非支配解的收敛性和多样性。

ϵ 约束法（Epsilon Constraint Method）是另一种常用的求解多目标优化问题的方法，它将多个目标函数优化转化为多个约束条件下的一个单一的目标函数优化。具体来说， ϵ 约束法通过在每个目标函数上引入一个容忍度 ϵ ，然后将每个目标函数作为约束条件，并将其中一个目标函数作为单一的目标函数，从而将多目标优化问题转化为单目标优化问题。在 SPOS^{II} 中，作者便使用此种方法将 FLOP 的约束转化为约束条件，只对子模型的精度进行优化。公式如下

$$\begin{aligned} \min \quad & f_p(x) \\ \text{s.t.} \quad & f_q(x) \leq \epsilon_q, q = 1, 2, \dots, m \text{ 且 } q \neq p \\ & x \in X \end{aligned}$$

这种方法可以保证非支配解的收敛性和多样性，但缺点是需要手动选择合适的容忍度 ϵ ，并且当目标函数的数量增加时，问题会变得更加复杂。

尽管上述两种朴素的方法在实践中大量使用，但是它们需要大量的超参数调试、目标函数的可微分化（如应用加权求和法于神经网络训练），并且并不能给出一组帕累托前沿供用户进行挑选。因此，在下一章中，我们将介绍能提供一组帕累托前沿解的多目标优化求解方法。

3.4 本章小结

本章主要介绍了神经架构搜索中的卷积神经网络及其变体，以及搜索空间的构建和多目标搜索的应用。在卷积操作小节中，我们介绍了常用的卷积神经网络，包括标准空间卷积层、扩张卷积、深度卷积、点卷积、深度可分离卷积等，并介绍了 SENet 模块、分组卷积、MBConv、ShuffleNet Block 等变体。在搜索空间构建小节中，我们介绍了基于单元或微结构和层次搜索空间的神经架构搜索方法。在多目标搜索小节中，我们介绍了多目标优化在硬件感知的神经架构搜索中的应用，包括支配关系、帕累托最优解集合和帕累托前沿的概念，并介绍了加权求和法和 ϵ 约束法等常用的多目标优化求解方法。总的来说，本章为神经架构搜索提供了基本的理论和支持。

第四章 单次超网络下的多目标优化搜索

4.1 问题假设和形式化

为进行软硬件协同的神经架构搜索中，我们首先需要定义我们的问题以及相应的基本假设。本文中，我们将优化的目标设置为模型精度 f_{acc} 、硬件性能 f_{hw} （如特定硬件时延、硬件能耗 $f_{hw} = (f_{lat}, f_{energy})$ ）。同时，为了使我们的方法能够有更好的扩展性，我们做出如下的假设：

- 模型精度在训练平台和实际硬件部署平台之间具有一致性，这有助于加速搜索过程的测试环节；
- 模型推断的时延和能耗可以能够直接测量得到，但是无法得到每一层的具体数据。

因此，我们的问题可定义为寻找一种架构 α 使得上述优化目标达到极值：

$$\min \quad f_{acc}(\alpha), f_{hw}(\alpha) \quad (4.1)$$

为了加速搜索过程且快速适配不同类型的硬件，我们提出使用多目标遗传算法，在预训练的单次超网络中进行搜索。通过使用超网络的参数代替从头训练的参数，我们可以将代理精度 $f_{proxy}(\alpha)$ 代替真实精度 $f_{acc}(\alpha)$ ，从而加快搜索过程：

$$\min \quad f_{proxy}(\alpha), f_{hw}(\alpha) \quad (4.2)$$

我们提出的方法包括以下步骤：

1. 训练单次超网络：在给定搜索空间条件下，我们将每个预选操作算子放置到一个超网络之中进行训练，此处我们仅训练模型网络的精度。
2. 多目标搜索：基于预训练的单次超网络，结合硬件性能，我们使用多目标遗传算法 NSGA-II 来搜索帕累托最优架构集合，为用户了解特定搜索空间在特定硬件中多目标最优边界的形状，供用户自行挑选符合其需求的模型架构。

为了实现上述搜索过程，我们需要建立单次超网络模型，并训练该模型来代理模型的精度，同时考虑硬件性能的约束。因此，我们将在接下来的三个小节中逐一介绍单次超网络建模、单次超网络训练以及多目标遗传算法搜索的具体实现过程。

4.2 单次超网络建模

单次超网络（One Shot Supernet）是现代神经架构搜索中常用的方法。通过共享权重的方式，只需单次训练，即可对子网络进行测试验证，大大加速了模型的验证环节。本小节中，我们将形式化单次超网络训练的过程，同时介绍几种相关的训练方式。

无论是对于基于单元结构的或者是基于层次的搜索空间，我们都可以将模型的前向推断视作由 N 节点组成的有向无环图（DAG）。每个节点 $x^{(i)}$ 是神经网络中的一个特征图，每条边 (i, j) 代表从候选操作集合 \mathcal{O} 中转换节点特征 $x^{(i)}$ 的操作符 $o^{(i,j)}$ 。所以我们可以将任何中间节点 $x^{(j)}$ 表述为：

$$x^{(j)} = \sum_{i < j} o^{(i,j)}(x^{(i)})。$$

对于层次搜索空间而言，我们遵循 SPOS 工作的设定，假定每层节点收到前一节点的特征，并输出一节点，也即图结构退化为链的形式^[20]；在单元搜索空间中，我们遵循 NASNet^[4]的设定，假设每个单元有两个来自前两个单元的输入节点和一个输出节点。

对于超网络训练时，每层特征节点的计算，即 $o^{(i,j)}$ 有以下三种建模方式：

- 假设候选操作服从均匀分布 $\mathcal{U}_{\mathcal{O}}$ ，每次前向推断时从候选操作集合 \mathcal{O} 中随机采样一种操作，即

$$o^{(i,j)} \sim \mathcal{U}_{\mathcal{O}}。$$

SPOS 等单路超网络训练的工作使用了这样的假设^[20]。

- 假设候选操作服从一可学习的分布 $\pi_{\mathcal{O}}^{(i,j)}(\alpha)$ ，在每次前向推断时，将离散的分布放松为连续的过程，将输出特征计算为根据 $\pi_{\mathcal{O}}(\alpha)$ 加权的

分布，即

$$o^{(i,j)}(\cdot) = \sum_{o \in \mathcal{O}} h_o^{(i,j)}(\alpha) o(\cdot).$$

DARTS 等工作使用了这样的假设^[5-6]，其中 $h_o^{(i,j)}$ 是操作算子 $o(\cdot)$ 的权重，由 α 表征架构分布大小的可学习参数得到。往往我们使用 Softmax 进行归一化表达概率，即

$$h_o^{(i,j)} = \text{Prob}[o^{(i,j)} = o] = \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})}, \quad (4.3)$$

其中 $\alpha_o^{(i,j)}$ 是一对节点 (i, j) 中算子 o 的权重参数。

- 假设候选操作服从一可学习的分布 $\pi_{\mathcal{O}}^{(i,j)}(\alpha)$ ，每次前向推断时从候选操作集合 \mathcal{O} 中根据学习的分布采样一种操作，即

$$o^{(i,j)} \sim \pi_{\mathcal{O}}^{(i,j)}$$

。GDAS 等工作使用了上述假设^[24]。为了参数化离散的分布并令计算可微分，我们需要使用 Gumbel-Softmax 对数据分布进行重参数化^[25]。具体来说，在我们进行前向计算的时候，我们可以使用下述(4.4)确定进行采样，

$$o^{(i,j)} = \arg \max_o \alpha_o^{(i,j)} + g, \quad (4.4)$$

其中 $g = -\log(-\log(u))$, $u \sim \text{Unif}[0, 1]$ 是采样自 $\text{Gumbel}(0, 1)$ 分布的值。而对于反向传播计算，我们使用(4.5)对(4.4)进行替换，

$$\tilde{h}_o^{(i,j)} = \frac{\exp((\log(\text{Prob}[o^{(i,j)} = o]) + g)/\tau)}{\sum_{o' \in \mathcal{O}} \exp(\log(\text{Prob}[o^{(i,j)} = o]) + g/\tau)}. \quad (4.5)$$

这里的 $\text{Prob}[o^{(i,j)} = o]$ 沿用了(4.3)的参数化形式，而 τ 是 Softmax 的温度。当 $\tau \rightarrow 0$ 时， $\tilde{h}_o^{(i,j)}$ 即符合前向 argmax 的分布；当 $\tau \rightarrow \infty$ 时， $\tilde{h}_o^{(i,j)}$ 近似均匀分布。有了 Gumbel-Softmax 的技巧，我们就可以使用梯度的方式进行优化。

以上三种建模方式都在不同的神经架构搜索工作中得到了应用，选择

何种方式需要根据具体任务和实际情况进行权衡和选择，而我们的框架均可以支持上述的建模过程。接下来，我们将介绍如何训练单次超网络。

4.3 单次超网络训练

对于仅建模模型参数的超网络，其训练过程是平凡的。本节我们将介绍如何训练同时建模模型参数和架构权重的超网络。

我们可以将单次超网络训练表述为一个二层优化问题，即同步学习架构 α ，以及网络中的权重 w 。优化的神经结构搜索 α^* 使验证损失最小化 $\mathcal{L}_{val}(w^*(\alpha), \alpha)$ ，其中 w^* 是使训练损失最小化的最佳网络权值 $\mathcal{L}_{train}(w, \alpha)$ 。具体来说，

$$\min_{\alpha} \quad \mathcal{L}_{val}(w^*(\alpha), \alpha) \quad (4.6)$$

$$s.t. \quad w^*(\alpha) = \arg \min_w \mathcal{L}_{train}(w, \alpha). \quad (4.7)$$

若要先进行内部优化(4.7)，再进行外部优化搜索，其开销是十分昂贵的。为了更高效的解决这个嵌套的优化问题，我们应用近似梯度，对(4.6)进行处理，即。

$$\begin{aligned} & \nabla_{\alpha} \mathcal{L}_{val}(w^*(\alpha), \alpha) \\ & \approx \nabla_{\alpha} \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(\alpha), \alpha) \\ & = \nabla_{\alpha} \mathcal{L}_{val}(w', \alpha) - \xi \nabla_{\alpha, w}^2 \mathcal{L}_{train}(w, \alpha) \nabla_{w'} \mathcal{L}_{val}(w', \alpha), \end{aligned}$$

其中 ξ 是内部模型权重优化的学习率， $w' = w - \xi \nabla_w \mathcal{L}_{train}(\alpha)$ 。如果我们将二阶项近似为 0，我们即获得了一阶形式的优化；或者，我们也可以用差分来近似二阶求导项。随后，我们采用交替优化的形式，对模型进行优化，如算法1所示。

算法 1: 单次超网络训练算法

```
1 while not converge do
2   Sample batch of training data  $x_{train}$  ;
3    $w \leftarrow w - \nabla \mathcal{L}_{train}(w, x_{train}; \alpha)$ 
4   Sample batch of validation data  $x_{val}$  ;
5    $\alpha \leftarrow \alpha - \nabla \mathcal{L}_{val}(\alpha, x_{val}; w)$ 
6 end
```

4.4 多目标遗传算法

为了解决软硬件协同的神经架构搜索问题，我们采用非支配排序遗传算法-II(Non-dominated Sorting Genetic Algorithm II)作为多目标搜索的方法。NSGA-II 是一种经典的多目标遗传算法，其核心思想是将解空间根据非支配排序划分为多个等级，并根据个体的等级和等级内的拥挤度来选择和保留较优的解，最终给出帕累托最优前沿。它具有较高的效率和准确性，广泛使用于离散空间的多目标搜索问题中^[26]。在算法2中，我们给出了 NSGA-II 的高层次伪代码，并将在下文做具体的方法介绍。

算法 2: 多目标遗传算法 NSGA-II

输入: 初始化候选种群 P_0 , 迭代总数 T

输出: 帕累托最优前沿 $\mathcal{F}_{\mathcal{T}}$

```

1    $t \leftarrow 0$ 
2   for  $t \leq T$  do
3        $Q_t \leftarrow \text{EA}(P_t)$ 
4        $R_t \leftarrow P_t \cup Q_t$ 
5        $\mathcal{F} = \text{non-dominated-sort}(R_t)$ 
6        $P_{t+1} \leftarrow \emptyset, i \leftarrow 1$ 
7       while  $|P_{t+1}| + |\mathcal{F}_i| \leq N$  do
8           crowding-distance-sorting( $F_i$ )
9            $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i$ 
10           $i \leftarrow i + 1$ 
11      end
12       $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$ 
13  end

```

首先, NSGA-II 算法遵循一般的遗传算法的规则, 通过优良个体的交叉变异, 产生新一代的种群。接着, 它通过非支配排序得到各级的非支配层级, 其中第一级即为帕累托最优前沿。在算法3中, 我们展现了一种 $O(MN^2)$ 的快速非支配排序, 其中 M 是目标的数量, N 是参与排序的种群总数。

其次, 在每一个非支配层级中, 我们计算每个候选架构的(归一化)性能之间的欧式距离, 根据目标空间中的密度对每层中的候选进行排序, 以此挑选出该层次中更具代表性的候选架构, 伪代码如算法4所示。

最后, 它采取了一种保留机制, 即精英策略 (Elitism), 始终保持着历史种群中的最优解, 以避免局部最优解。同时, 由于该算法给出的是一组解集, 我们可以更好地分析搜索空间在特定数据集、硬件上的最优边界, 有助于模型的调整和优化。

算法 3: 非支配集快速排序

输入: 待排序集合 P

输出: 正序非支配集 $\mathcal{F} = [\mathcal{F}_1, \mathcal{F}_2, \dots]$

```

1 foreach  $p \in P$  do
2    $S_p \leftarrow \emptyset, n_p \leftarrow 0$ 
3   foreach  $q \in P$  do
4     if  $p \prec q$  then
5        $S_p \leftarrow S_p \cup \{q\}$ ; //若  $p$  支配  $q$ , 则  $q$  加入  $p$  的支配
6       集
7     else
8       if  $q \prec p$  then
9          $n_p \leftarrow n_p + 1$ ; //反之, 则  $p$  的支配计数增加一位
10      end
11    end
12    if  $n_p = 0$  then
13       $\mathcal{F}_1 \leftarrow \mathcal{F}_1 \cup \{p\}$ ; // $p$  属于第一帕累托前沿队列
14    end
15  end
16   $i \leftarrow 1$ 
17  while  $\mathcal{F}_i \neq \emptyset$  do
18     $Q \leftarrow \emptyset$ 
19    foreach  $p \in \mathcal{F}_i$  do
20      foreach  $q \in S_p$  do
21         $n_q \leftarrow n_q + 1$ 
22        if  $n_q = 0$  then
23           $Q \leftarrow Q \cup \{q\}$ 
24        end
25      end
26    end
27     $i \leftarrow i + 1$ 
28     $\mathcal{F}_i \leftarrow Q$ 
29  end

```

算法 4: 密集距离指派

输入: 集合 \mathcal{F} , 目标函数集合 $O = \{f_1, f_2, \dots, f_m\}$

输出: 输入集合密集距离 $Dist(\mathcal{F})$

```

1   $n = |\mathcal{F}|$ 
2  foreach  $p \in \mathcal{F}$  do
3       $Dist(p) \leftarrow 0$ 
4  end
5  foreach  $f \in O$  do
6       $Sort(\mathcal{F}, f)$ ; // 根据目标函数  $f$  进行集合排序
7       $Dist(\mathcal{F}[1]), Dist(\mathcal{F}[n]) \leftarrow \infty$ ; // 边界点密度为无穷大
8      for  $i \leftarrow 2$  to  $n - 1$  do
9           $Dist(\mathcal{F}[i]) \leftarrow Dist(\mathcal{F}[i]) + \frac{Dist(\mathcal{F}[i+1]) - Dist(\mathcal{F}[i-1])}{\max f(\mathcal{F}) - \min f(\mathcal{F})}$ ; // 归一化密集
10         距离
11     end
12 end

```

4.5 本章小结

在本章中, 我们提出了一种基于多目标遗传算法和单次超网络的神经架构搜索方法。通过定义问题假设和形式化, 我们建立了一个优化模型, 使用多目标遗传算法来搜索帕累托最优前沿, 同时使用单次超网络来加速搜索过程。通过获得帕累托最优解集, 我们可以更好地分析搜索空间在特定数据集、硬件上的最优边界, 有助于模型的调整和优化。

第五章 实验

5.1 搜索空间介绍

我们使用 NAS-bench-201^[27]对我们的框架进行实验。NAS-bench-201 共有 $5^6 = 15625$ 个可能的候选架构，支持 3 种数据集的查找。

HWNAS-Bench^[28] 支持对 NAS-Bench-201 空间中的任意架构、获得 6 种不同的硬件上时延（和部分能耗）的性能，它们分别是 NVIDIA Edge GPU Jetson TX2，Raspberry Pi 4 (Raspi 4)，Google Edge TPU Dev Board (Edge TPU)，Pixel 3，ASIC 加速器 Eyeriss 和 Xilinx ZC706 (FPGA)。

NAS-bench-201 提供的是基于单元的搜索空间。它的“搜索单元”由一个密集连接的有向无环图 (DAG) 组成，其中每条边都与一个操作相关联，该操作将源节点的特征图转换为目标节点。DAG 具有 $V=4$ 个节点，预定义的操作集 \mathcal{O} 具有 5 个代表性操作，包括取零 (zeroize)、跳连 (skip connection)、 1×1 卷积、 3×3 卷积和 3×3 平均池化层。此处的卷积操作实际对应的是 ReLU、卷积和批量归一化的操作序列的缩写。由于 DAG 的设计共有 4 个节点，这允许基本残差块样式的单元存在。同时，该搜索空间不限于密集连接的单元，因为操作集中包括 zeroize，可以删除关联的边缘。

5.2 实验数据集

CIFAR-10：这是一个标准的图像分类数据集，包含 60000 张共 10 类别的 32×32 彩色图像，其中原始训练集包含 50000 张图片，每个类别有 5000 张图像；原始测试集包含 10000 张图像，每个类别有 1000 张图像。由于原数据集并不提供验证集，我们将 CIFAR-10 的所有 50000 个训练图像分为两组。每组包含 25000 个带有 10 个类别的图像。我们将第一组视为新的训练集，另一组视为验证集。

CIFAR-100：这个数据集与 CIFAR-10 类似。它使用与 CIFAR-10 相同的图像，但将每个图像分类为 100 个更细粒度的类别。原始 CIFAR-100 训练集有 50000 个图像，原始测试集有 10000 个图像。我们随机将原始测试集分为两个大小相等的组——每组 5000 张图像。其中一个组被视为验证集，另一个组被视为新的测试集。

ImageNet-16-120: ImageNet 是一个大规模的图像识别数据集，被广泛用于深度学习视觉领域的研究和评价。它拥有超过 1400 万张、超过 2 万种类别的图像，用于训练、验证和测试。其中，训练集由超过 100 万张图像构成，验证集和测试集各包含 50000 张图像。每个图像都被标记为一个类别，大规模、多种类的自然图像使得 ImageNet 成为了深度学习视觉领域中最广泛使用的数据集之一。Chrzaszcz 等人 (2017) 将原始 ImageNet 下采样至 16×16 像素，制作了 ImageNet16×16。他们指出，下采样后 ImageNet 中的图像可以大大降低模型的计算成本，同时保持类似的模型超参数搜索结果。我们从经过下采样的 ImageNet16×16 构建 ImageNet-16-120，即我们从 ImageNet16×16 中选择所有标签从序号 1 至序号 120 的图像来创建 ImageNet-16-120。于是，经统计，ImageNet-16-120 包含 1517000 张带有 120 种类别的训练图像，以及 3000 张验证图像和 3000 张测试图像。

5.3 实验基线及训练细节

尽我们所知，在 Nas-Bench-201 上基于软硬件协同的神经架构搜索方法并没有公认的基准模型。于是，我们精心挑选了如下基于单次超网络训练的方法作为参考基线，分别为共享参数的随机搜索 (Random Search with Parameter Sharing，简称 RSPS)^[29]、二阶梯度优化的可微分神经架构搜索 (DARTS-v2)^[5] 和 GDAS^[24]。表5-1展示了超网络训练的超参数情况。

另外，我们挑选其中表现最好的超网络训练方法 GDAS 作为预训练超网络，将加权求和的遗传搜索（后文表格中我们记为 WSEA）、 ϵ 约束的遗传搜索（后文表格中我们记为 CEA）作为与 NSGA-II 作对比。其中，在加权求和的遗传搜索中，我们通过统计 1000 个随机采样的候选架构的精度、硬件性能，将遗传搜索的目标函数定义为各指标归一化的平均值；对于 ϵ 约束的遗传搜索，我们将硬件性能（如时延、能耗）改写为约束形式，统计 1000 个随机采样的候选模型的 60 分位数，作为约束条件，因而，该方法相对其他方法在事实上要多统计 1.6 倍的候选架构。表5-2展示了遗传算法中的相关超参数。

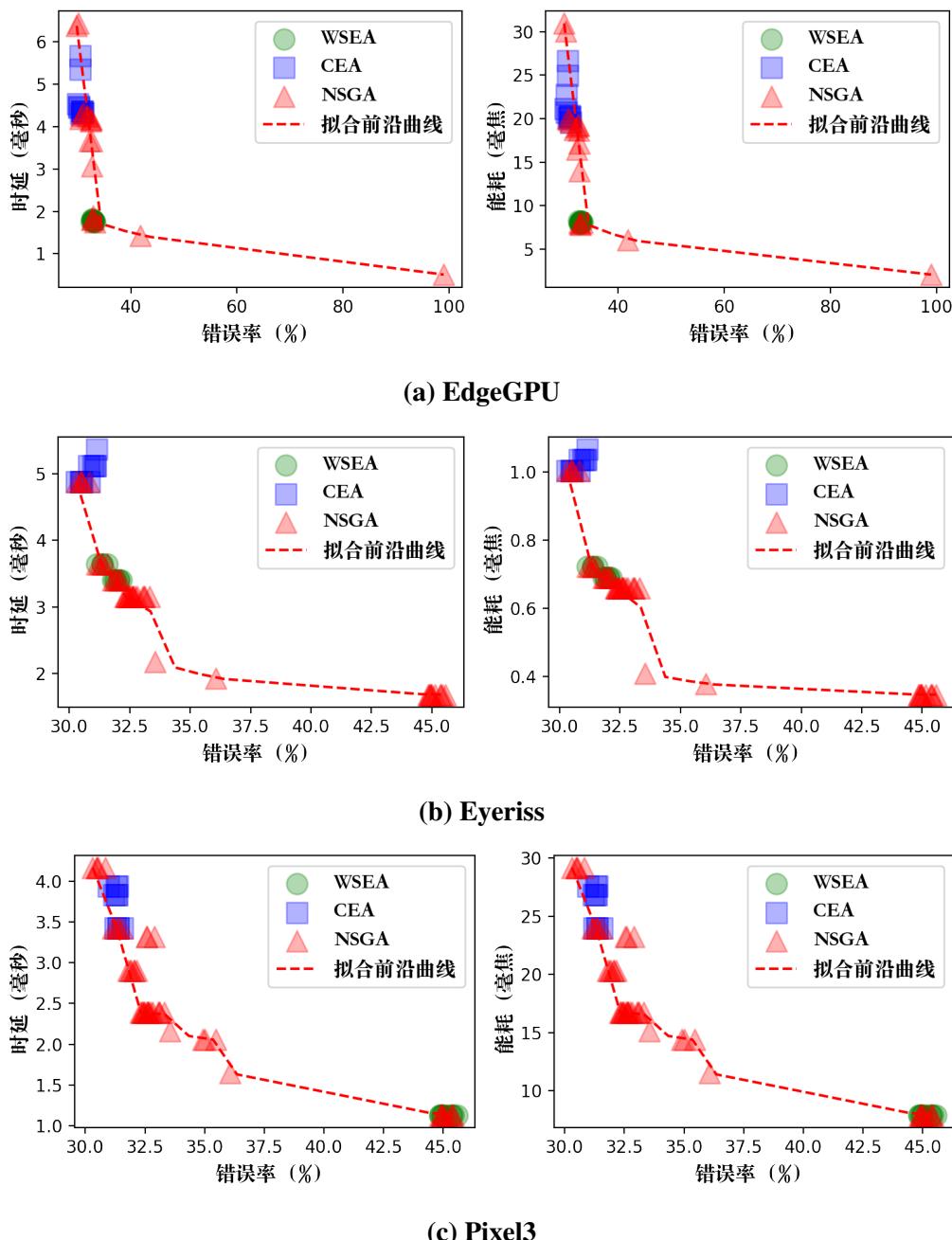
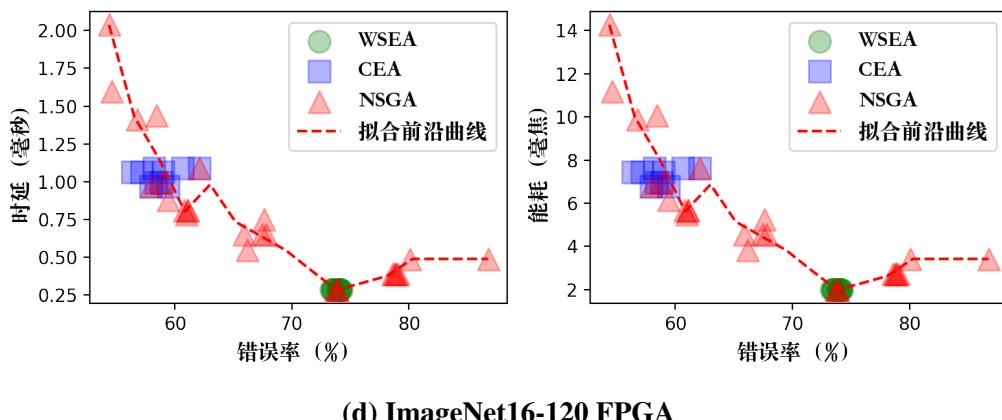


图 5-2 NSGA-II 在不同硬件设备上对 CIFAR100 数据集搜索 GDAS 超网络中。其中红色代表使用超网络进行搜索后得到的帕累托前沿。

表 5-1 超网络训练超参数

| | |
|-------------------------------|--------------|
| 批量大小 Batch Size | 64 |
| 随机翻转 Random Flip | 0.5 |
| 随机裁剪 Random Crop | 是 |
| 归一化 Normalization | 是 |
| 网络初始学习率 Initial Learning Rate | 0.025 |
| 网络权重衰减率 Weight decay | 0.0005 |
| 网络终止学习率 End Learning Rate | 0 |
| 网络优化器 Optimizer | 随机梯度下降 (SGD) |
| 动量值 Momentum | 0.9 |
| Nesterov 加速梯度法 | 是 |
| 学习率调度器 Scheduler | 余弦 |
| 架构初始学习率 Initial Learning Rate | 0.0003 |
| 架构权重衰减率 Weight decay | 0.001 |
| 架构终止学习率 End Learning Rate | 0 |
| 架构优化器 Optimizer | Adam |
| 迭代次数 Epoch | 200 |



(d) ImageNet16-120 FPGA

图 5-2 (续) NSGA-II 在不同硬件设备上对 CIFAR100 数据集搜索 GDAS 超网络中。其中红色代表使用超网络进行搜索后得到的帕累托前沿。

表 5-2 NSGA-II 搜索参数

| | |
|----------|-----|
| 迭代次数 | 20 |
| 种群大小 | 50 |
| 变异概率 | 0.1 |
| 交叉种群 | 10 |
| 基于交叉后代数量 | 20 |
| 基于变异后代数量 | 25 |
| 随机生成后代数量 | 5 |

5.4 实验结果分析

表5-3,5-4,5-5,5-6分别展示了各类基线方法在 EdgeGPU、Eyeriss、FPGA、Raspi 上的表现。我们将分别分析不同基线超网络模型训练方法在不同硬件和数据集上的搜索结果，以及分析不同种多目标优化方法在 GDAS 上的搜索情况。

首先，在基线的超网络模型训练方法中，在精度方面，GDAS 超过了 DARTS 和 RSPS，取得了最佳的精度；但与此同时，其时延与能耗在 4 个设备上往往有更高的要求。当然，其中也不乏模型精度和硬件性能均优的架构，如 GDAS 在 EdgeGPU、Cifar10 的搜索结果，这为我们的多目标搜索指明了可行性的方向。其次，对于相同的候选架构，其在不同硬件上的性能是不同的，而且，架构间性能的排序在不同的硬件上是不同的，比如，RSPS 和 GDAS 关于时延和能耗的排序上，cifar10 和 ImageNet16-120 的排序是相反的。因而，在不同数据集、不同硬件上进行定制化的搜索是颇有必要。

随后，我们分析在最佳的预训练超网络 GDAS 中，加入我们的遗传算法搜索结果。由于 NSGA 搜索的是一个帕累托最优集合，因而，我们仅挑选一代表性的结果放置于表格中。图5-2展示了基于 NSGA 得到的帕累托最后前沿的分布，以及基于 WSEA 和 CEA 下前 10 模型的分布。可以看到，无论是 WSEA 或是 CEA，前 10 的结果均会集中分布在一小块区域。这限制了用户在多目标下候选模型的选择，而且对于不同目标间权重的设计其实是费劲的。对于 WSEA 而言，由于候选模型的均值方差统计本身具有较大的波动性、而且某些目标间可能存在相关性（如能耗和时延在某些设备上几乎完全呈现正相关的关系），故而有可能存在得到较为极端的结果，如在 Pixel3 的结果，在时延、能耗达到最低的同时，模型精度被大大放弃

表 5-3 EdgeGPU 上的多目标搜索结果

| | Cifar10 | | | Cifar100 | | | ImageNet16-120 | | |
|-----------|---------|---------|---------|----------|---------|---------|----------------|---------|---------|
| | 精度 (%) | 时延 (ms) | 能耗 (mJ) | 精度 (%) | 时延 (ms) | 能耗 (mJ) | 精度 (%) | 时延 (ms) | 能耗 (mJ) |
| DARTS | 54.30 | 3.737 | 16.044 | 55.15 | 3.327 | 14.736 | 26.07 | 2.405 | 10.094 |
| RSPS | 87.08 | 5.224 | 23.345 | 60.64 | 4.968 | 21.933 | 25.77 | 2.441 | 10.185 |
| GDAS | 93.67 | 6.162 | 31.083 | 66.76 | 3.172 | 14.847 | 41.02 | 6.288 | 28.329 |
| GDAS+WSEA | 10.00 | 0.501 | 2.059 | 66.80 | 1.753 | 8.051 | 0.830 | 0.533 | 2.192 |
| GDAS+CEA | 92.99 | 4.967 | 23.204 | 70.05 | 4.542 | 21.146 | 43.25 | 5.551 | 24.223 |
| GDAS+NSGA | 92.95 | 4.734 | 22.294 | 70.20 | 6.375 | 30.929 | 41.45 | 4.082 | 17.447 |

表 5-4 Eyeriss 上的多目标搜索结果

| | Cifar10 | | | Cifar100 | | | ImageNet16-120 | | |
|-----------|---------|---------|---------|----------|---------|---------|----------------|---------|---------|
| | 精度 (%) | 时延 (ms) | 能耗 (mJ) | 精度 (%) | 时延 (ms) | 能耗 (mJ) | 精度 (%) | 时延 (ms) | 能耗 (mJ) |
| DARTS | 54.30 | 1.679 | 0.347 | 55.15 | 1.683 | 0.347 | 26.07 | 0.484 | 0.097 |
| RSPS | 87.08 | 6.103 | 1.215 | 60.64 | 5.123 | 0.965 | 25.77 | 0.792 | 0.161 |
| GDAS | 93.67 | 9.052 | 1.913 | 66.76 | 4.632 | 0.974 | 41.02 | 2.266 | 0.489 |
| GDAS+WSEA | 87.46 | 1.925 | 0.378 | 68.08 | 3.402 | 0.692 | 26.68 | 0.423 | 0.089 |
| GDAS+CEA | 91.77 | 4.874 | 0.969 | 69.02 | 5.123 | 1.036 | 45.40 | 1.344 | 0.274 |
| GDAS+NSGA | 92.64 | 3.399 | 0.691 | 69.69 | 4.877 | 1.005 | 45.40 | 1.344 | 0.274 |

了。对于 CEA，由于其将多目标问题转为单目标优化，且其实际搜索数量大于其他方法，能较好权衡准确率和硬件性能的关系，在与原 GDAS 搜索的精度相当的情况下，获得了更优的硬件性能水平。对于 NSGA-II，则可以很好地探索到整个帕累托最优边界，其解空间覆盖上述两种方法的空间，提供给用户更加灵活的架构选择空间。

5.4.1 不同数据集与硬件下算法分析

图5-3展示了在不同数据集上使用 NSGA-II 算法基于 GDAS 预训练的超网络，在硬件 Eyeriss 上进行搜索的结果。而图5-4则展示了在不同硬件上使用 NSGA-II 算法基于 GDAS 预训练的超网络，在 Cifar100 数据集上进行搜索的结果。在这两张图中，代理错误率表示基于预训练超网络权重进行

表 5-5 FPGA 上的多目标搜索结果

| | Cifar10 | | | Cifar100 | | | ImageNet16-120 | | |
|-----------|---------|---------|---------|----------|---------|---------|----------------|---------|---------|
| | 精度 (%) | 时延 (ms) | 能耗 (mJ) | 精度 (%) | 时延 (ms) | 能耗 (mJ) | 精度 (%) | 时延 (ms) | 能耗 (mJ) |
| DARTS | 54.30 | 1.132 | 7.931 | 55.15 | 1.132 | 7.931 | 26.07 | 0.415 | 2.908 |
| RSPS | 87.08 | 3.209 | 22.468 | 60.64 | 2.982 | 20.877 | 25.77 | 0.387 | 2.707 |
| GDAS | 93.67 | 7.421 | 51.946 | 66.76 | 3.648 | 25.537 | 41.02 | 2.922 | 20.451 |
| GDAS+WSEA | 84.01 | 1.132 | 7.931 | 54.64 | 1.132 | 7.931 | 26.10 | 0.284 | 1.990 |
| GDAS+CEA | 93.17 | 3.725 | 26.074 | 68.67 | 3.831 | 26.814 | 37.93 | 1.087 | 7.612 |
| GDAS+NSGA | 91.51 | 3.315 | 23.207 | 69.69 | 4.162 | 29.143 | 42.08 | 0.969 | 6.787 |

表 5-6 Raspi4 上的多目标搜索结果

| | Cifar10 | | Cifar100 | | ImageNet16-120 | |
|-----------|---------|---------|----------|---------|----------------|---------|
| | 精度 (%) | 时延 (ms) | 精度 (%) | 时延 (ms) | 精度 (%) | 时延 (ms) |
| DARTS | 54.30 | 3.839 | 55.15 | 2.886 | 26.07 | 1.699 |
| RSPS | 87.08 | 27.740 | 60.64 | 12.916 | 25.77 | 1.034 |
| GDAS | 93.67 | 82.077 | 66.76 | 33.332 | 41.02 | 15.853 |
| GDAS+WSEA | 91.89 | 24.527 | 69.03 | 20.509 | 41.30 | 3.794 |
| GDAS+CEA | 92.07 | 24.569 | 68.59 | 26.375 | 43.25 | 6.189 |
| GDAS+NSGA | 93.36 | 38.414 | 70.14 | 33.975 | 45.64 | 8.767 |

模型测试的错误率，而真实错误率则表示对候选网络进行从头训练后的模型错误率。我们可以发现以下几点事实：

首先，代理错误率和真实错误率的分布形状存在较大偏差。代理错误率的分布大量集中在错误率较高的截面，而真实错误率的分布主要集中在错误率较低的一侧。这表明超网络的预训练不充分，导致许多候选网络的性能被低估。

其次，上述现象进一步导致帕累托最优前沿的分布不同。因为搜索算法根据代理错误率进行优化，因此在代理错误率分布的截面上，帕累托最优前沿的密度较高。但是，由于真实错误率的分布不同，实际帕累托最优前沿的分布也随之不同。

然而，即使存在上述事实，我们仍然可以通过权重共享模型的测试得到帕累托最优前沿。具体来说，我们发现在边界的中间部分，权重共享模型得到的帕累托最优前沿几乎与实际前沿重合。这说明基于大模型预训练共享参数的方法在遗传算法搜索阶段具有一定的有效性。

综上所述，我们的实验结果表明，在使用遗传算法搜索神经网络架构时，应该充分考虑超网络预训练的不足，并且需要注意代理错误率和真实错误率的分布差异。此外，我们的实验还表明，在搜索算法中使用权重共享模型的方法是可行的，可以在一定程度上提高搜索效率。

5.4.2 NSGA 搜索过程分析

当使用 NSGA-II 算法搜索神经网络架构时，帕累托最优前沿的变化情况可以提供有用的信息。在每一轮迭代中，NSGA-II 算法会根据当前帕累

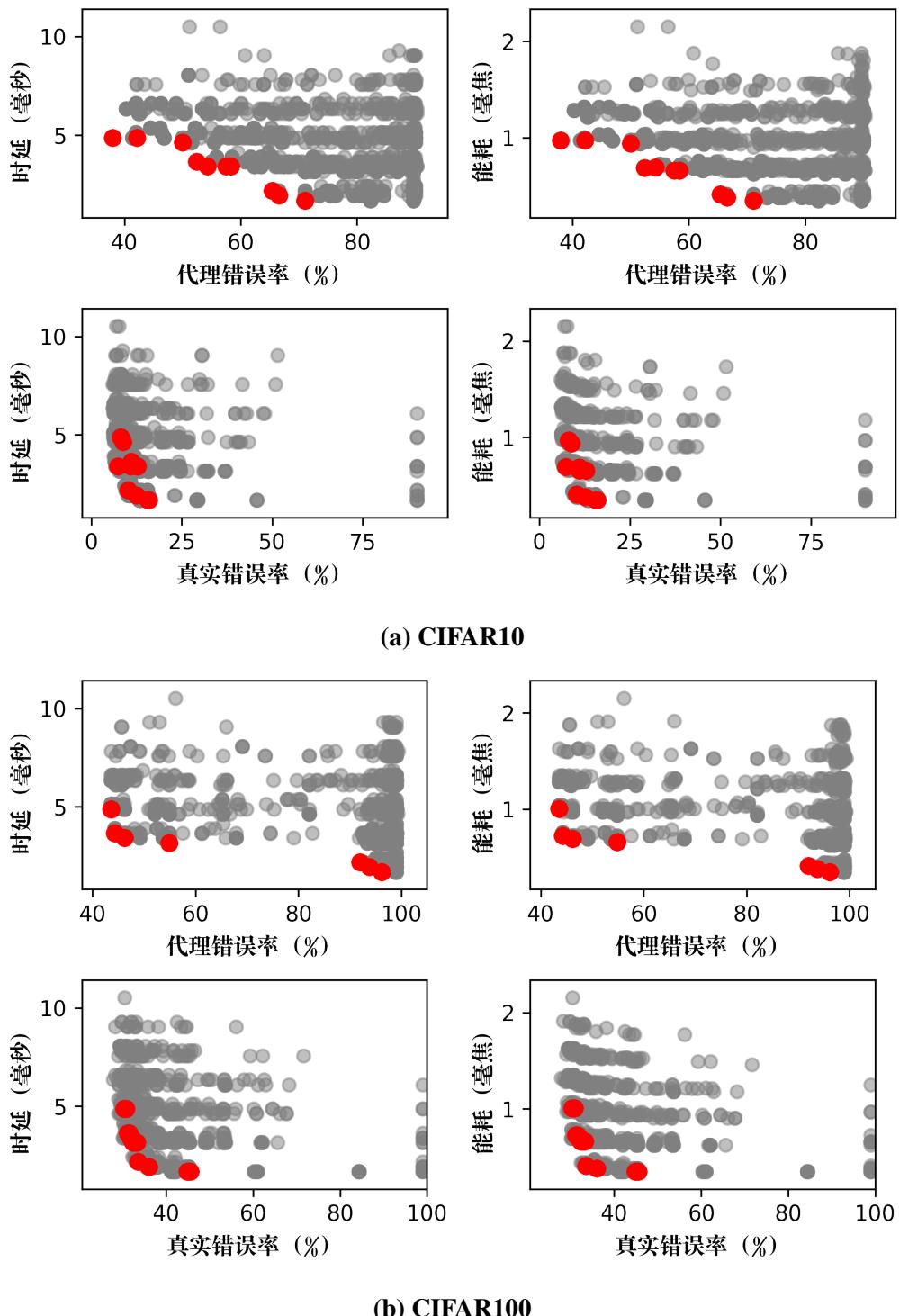


图 5-3 NSGA-II 在不同数据集上对 Eyeriss 搜索 GDAS 超网络中。其中红色代表使用超网络进行搜索后得到的帕累托前沿。

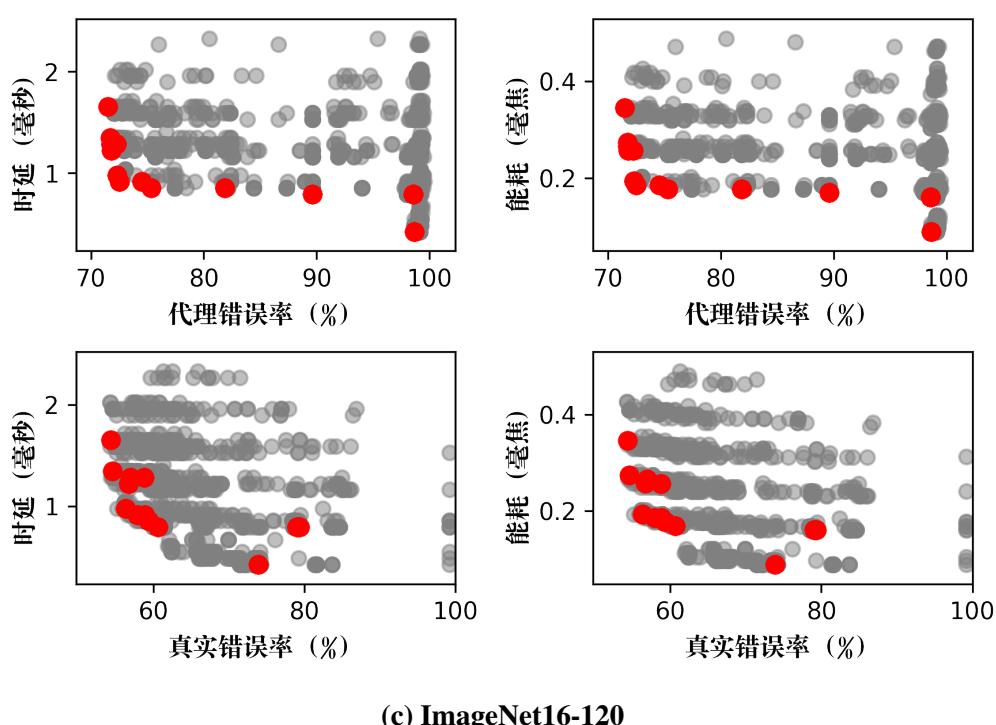


图 5-3 (续) NSGA-II 在不同数据集上对 Eyeriss 搜索 GDAS 超网络中。
其中红色代表使用超网络进行搜索后得到的帕累托前沿。

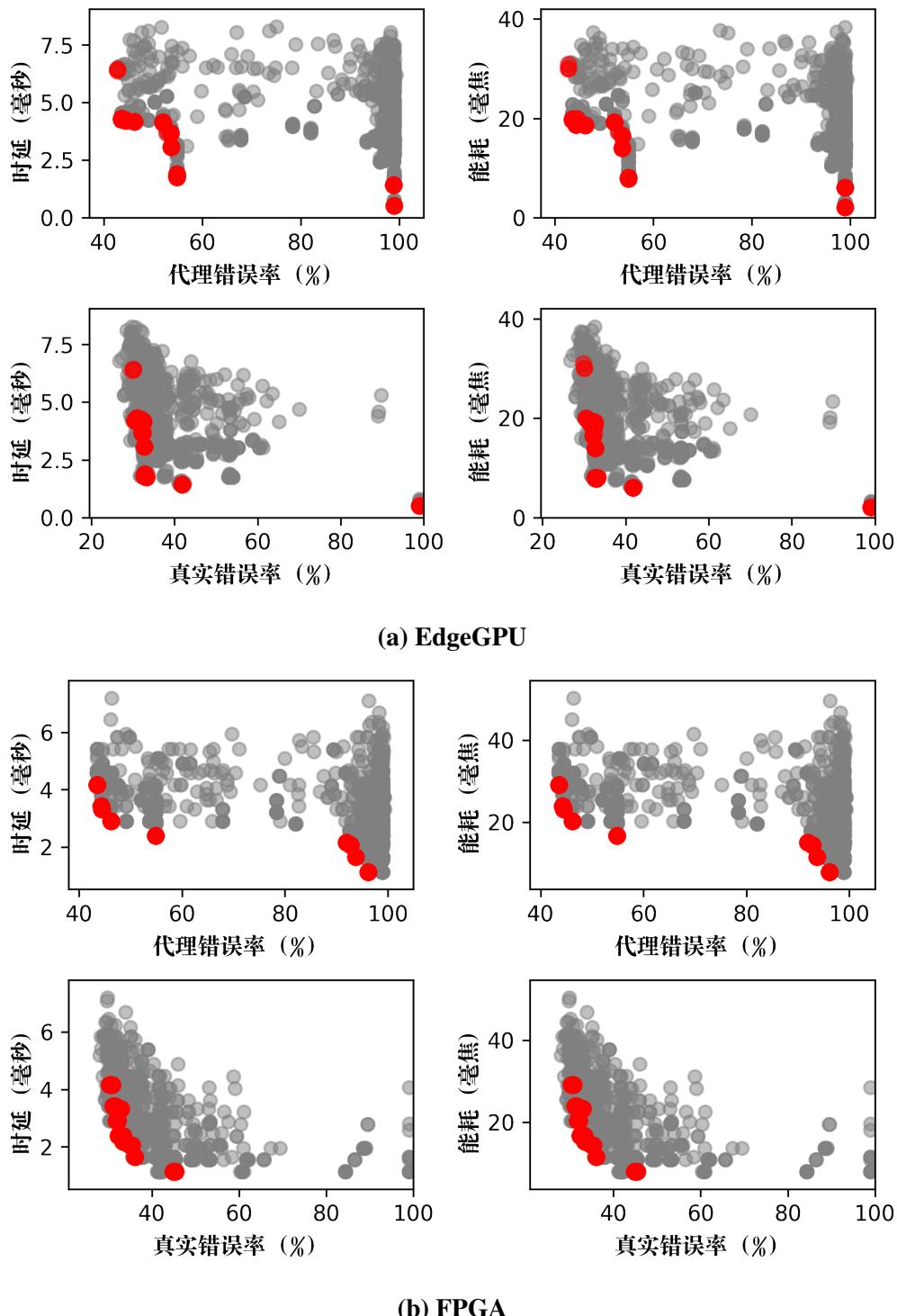


图 5-4 NSGA-II 在不同硬件设备上对 CIFAR100 数据集搜索 GDAS 超网络中。其中红色代表使用超网络进行搜索后得到的帕累托前沿，灰色点代表搜索过程中出现的数据点。

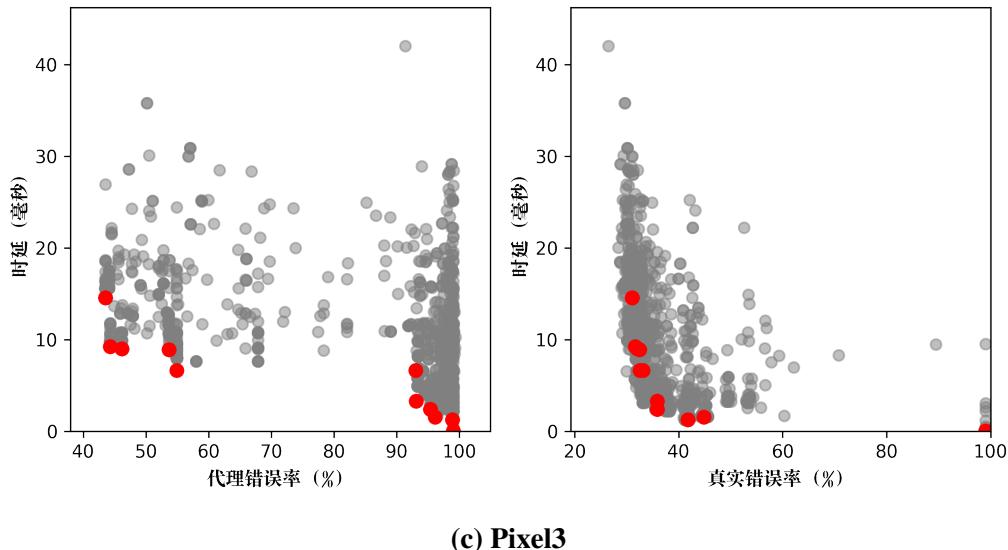


图 5-4 (续) NSGA-II 在不同硬件设备上对 CIFAR100 数据集搜索 GDAS 超网络中。其中红色代表使用超网络进行搜索后得到的帕累托前沿，灰色点代表搜索过程中出现的数据点。

托最优前沿中的个体，生成下一代个体，以期望获得更好的性能。因此，帕累托最优前沿的变化情况可以反映搜索过程中的性能提升和局限性。

为了更直观地展示这一过程，我们提供了一个可视化的结果，如图5-5所示。该图显示了 NSGA-II 算法在 Cifar100 数据集上对 Eyeriss 硬件上搜索 GDAS 超网络迭代过程的结果。每个红色圆点代表帕累托最优前沿中的一个个体，其坐标表示候选架构待优化的目标（模型误差、时延、能耗），我们分别提供了 3D 视角和投影至 2D 截面的视角。可以看到，随着迭代次数的增加，帕累托最优前沿逐渐向着左下方移动。这表明 NSGA-II 算法逐步寻找到更小、更精确的神经网络架构，并且在搜索过程中不断提高性能。

5.5 本章小结

本章中，我们在一个公开的神经架构搜索基准 NAS-BENCH-201 中进行实验，展现了基于我们的框架有效且快速地针对特定数据集、硬件下的特定搜索空间进行软硬件多目标搜索，并由用户自由地选择合适的候选框架。

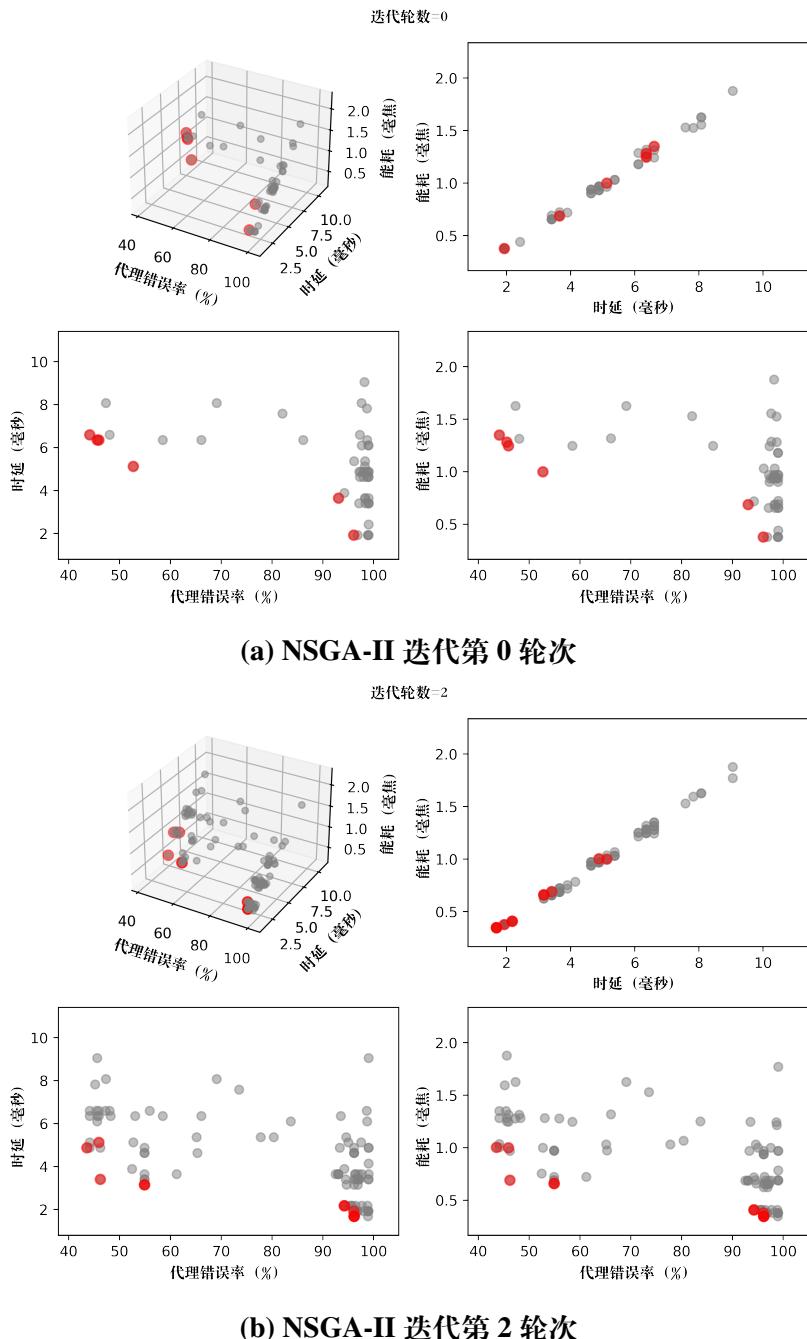


图 5-5 NSGA-II 在 CIFAR100 数据集、Eyeriss 硬件上搜索 GDAS 超网络迭代过程的种群变化。其中红色代表帕累托前沿。

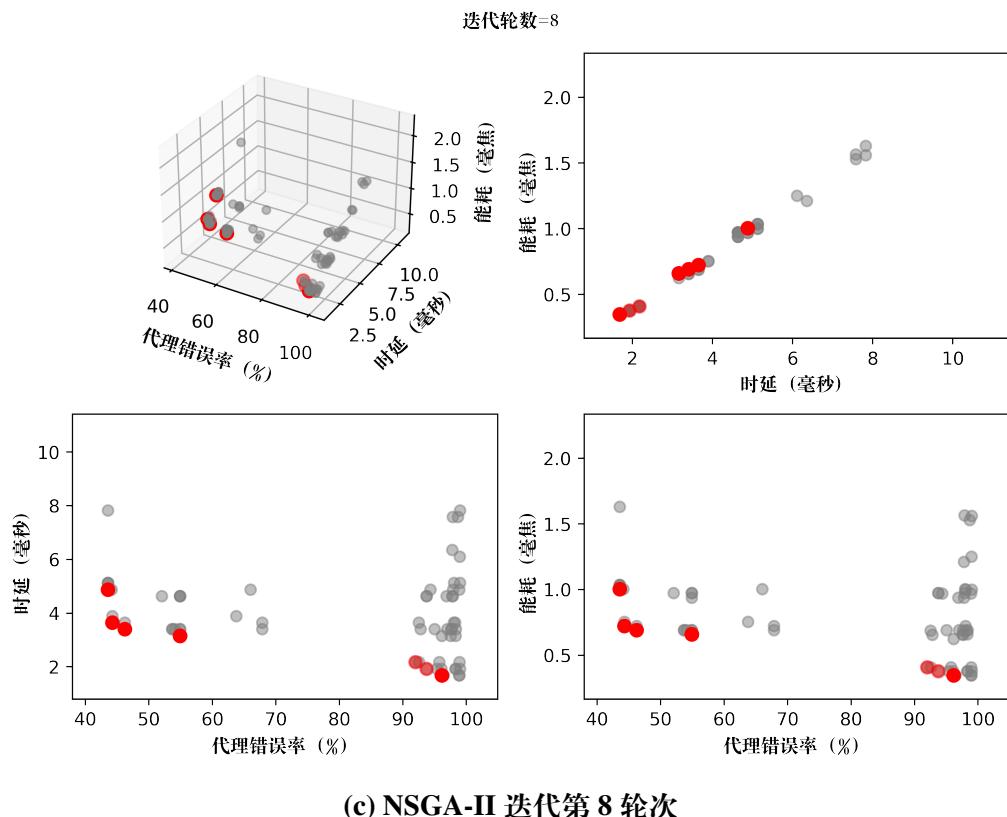


图 5-5 (续) NSGA-II 在 CIFAR100 数据集、Eyeriss 硬件上搜索 GDAS 超网络迭代过程的种群变化。其中红色代表帕累托前沿。

第六章 国产芯片应用示例

6.1 应用芯片介绍和目标设定

为了在实际应用中测试我们的框架，我们使用了 RockChip RK3588S 开发板作为硬件平台进行应用实验。该开发板采用 8nm 先进工艺，配备了八核 64 位处理器（包括 4 个 Cortex-A76 核和 4 个 Cortex-A55 核），主频高达 2.4GHz，拥有 32GB 运行内存。此外，该开发板还配备了 NPU，其算力高达 6 TOPS，支持 INT4/INT8/INT16 混合运算，这使得我们可以在该硬件平台上进行高效的神经网络计算。

为了进一步加速模型的吞吐率，我们运用到了该硬件平台的量化功能。具体地，我们支持将浮点模型量化为定点模型，目前支持的量化方法为非对称量化。这使得我们可以将浮点模型转换为更适合于硬件平台上进行推理的定点模型，并在该开发板上进行高效的推理。我们通过统计 400 次模型前向推理的时间的平均值作为模型单次推断的时延。这样的统计量，相比以往工作中统计 FLOPs、MACs、模型参数大小，更能直接反映实际芯片中开发者对模型的需求。

6.2 搜索空间和大模型训练方式

为了同人工设计的框架进行更好的对比以及更便捷的端侧模型搭建，我们采用层级搜索空间进行实验。因而，我们使用单路单次大模型训练方法（Single Path One Shot，简称 SPOS）对单次大模型进行训练。

我们对 3 个搜索空间进行了搜索，以期望达到最佳的效果：

- 基于 ShuffleNet^[16]的搜索空间（后文简称 ShuffleSpace），即原 SPOS 中使用的搜索空间，模型深度 20 层。该空间共有 4 个候选操作算子， 3×3 、 5×5 、 7×7 通道重排网络模块和 3×3 Xception 模块。
- 在 ShuffleNet 空间基础上，加入不变层（Identity），等价于减小网络深度（后文简称 ShuffleSpace+I）。后文我们将展示这个算子的加入可以极大提升模型的硬件性能；

表 6-7 网络训练超参数

| | |
|-----------------------------|--------------|
| 批量大小 Batch Size | 96 |
| 随机翻转 Random Flip | 0.5 |
| 随机裁剪 Random Crop | 是 |
| 归一化 Normalization | 是 |
| 初始学习率 Initial Learning Rate | 0.025 |
| 终止学习率 End Learning Rate | 0 |
| 权重衰减率 Weight decay | 0.0003 |
| 优化器 Optimizer | 随机梯度下降 (SGD) |
| 动量值 Momentum | 0.9 |
| Nesterov 加速梯度法 | 是 |
| 学习率调度器 Scheduler | Lambda |

- 基于 MobileNetv2^[15]的搜索空间, 模型深度 20 层(后文简称 MobileSpace+I)。我们共设定共有 7 个候选操作算子, 卷积核有 $k \in \{3, 5, 7\}$ 的选择, 扩张比 $e \in \{3, 6\}$, 以及不变层。

我们 NSGA-II 搜索的超参数与表5-2一致, 单次超网络和候选网络重新训练的超参数如表6-7, 单次超网络训练 900 轮次, 而重新训练仅需要 300 轮次。

6.3 实验结果

图6-6展示了我们在 Cifar10 上的搜索结果。相比人工设计的 ShuffleNet 和经过搜索的 EfficientNet, 原 SPOS 的搜索空间在错误率和时延上均处于劣势。通过比较上述基线模型和 SPOS 搜索空间在宏观架构上的区别, 我们在搜索空间中加入了不变层, 使得模型深度可变, 于是便能达到和基线相近的性能, 同时还找到了延时更低的模型。但上述空间在 Rockchip 的性能仍与 MobileNetv2 有较大的差距, 因而我们设计了第三类基于 MB Conv Block 的搜索空间, 于是找到了比 MobileNetv2 更优的模型架构。以上我们发现, 在寻找针对特定硬件的模型结构时, 我们的框架提供了很好的修正指导作用。

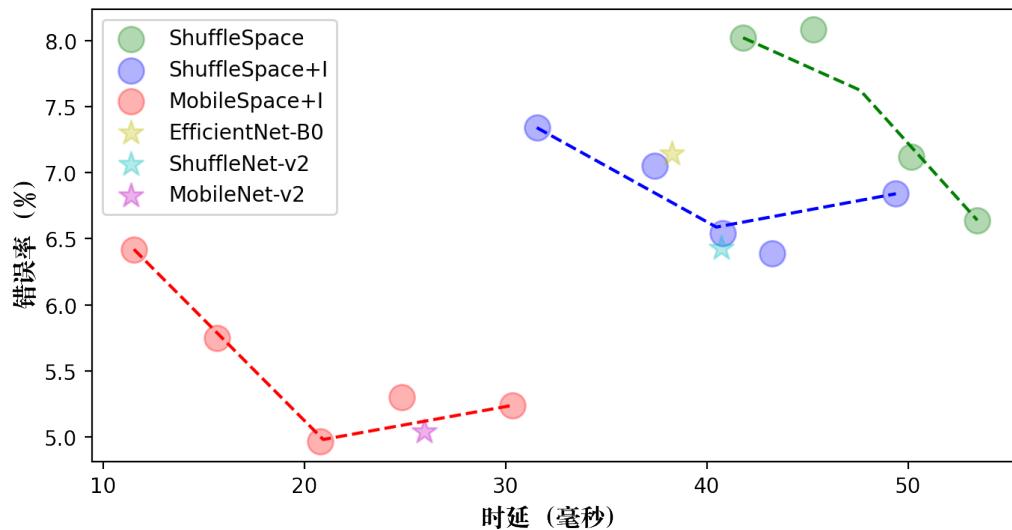


图 6-6 Cifar10 上不同搜索空间的帕累托最优前沿与常用模型的比较

6.4 本章小结

本章介绍了在 RockChip RK3588S 开发板上进行的国产芯片应用示例。首先，我们介绍了硬件平台的基本参数和量化功能，并说明了如何通过统计模型前向推理的时间来反映实际芯片中开发者对模型的需求。其次，我们采用层级搜索空间进行实验，并使用单路单次大模型训练方法进行训练。我们对三个搜索空间进行了搜索，并比较了与不同已有常见模型架构的性能。最后，我们展示了在特定硬件上针对性地设计的搜索空间，能够找到比 MobileNetv2 更优的模型架构。通过该应用示例，我们验证了我们的框架在实际硬件平台上的有效性和可用性。

第七章 全文总结

7.1 主要结论

本文提出了一种软硬件协同设计框架，旨在解决传统神经网络架构设计难以满足各种设备和应用场景需求的问题。该框架采用了一种在单次预训练超网络上采样子网络进行多目标优化的方法，结合权重共享技术和 NSGA-II 遗传算法，以快速、精准地找到适合特定硬件的高效神经架构。本文的贡献在于提出了一种硬件扩展性好、非硬件专家友好的多目标输出方法，同时作为在特定硬件下搜索空间的分析工具，使用户可以根据需求自主选择模型。

本文在多个硬件平台上进行了评估，实验结果表明，该方法可以提高神经架构搜索的效率，而基于多目标优化的 NSGA-II 遗传算法也能够在硬件性能与模型准确性之间实现平衡。同时，在一国产 AI 加速芯片上进行的实验表明，该方法可以不断改进搜索空间，以获得软硬件下的最优模型，同时在不同硬件上具有良好的泛化性能，为其他领域的研究和应用提供参考和借鉴。因此，本文的方法具有较高的实际应用价值，并能够为神经架构搜索和优化领域的研究提供重要参考。

7.2 研究展望

本文的方法为神经架构搜索和优化领域提供了一种新的思路和解决方案。然而，在未来的研究中，还有许多值得深入探讨的问题和方向：

首先，可以进一步提升单次超网络训练的充分性和有效性，以使基于共享权重的代理错误率更好地表征候选模型在真实错误率中的排名位次和分布。例如，可以通过更好的训练策略、更优秀的超网络结构等方式来提升预训练的质量和准确性，进而增强搜索的效率和精度。

其次，除了对模块搜索，还可以进一步探索如特征通道数、模型深度等搜索空间的更好方式，以期达到更好的模型性能。同时，可以探索不同搜索空间的结合方式，如对搜索空间进行分层、分阶段等策略，以期更高效、更精准地搜索出最优模型。

第三，需要探索多目标优化下的更公平的评价标准，以便对各类方法

做更公正的比较。当前多目标优化存在许多评价标准，如精度、时延、能耗等，但它们之间存在着相互制约的关系，因此需要进一步研究和制定更合理的评价标准和权重设计，以便在不同的应用场景下实现最佳的软硬件协同设计。

最后，在实际应用中，对候选架构进行硬件性能的统计仍然是搜索的瓶颈。未来可以通过硬件加速或分布式计算等方式来加速搜索过程，同时还可以进一步探索更有效的算法和策略，以期更快地找到最优的神经架构。

参 考 文 献

- [1] ZOPH B, LE Q. Neural architecture search with reinforcement learning [C/OL]//International Conference on Learning Representations. 2017. <https://openreview.net/forum?id=r1Ue8Hcxg>.
- [2] REAL E, MOORE S, SELLE A, et al. Large-scale evolution of image classifiers[C]//ICML'17: Proceedings of the 34th International Conference on Machine Learning - Volume 70. Sydney, NSW, Australia: JMLR.org, 2017: 2902–2911.
- [3] KANDASAMY K, NEISWANGER W, SCHNEIDER J, et al. Neural architecture search with bayesian optimisation and optimal transport[C]//NIPS, 2018.
- [4] ZOPH B, VASUDEVAN V, SHLENS J, et al. Learning transferable architectures for scalable image recognition[C/OL]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Los Alamitos, CA, USA: IEEE Computer Society, 2018: 8697-8710. <https://doi.ieeecomputer-society.org/10.1109/CVPR.2018.00907>.
- [5] LIU H, SIMONYAN K, YANG Y. DARTS: Differentiable architecture search[C/OL]//International Conference on Learning Representations. 2019. <https://openreview.net/forum?id=S1eYHoC5FX>.
- [6] JIANG Y, HU C, XIAO T, et al. Improved differentiable architecture search for language modeling and named entity recognition[C/OL]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019: 3585-3590. <https://aclanthology.org/D19-1367>. DOI: [10.18653/v1/D19-1367](https://doi.org/10.18653/v1/D19-1367).
- [7] XU Y, XIE L, ZHANG X, et al. PC-DARTS: partial channel connections for memory-efficient architecture search[C/OL]//8th International Conference

- on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020. <https://openreview.net/forum?id=BJIS634tPr>.
- [8] CHEN X, XIE L, WU J, et al. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation[C]//Proceedings of the IEEE International Conference on Computer Vision. 2019: 1294-1303.
- [9] WU B, DAI X, ZHANG P, et al. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search[C/OL]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019: 10726-10734. DOI: [10.1109/CVPR.2019.01099](https://doi.org/10.1109/CVPR.2019.01099).
- [10] LIN J, CHEN W M, COHN J, et al. Mcunet: Tiny deep learning on iot devices[C]//Annual Conference on Neural Information Processing Systems (NeurIPS). 2020.
- [11] NAYMAN N, AFLALO Y, NOY A, et al. Hardcore-nas: Hard constrained differentiable neural architecture search[C]//MEILA M, ZHANG T. Proceedings of Machine Learning Research: volume 139 Proceedings of the 38th International Conference on Machine Learning. PMLR, 2021: 7979-7990.
- [12] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[C]//PEREIRA F, BURGES C, BOTTOU L, et al. Advances in Neural Information Processing Systems: volume 25. Curran Associates, Inc., 2012.
- [13] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[J]. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015: 770-778.
- [14] HOWARD A G, ZHU M, CHEN B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J/OL]. CoRR, 2017, abs/1704.04861. <http://arxiv.org/abs/1704.04861>.

- [15] 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 (CVPR). 2018.
- [16] MA N, ZHANG X, ZHENG H T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018.
- [17] PHAM H, GUAN M, ZOPH B, et al. Efficient neural architecture search via parameters sharing[C]//DY J, KRAUSE A. Proceedings of Machine Learning Research: volume 80 Proceedings of the 35th International Conference on Machine Learning. PMLR, 2018: 4095-4104.
- [18] BROCK A, LIM T, RITCHIE J, et al. SMASH: One-shot model architecture search through hypernetworks[C/OL]//International Conference on Learning Representations. 2018. <https://openreview.net/forum?id=rydeCEhs->.
- [19] BENDER G, KINDERMANS P J, ZOPH B, et al. Understanding and simplifying one-shot architecture search[C]//DY J, KRAUSE A. Proceedings of Machine Learning Research: volume 80 Proceedings of the 35th International Conference on Machine Learning. PMLR, 2018: 550-559.
- [20] GUO Z, ZHANG X, MU H, et al. Single path one-shot neural architecture search with uniform sampling[C/OL]//Computer Vision –ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI. Berlin, Heidelberg: Springer-Verlag, 2020: 544–560. https://doi.org/10.1007/978-3-030-58517-4_32.
- [21] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[C]//International Conference on Learning Representations. 2015.
- [22] CAI H, ZHU L, HAN S. ProxylessNAS: Direct neural architecture search on target task and hardware[C/OL]//International Conference on Learning Representations. 2019. <https://arxiv.org/pdf/1812.00332.pdf>.

- [23] HU J, SHEN L, SUN G. Squeeze-and-excitation networks[C/OL]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018: 7132-7141. DOI: [10.1109/CVPR.2018.00745](https://doi.org/10.1109/CVPR.2018.00745).
- [24] DONG X, YANG Y. Searching for a robust neural architecture in four gpu hours[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.
- [25] JANG E, GU S, POOLE B. Categorical reparametrization with gumbel-softmax[C/OL]//Proceedings International Conference on Learning Representations (ICLR). 2017. <https://openreview.net/pdf?id=rkE3y85ee>.
- [26] DEB K, AGRAWAL S, PRATAP A, et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii[J]. IEEE Trans. Evol. Comput., 2002, 6: 182-197.
- [27] DONG X, YANG Y. Nas-bench-201: Extending the scope of reproducible neural architecture search[C/OL]//International Conference on Learning Representations (ICLR). 2020. <https://openreview.net/forum?id=HJxyZkBKDr>.
- [28] LI C, YU Z, FU Y, et al. {HW}-{nas}-bench: Hardware-aware neural architecture search benchmark[C/OL]//International Conference on Learning Representations. 2021. https://openreview.net/forum?id=_0kaDkv3dVf.
- [29] LI L, TALWALKAR A. Random search and reproducibility for neural architecture search[C/OL]//ADAMS R P, GOGATE V. Proceedings of Machine Learning Research: volume 115 Proceedings of The 35th Uncertainty in Artificial Intelligence Conference. PMLR, 2020: 367-377. <https://proceedings.mlr.press/v115/li20c.html>.

攻读学位期间学术论文和科研成果目录

- [1] Qibing Ren, Qingquan Bao, Runzhong Wang, Junchi Yan. Appearance and Structure Aware Robust Deep Visual Graph Matching: Attack, Defense and Beyond[C]. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2022. (已录用)
- [2] Shenyu Zhang, Zichen Zhu, Qingquan Bao. Rb-PaStaNet: A Few-Shot Human-Object Interaction Detection Based on Rules and Part States[C]. Irish Machine Vision and Image Processing Conference, 2020. (已录用)

致 谢

我非常荣幸能够在我的毕业设计中获得严骏驰老师的指导和支持，严老师在整个过程中一直以其开放温和的态度和深厚的学识为我提供了宝贵的指导和建议。同时，我还要感谢我的学长王晓星同学，在整个项目期间一直和我进行深入的讨论和交流，不断提供宝贵的建议和意见，使我能够更好地把握神经架构搜索问题的本质和当下该领域面临的难点与挑战，从而取得了更好的研究成果。没有严老师和王学长的帮助和支持，我的毕业设计无论是在理论还是实践方面都难以取得这样的成果。我十分感激严老师和王学长的悉心指导和支持！

回顾四年，无数交大优秀的学长老师为我奠定了扎实的机器学习和深度学习的理论实践基础。我十分有幸能受到江波、张驰豪、严骏驰、俞凯、卢策吾、张伟楠、沈为等老师的教学传授。在卢老师的课上，经李永露、徐良两位学长的指导，我在大一便走入科研的大门，同时收获了扎实的深度学习基础代码能力和论文写作能力；在严老师的 ReThinklab，我有幸与汪润中、任麒冰学长一同，在图匹配的前沿领域中不断磨练自身的代码能力和科研能力。我也颇能有幸在本科期间前往 MIT 进行科研实习，Leslie Kaelbling、Tomás Lozano-Pérez、Josh Tenenbaum、Chuang Gan、Yilun Du，他们让我见识到了何为真正的热爱、何为求知不尽的学者、何为徐徐向前却不气馁般得开拓知识前沿。

最后，感谢人工智能-致远专业的朋辈、学生会结识的战友、核心社交群的挚友、范绪箕奖学金相遇的同侪良师、远在波士顿结识的兄长姐、以及我的家人，在我最失意之时给予我支持，在我最困顿之时给予我鼓励，并以某些方式提醒着我，这世上依然有更重要的事去达成。

凡是过往，皆为序章。希望在未来的人生道路上，我能够有 George Mallory 的勇气，面对高峰，以三个字坚决地走下去，“Because it's there”。

VISUAL MODEL SEARCH AND OPTIMIZATION FOR SOFTWARE AND HARDWARE CO-DESIGN

With the widespread use of Convolutional Neural Networks (CNNs), there is an increasing demand for neural networks with fast inference speed and high accuracy. Initially, neural network architectures were manually designed, such as VGGNet and ResNet, and optimized for powerful GPUs, which are the main computing platform for deep CNNs. However, these architectures and their variants were considered standard until the need to deploy them on edge devices and standard CPUs became a priority. Designing the best neural architecture for various devices manually can be expensive. Therefore, more and more researchers have turned their attention to automating neural architecture design.

Neural Architecture Search (NAS) is an automated deep learning model design and training method that can achieve optimal performance with minimal human intervention. NAS uses various search algorithms such as evolutionary algorithms or reinforcement learning to explore a vast space of possible architectures and determine the ones that perform well on a given task. Compared to manually designed architectures, NAS can save a lot of time and effort and produce models with better performance.

In the early stages of NAS, a predefined search space, typically a set of operation sets, was used along with a controller to generate candidate neural architectures. After training the candidate architectures on the training set, they were evaluated and ranked on the validation set, with the ranking providing feedback to guide the controller to generate new candidates. Finally, the best neural structure was selected and tested on the test set.

Early NAS methods used discrete search strategies such as random search, reinforcement learning, evolutionary algorithms, and Bayesian optimization to search all possible components in the global space. However, global space search requires too much computational cost. For example, a reinforcement learning-based neural architecture search requires 2,000 GPU days to obtain state-of-the-art architectures for CIFAR-10 and ImageNet, while an evolutionary algorithm

requires 3,150 GPU days. Meanwhile, they ignored human prior factors in excellent neural architecture designs.

To address these issues, we propose a software and hardware co-design framework that can achieve automated neural architecture search and optimization, achieving efficient inference performance on specific hardware. Specifically, we focus on the following issues: (1) How to efficiently evaluate the performance of the architecture during the search process to achieve faster convergence and avoid unnecessary computational costs? (2) How to incorporate hardware performance into the search objective to generate efficient architectures suitable for specific hardware, while ensuring scalability and ease of use?

Previous work on hardware-aware NAS shows a similar interest with us. However, existing methods still have some drawbacks when searching for neural architectures under hardware constraints. For example, they often use soft constraints, like FBNet and ProxylessNAS, which may violate hard resource constraints. Additionally, the final discretization step projects the architecture from a differentiable search space to a discrete architecture space, which further violates hard resource constraints. To address these issues, recent methods such as HardCoRe-NAS have emerged, but they require specialized hardware knowledge and still have some limitations.

Our proposed method is a multi-objective optimization with sampled subnetworks on a one-shot pretrained supernet, which uses weight-sharing technology to accelerate the search process. Specifically, our proposed method consists of the following steps: (1) Training a one-shot pretrained supernet: Under the given search space conditions, we train each candidate operation operator in a single supernet, where we only train the accuracy of the model network. Our framework supports both cell-wise and layer-wise search spaces. (2) Multi-objective search: Based on the pre-trained single supernet, combined with hardware performance, we use the multi-objective genetic algorithm NSGA-II to search for the Pareto-optimal architecture set.

Our proposed method has several advantages. Firstly, it can decouple network weight training from hardware performance and search for models deployed

on different hardware devices based on the same pre-trained hyper-network parameters. Secondly, our approach uses the final hardware performance indicators directly, leading to better search results and avoiding the challenges of obtaining hardware information at the block level and expensive calculations. Thirdly, our approach can search for Pareto-optimal solutions across multiple objectives without much manual hyperparameter tuning. This allows users to understand the shape of the Pareto optimal boundary of a specific search space in a specific hardware, and choose the model architecture that meets their needs.

The proposed method was evaluated on several public datasets, including CIFAR-10, CIFAR-100, and ImageNet, as well as on multiple hardware platforms, such as EdgeGPU, Eyeriss, and FPGA. The experimental results demonstrate that our method achieves efficient inference performance while maintaining good accuracy and generalization performance across different hardware platforms. Additionally, the proposed framework was applied to a commercial AI-accelerator chip (Rockchip) and compared with state-of-the-art models, such as ShuffleNet, EfficientNet, and MobileNetv2, on the CIFAR-10 dataset. The findings indicate that the proposed framework achieves comparable performance to ShuffleNet and EfficientNet while discovering models with lower latency. However, the performance of the search space lags behind that of MobileNetv2 on the Rockchip platform. To address this limitation, the authors design a third search space based on the MB Conv Block, which discovers a model architecture superior to MobileNetv2. These results suggest that the proposed framework provides valuable guidance for correcting the search space while looking for model architectures tailored to specific hardware.

Furthermore, the experiments reveal some interesting observations, such as the significantly different distribution of proxy errors and true errors, which affect the Pareto frontiers obtained by the multi-objective search algorithm. Despite these differences, the Pareto frontier obtained by weight-sharing models tested in the middle of the boundary almost overlaps with the actual frontier, indicating that weight-sharing methods can improve the search efficiency in the genetic algorithm search phase. Overall, the experiments suggest that when using genetic

algorithms to search for neural network architectures, researchers should consider the shortcomings of the pretrained supernet and the differences between proxy errors and true errors.

Moreover, the Pareto frontiers searched by the proposed methods provide a useful way to analyze the search space at specific hardware and allow users to adjust the search space according to the feedback from the frontiers. By incorporating hardware performance indicators into the search objective, the framework can generate models optimized for specific hardware platforms while ensuring scalability and ease of use. This is particularly useful for optimizing deep learning models for emerging hardware platforms such as AI accelerators and neuromorphic computing devices.

In conclusion, the proposed software and hardware co-design framework for automated neural architecture search and optimization offers a promising approach for designing efficient and optimized neural architectures tailored to specific hardware platforms. The framework can accelerate the search process, reduce human intervention, and improve the accuracy and efficiency of deep learning models. It can also help researchers and practitioners explore the design space of neural architectures for hardware platforms and provide insights into the impact of hardware constraints on model performance. With the increasing demand for deep learning models in various applications and devices, the proposed framework provides a reliable and efficient way to design and optimize neural architectures that can meet the needs of different hardware platforms.