中图分类号: 学校代码: 10055
UDC: 密级: 公开

南開大學

硕 士 学 位 论 文

轻量化视觉模型的自动搜索

Automatic Search for Lightweight Vision Models

论文作者 顾宇超 指导教师 程明明教授
申请学位 硕士 培养单位 南开大学
学科专业 计算机技术 研究方向 计算机视觉
答辩委员会主席 评 阅 人

南开大学研究生院

二〇一八年四月

南开大学学位论文使用授权书

本人完全了解《南开大学关于研究生学位论文收藏和利用管理办法》关于南开大学 (简称“学校”) 研究生学位论文收藏和利用的管理规定，同意向南开大学提交本人的学位论文电子版及相应的纸质本。

本人了解南开大学拥有在《中华人民共和国著作权法》规定范围内的学位论文使用权，同意在以下几方面向学校授权。即：

1. 学校将学位论文编入《南开大学博硕士学位论文全文数据库》，并作为资料在学校图书馆等场所提供阅览，在校园网上提供论文目录检索、文摘及前 16 页的浏览等信息服务；

2. 学校可以采用影印、缩印或其他复制手段保存学位论文；学校根据规定向教育部指定的收藏和存档单位提交学位论文；

3. 非公开学位论文在解密后的使用权同公开论文。

本人承诺：本人的学位论文是在南开大学学习期间创作完成的作品，并已通过论文答辩；提交的学位论文电子版与纸质本论文的内容一致，如因不同造成不良后果由本人自负。

本人签署本授权书一份（此授权书为论文中一页），交图书馆留存。

学位论文作者暨授权人 (亲笔) 签字：_____

20　　年　　月　　日

南开大学研究生学位论文作者信息

论 文 题 目	轻量化视觉模型的自动搜索				
姓　　名	顾宇超	学号	2120190495	答辩日期	
论 文 类 别	博士 ☐　学历硕士 ☐　专业学位硕士 ☑　同等学力硕士 ☐　划☑选择				
学院（单位）	网络空间安全学院		学科/专业 (专业学位) 名称	计算机技术	
联 系 电 话	18559855803		电子邮箱	ycgu@mail.nankai.edu.cn	
通讯地址 (邮编)：300350					
非公开论文编号			备注		

注：本授权书适用我校授予的所有博士、硕士的学位论文。如已批准为非公开学位论文，须向图书馆提供批准通过的《南开大学研究生申请非公开学位论文审批表》复印件和“非公开学位论文标注说明”页原件。

南开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，是本人在导师指导下进行研究工作所取得的研究成果。除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

学位论文作者签名：_____ 年 月 日

非公开学位论文标注说明

(本页表中填写内容须打印)

根据南开大学有关规定，非公开学位论文须经指导教师同意、作者本人申请和相关部门批准方能标注。未经批准的均为公开学位论文，公开学位论文本说明为空白。

论文题目			
申请密级	☐ 限制 (⩽2 年) ☐ 秘密 (⩽10 年) ☐ 机密 (⩽20 年)		
保密期限	20 年 月 日至 20 年 月 日		
审批表编号		**批准日期**	20 年 月 日

南开大学学位评定委员会办公室盖章 (有效)

注：限制 ★2 年 (可少于 2 年); 秘密 ★10 年 (可少于 10 年); 机密 ★20 年 (可少于 20 年)

摘要

轻量化视觉模型设计旨在构建高性能、低时延的模型。由于不同的部署设备，不同的时延限制和不同的视觉任务，视觉模型的探索面临巨大的模型设计空间。人工为不同需求设计专有模型需要消耗巨大的人力资源。受到神经网络架构搜索的启发，本文探索了轻量化视觉模型的自动搜索。自动搜索的难点在于现有搜索算法的模型性能评估和模型真实性能存在偏差，导致其不能搜索出适配的架构。本文对造成偏差的原因进行了深入探索。

首先，本文发现现有的可微网络搜索耦合了操作和拓扑搜索，这种模式会带来模型性能排序的偏差。为了解决该问题，本文提出了一个解耦操作和拓扑共享的可微搜索框架 (DOTS)。DOTS 为拓扑搜索定义了独立的搜索空间和重要性表示。同时，DOTS 将操作和拓扑的搜索过程进行解耦。现有的可微搜索方法可以集成进 DOTS，通过独立的拓扑搜索得到进一步改进。通过解耦了操作和拓扑共享，DOTS 大幅提高了拓扑结构性能排序的准确性。相比于现有的可微搜索方法，DOTS 可以用较小的搜索时长搜索到更优的单元结构，并应用于下游视觉任务进行轻量化模型的自动设计。

其次，本文发现现有的搜索方法基于代理指标 (GPU 时延或 FLOPs) 来搜索轻量化模型。然而，当部署到不同硬件架构的端侧设备时，例如嵌入式设备或手机，代理指标会带来时延评估偏差。为了解决该问题，本文提出了一种基于设备感知的整体性搜索框架 (iNAS)。具体来说，iNAS 首先构建了一个用于显著性目标检测的整体性搜索空间，并将搜索空间组织为一个通过动态采样训练的嵌套权重共享超网。经过训练的超网中的采样架构可以实现与其独立训练相似的性能。然后，iNAS 在部署设备上构建设备专用的时延查找表，用以在搜索中评估采样模型在部署设备上的时延。iNAS 解决了搜索设备和部署设备差异导致的性能评估偏差，可为多种设备自动设计轻量化显著性目标检测模型。

关键词： 轻量化视觉模型设计；神经网络架构搜索；拓扑搜索；硬件感知搜索

Abstract

Lightweight vision model design aims to build high-performance, low-latency models. Due to different deployment devices, latency constraints, and vision tasks, the exploration of vision models faces a huge model design space. Manually specializing models for different scenarios requires large human labor. Inspired by neural architecture search (NAS), this thesis explores the automatic search for lightweight vision models. The difficulty of searching for lightweight vision models is the gap between their performance ranking and actual performance ranking, hindering NAS from finding the suitable architecture. This thesis dives into the root cause most responsible for the gap.

First, this thesis observes that existing differentiable search methods couple the operation and topology search, leading to poor model ranking correctness. To address this issue, this thesis proposes a framework to Decouple Operation and Topology Search (DOTS) in differentiable architecture search. DOTS defines an independent search space and importance representation for topology search. Besides, DOTS decouples the search process of operation and topology. Existing differentiable search methods can be integrated into DOTS and further improved by independent topology search. By decoupling operation and topology search, DOTS greatly improves performance ranking correctness. Compared to existing differentiable search methods, DOTS can search for better cell structures with smaller search costs and can be applied to downstream vision tasks for the automatic design of lightweight models.

Second, this thesis finds that existing search methods are based on proxy metrics (GPU latency or FLOPs) to search for lightweight models. However, proxy metrics can bring the speed evaluation gap when deployed to devices with different hardware architectures, such as embedded devices or mobile phones. To solve this problem, this thesis proposes an integrated search for device-aware salient object detection (iNAS). Specifically, iNAS first constructs an integrated search space for salient object detection and organizes the search space as a nested weight-sharing supernet trained by dynamic

sampling. The sampling architecture in the trained supernet can achieve a similar performance to its stand-alone training. Then, iNAS builds a specialized latency lookup table on the deployment device. The latency lookup table helps evaluate the latency of the sampled model on the deployment device during searching. iNAS solves the performance evaluation gap caused by differences in search and deployment devices and can automatically search lightweight salient object detection models on different devices.

Key Words: Lightweight vision model design; neural architecture search; topology search; hardware-aware search

目录

第一章 绪论

近年来，深度学习技术推动了计算机视觉的应用研究，深度神经网络模型在不同的视觉任务中都取得了较为理想的性能。但是这些模型的精度是以其较大的计算开销和内存消耗为代价。当需要将这些应用部署到真实场景时，视觉模型轻量化起到了至关重要的作用。由于不同的视觉任务，不同的部署设备和不同的时延限制，人工为不同需求设计专有轻量化模型需要消耗过多的人力资源，因此如何自动搜索适配的轻量化视觉模型成为了一个充满挑战性同时也具有很强实用意义的方向。

第一节 研究背景和意义

近年来，深度学习技术引起计算机视觉社区的广泛关注，神经网络模型已用于不同的视觉任务中。相比于手工设计的特征模式，例如尺度不变特征变换 (Scale-Invariant Feature Transform，SIFT) 方向梯度直方图 (Histogram of Oriented Gradient，HOG)、局部二值模式 (Local Binary Pattern，LBP) 等，深度神经网络通过数据驱动的方式自动学习任务适配的特征提取方式，将不同的视觉任务性能提升到了新的高度。虽然深度神经网络在不同视觉任务中达到了良好的性能，其模型复杂度限制了其部署应用到现实场景中。因此，轻量化视觉模型设计 (Light-Weight Vision Model Design) 是成为了深度学习时代中一个热门的研究方向。

早期轻量化的网络设计主要是手工设计连续的小卷积核来替换大卷积核来减少参数量[93]。后续的研究探索了不同的轻量化卷积结构，例如可分离卷积[90]，八度卷积[9]，组卷积[49] 等。这些卷积结构大幅减少了普通卷积的参数和计算量，可以应用于轻量化网络设计中。除了网络结构，另一系列工作探索了不同的模型后处理方式来降低计算量。其中，模型剪枝[58] 通过发现冗余通道并进行去除，可以保持性能并降低计算消耗。考虑到目前的神经网络参数一般是用高精度浮点数进行表示的，为了减少模型的存储消耗和计算消耗，模型量化[111] 通过控制存储精度，来大幅减少模型的推理开销。此外，为了得到紧凑的高性能小模型结构，知识蒸馏[38] 探索如何将大网络所学习到的知识逐步迁移至小模型中。

虽然，上述研究可以在不同角度上对网络进行轻量化，但是人工为不同视觉任务，不同时延限制和不同部署设备定制模型需要消耗大量的人力资源。此外，上述研究主要关注于模型的计算量，参数量等硬件无关的指标。这些指标虽然可以一定程度上反映模型的复杂度，但其和模型在硬件设备上的真实推理时延并没有完全一致的联系。因此，通过神经网络结构搜索 (NAS) 自动搜索轻量化模型引起了研究者们的注意力。NAS 通过设计搜索空间，性能评估策略，和搜索策略，可以自动地探索模型设计空间。为了设计轻量化高性能视觉模型，NAS 通过多目标搜索来探索性能和速度俱佳的网络结构。但是，网络搜索需要对采样结构进行准确的性能评估来确保搜索的准确性。早期的基于采样的网络搜索算法[115,140] 是将采样的网络经过一次完整的训练流程来得到其真实性能.。这种方式虽然可以准确的获得模型性能，但是需要消耗大量的计算资源。现有方法通过权重共享[33,69]，将不同网络结构合并成一个超网来降低性能评估需要消耗的时间。这种权重共享虽然减少了耗时，但是为模型性能的准确性带来了偏差。带来这种偏差的原因和解决方式是一个开放性问题，引起了学术界的广泛关注和讨论。

自动轻量化视觉模型设计具有较多的应用场景和重要的学术研究价值。首先，针对实时性要求较高的任务，如人脸识别，目标识别，显著性目标检测等，自动轻量化视觉模型设计可以快速找到适配的低时延模型。其次，针对存储空间和运行空间较小的端侧设备，自动轻量化视觉模型设计可以搜索到符合端侧设备需求大小的模型。最后，对自动轻量化模型的研究，可以反过来启发人工神经网络结构设计。因此，如何自动化地设计轻量化视觉模型无论在学术界还是工业界都有深远的意义和影响。

第二节　国内外研究现状

近年来，神经网络结构搜索 (NAS) 展示出为各种任务自动设计轻量化网络的潜力[31,63,67,89,127]。早期的基于采样的方法通过强化学习[140,141] 和进化算法[88,115] 来探索搜索空间。这类方法需要将数千个候选架构进行完整的训练，得到他们的真实性能，进一步来学习元控制器进行高性能结构的采样。这种设计模式需要花费上千 GPU/天的搜索时间消耗。为了降低搜索成本，近期的搜索方法利用权重共享[85] 的思想，将搜索空间构建为参数共享的超网。基于权重共享的方法不需要为每个采样模型进行完整的训练，显著降低了搜索时长，成为了

目前的主流搜索方案。在本节，本文分别在第 1.2.1节和第 1.2.2节中介绍两类基于权重共享的搜索方法，可微分网络搜索和 one-shot 搜索。

1.2.1 可微网络搜索

早期的基于采样的搜索算法[88,115,140,141] 需要通过大量的采样并完整训练模型，来训练元控制器进行进一步的采样。这种方式由于需要从头训练采样模型，消耗了大量的计算资源。为了减少搜索的资源消耗，可微网络搜索构建了参数共享超网，并将离散结构选择进行参数化，结合进超网的优化中。结构参数随着超网参数通过梯度优化进行更新。这种方式可以大幅减少搜索时长。

基于可微网络搜索，一系列工作[7,14,15,35,64,118,128] 尝试解决可微搜索中的不稳定性。Chen 等人[7] 通过在不同的可微搜索空间进行验证，发现其搜索存在不稳定性。随着搜索轮数迭代，可微网络搜索的结构性能会退化。为了解决可微网络搜索的不稳定性，大量的工作从不同的角度出发来解决该问题。其中，Chu 等人[15] 指出这是由于候选操作中不公平的排外竞争造成的，并提出通过将排外竞争变为协同竞争来提升搜索的稳定性。Liu 等人[68] 发现候选操作中跳接操作会主导搜索后的结构，并提出在跳接操作后加入随机丢弃模块[94] 来防止过拟合。Hong 等人[40] 发现了操作搜索的不稳定性是由于搜索中存在马太效应，无参数化的操作在搜索初期相对于随机初始化的操作存在优势，这种优势导致这些无参数化的操作在搜索初期有较大的初始权重，并得到更多的梯度更新。本文受上述工作启发，通过设计组搜索策略，克服操作搜索中马太效应带来的不稳定性。

除了稳定搜索过程，另一系列工作[24,68,120] 尝试进一步缩短搜索时长。Liu 等人[68] 通过多阶段搜索，逐步地剪除不重要的操作。靠后阶段的搜索基于缩小后的搜索空间，因此减少了搜索时长。Xu 等人[120] 通过将通道分为两个部分，一个部分由跳接运算得到，另一个部分由连边上定义的操作得到。基于这种操作空间构建方式，网络更新只需要处理部分的通道，因此大幅减少搜索时间。Dong 等人[24] 通过 Gumbel-Max 技巧，每次仅采样一个操作进行更新，将搜索时间减少到 4 GPU/小时。本文受多阶段搜索的启发，将可微网络搜索重定义为操作搜索和拓扑结构搜索。

最近的研究[26,30,92,117] 揭示了连接拓扑在神经网络中的重要性。随机连接网络[117] 发现随机图算法生成的网络可以获得有竞争力的结果。Shu 等人[92] 发现，与可微网络搜索中的操作相比，搜索单元中的拓扑结构对网络收敛的影响更大。这是因为网络拓扑主导了梯度传导的路径。DenseNAS[26] 提出了一个密集连接

的搜索空间，专注于搜索宏观的拓扑结构。本文的工作受上述工作启发，关注于可微网络搜索中的拓扑结构搜索。

1.2.2 one-shot 网络搜索

可微网络搜索将搜索过程和网络参数优化过程耦合在一起，会造成优化的不稳定性。同时，可微网络搜索通过一次搜索仅能获得一个最优架构。相较于可微网络搜索，one-shot 网络搜索[33] 采用两阶段的搜索范式。第一阶段，one-shot 搜索方法先将搜索空间构建为参数共享的超网并进行训练。接着，搜索空间内的采样模型可以通过超网获得其对应性能，并配合网络参数量进行搜索。

one-shot 方法相对于基于采样的区别在于其构建的搜索空间仅需要一次训练就可以获得超网中所有架构的性能，而基于采样的方法需要对每个架构进行完整的训练来获得其性能。one-shot 的方法以架构性能评估的准确性为代价节省大量的架构性能评估时间。最近的权重共享方法[45,52,57,59,126,136,137] 试图提高架构性能评估的正确性。Yu 等人[126] 指出，由于广泛使用的权重共享策略，最近的神经网络结构搜索算法没有显著强于随机搜索。PCNAS[59] 识别并解决权重共享方法中的后验衰减问题。Block-wise NAS[52] 通过将整体搜索空间分为不同的模块，并对模块进行单独搜索来提高架构评估的准确性。FairNAS[16] 通过提高超网训练阶段不同组件的采样公平性来提高整体架构评估的准确性。Yu 等人[125] 和 Cai 等人[4] 将搜索空间组织为嵌套的超网并辅以动态采样的训练策略，搜索空间中的结构通过一次训练就可以达到类似其单独训练的性能。这种方法搜索出的结构可以不需要进行重训和微调，就可以直接进行部署，规避了性能评估准确性的问题。本文受上述工作的启发，将嵌套的超网设计扩展到多尺度搜索单元以及多尺度分支设计，以适配需要多尺度信息的显著性检测任务。

基于 one-shot 的搜索方法解耦了搜索空间的训练和其搜索过程，适用于在搜索过程中引入多目标搜索策略。在这条研究线上，不同的工作引入了不同的多目标搜索方案。[34] 基于 shuffleNet[132] 的基本搜索单元构建了轻量化搜索空间，并通过搜索 FLOPs 和性能的帕累托前沿得到计算量小的模型。Cai 等人[3] 提出直接基于部署设备的时延查找表来搜索符合目标时延的架构。Cai 等人[4] 提出在部署设备上采样操作和其对应时延，训练时延预测模型，以期能够在搜索时直接为采样网络预测其推理时延。本文受上述工作启发，构建了搜索设备和部署设备一致的时延评估，并辅以无需重训的超网设计，解决由于不同设备带来的性能评估偏差。

第三节 本文研究内容

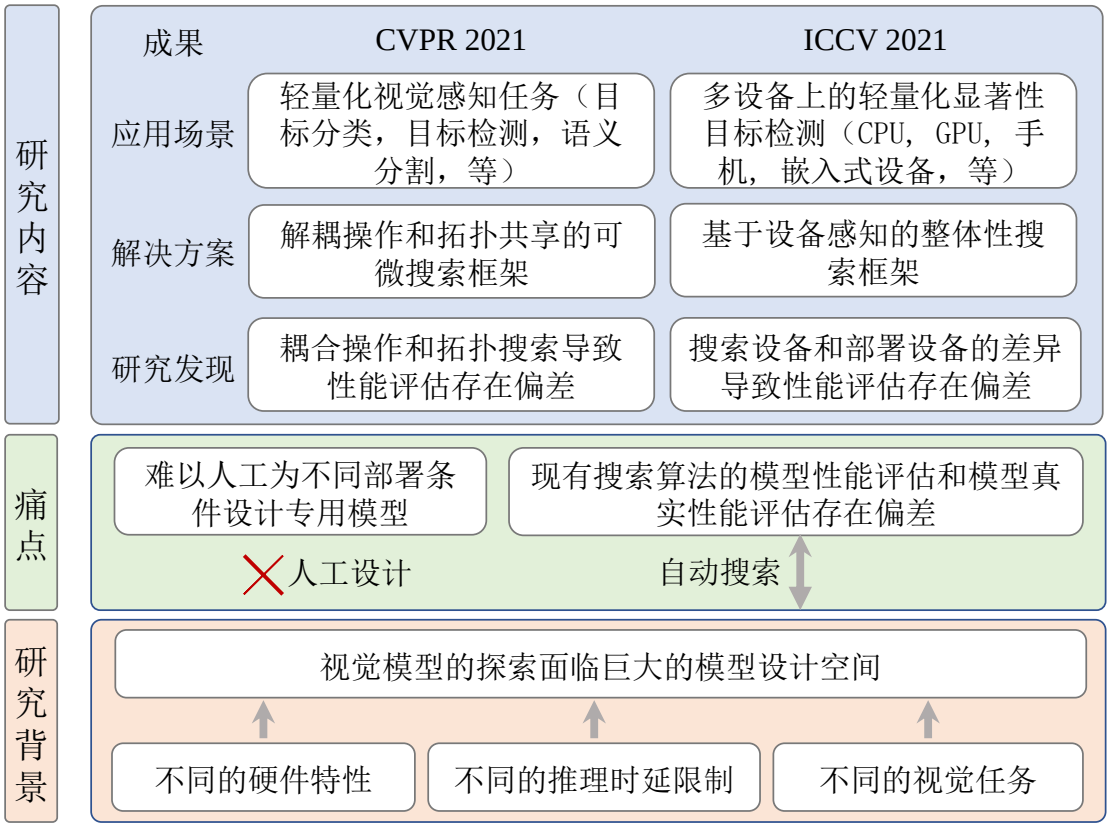


图 1.1 **研究背景和本文研究内容**。本文重点研究神经网络结构搜索中造成性能评估偏差的原因，并提出相应的解决方案。

如图 1.1所示，本文针对神经网络结构搜索在轻量化模型设计中的应用进行研究。神经网络结构搜索的主要问题在于现有搜索算法的模型性能评估和模型真实性能评估存在偏差。本文发现造成该问题的两个原因，并加以讨论和解决。

(1) 本文观察到可微网络搜索中由于耦合了操作搜索和拓扑搜索，不同拓扑结构的网络的性能评估排序和其真实性能排序不存在正相关性。为了解决这个问题，本文提出一个新颖的解耦拓扑和操作共享的可微搜索框架 (DOTS)。首先，DOTS 解耦了操作和拓扑的重要性表示。然后，DOTS 构建了独立的拓扑搜索过程，将原搜索空间分解为操作搜索和拓扑搜索。通过解耦了拓扑的表示和拓扑搜索的过程，DOTS 的拓扑搜索可以得到更为准确的性能评估。最后，考虑到一些操作可能会影响拓扑结构，为了保留潜在可能的拓扑结构，DOTS 采用了组操作搜索策略进行操作搜索。DOTS 将操作划分为无参数和有参数的两个操作组，并单独进行搜索。该方法保存了潜在的拓扑可能性的同时并克服了搜索中的不

稳定性。DOTS 可为下游视觉任务 (例如目标分类，目标检测，语义分割等) 自动搜索轻量化模型。

(2) 本文观察到在视觉显著性检测中，由于手工设计的模型和之前基于搜索的方法都是基于 GPU 进行设计，并将结构迁移到其他设备上。或者其基于硬件无关的计算量 FLOPs 进行搜索，而 FLOPs 与真实设备时延不存在很好的相关性。这个观察指出了搜索设备和部署设备间的不同导致其搜索结果的性能评估存在偏差。为了解决该问题，本文提出了基于设备感知的整体性搜索框架 (iNAS)。首先，iNAS 构建了适配显著性检测的整体性搜索空间。该空间涵盖了手工的显著性检测模型的设计模式。其次，为了增加搜索空间对多尺度结构的搜索能力，iNAS 构建了多尺度基本搜索单元。通过将训练时的多分支结构重参数化为部署时的单分支结构，iNAS 可以搜索出不同的尺度组合并不显著增加推理时延。最后，iNAS 将搜索空间构建为耦合参数共享的超网，该超网仅需要一次训练，搜索空间中不同的子模型都可以得到和其单独训练类似的性能。iNAS 在不同的设备上构建了时延查找表来得到采样模型在部署设备上的真实时延。基于构建的超网和时延查找表，iNAS 可以搜索在不同设备上的低时延高性能模型，并无需任何重训和微调就可以直接部署。iNAS 可应用于在不同设备上 (CPU，GPU，手机，嵌入式设别等) 快速地构建专用的轻量化视觉显著性检测模型。

第四节　论文章节安排

本文的主要工作是探索了自动搜索轻量化视觉模型中存在的性能评估偏差问题，一共分为四章，具体安排如下：

第一章为绪论，首先介绍了神经网络结构搜索在轻量化视觉模型设计中的研究背景和意义。然后就参数共享网络搜索中的可微网络搜索和 one-shot 网络搜索阐述了与本文工作相关的国内外研究现状。最后介绍了本文研究内容和后序论文章节安排。

第二章和第三章探讨了两个造成性能评估偏差的原因，并介绍了基于解耦操作和拓扑共享的可微搜索框架和基于设备感知的整体性搜索框架来解决评估偏差。

第四章是总结和展望，总结了本文工作内容并对未来工作进行了展望。

第二章　基于解耦操作和拓扑共享的可微网络搜索

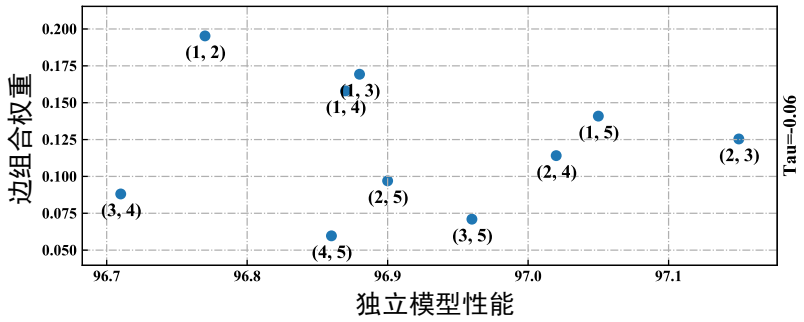
第一节　研究动机以及贡献

神经网络结构搜索 (NAS) 因其在大型搜索空间中能自动找到最佳架构的潜力而引起了广泛关注。早期的基于采样的方法[70,88,141] 通过强化学习和基于进化算法探索搜索空间。基于采样的方法需要完整的训练过程来获得采样模型性能，以训练元控制器进行采样。虽然这种方式可以得到准确的性能评估，但是需要花费成百上千个 GPU/天的搜索时长。为了降低搜索成本，one-shot 方法[16,23,33,84] 采用权重共享策略，对超网进行一次训练，并直接从超网中获得子网的性能。最近的方法[3,8,118,120] 基于可微网络搜索 (DARTS)[69] 也采用权重共享策略，通过统一超网训练和子网搜索，进一步降低搜索成本。

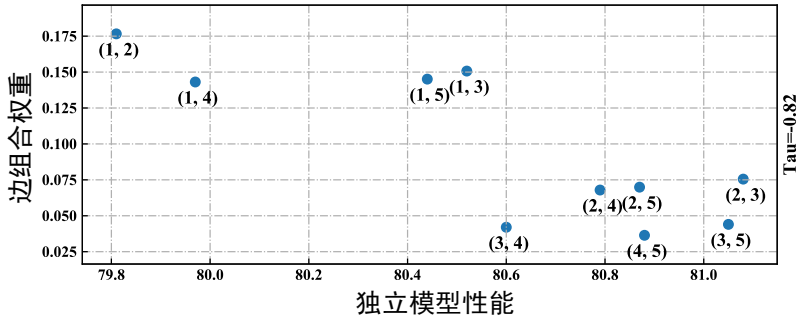
DARTS 将操作的离散选择进行参数化，为不同的操作选择前加入权重系数，并随着超网训练通过梯度下降进行更新。训练后，操作权重用于对操作和拓扑的重要性进行排序，并导出排序中最优的架构。DARTS 对操作的重要性排序就是操作权重本身，但对拓扑结构的重要性是将该边上的最大操作权重表示为该边的重要性。在导出最优结构的时候，DARTS 为每个中间节点保留两条最重要的边，并剪枝剩下的边，以导出搜索到网络的拓扑结构。接着，其保留操作权重最大的操作，去除其他的操作，来获得搜索到的网络的操作。

DARTS 的搜索模式引出一个问题：用操作权重是否可以准确地表示边的重要性并准确地对独立模型的性能进行了排序。如图 2.1所示，本文构建了一个简单的实验来验证操作权重是否可以用来表示边的重要性排序。依据 DARTS 采用固定边数的策略，即为每个搜索后的单元保留两条边，本文用边组合权重 (两条留存边权重的加和) 来表示这种拓扑结构的重要性。不同拓扑结构的重要性和其经过独立训练后真实性能的相关性如图 2.1所示。从实验中，本文没有发现明显的排序相关性，这意味着 DARTS 相较于随机选择边没有明显的优势 (请参阅第 2.2.2节中的更多详细信息)。此外，DARTS 的固定边数的策略限制了其找到更灵活的网络结构的潜力。

为了解决上述问题，本文提出了基于解耦操作和拓扑共享的可微网络搜索



(a) CIFAR10



(b) CIFAR100

图 2.1 **DARTS 中的拓扑表示与独立模型性能相关性**。*Kendall Tau* 指标[47] 用于衡量拓扑重要性表示和模型真实性能排序相关性。

框架 (DOTS)。DOTS 中解耦的含义有两方面。一方面，本文将拓扑的重要性表示与操作权重解耦。详细地，本文引入了一个包含边的成对组合的拓扑搜索空间。离散的拓扑搜索空间通过引入拓扑权重进行连续性松弛，松弛的拓扑权重对候选边的组合分布的重要性进行衡量。本文提出的拓扑搜索空间直接反映了搜索目标，并且可以轻松扩展以支持任意数量的边的自动化选择。另一方面，本文将操作搜索和拓扑搜索的搜索过程进行解耦。整个搜索过程分为操作搜索阶段和拓扑搜索阶段，分别更新操作权重和提出的边组合权重。通过将两个搜索过程解耦，现有的基于梯度的可微搜索可以直接用作 DOTS 的操作搜索，并通过拓扑搜索得到进一步改进。此外，拓扑搜索基于缩小的操作搜索空间，其搜索过程需要更少的资源消耗，更加高效和准确。考虑到某些操作 (例如, 跳接) 会影响拓扑结构，DOTS 在操作搜索中采用分组策略来保留这些与拓扑相关的操作，以便更好地进行拓扑搜索。

综上，本文的贡献总结如下：

- 本文发现耦合的操作和拓扑搜索会带来拓扑结构评估的偏差。这种偏差导

致 DARTS 中的拓扑选择不能显著由于随机拓扑选择。

- 本文提出将操作搜索和拓扑搜索解耦，包括了拓扑重要性表示解耦和搜索过程解耦。这种解耦可以对具有不同拓扑结构的独立模型进行正确的重要性排序。
- DOTS 仅需 0.26 GPU/天和 1.3 GPU/天在 CIFAR 和 ImageNet 数据集上搜索到当前性能最优的结构，并可以迁移到下游视觉任务中。

第二节 可微网络搜索中的耦合问题分析

2.2.1 可微网络搜索

本文首先回顾基准算法 DARTS[69]。DARTS 旨在搜索神经网络的基本构成单元，并重复叠加搜索到的基本单元来构建整体的网络结构。一个基本搜索单元可以表示为一个有向循环图。该单元有 N 个节点 $\{x_i\}_{i=1}^{N}$，包括两个输入节点、一个输出节点和若干个中间节点。每个节点表示一个由边上的操作进行编码得到的特征图。第 j 个中间节点 x_j 通过边 (i,j) 连接到它的所有前驱节点 x_i。每条边 (i,j) 包含由操作权重 $\alpha^{(i,j)}$ 加权的候选操作，可以定义为：

$$\bar{o}^{(i,j)}(x) = \sum_{o \in O} \alpha_o^{(i,j)} o^{(i,j)}(x_i), \tag{2.1}$$

其中 $o(x) \in \mathcal{O}$ 和 $\mathcal{O}$ 是包含八个候选操作的操作搜索空间，包括 *Zero*, *Skip-Connection*, *Avg-Pooling*, *Max-Pooling*, *Sep3x3Conv*, *Sep5x5Conv*, *Dil3x3Conv*, 和 *Dil5x5Conv*。操作的重要性权重经过 softmax 标准化：

$$\alpha_o^{(i,j)} = \frac{\exp(\alpha'^{(i,j)}_o)}{\sum_{o' \in \mathcal{O}} \exp(\alpha'^{(i,j)}_{o'})}, \tag{2.2}$$

其中 α' 是未标准化的操作权重。操作权重随着超网训练一起更新，逐渐聚焦于最优架构。本文为边 (i,j) 定义了混合操作 $\bar{o}^{(i,j)}$，其表示通过操作权重加权的所有操作输出结果的平均特征。定义了混合操作 $\bar{o}^{(i,j)}$ 之后，中间节点的特征 x_j 就可以从它的所有前驱节点 x_i 计算出来：

$$x_j = \sum_{i<j} \bar{o}^{(i,j)}(x_i). \tag{2.3}$$

本文用 $\mathcal{L}_{cls}^{train}$ 和 $\mathcal{L}_{cls}^{val}$ 表示训练集和验证集的交叉熵损失。然后，本文定义操作权重 α 和网络权重 w 的双层优化公式化为

$$\begin{aligned} &\min_{\alpha} \ \ \mathcal{L}_{cls}^{val}(w^*(\alpha),\alpha), \\ &\text{s.t. } \ w^*(\alpha) = \arg\min_{w}(\mathcal{L}_{cls}^{train}(w,\alpha)). \end{aligned} \tag{2.4}$$

经过训练，最终搜索到的架构从操作权重 α 经过两次剪枝得到：

1. 在每条边上保留权重最大的操作，并剪除其他操作, 即 $o^{(i,j)} = \arg\max_{o\in\mathcal{O},o\neq Zero}\alpha_o^{(i,j)}$.
2. 为每个中间节点保留两条边重要性最大的入边，并剪除其他边。边重要性定义为每条边上的最大操作权重 (i,j), 即 $\max_{o\in\mathcal{O},o\neq Zero}\alpha_o^{(i,j)}$.

2.2.2 可微分网络搜索中的耦合问题

之前基于 DARTS 的工作[7,8,14,24,40,56,64,80,118,120] 通过边上的最大操作权重 (不包括 *Zero* 操作) 来表示边的重要性。本文进行了排序相关性分析，以确定操作权重指示的边重要性是否可以准确地对不同拓扑结构的独立模型进行性能排序 (由 *Kendall Tau*[47] 衡量)。本文遵循 DARTS 固定边数的策略，将两条边的重要性相加以获得其边组合重要性。本实验中有五条边，从而产生十种不同的边组合。独立模型使用与第 2.4.1节中相同的设置进行训练，除了将训练轮次减少到 300。本文画出了边组合的重要性以及其真实性能的散点图，如 图 2.1所示。DARTS 的拓扑表示策略在 CIFAR10 和 CIFAR100 上的 *Kendall Tau* 分别为 $Tau = 0.06$ 和 $Tau = -0.82$。该结果表明 DARTS 的拓扑重要性表示与独立模型精度没有明显的排序相关性。

其中，DARTS 的拓扑搜索策略在 CIFAR100 数据集上呈现了负相关性。这种负相关是因为 DARTS 在 CIFAR100 上搜索不稳定，越浅的边具有越大的操作权重。因此，较浅的两条边的组合在所有的中间节点都具有较大的操作权重。然而，该数据集需要更深的网络结构。因此，DARTS 的拓扑重要性表示与独立模型性能呈负相关。

上述分析意味着 DARTS 搜索到的结构是次优的，因为它无法收敛到最佳的拓扑结构，这与之前 DARTS 在某些情况下无法超越随机搜索的发现一致[128]。直观上来说，较大的操作权重只能表明操作在特定边上的重要性，并不意味着该边应该保留在搜索到的拓扑中。

第三节 基于解耦操作和拓扑共享的可微搜索框架

以上分析指出了耦合操作和拓扑搜索的局限性。为了解决该问题，本文提出了基于解耦操作和拓扑共享的可微搜索框架 (DOTS) 来解决这个问题。如图 2.2所示，本文将搜索解耦为操作搜索和拓扑搜索两个部分。在操作搜索阶段，本文在每条边上搜索最优的操作。在拓扑搜索阶段，本文搜索候选边的最佳组合。在第 2.3.1节中，本文将介绍如何构建拓扑搜索空间并支持自动搜索不同数量的边。在第 2.3.2节中，本文描述了如何将现有的可微搜索方法集成到 DOTS 的操作搜索中，并提出了一个组操作搜索方案来保留与拓扑相关的操作，以便更好地进行拓扑搜索。

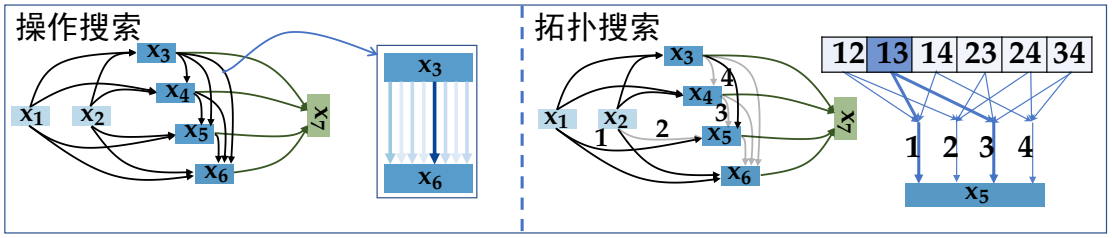


图 2.2 **基于解耦操作和拓扑共享的可微搜索框架 (DOTS)**。DOTS 首先进行操作搜索。接着，DOTS 基于操作搜索导出的子空间进行拓扑搜索。

2.3.1 拓扑搜索

2.3.1.1 固定边数的策略

第 2.2.2节讨论了耦合操作和拓扑搜索的局限性。因此，DOTS 将边的重要性表示与操作权重分离。为了实现这一点，除了操作搜索空间之外，本文定义了一个拓扑搜索空间。形式上，第 j 个中间节点 x_j 通过边 (i,j) 连接到它的所有前驱节点 x_i。本文遵循 DARTS 的固定边数策略，限制搜索到的结构中，每个中间节点保留两条边。本文将中间节点 x_j 的拓扑搜索空间 $\mathcal{E}_{x_j}$ 表示为其传入边的所有成对组合：

$$\mathcal{E}_{x_j} = \{\langle (i_1,j),(i_2,j)\rangle \,|0 < i_1 < i_2 < j\}. \tag{2.5}$$

假设节点 x_j 总共有 n 条入边，那么拓扑搜索空间 $\mathcal{E}_{x_j}$ 包含 $C_n^2 = \frac{n!}{2!(n-2)!}$ 种候选组合。对于每个节点 x_j，本文将其离散的拓扑搜索空间通过引入边组合权重松

弛为连续的表示，可以定义为

$$\beta_{x_j}^{c} = \frac{\exp(\beta'^{x_j}_{c} / T_\beta)}{\sum_{c' \in \mathcal{E}_{x_j}} \exp(\beta'^{x_j}_{c'} / T_\beta)}, \tag{2.6}$$

其中 $\beta_{x_j}^{c}$ 表示选择边组合 c 的归一化概率。虽然拓扑搜索空间是在边组合上定义的，但实际上本文不需要获取每个边组合的特征。为了减少内存消耗，本文从包含这条边的所有组合中聚合边 e_j^i 的权重，可以表示为

$$\gamma^{(i,j)} = \sum_{c \in \mathcal{E}_{x_j}, (i,j) \in c} \frac{1}{N(c)} \beta_{x_j}^{c}, \tag{2.7}$$

其中 $\gamma^{(i,j)}$ 是每条边的权重，$N(c)$ 是边组合 c 中的边数。本文将 x_j 的所有入边按边重要性权重 γ 相加得到它的特征：

$$x_j = \sum_{i<j} \gamma^{(i,j)} \cdot \bar{o}^{(i,j)}(x_i), \tag{2.8}$$

其中 $\bar{o}^{(i,j)}$ 表示边 (i,j) 上的混合操作。由于本文将操作和拓扑搜索过程解耦，$\bar{o}^{(i,j)}$ 仅包含了操作搜索保留的候选操作，这将在第 2.3.2节中讨论。

正如在 ASAP[80] 和 SNAS[118] 中讨论的那样，超网和子网之间的优化差距会导致性能下降。这两项工作都利用架构参数退火来弥合搜索过程中的优化差距。本文将退火思想推广到拓扑搜索。在式 2.6中，T_β 相当于退火温度。本文采用指数进行退火：

$$T(t) = T_0 \theta^t, \tag{2.9}$$

它从初始温度 T_0 开始并随着训练步骤 t 的增加而衰减。

DARTS 使用双层优化来避免过拟合[69]。然而，先前的工作[33,56] 表明单层优化会更加稳定和准确。在本文的拓扑搜索阶段，每条边上的操作大大减少，消除了过拟合的风险。因此，本文使用单层优化来更新网络权重 w 和拓扑权重 β，表示为

$$\begin{aligned} w_t &= w_{t-1} - \eta_t \partial_w L_{train}(w_{t-1}, \beta_{t-1}), \\ \beta_t &= \beta_{t-1} - \delta_t \partial_w L_{train}(w_{t-1}, \beta_{t-1}), \end{aligned} \tag{2.10}$$

其中 η_t 和 δ_t 分别是网络权重和拓扑权重的学习率。

2.3.1.2 自动化边数搜索的策略

在第 2.3.1节中，本文按照 DARTS 固定边数的策略构建拓扑搜索空间，该空间限制搜索单元中的每个中间节点连接两条边。这种固定策略无法自动学习边的数量。因此，本文扩展拓扑搜索空间以支持任意数量的边。具体来说，本文采用二进制代码来描述这样的搜索空间。对于具有 n 条传入边的中间节点 x_j，第 m 条边组合的二进制代码可以表示为：

$$c_m = \{e_1, e_2, \ldots, e_n\}, \tag{2.11}$$

其中 $e_i \in \{0,1\}$ 表示边 i 是否存在于边组合中。节点 x_j 的拓扑搜索空间可以定义为：

$$\mathcal{E}_{x_j} = \{c^1, c^2, \ldots, c^M\}, \tag{2.12}$$

其中 M 是有效边组合的数量。本文可以通过下式计算 M：

$$M = \sum_{k=1}^{n} C_n^k = 2^n - 1. \tag{2.13}$$

本文排除了所有边都不在边组合中的极端情况。利用上述新的搜索空间搜索任意数量的边将很容易实现，因为本文只用将 式 2.12的搜索空间替换为式 2.5 的搜索空间，并保持架构松弛和优化相同。

2.3.2 操作搜索

本节介绍 DOTS 的操作搜索。操作搜索旨在得到每条边上的最佳操作。本文在第 2.3.2节中介绍了如何将现有的基于梯度的神经网络搜索方法集成进 DOTS 操作搜索。现有方法只保留每条边上的最佳操作，但是剪除掉的某些操作 (例如，跳接) 可能会影响最终的拓扑结构。这是因为网络的拓扑结构主要对梯度的传导产生影响，而无参数化的操作 (例如，跳接)，相当于一条节点间传导梯度的连边。为了防止与拓扑相关的操作在操作搜索中被修剪，本文在第 2.3.2节中提出了一个组操作搜索方案。

2.3.2.1 结合基于梯度的神经网络搜索方法

现有的可微网络搜索方法可以很容易地集成到 DOTS 的操作搜索中。DARTS 为每条边 (i,j) 上的候选操作引入了一组权重 $\alpha = \{\alpha^{(i,j)}\}$。本文按照这些方法自己的搜索策略得到训练好的操作权重 α。然后，在每条边 (i,j) 上保留权重最大的操作：

$$o^{(i,j)} = \arg\max_{o \in \mathcal{O}} \alpha_o^{(i,j)}. \tag{2.14}$$

最后一步是将式 2.8中的混合操作 $\bar{o}^{(i,j)}$ 替换为 $o^{(i,j)}$。总的来说，现有的可微网络搜索方法可以集成到 DOTS 的操作搜索中，并通过拓扑搜索得到进一步的改进。

2.3.2.2 基于组策略的操作搜索

通常，操作可以分为拓扑相关 (例如，跳接) 和拓扑不可知 (例如，可分离卷积)。在第 2.3.2节中，本文将现有的基于梯度的方法集成到 DOTS 的操作搜索中，每条边上仅保留了最佳操作。这种策略由于在拓扑搜索之前删除了一些与拓扑相关的操作，减少了潜在可能的拓扑选项。为此，本文采用基于组策略[40,56] 的操作搜索。基于组策略的操作搜索可以确保为拓扑搜索保留与拓扑相关的操作。

形式上，在基于组策略的操作搜索中，操作搜索空间 $\mathcal{O}$ 被分为几个子空间 $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots \mathcal{O}_p\}$, 其中 p 是组数。每个操作子空间进行独立的连续松弛。搜索后，从每组中选出权重最大的操作，在每条边 (i,j) 上构造一个新的操作搜索空间 $\mathcal{O}_n$，可以表示为

$$o_p^{(i,j)} = \arg\max_{o_p \in \mathcal{O}_p} \alpha_{o_p}^{(i,j)}, \tag{2.15}$$

$$\mathcal{O}_n = \{o_1^{(i,j)}, o_2^{(i,j)}, \ldots, o_p^{(i,j)}\}. \tag{2.16}$$

本文在实验中评估不同的分组策略并讨论它们。由于 O_n 包含不止一个操作，本文需要在拓扑搜索阶段进行进一步的搜索。同时更新操作和拓扑是不准确的，因为这样增加了权重共享的子模型[16,59]。为了解决这个问题，本文在拓扑搜索中使用比 T_β 更低的温度 $T_{\alpha_{o_n}}$ 对操作权重进行退火。如图 2.3所示，操作的选择在前几个训练轮数中就逐渐固定下来。一旦操作固定下来，进一步的拓扑搜索与 第 2.3.2节中的类似。

第四节 实验对比以及结果分析

2.4.1 CIFAR10/100 上的实验

2.4.1.1 搜索设置

本文基于 Pytorch[83]、Mindspore[79] 和 Jittor[44] 深度学西框架实现了 DOTS。CIFAR10/100 上的整个搜索过程需要 70 个搜索轮次，前 30 个轮次用于操作搜索，后 40 个轮次用于拓扑搜索。整体网络结构由 8 个操作搜索单元和 20 个拓扑搜索单元组成。本文采用 SGD 优化器优化网络权重 w，初始学习率为 0.025(本

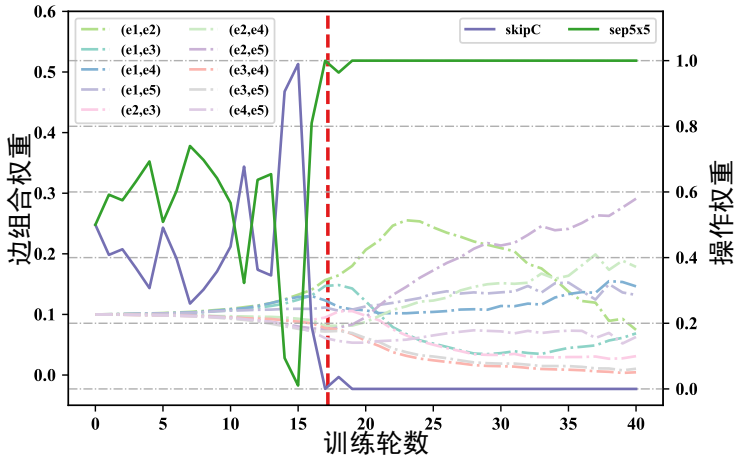


图 2.3 **基于多个留存操作的拓扑搜索**。操作权重比拓扑权重的退火温度低，即 $T_{\alpha_{o_n}} = \frac{1}{1000} T_\beta$。因此，多个留存操作在前几个训练轮次 (由红色虚线表示) 就将固定下来。

文采用余弦学习率衰减，在 70 轮次内衰减到 0.001)。SGD 优化器中的权重衰减为 3e-4，动量为 0.9。对于更新操作权重 α 和边组合权重 β，本文使用 Adam 优化器，学习率为 1e-4，权重衰减为 1e-3。为了在拓扑搜索中对边组合权重进行退火，本文设置初始温度 $T_0 = 10$ 和衰减率 $\theta = 0.72$。搜索过程在一个 NVIDIA Tesla V100 GPU 上花费 6.3 小时 (0.26 GPU/天)。

2.4.1.2 评估设置

评估网络由 20 个基本单元 (18 个普通单元和 2 个递减单元) 组成，初始通道数为 36。其中递减单元会带来特征大小的缩减。本文以批大小 96 从头开始训练评估网络 600 轮。该网络通过 SGD 优化器进行优化，初始学习率为 0.025(余弦退火为 0)，动量为 0.9，权重衰减为 3e-4，梯度裁剪为 5。本文使用 Cutout 增强[20] 和概率为 0.2 的 DropPath[51] 用于防止过拟合。

2.4.1.3 主要结果

在 CIFAR10/100 上的评估结果显示在表 2.1和表 2.2中。DOTS 在 CIFAR10 和 CIFAR100 上仅花费 0.26 GPU/天的搜索消耗，小于仅包含操作搜索的基线方法 DARTS(0.4 GPU/天)。这是因为本文将网络搜索分解为两个阶段。由于操作和拓扑搜索的解耦，两个阶段都可以快速收敛。其中拓扑搜索是构建在操作搜索的结果构建的子网上。因此拓扑搜索无需过多的时间损耗。DOTS 在 CIFAR10 和 CIFAR100 上分别达到了 97.51% 和 83.52% 的准确度，相对于 DARTS 分别提

表 2.1 在 CIFAR10 数据集上与最先进模型的对比实验。

架构	Top-1 准确率 (%)	参数量 (M)	搜索时长 (GPU/天)
DenseNet-BC[46]	96.54	25.6	N/A
NASNet-A[141]	97.35	3.3	1800
ENAS[84]	97.11	4.6	0.5
PNAS[68]	96.59± 0.09	3.2	225
DARTS[69]	97.00	3.4	0.4
SNAS[118]	97.15	2.8	1.5
GDAS[24]	97.07	2.5	0.2
P-DARTS[8]	97.50	3.4	0.3
FairDARTS[15]	97.46	2.8	0.4
PC-DARTS[120]	97.43 ± 0.07	3.6	0.1
DropNAS[40]	97.42 ± 0.14	4.1	0.6
MergeNAS[109]	97.27 ± 0.02	2.9	0.2
ASAP[80]	97.32 ± 0.11	2.5	0.2
SDARTS-ADV[7]	97.39 ± 0.02	3.3	1.3
DARTS-[14]	97.41 ± 0.08	3.5	0.4
DOTS (avg)*	97.51±0.06	3.5	0.26

升了 0.51% 和 1.06%。搜索结果展示在图 2.4(a,b)。

2.4.2 ImageNet 上的评估

2.4.2.1 搜索设置。

本文按照 PC-DARTS[120] 的实验设置，从 ImageNet[18] 中随机抽取 10% 和 2.5% 图像来构建训练和验证集。该网络由 14 个基本单元组成。整个搜索过程需要 70 个轮次，其中操作搜索需要 30 个轮次，拓扑搜索需要 40 个轮次。SGD 优化器用于优化网络权重 w 初始学习率为 0.25(余弦衰减到 1e-2 在 70 时期)，权重衰减为 3e-4，动量为 0.9。SGD 的批量大小设置为 512。对于更新操作权重 α 和边组合权重 β，本文使用 Adam 优化器，两个阶段的学习率为 3e-3，权重衰减为 1e-3。本文设置初始温度 $T_0 = 10$ 和衰减率 $\theta = 0.72$ 用于在拓扑搜索中对边缘组合权重进行退火。整个搜索过程在四个 NVIDIA Quadro RTX 8000 GPU 上花费 7.8 小时 (1.3 GPU/天)。

2.4.2.2 评估设置

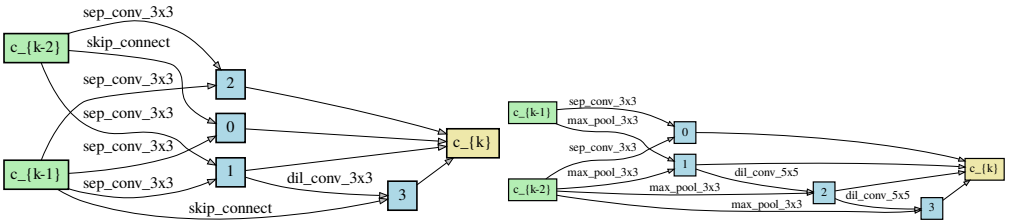
表 2.2 在 CIFAR100 数据集上与最先进模型的对比实验。

架构	Top-1 准确率 (%)	参数量 (M)	搜索时长 (GPU/天)
DenseNet-BC[46]	82.82†	25.6	N/A
NASNet-A[141]	83.18	3.3	1800
ENAS[84]	80.57†	4.6	0.5
PNAS[68]	80.47†	3.2	225
DARTS[69]	82.46†	3.4	0.4
SNAS[118]	82.45	2.8	1.5
GDAS[24]	81.62†	3.4	0.2
P-DARTS[8]	82.51†	3.6	0.3
FairDARTS[15]	82.39	2.8	0.4
PC-DARTS[120]	83.10	3.6	0.1
DropNAS[40]	83.13	4.0	0.6
MergeNAS[109]	82.42	2.9	0.2
ASAP[80]	82.69	2.5	0.2
SDARTS-ADV[7]	83.27	3.3	1.3
DARTS-[14]	82.84†	3.4	0.4
DOTS (avg)*	83.52±0.13	4.1	0.26

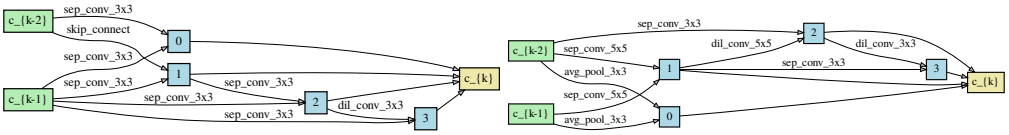
模型评估遵循以下设定, 输入图像的大小设置为 224×224，乘加运算次数限制在 600M 以内。网络由 14 个单元组成 (12 个普通单元和 2 个递减单元)，初始通道数为 46。训练的批大小为 1024，训练轮次为 250。本文使用初始学习率为 0.5(在前 5 个时期预热，余弦退火为 0)、0.9 的动量和 $3e$-5 的权重衰减的 SGD 优化器。其他优化操作遵循 P-DARTS[8] 和 PC-DARTS[120]，包括标签平滑和辅助损失。

2.4.2.3 主要结果

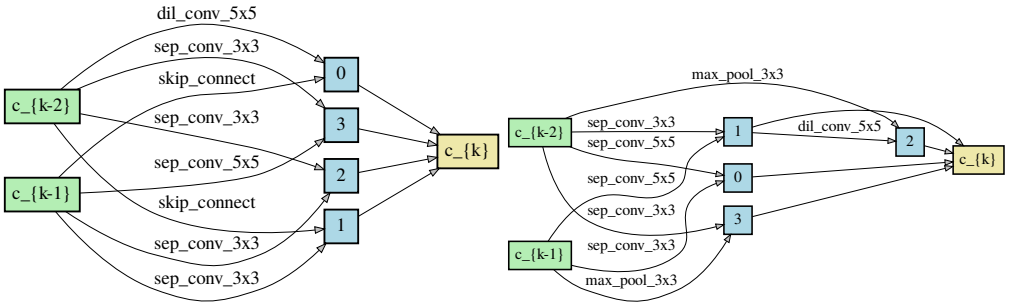
评估结果展示在表 2.3中。大多数基于梯度的方法在代理数据集上进行搜索，例如 CIFAR10，并将搜索到的单元转移到 ImageNet。这是由于这些方法在 ImageNet 上的搜索成本过高。而 DOTS 可以直接在 ImageNet 上进行搜索，并且需要极少的搜索成本 (1.3 GPU/天)。相对于 DARTS(2nd order)，DOTS 在 ImageNet 数据集上 DOTS 将 top-1 准确率提高了 2.7%。搜索结果展示在图 2.4。



(a) CIFAR10 上搜索到的普通单元 (左) 和递减单元 (右)



(b) CIFAR100 上搜索到的普通单元 (左) 和递减单元 (右)



(c) ImageNet 上搜索到的普通单元 (左) 和递减单元 (右)

图 2.4 DOTS 搜索的结果可视化

2.4.3 消融研究

2.4.3.1 拓扑搜索有效性验证

本文将五种可微搜索方法 (即 DARTS，DARTS(2nd)，GDAS，SNAS 和 PC-DARTS) 集成进 DOTS 框架，并验证拓扑搜索的有效性。基线是通过原 DARTS 策略导出搜索单元的拓扑结构。对比结果总结在表 2.4中。从结果中可以观察到添加拓扑搜索后，基线性能得到了明显改进，该结果说明了所提出的拓扑搜索的有效性。DOTS 提出新的拓扑重要性表示，即边组合权重。为了进一步验证所提出的边组合权重的有效性，本文按照之前的实验设定，得到独立模型的真实性能，以及对应的边组合权重，绘制散点图如图 2.5所示。DOTS 的边组合权重和真实模型性能在 CIFAR10 和 CIFAR100 上的 *Kendall Tau* 分别为 $tau = 0.73$ 和 $tau = 0.71$。这表明 DOTS 引入的边组合权重与真实模型性能存在良好的排序相关性。

表 2.3 **在 ImageNet 数据集上与最先进模型的对比实验**。† 表示该方法将 CIFAR10 上搜索得到的结果迁移到 ImageNet 数据集。

架构	准确率 (%)		参数量	乘加计算量	搜索时长
	top-1	top-5	(M)	(M)	(GPU/天)
Inception-v1[97]	69.8	89.9	6.6	1448	N/A
MobileNet[43]	70.6	89.5	4.2	569	N/A
ShuffleNet 2× (v1)[132]	73.6	89.8	5.4	524	N/A
ShuffleNet 2× (v2)[78]	74.9	92.4	7.4	591	N/A
NASNet-A[141]	74.0	91.6	5.3	564	1800
AmoebaNet-C[88]	75.7	92.4	6.4	570	3150
PNAS[68]	74.2	91.9	5.1	588	225
MnasNet-92[98]	74.8	92.0	4.4	388	1667
DARTS (2nd order)†[69]	73.3	91.3	4.7	574	4.0
SNAS[118]	72.7	90.8	4.3	522	1.5
P-DARTS†[8]	75.6	92.6	4.9	557	0.3
GDAS†[24]	74.0	91.5	5.3	581	0.2
FairDARTS†[15]	75.1	92.5	4.8	541	0.4
PC-DARTS†[120]	74.9	92.2	5.3	586	0.1
SDARTS-ADV†[7]	74.8	92.2	5.4	594	1.3
DropNAS†[40]	75.5	92.6	5.2	572	0.6
ASAP†[80]†	73.7	91.5	3.8	427	0.2
ProxylessNAS[3]	75.1	92.5	7.1	465	8.3
FairDARTS[15]	75.6	92.6	4.3	440	3
PC-DARTS[120]	75.8	92.7	5.3	597	3.8
DOTS	76.0	92.8	5.3	596	1.3

图 2.6中展示了一个通过 DOTS 改进拓扑搜索的示例。首先，本文通过 DARTS 中的策略进行操作搜索，得到的操作搜索结果如图 2.6(a) 所示。接着，本文通过 DARTS 中的拓扑推导策略 (即根据操作权重直接导出其拓扑结构) 得到的结构如图 2.6(c) 所示。这个结构以跳接为主，缺少可学习的卷积操作，在 CIFAR100 上只能达到 80.74% 的准确率。与之相比，基于相同的操作搜索结果，使用 DOTS 的拓扑搜索得到的结构如图 2.6(b) 所示。该结构在 CIFAR100 上获得了 2.33% 的改进 (83.07% *vs.* 80.74%)。之前的研究表明，DARTS 搜索具有马太效应，即无参数的操作在搜索开始时优于随机初始化的卷积操作并累积较大的权重。如果依照 DARTS 的搜索方式，这些权重会直接反应到搜索的拓扑结构上，即导出这些较大权重操作所在的边。而 DOTS 提出的拓扑搜索有助于通过

表 2.4 通过拓扑搜索改进现有的基于梯度的搜索方法。

架构	拓扑搜索	CIFAR10	CIFAR100
DARTS[69]	✘	97.02±0.12	80.74
	✔	97.40±0.09	83.07
DARTS (2nd)[69]	✘	97.01±0.15	81.37
	✔	97.12±0.11	83.28
GDAS[24]	✘	96.84±0.06	82.75
	✔	97.06±0.08	83.01
SNAS[118]	✘	97.05±0.10	81.92
	✔	97.26±0.12	83.25
PC-DARTS[120]	✘	97.28±0.08	81.74
	✔	97.45±0.06	82.36

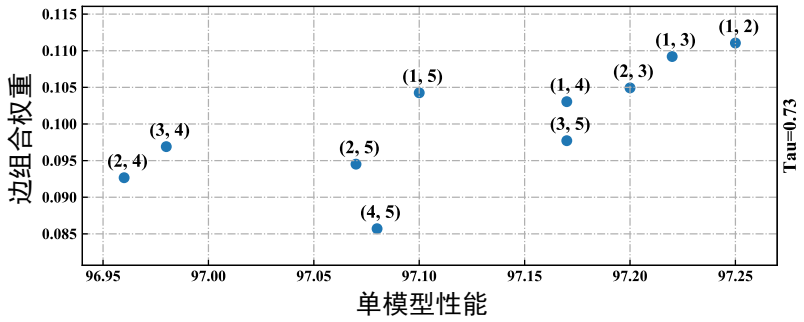
进一步选择边来弥补操作搜索的不稳定结果。

2.4.3.2 拓扑参数化策略的影响

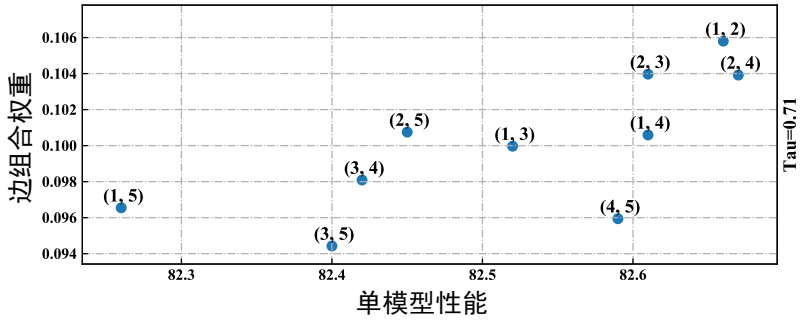
在之前的实验中，本文验证了所提出的边组合权重的有效性。本文将边组合权重与其他的拓扑结构参数化策略进行了比较。首先，本文与直接在边上增加可学习权重的 PC-DARTS 进行了比较。PC-DARTS 为每一条边添加一个权重，这种方式无法用于自动化搜索边的数量，因此本文在固定边数的设置下来对比 DOTS 和 PC-DARTS 的策略。从表 2.5中的结果来看，所提出的 DOTS 在 CIFAR10 和 CIFAR100 上将 PC-DARTS 的性能分别提高了 0.13% 和 0.74%。这主要是因为 PC-DARTS 中边权重的最初目的是为了稳定训练，而不是为了拓扑搜索。相比之下，所提出的边组合权重可以直接反映边选择的目标。接着本文验证 DOTS 在自动边数设置中的有效性。通过去除边数约束，DOTS 在 CIFAR100 上的性能从 83.72% 提升到 83.92%。在任意边数设置中，一种直接的方法是在边权重上用 sigmoid 激活替换 softmax 激活，并使用阈值对选择进行二值化。本文将此策略命名为边二值搜索。从表 2.5 来看，DOTS 比边二值搜索策略有明显的优势。

2.4.3.3 操作搜索策略的影响

本文研究了不同的操作搜索策略并讨论了它们对拓扑搜索的影响，结果如表 2.6所示。基线是 DARTS-Top1，它在拓扑搜索的每条边上保留一个最优的操



(a) CIFAR10



(b) CIFAR100

图 2.5 **DOTS 中的拓扑表示与独立模型性能相关性**。*Kendall Tau* 指标[47] 用于衡量拓扑重要性表示和模型真实性能排序相关性。

作。DARTS-Top2，即每条边留存重要性最大的两个操作。相对于 DARTS-Top1，DARTS-Top2 增加了时间开销，这是由于其在一个相对大的搜索空间进行拓扑搜索。但是其性能相对于 DARTS-Top1 没有提升，这表明简单地扩大拓扑搜索的空间不能带来拓扑搜索性能地提升。

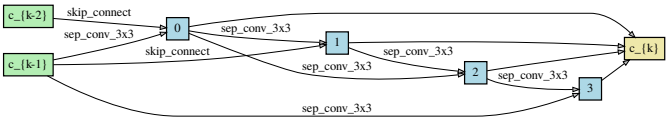
DOTS 引入了两种操作搜索策略，即，1) 结合现有的基于梯度的方法，2) 采用基于组策略的操作搜索从头开始搜索。

第一种策略继承了基于梯度的方法[8,15] 的不稳定性，因此保留的操作可能不是最优的。此外，第一种策略忽略了一些与拓扑相关的操作，最好让它们参与拓扑搜索。最近的研究[14] 表明跳接操作有两个作用：1) 作为搜索单元中的操作和 2) 稳定网络的连接。后一个角色使得跳接操作会影响网络拓扑结构。此外，DARTS[69] 中提出了 *Zero* 操作，用于在搜索阶段缩放边的重要性，这也与网络拓扑有关。在操作搜索阶段剪除这些与拓扑相关的操作消除了拓扑搜索中潜在的拓扑选择。

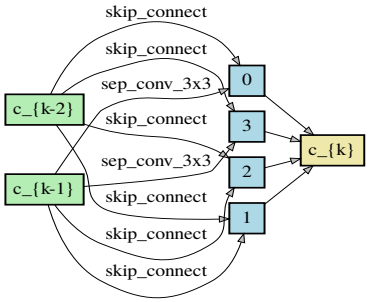
第二种策略，即，使用组策略的操作搜索，有助于稳定操作搜索并保留更

EdgeID	1	2	3	4	5	6	7
OP	SkipC	Sep3x3	SkipC	SkipC	Sep3x3	SkipC	SkipC
EdgeID	8	9	10	11	12	13	14
OP	Sep3x3	Sep3x3	SkipC	Sep3x3	Sep3x3	Sep3x3	Sep3x3

(a) 操作搜索结果



(b) 基于 DOTS 导出的拓扑结构



(c) 基于 DARTS 导出的拓扑结构

图 2.6 不同拓扑搜索策略的结果对比。

表 2.5 不同拓扑参数化策略的消融研究。

拓扑参数化策略	留存边 数量限制	**CIFAR10** **测试准确率 (%)**	**CIFAR100** **测试准确率 (%)**
PC-DARTS	2	97.38 ±0.09	82.98
DOTS	2	97.51 ±0.06	83.72
Edge-Level Sigmoid	任意	97.26 ±0.14	81.02
DOTS	任意	97.53 ±0.08	83.92

多潜在的拓扑选择。本文在实验中比较了两种分组策略，即 Group-V1 策略和 Group-V2 策略。Group-V1 策略[56] 通过将操作分为四组来考虑相似操作的多重共线性：

- Group1: *Skip-Connection*
- Group2: *Max-Pooling*, *Avg-Pooling*
- Group3: *SepConv3×3*, *SepConv5×5*
- Group4: *DilConv3×3*, *DilConv5×5*

Group-V2 策略[40] 在操作搜索中考虑马太效应。具体来说，具有可学习参数的操作在搜索开始时表现不佳，因此会通过降低其重要性来进行惩罚。较低的重要性使这些操作更新更慢，从而导致更小的重要性。因此，Group-V2 策略根据操作是否具有可学习的参数将操作分为两组：

- Group1: *Zero*, *Skip-Connection*, *Max-Pooling*, *Avg-Pooling*
- Group2: *SepConv3×3*, *SepConv5×5*, *DilConv3×3*, *DilConv5×5*

实验结果如表 2.6所示。Group-V2 策略相对于 DARTS-Top1 在 CIFAR10 和 CIFAR100 上分别提升了 0.11% 和 0.65%。在本文中，Group-V2 策略有助于避免

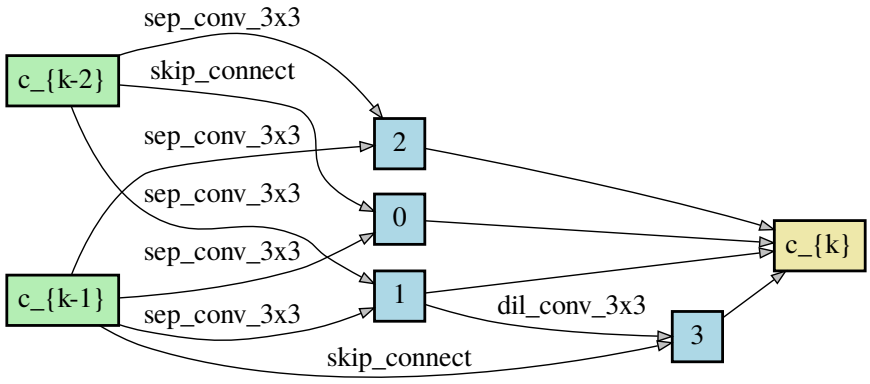
表 2.6 不同操作搜索策略的消融研究。

操作搜索策略	留存操作数量限制	CIFAR10 测试准确率 (%)	CIFAR100 测试准确率 (%)	搜索时长 (GPU/天)
DARTS-Top1	1	97.40 ±0.09	83.07	0.22
DARTS-Top2	2	97.42 ±0.11	82.96	0.26
Group-V1[56]	4	97.48 ±0.11	83.55	0.35
Group-V2[40]	2	97.51 ±0.06	83.72	0.26

马太效应，并为拓扑搜索保留更多潜在的拓扑选择。从这个实验中可以发现，更细的分组 Group-V1 对性能提升没有帮助，所以保留两个操作 (一个与拓扑相关，一个与拓扑无关) 足以用于拓扑搜索。因此，DOTS 使用 Group-V2 策略作为默认的操作搜索方法。

边序号	1	2	3	4	5	6	7
操作 1 组	SkipC	SkipC	SkipC	SkipC	SkipC	SkipC	SkipC
操作 2 组	Dil3x3	Sep3x3	Sep3x3	Sep3x3	Sep3x3	Sep3x3	Sep3x3
边序号	8	9	10	11	12	13	14
操作 1 组	SkipC	SkipC	SkipC	SkipC	SkipC	SkipC	Zero
操作 2 组	Dil3x3	Dil3x3	Dil5x5	Sep3x3	Sep3x3	Dil3x3	Sep3x3

(a) 操作搜索结果



(b) 拓扑搜索结果

图 2.7 基于 Group-V2 策略的操作和拓扑搜索结果。

图 2.7 中展示了一个基于组策略的操作搜索的实例。从图 2.7(a) 中可见，拓扑相关和拓扑不相关组中的最佳操作都被保留用于拓扑搜索。虽然拓扑相关组保留了大部分的跳接操作 (SkipC)，而拓扑无关组不会受到马太效应的影响，可以留存合适的卷积操作。通过这个搜索空间进行拓扑搜索不会被跳接操作主导，

可以验证拓扑搜索对稳定性的提升。

2.4.3.4 DOTS 与随机边搜索的比较

通过第 2.2.2节的分析，DARTS 的策略相对于随机选择边没有优势。为了验证该结论，本文将 DARTS 和 DOTS 的选择策略与随机选择边进行比较。结果总结在表 2.7。DOTS 分别在 CIFAR10 和 CIFAR100 数据集上优于随机搜索 0.56% 和 2.14%，而 DARTS 的性能在 CIFAR100 上相对于随机选择差 0.6%。该结果表明 DOTS 的拓扑搜索策略优于 DARTS 和随机边选择。

表 2.7 **不同拓扑搜索策略与随机搜索的对比。**

	随机搜索	DARTS	DOTS
CIFAR10	96.98	97.12	97.54
CIFAR100	81.52	80.92	83.66

DOTS 的稳定性。本文通过对 CIFAR10 和 CIFAR100 执行五次独立搜索来验证 DOTS 的稳定性。结果总结如表 2.8所示。由于本文将操作和搜索解耦，操作搜索的不稳定性不会影响到拓扑搜索。正如在[120] 中发现的，跳接操作通常占据较大的操作权重。如果本文遵循 DARTS 策略根据操作权重选择边，则将选择具有跳接的边，并且跳接将主导搜索到的结构。在 DOTS 中，独立拓扑搜索抑制了操作搜索的不稳定性。

表 2.8 **DOTS 搜索稳定性验证实验**。本文进行了五次的独立搜索。

	#1	#2	#3	#4	#5	平均
CIFAR10	97.54	97.51	97.49	97.58	97.62	97.55 ± 0.05
CIFAR100	83.59	83.79	83.57	83.66	83.76	83.67 ± 0.09

第五节 应用

本文将 DOTS 搜索的架构应用于物体检测和语义分割以验证其对下游任务的迁移性能。本文使用在 ImageNet 上搜索的架构替换基线方法[65,123] 的主干网络。本文与手工设计和自动搜索的轻量化主干网络进行对比。

2.5.1 目标检测

目标检测基准模型基于 RetinaNet[65]。本文使用 MMDetection 工具箱[5] 与[15] 进行公平比较。所有模型都在 MS-COCO[66] 进行训练和评估。MS-COCO 包含 115k 训练图像 (trainval35K) 和 5k 验证图像 (minival)，每个图像都用边界框进行标注。

本文使用 SGD 优化器，训练持续 12 个轮次，批量大小为 16. 初始学习率为 0.01。学习率在第 8 轮次和第 11 轮次除以 10。除非另有说明，本文仅使用水平图像翻转作为数据增强。

结果汇总在表 2.9中。从结果中，本文可以发现提出的 DOTS 作为主干网络比人工设计的轻量化神经网络 MobileNet-V3 有着相似的参数，但是性能提高了 5.8% AP。相比于基于网络搜索的 SinglePath NAS, MnasNet 和 FairDARTSC 的主干网络，DOTS 保持相似的参数，性能分别提高 5%，5.2%，3.8% AP。相比于手工设计的 ResNet-50，DOTS 的性能与 ResNet-50 相当，但仅需 ResNet-50 的 20.7% 参数和 14.5% FLOPs。上述结果表明了 DOTS 搜索的模型作为主干网络可以降低手工设计模型的复杂度，并取得类似的性能。相较于其他手工设计的和搜索轻量化模型，DOTS 能取得更好的效果。

表 2.9 **DOTS 在目标检测任务上的应用**。

主干网络	参数量 (M)	FLOPs (M)	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50[37]	25.6	4120	36.3	55.3	38.6	19.3	40.0	48.8
MobileNet-V2[91]	3.4	300	28.3	46.7	29.3	14.8	30.7	38.1
SinglePath NAS[33]	4.3	365	30.7	49.8	32.2	15.4	33.9	41.6
MobileNet-V3[42]	5.4	219	29.9	49.3	30.8	14.9	33.3	41.1
MnasNet[98]	4.8	340	30.5	50.2	32.0	16.6	34.1	41.1
FairDARTSC[15]	5.0	386	31.9	51.9	33.0	17.4	35.3	43.0
DOTS	5.3	596	35.7	55.2	37.8	19.9	39.3	47.8

2.5.2 语义分割

语义分割基准模型基于 BiSeNet[123]。所有模型都在 Cityscape 数据集[17] 上进行训练和评估。Cityscapes 数据集是来自汽车的大型城市街景数据集看法。它包含 2975 个用于训练的精细注释图像和另一个 500 张图像用于验证。在本文的实验中，本文只使用精细标注的图片。为了进行测试，它提供了 1525 张没有真

实情况的图像以进行公平比较。这些图像的分辨率均为 2048×1024，其中每个像素为注释为预定义的 19 个类。

本文使用小批量随机梯度下降 (SGD) 进行训练。训练中本文设置批量大小为 16，动量为 0.9 和权重衰减 1e-4 。本文应用“poly”学习率策略，其中初始速率乘以 $(1-\frac{iter}{max_iter})^{power}$, 每次迭代的幂为 0.9。初始学习率为 $2.5e-2$。本文采用如下数据增强：本文减去平均均值，随机水平翻转并在输入图像上随机缩放以在训练过程中扩充数据集。缩放尺度包含 {0.75, 1.0, 1.5, 1.75, 2.0}。最后，本文随机裁剪图像至固定大小 (1024×512) 以进行训练。

本文不采用任何复杂的测试技术，例如多尺度或多裁剪测试。结果如表 2.10所示，本文可以观察到 DOTS 相比于手工设计的轻量化网络 Xception-39 在验证集和测试集上分别提升了 10.3% 和 9.2% mIOU。相对于搜索得到的轻量化网络 MnasNet，DOTS 以类似的计算消耗在验证集和测试集上分别提升了 2.5% 和 2.6% mIOU。相对于 ResNet-18，DOTS 仅需 56.7% 和 64.2% 的参数量和乘加运算量，并在验证集和测试集上分别提升了 4.5% 和 2.9% mIOU。

表 2.10 **DOTS 在语义分割任务上的应用。**

主干网络	参数量 (M)	FLOPs (G)	mIOU(%) 验证集	mIOU(%) 测试集
ResNet-18	14.1	20.1	74.8	74.7
Xception-39	1.9	4.1	69.0	68.4
MnasNet	6.8	11.0	76.8	74.2
DOTS	8.0	12.9	**79.3**	**77.6**

第六节 本章小节

本章通过对可微神经网络搜索算法的研究，发现其操作和拓扑搜索的耦合导致其预测的拓扑性能排序和独立模型的性能排序相关性较差。针对该问题，本文提出了一种解耦操作和拓扑共享的可微搜索框架 (DOTS)。DOTS 将拓扑表示从操作权重中解耦，并进行显式拓扑搜索。本文提出的搜索空间直接反映了搜索目标，同时易于扩展以支持自动搜索不同数量的边。现有的基于可微搜索方法可以整合到 DOTS 中，通过拓扑搜索得到进一步改进。考虑到一些操作 (如跳接) 会影响拓扑，本文提出了一种基于分组策略的操作搜索。该策略保留与拓扑相关的操作，以便更好地进行拓扑搜索。CIFAR 和 ImageNet 实验证明了 DOTS

可以搜索出更优的结构，并且可以稳定搜索过程。DOTS 搜索到的结构可以迁移到其他视觉任务中，例如语义分割和目标检测，并用较低的复杂度取得良好的性能。

第三章　基于设备感知的整体性网络搜索

第一节　研究动机以及贡献

显著性物体检测 (SOD) 旨在分割出图像中最吸引人的物体[2,106]. SOD 作为预处理步骤，可以应用于众多下游应用，例如图片编辑[13]，图像检索[12,36]，视觉追踪[39]，图片质量评估[108]，和视频对象分割[32]。这些应用通常要求以低推理延迟将 SOD 模型部署在多个不同设备上，例如 GPU，CPU，手机和嵌入式设备。每个设备都有不同的硬件环境。例如，GPU 适用于进行大规模并行计算[81]，而嵌入式设备则以低计算功效[50] 为代价减少了计算能源消耗。正是因为不同设备有独特的运行环境，不同的部署方案需要完全不同的 SOD 模型设计。

最新的 SOD 方法大多手工设计显著性分割头[72,82,86,133,138]，来聚合来自预训练主干网 (例如 VGG[93]，ResNet[37]) 的不同层级的图像特征。由于这些方法主要在 GPU 上进行设计，这些模型通常需要较大的计算开销。过高的计算开销通常会限制将它们应用于 GPU 以外的其他设备。另一方面，近期的工作尝试为资源受限的情况手工设计低延迟的 SOD 模型[29,87]。这些方法通过设计新的主干网络来降低主干网络的复杂度。但是手工设计的较弱的主干网带来表达能力的降低，导致手工设计的轻量化 SOD 模型[29,87] 会遭受较大的性能下降。模型性能和推理延迟之间的权衡需要大量的人力资源为不同设备设计高效的 SOD 模型，导致大量的人力消耗。因此，本文旨在提供一种设备感知型的搜索方案，以便在多个设备上快速找到适配的低延迟 SOD 模型。

如图 3.1所示，在不同设备上设计低延迟 SOD 模型存在多个挑战。首先，由于不同的并行计算能力，IO 瓶颈和实现方式，各种操作的相对延迟在不同的设备之间有所不同。将为一台设备设计的 SOD 模型转移到另一台设备会导致次优的推理延迟和运行性能。其次，传统的手工设计的 SOD 模型主要依靠设计功能更强大的显著性分割头[72,82,86,138] 和设计更高效的主干网[29,87]，而忽略了它们之间的关系。类似地，大多数神经网络结构搜索 (NAS) 方法都专注于搜索分类任务的主干网络[69,98]，并将搜索到的主干网络和一个固定的分割头[61,67] 进行集成。这种方法忽略了主干网络部分和分割头部分间的关系。本文观察到，功能

强大的骨干网在较弱的显著性分割头的情况下无法达到理想的性能，反之亦然。这些障碍使得学界无法简单地通过手工或神经网络搜索方案为不同设备设计专属的低延迟 SOD 模型。

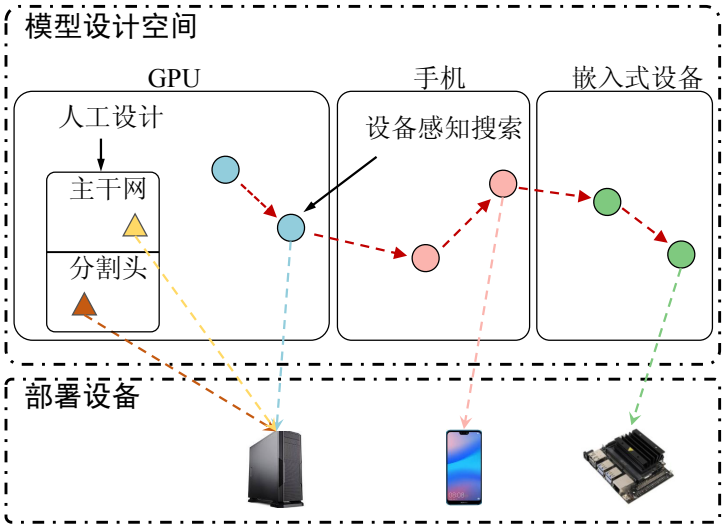


图 3.1 **iNAS 与现有的手工设计 SOD 模型方法之间的比较**。iNAS 将主干网设计和分割头设计统一到一个完整的设计空间中，并为不同的设备设计专属的 SOD 模型。

为了解决上述问题，本文提出了一种基于设备感知的整体性搜索框架 (iNAS)，仅需要一次训练就可以快速在多个设备上找到高性能低延迟的 SOD 模型。具体来说，为了整体考虑主干网和显著性分割头间的时延和性能权衡，iNAS 首先提出了一个整体性的 SOD 搜索空间，该搜索空间囊括了最先进的手工设计的 SOD 模型的设计模式。iNAS 参考了最新的手工设计的显著性分割头[41,72,76,82,130]，将其统一化为可搜索的传输器和解码器部分，从而为与主干网络进行特征交互提供了丰富的显著性分割头搜索空间。之前的工作证明了 SOD 模型需要有较强的多尺度能力，因此，这些工作设计了多分支结构捕捉不同尺度的信息。但是多分支结构会显著增加模型的推理耗时和内存消耗。为了满足 SOD 模型的多尺度需求，同时又避免了多分支结构增加的推理时延，本文构建了一个多尺度基本搜索单元。该单元支持搜索具有不同卷积核大小的多分支结构，但在推理时将搜索到的多分支卷积结构重参数化为单分支结构，以降低推理延迟。

本文提出的整体性 SOD 搜索空间包含了主干网络和显著性分割头，并利用多尺度基本单元增加了可搜索的多尺度结构。相对于仅在分类任务[4,125] 上搜索主干网络，本文提出的整体性 SOD 搜索空间更为庞大，因此之前基于分类任

务的主干网络搜索算法不适用于本文的搜索空间。之前基于分类任务的主干网络搜索算法，在训练完超网后，采用均匀采样的进化搜索[4,34,125] 来探索搜索空间里的不同结构。虽然均匀采样可以保证在同一层级搜索空间内以等概率选择候选结构。但是，当以推理时延进行衡量时，采样模型的整体时延服从多项式分布，这会导致极低时延或极高时延区域被欠采样。这种不平衡的采样模式阻碍了均匀采样完整地探索本文提出的搜索空间。为了克服这种不平衡采样问题，本文提出了一种时延组采样 (LGS)，它引入了设备时延来指导采样过程。通过将层级搜索空间划分为多个时延组，并在特定时延组中获得样本，LGS 可以保留欠采样区域的后代，同时控制过采样区域的样本数量。与均匀采样相比，使用 LGS 的进化搜索可以探索整个完整的搜索空间，并可以得到更优的时延-性能的帕累托前沿。

本文的主要贡献是：

- 一个整体性的 SOD 搜索空间，该空间考虑了主干网络-显著性分割头的整体联系，并涵盖了现有的手工 SOD 模型的设计模式。
- 提出基于时延组采样的设备感知进化搜索算法，用于完整地探索搜索空间的不同时延区域。
- 在五个常用的 SOD 数据集上对本文提出的方法进行了全面评估。本文的方法可以达到与手工构建的 SOD 方法类似的性能，但是大幅减少在不同设备上的推理时延，这有助于将 SOD 的应用扩展到不同的部署设备上。

第二节 基于设备感知的整体性网络搜索框架

3.2.1 整体性搜索空间

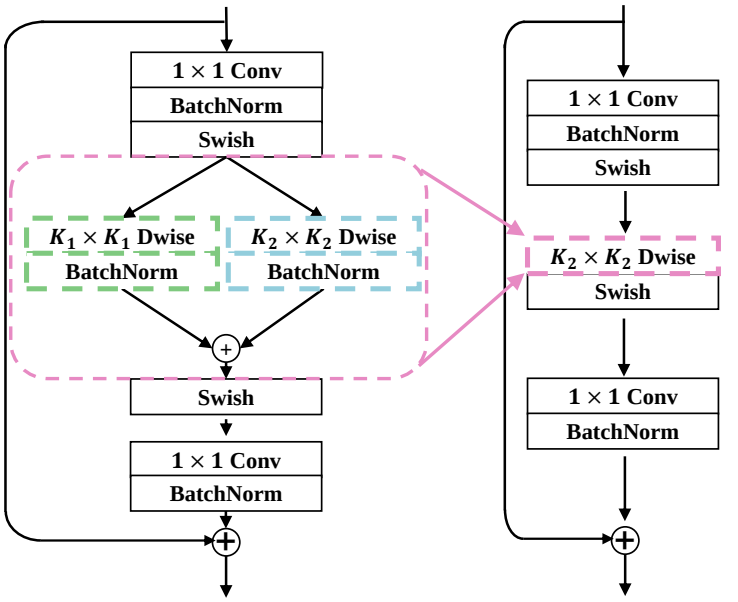
先前手工设计的 SOD 模型[2,55,74,106] 主要基于固定的预训练主干网络 (例如 VGG[93] 和 ResNet[37])，并且设计不同的显著性分割头，用以融合来自主干网络的多层级图像特征。最近的一些工作[29] 注意到，预训练的主干网在 SOD 模型中占据了大部分运算消耗。因此他们没有采用现有的繁重的主干网络，而是为 SOD 设计了专有的轻量化主干网络。但是，两种设计策略都将主干网络和显著性分割头的设计分离开来，阻碍了在整体化的模型空间中找到低延迟高性能 SOD 模型。

为了解决上述问题，本节介绍了整体性 SOD 模型搜索空间，由多尺度搜索单元和可搜索的显著性分割头组成。

3.2.1.1 多尺度搜索单元

由于通用主干网占据了大部分运算开销，因此，最新设计的显著性检测主干网[29,87] 通过分组卷积[116] 或可分离卷积[90] 代替了普通卷积以减少推理时延。为了捕获图像中的多尺度特征，他们设计了多分支结构，以不同感受野对特征进行编码，并进行融合以得到多尺度特征表示。但是，多分支结构对硬件不友好[90,110,132]，会降低推理速度并增加运行内存消耗。例如，CSNet[29] 比 ITSD-R[138] 减少了 13.4 倍计算量，但在 GPU 上的推理时延并未有所减少。因此，本文提出了多尺度搜索单元 (SMSU)，可以自动搜索合适的多尺度组合方式。

SMSU 利用多分支结构来捕获不同尺度的特征表示，每个分支有不同卷积核大小的可分离卷积组成，因此每个分支拥有不同的感受野。SMSU 能够在训练中捕获多尺度特征表示。在进行推理部署时，SMSU 利用重参数化策略[21,22] 将多个分支融合到单个分支中以进行快速推理。本文在图 3.2中展示了 SMSU 在两个分支时候的例子。



(a) 多分支结构 (训练时)　　(b) 单分支结构 (部署时)

图 3.2 多尺度搜索单元 (SMSU)。

如图 3.3所示，假设有 3×3 卷积和 5×5 卷积，本文使用 $W_1 \in \mathcal{R}^{C\times1\times3\times3}$ 和 $W_2 \in \mathcal{R}^{C\times1\times5\times5}$ 指代深度可分离卷积的卷积核。这两个卷积后面跟随着批正则化 (BatchNorm)，本文分别使用 $\mu_1,\sigma_1,\gamma_1,\beta_1$ 和 $\mu_2,\sigma_2,\gamma_2,\beta_2$ 分别指代紧接着 3×3 卷积和 5×5 卷积的批正则化参数。给定输入特征 $F_{in} \in \mathcal{R}^{C\times H\times W}$，本文使

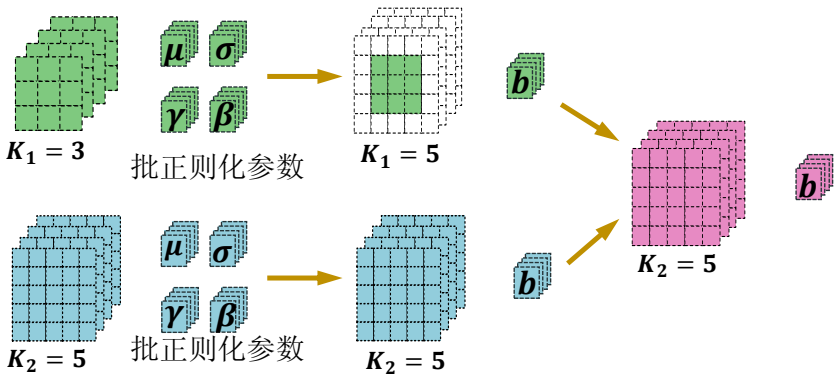


图 3.3 **多尺度搜索单元 (SMSU) 的重参数化。**

用 $M = F_{in} * W$ 指代卷积输出特征，其中 $*$ 代表卷积操作。两个分支的融合可以表示为：

$$\begin{aligned} F_{out}^{(i)} =& (M_1^{(i)} - \mu_1^{(i)}) \frac{\gamma_1^{(i)}}{\sigma_1^{(i)}} + \beta_1^{(i)} \\ &+ (M_2^{(i)} - \mu_2^{(i)}) \frac{\gamma_2^{(i)}}{\sigma_2^{(i)}} + \beta_2^{(i)}, \end{aligned} \tag{3.1}$$

其中 i 代表第 i 个通道。(3.1) 描述了训练时 SMSU 中多分支结构的融合方式。在部署时，SMSU 先将批正则化参数融合进其对应的卷积权重和偏差：

$$V^{(i)} = \frac{\gamma^{(i)}}{\sigma^{(i)}} W^{(i)}, \qquad b^{(i)} = -\frac{\mu^{(i)} \gamma^{(i)}}{\sigma^{(i)}} + \beta^{(i)}, \tag{3.2}$$

其中 V 是合并后的卷积权重，b 是合并后的卷积偏差。然后，SMSU 将较小的卷积核用零填充，扩展卷积核以匹配所有分支中最大的卷积核大小。最后，SMSU 对这两个分支的卷积核求平均值，以获得一个单分支的卷积核。

上述的双分支融合可以轻松扩展到任意数量的分支。因此，本文可以在 SMSU 中搜索不同大小的卷积组合，并且不显著增加运算消耗。本文用 SMSU 代替了 MobileNet 搜索空间[90] 的基本搜索单元，并在表 3.1和表 3.2总结了搜索空间的具体配置。

3.2.1.2 可搜索的显著性分割头

手工设计的显著性分割头通常包含传输器和解码器，以融合主干网中编码的多层级特征。高层特征提供了显著性物体的粗略位置，低层特征提供了恢复边缘和边界所需的细节特征。如图 3.4所示，

表 3.1 **整体性 SOD 搜索空间中主干网的详细配置**。本文采用 MobileNet 搜索单元[90](MBconv) 的宏观结构，并采用 SMSU 替换其中的可分离卷积。

主干网					
层级	操作	分辨率	通道数	层数	卷积核
stem	Conv	256x256-384x384	32-40	1	3
1	MBconv1	128x128-192x192	16-24	1-2	3
2	MBconv6	128x128-192x192	24-32	2-3	3
3	MBconv6	64x64-96x96	32-48	2-3	3,5,7,9
4	MBconv6	32x32-48x48	64-88	2-4	3,5,7,9
5	MBconv6	32x32-48x48	96-128	2-6	3,5,7,9
6	MBconv6	16x16-24x24	160-216	2-6	3,5,7,9
7	MBconv6	16x16-24x24	320-352	1-2	3,5,7,9

表 3.2 **整体性 SOD 搜索空间中分割头的详细配置**

传输器			解码器		
层级	卷积核	多尺度融合	层级	卷积核	多尺度融合
1	3,5,7	1-5	1	3,5,7	2-5
2	3,5,7	1-5	2	3,5,7	2-4
3	3,5,7	1-5	3	3,5,7	2-3
4	3,5,7	1-5	4	3,5,7	2
5	3,5,7	1-5	5	3,5,7	1

典型的传输器设计[82,130] 可实现自下而上和自上而下的多层特征融合。本文提出的可搜索的传输器连接到骨干网的相应分辨率层级。最大传输器的每个层级都可以聚合来自所有分辨率层级的特征 (在手工构建的传输器中对应的是 Amulet[130])，而最小传输器的每个层级仅保留跳接分支 (在手工构建的传输器中对应的是 FCN[76])。不同层级特征间的尺度变换由 1×1 卷积和池化下采样/插值上采样操作组成。本文提出的可搜索的传输器涵盖了最先进的 SOD 传输器设计[27,71,82,107,135]。

与传输器不同，解码器[41,72] 仅支持自下而上的预测细化，并逐渐融合低层特征以恢复边缘细节。因此，解码器不支持自上而下的融合分支。相邻分辨率层级中的跳接分支和上采样分支是固定的，而其他分支则是可搜索的。最大的子解码器具有与手工设计的 DSS 相似的结构[41]，而最小的子解码器像 FCN[76]。本文提出的解码器搜索空间中包含了许多手工设计的 SOD 解码器设计模式[6,73,104,113,119]。考虑到多尺度融合的最佳感受野在不同的分辨率层级上可能不同，本文使用 SMSU 作为传输器和解码器中的基本搜索单元。尽管大量工作

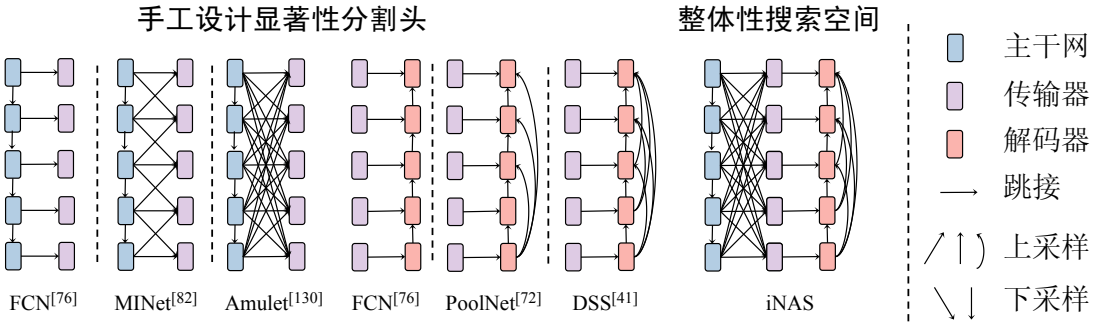
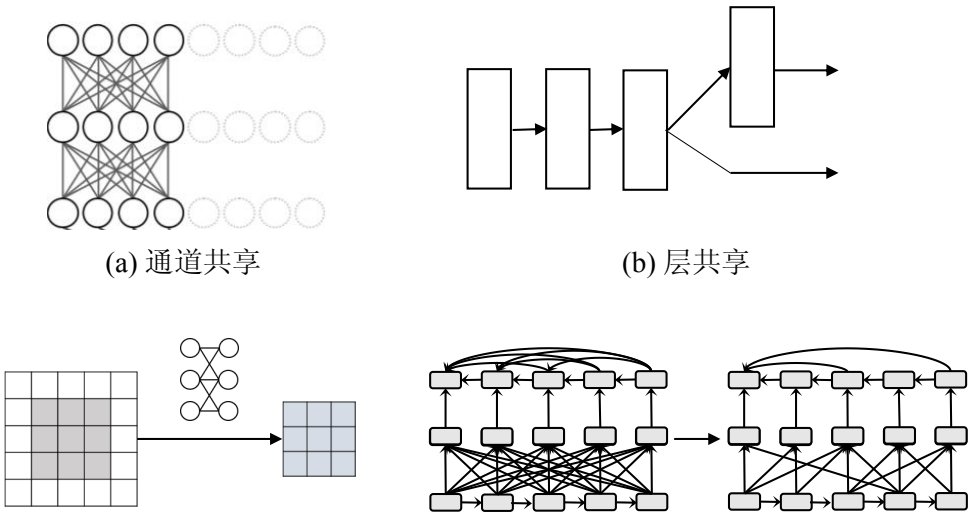


图 3.4 手工设计的 SOD 模型和提出的整体性 SOD 搜索空间。

已证明多尺度融合在 SOD 中是有效的，但是如何修剪冗余融合分支，并根据推理时延的限制选择合适的多尺度融合是一项劳动密集型的工作。本文提出的可搜索显著性分割头使这些关键组件可被搜索，并可根据推理时延限制自动设计出合适的结构。

3.2.2 动态超网训练

上一节本文引入了整体 SOD 搜索空间，整体化考虑主干网和显著性分割头的设计。这一节本文将介绍如何训练所提出的搜索空间。



(a) 通道共享　(b) 层共享

(c) 卷积核共享　(d) 多层级融合共享

图 3.5 耦合参数共享超网的构建。

首先，本文将搜索空间构建为耦合参数共享的超网。如图 3.5所示，参数共享有四个维度，分别也是该搜索空间中可搜索的维度：

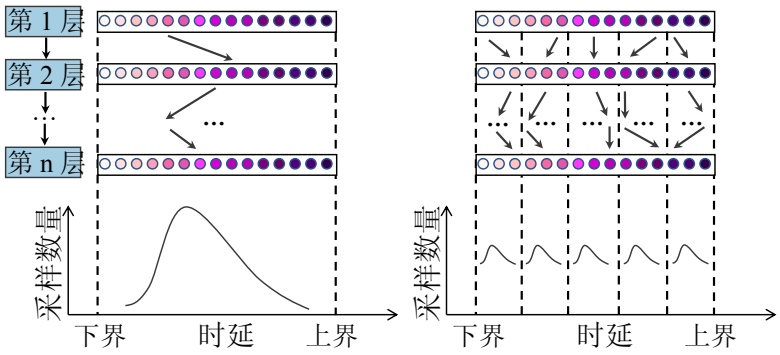
- 通道共享: 如图 3.5(a) 所示，本文仅需构建搜索空间中最大的通道数。对于

其他的通道配置，本文从最大通道的参数中截取靠前的若干通道来构建。

- 层共享: 如图 3.5(b) 所示，本文仅需构建了搜索空间中最大的层数。对于其他的层数配置，本文从最大层数的参数中截取靠前的若干层来构建
- 卷积核共享: 如图 3.5(c) 所示，本文仅需构建了搜索空间中最大的卷积核。对于其他的卷积核配置，本文从最大卷积核的参数中截取中心的特定大小的卷积核，并经过一个全连接网络来获得。
- 多层级融合共享: 如图 3.5(d) 所示，本文仅需构建了搜索空间中最密集的多层级融合连接。对于其他的多层级融合配置，本文从最大层级融合的参数中来构建。

通过构建参数共享超网络，本文仅需要构建维护最大的超网络对应的参数，其他的网络配置可以通过动态的参数选取从超网络中获得。

接着本文通过动态训练来对本文构建的超网络进行训练。本文按照[125] 中提出的三明治规则进行训练。在每次训练中，本文采样选择最小的，最大的，和两个中间大小的网络。本文将这四个网络的梯度进行混合并更新参数。三明治规则通过每次优化搜索空间的性能下界 (最小的网络)，和提高搜索空间的性能上界 (最大的网络) 和提高中间结构的性能来更新整个搜索空间。通过三明治规则训练本文构建的权重共享的搜索空间，每个搜索空间中的结构都可以具有和其单独训练类似的性能。



(a) 均匀采样　　(b) 时延组采样 (LGS)

图 3.6　**均匀采样和时延组采样 (LGS) 的对比。**

3.2.3　基于时延组采样的进化搜索算法

先前的 one-shot 搜索方法采用逐层均匀采样的进化算法，这会导致不平衡采样问题。如图 3.6所示，整体的搜索空间由分层搜索空间组成。分层搜索空间

中包含了若干个不同时延的候选模块。假设本文逐层均匀地进行采样，则整个采样模型的累计时延将服从多项式分布，即极低时延区域或极高时延区域的采样数量不足，而中等时延区域被过抽样。为了平衡地探索整体性 SOD 搜索空间的所有时延区域，本文提出了基于时延组采样 (LGS) 的进化搜索算法。LGS 通过在部署设备上构建专用的时延查找表 (LUT)，计算采样结构在部署设备上真实的运行时延。LGS 将层级搜索空间根据时延划分为几个时延组。为了获得特定时延组中的模型，LGS 在逐层空间内仅对该时延组内的候选模块进行采样。尽管采样过程在每个时延组内不平衡，但如果划分足够多的时延组，LGS 就可以平均地获得全局时延范围内的网络结构。

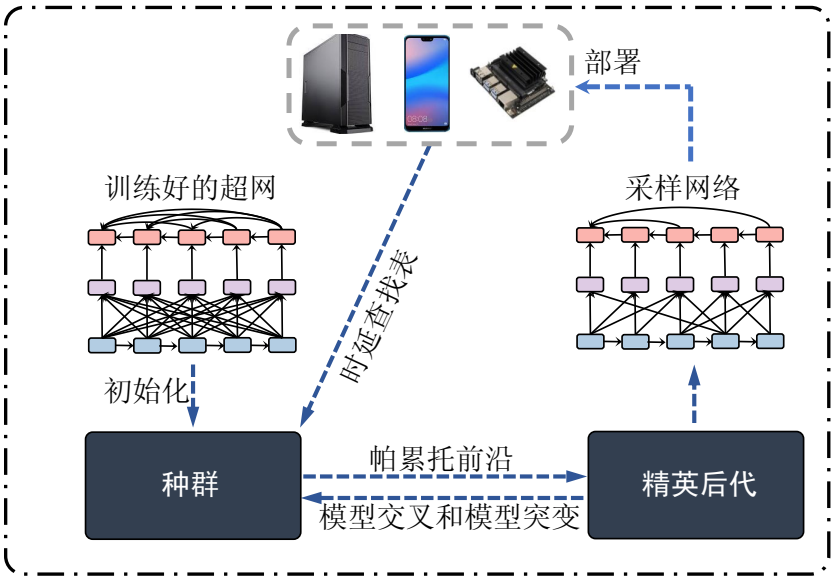


图 3.7 **iNAS 的搜索和部署流程**

基于时延组采样的进化搜索算法的整体流程见算法 1 和图 3.7。iNAS 首先在待部署设备上构建时延查找表 (LUT)。然后进行基于 LGS 的进化搜索。搜索后，搜索到的模型继承超网权重，无需重新训练即可直接部署。如所示，基于 LGS 的进化搜索包含四个阶段：

- **S1: 初始化**。LGS 将层级搜索空间中模型的时延范围划分为 G 延迟组。本文为初始群体 **P** 采样 N 个候选模型，其中每个时延组都有 $\frac{n}{G}$ 个样本。
- **S2: 选择**。LGS 从 P 的帕累托前沿中选择 k 模型进入候选模型集合 S，其中每个时延组包含 $\frac{k}{G}$ 样本。
- **S3: 交叉**。对于 S 中的每个模型，它都有 p_c 的概率与 S 中的另一个模型交

算法 1: 基于时延组采样的进化搜索

输入: 训练好的超网, 初始化种群大小 N, 时延查找表 (LUT), 时延组 G, 留存后代数量 k, 模型交叉概率 p_c, 模型突变概率 p_m, 迭代次数 $iter$.
输出: 种群的帕累托前沿 P.

从搜索空间中采样最小和最大的子模型 (即 $\mathbf{arch}_{min}$ 和 $\mathbf{arch}_{max}$);
通过时延查找表计算搜索空间的时延下限和上限 (即 $\mathbf{LAT}_{min}$ 和 $\mathbf{LAT}_{max}$);
将时延范围 $(\mathbf{LAT}_{min}, \mathbf{LAT}_{max})$ 划分为 G 组;
从每个时延组 $\{P_i | i = 1 \dots G\}$ 采样 $\frac{N}{G}$ 个子模型;
将多个时延组中采样的模型合并构成新种群 $P = P_1 \cup ... \cup P_G$;
为 P 中的所有模型计算准确率;
for $j = 1...iter$ **do**
    for 每个时延组 P_i **do**
        $S_i \leftarrow$ 从每个时延组 P_i 的帕累托前沿采样 $\frac{k}{G}$ 个模型;
    $S = S_1 \cup ... \cup S_G$;
    for S 中的每个模型 **do**
        分别以概率 p_c, p_m 进行模型交叉和模型突变;
    为 S 中的每个模型计算其性能;
    $P = P \cup S$
$P \leftarrow$ 选择种群 P 的帕累托前沿;
返回 P

叉。LGS 允许交换主干网络中的层级配置和显著性分割头中的层级配置。

- **S4: 突变**。对于 S 中的每个模型，每个配置都有 p_m 的概率发生变异。然后 LGS 将 S 合并到总体 P 中并继续到 S2，直到目标迭代 $iter$ 次结束。

LGS 和均匀采样的主要区别在于初始化和选择。在初始化步骤中，LGS 平衡了不同时延区域的样本，而均匀采样则对中间时延区域进行过采样。然后在选择步骤中，LGS 在不同的时延的种群中保留了一定数量的精英后代，这使得进化搜索能够在不同的时延区域找到更好的模型。

第三节 实验

3.3.1 实现细节

3.3.1.1 超网训练细节

本文使用 Pytorch 库[95] 和 Jittor 库[44] 实现 iNAS 。按照第 3.2.2节所描述的，

iNAS 将搜索空间组织为一个嵌套的权重共享超网。多尺度基本搜索单元的分支数量可以是 1-4，其中包含卷积核大小 (3,5,7,9) 的不同组合。在显著性分割头中，每一层的融合分支可以是 1-5，这意味着融合 (2×, 4× ,8×, 16×, 32×) 下采样分辨率的特征。本文在图像分类数据集 ImageNet 上对构建的超网进行预训练。预训练后，本文在显著性检测数据集 DUTS-TR 上对超网进行了 100 轮的训练，训练的批大小设为 40。本文使用学习率为 1e-4 和 poly 学习率衰减[75] 的 Adam 优化器。本文为每次迭代采样最大，最小和两个中间模型，并融合其梯度以更新超网。最大的子模型使与标注计算二元交叉熵商损失，其他模型则与最大子模型预测的结果做均方误差损失。这种优化方式类似知识蒸馏，使得本文搜索空间中的小模型迁移了最大模型的知识，很好地提升了搜索空间中小模型的优化效率。遵循[41] 提出的深层监督的做法，本文在每个解码器层级上都添加了监督信号。超网训练需要在 4 个 Tesla V100 GPU 上耗时 17 个小时。

3.3.1.2 搜索和部署细节

本文将初始种群规模 N 设置为 1000，并将时延组 G 设置为 10。进化迭代次数 $iter$ 设置为 20。每次筛选都保留 $k = 100$ 个后代。交叉和变异概率 (p_c 和 p_m) 设置为 0.2。由于采样模型是从超网继承的权重，超网权重中的批正则化参数是由不同采样子模型一起获得的，对于单个采样模型并不准确。因此，本文需要单独为采样模型计算其在数据集上的批正则化参数。本文通过将采样模型在训练集上进行 200 次前传，得到该采样模型在训练集上的批正则化参数[124]。本文使用 Pytorch-Mobile 库[95] 来构建手机设备的延迟表。每个设备仅需在一个 Tesla V100 GPU 上的搜索 0.8 天。

3.3.2 任务与相关工作介绍

3.3.2.1 相关工作

SOD 旨在通过二值分割提取图像中吸引人注意力的部分。传统的 SOD 方法[11,28,48,99,102,122,139] 主要依靠手工特征和启发式先验。[53,54,134] 最早尝试使用卷积神经网络 (CNN) 提取图像分块的特征。受全卷积神经网络[76] 的启发，最近的 SOD 方法[55,74,101,103,105,131] 将 SOD 定义化为像素级的预测任务，与传统方法或基于 CNN 的方法相比，取得了很大的进步。

大多数 SOD 方法都是通过手工设计显著性分割头来高效地融合预训练

表 3.3 与当前最先进的 SOD 方法在 ECSSD 和 DUT-O 数据集上的性能和速度对比。

方法	FLOPs (G)	时延 (毫秒) GPU	ECSSD(1000)			DUT-O(5168)		
			F_β	MAE	S_m	F_β	MAE	S_m
VGG-16/VGG-19								
NLDF[77]	66.68	9.48	90.5	0.063	87.5	75.3	0.080	77.0
DSS[41]	48.75	5.85	92.1	0.052	88.2	78.1	0.063	79.0
PiCANet[74]	59.82	34.21	93.1	0.046	91.4	79.4	0.068	82.6
CPD-V[113]	24.08	3.78	93.6	0.040	91.0	79.3	0.057	81.8
ITSD-V[138]	17.08	9.97	93.9	0.040	91.4	80.7	0.063	82.9
PoolNet-V[72]	48.80	8.81	94.1	0.042	91.7	80.6	0.056	83.3
EGNet-V[133]	120.15	11.58	94.3	0.041	91.9	80.9	0.057	83.6
MINet-V[82]	71.76	14.78	94.3	0.036	91.9	79.4	0.057	82.2
ResNet-34/ResNet-101/ResNetXt-101								
R3Net[19]	26.19	6.70	93.4	0.040	91.0	79.5	0.063	81.7
CPD-R[113]	7.19	2.52	93.9	0.037	91.8	79.7	0.056	82.5
BASNet[86]	97.51	16.37	94.2	0.037	91.6	80.5	0.056	83.6
PoolNet-R[72]	38.17	9.13	94.4	0.039	92.1	80.8	0.056	83.6
EGNet-R[133]	120.85	12.01	94.7	0.037	92.5	81.5	0.053	84.1
MINet-R[82]	42.68	7.38	94.7	0.033	92.5	81.0	0.056	83.3
ITSD-R[138]	9.65	3.57	94.7	0.034	92.5	82.0	0.061	84.0
手工设计 SOD 主干网								
CSNet[29]	0.72	3.63	91.6	0.065	89.3	77.5	0.081	80.5
U^2-Net[87]	9.77	4.45	94.3	0.041	91.8	81.3	0.060	83.7
基于搜索的方法								
iNAS(GPU)-S	**0.43**	**1.32**	94.4	0.037	92.1	81.9	0.055	84.2
iNAS(GPU)-L	0.70	1.94	94.7	0.036	92.4	82.4	0.052	84.6

的主干 (例如 ResNet[37] 和 VGG[93]) 提取的多层级特征中的多尺度信息。这些方法[6,41,72,73,104,119] 采用了全卷积网络中的编码器-解码器结构，其中解码器负责自下而上的特征融合。[27,82,107,129,130,135] 在显著性分割头中引入了传输器，从而得到自底向上和自顶向下的特征融合。为了得到更为精细的边缘分割结果，[60,96,112,114,133] 将边缘信息引入显著性分割头以进行精确边界细化。逐渐复杂的 SOD 模型会稳定地提高性能，但是会同时增加大量的推理时延。

最近的工作[29,113,138] 尝试设计轻量化模型以减少推理时延。其中，CPD[113] 和 ITSD[138] 设计了轻量化的显著性分割头，分别在 CPU 和 GPU 上实现了提速。CSNet[29] 设计了轻量化显著性检测主干网络以在手机和嵌入式设备上实现低延时推理。但是，当硬件特性完全不同时，将设计设备和部署设备分开考虑会导致次优推理延时。在这项工作中，本文引入了一个整体性的 SOD 搜索空间，其

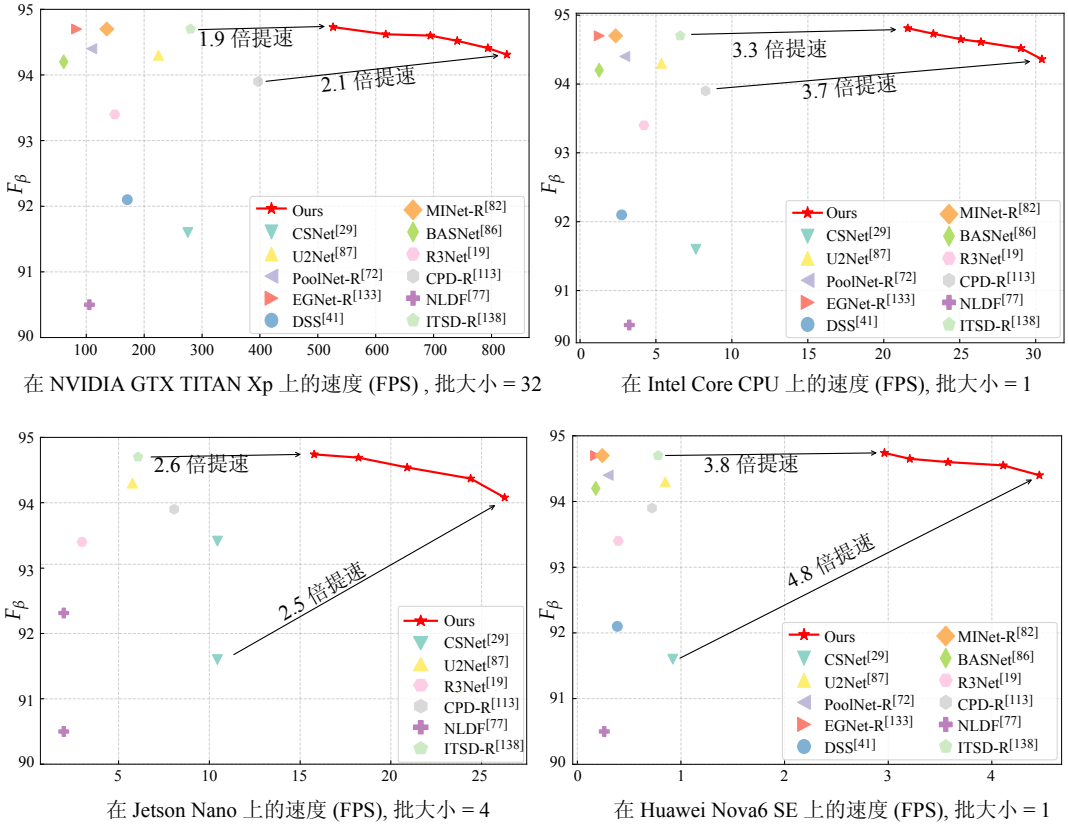
表 3.4 与当前最先进的 SOD 方法在 DUTS-TE，HKU-IS 和 PASCAL-S 数据集上的对比。

方法	DUTS-TE(5019)			HKU-IS(4447)			PASCAL-S(850)		
	F_β	MAE	S_m	F_β	MAE	S_m	F_β	MAE	S_m
VGG-16/VGG-19									
NLDF[77]	81.3	0.065	80.5	90.2	0.048	87.9	82.2	0.098	80.5
DSS[41]	82.5	0.056	81.2	91.6	0.040	87.8	83.1	0.093	79.8
PiCANet[74]	85.1	0.054	86.1	92.1	0.042	90.6	85.6	0.078	84.8
CPD-V[113]	86.4	0.043	86.6	92.4	0.033	90.4	86.1	0.072	84.5
ITSD-V[138]	87.6	0.042	87.7	92.7	0.035	90.6	86.9	0.068	85.6
PoolNet-V[72]	87.6	0.042	87.8	-	-	-	86.5	0.072	85.2
EGNet-V[133]	87.7	0.044	87.8	93.0	0.034	91.2	85.8	0.077	84.8
MINet-V[82]	87.7	0.039	87.5	93.0	0.031	91.2	86.5	0.064	85.4
ResNet-34/ResNet-101/ResNetXt-101									
R3Net[19]	83.1	0.057	83.5	91.6	0.036	89.5	83.5	0.092	80.7
CPD-R[113]	86.5	0.043	86.9	92.5	0.034	90.6	85.9	0.071	84.8
BASNet[86]	85.9	0.048	86.5	92.8	0.032	90.9	85.4	0.076	83.8
PoolNet-R[72]	88.0	0.040	88.3	93.2	0.033	91.6	86.3	0.075	84.9
EGNet-R[133]	88.8	0.039	88.7	93.5	0.031	91.7	86.5	0.074	85.2
MINet-R[82]	88.4	0.037	88.4	93.5	0.029	91.9	86.7	0.064	85.6
ITSD-R[138]	88.2	0.041	88.4	93.4	0.031	91.7	87.0	0.066	85.9
手工设计 SOD 主干网									
CSNet[29]	81.3	0.075	82.2	89.8	0.059	88.1	82.8	0.103	81.3
U^2-Net[87]	85.2	0.054	85.8	92.8	0.037	90.8	84.7	0.086	83.1
基于搜索的方法									
iNAS(GPU)-S	87.2	0.043	87.5	93.0	0.033	91.4	86.4	0.071	85.2
iNAS(GPU)-L	87.9	0.040	88.1	93.5	0.031	91.8	86.7	0.071	85.2

中涵盖了大多数手工 SOD 模型设计模式。基于本文的整体搜索空间，本文提出了一种设备感知的搜索方案，该方案直接基于待部署设备的硬件特性进行搜索，得到符合不同硬件的轻量化模型。

3.3.2.2 数据集

超网在 DUTS-TR[100] 数据集上训练。本文在五个流行的 SOD 数据集上对 iNAS 进行性能评估，即 ECSSD[121]，DUT-O[122]，DUTS-TE[100]，HKU-IS[54]，PASCAL-S[62]，分别包含 1000，5168，5019，4447，和 850 张图片和对应的显著图。



在 NVIDIA GTX TITAN Xp 上的速度 (FPS) , 批大小 = 32

在 Intel Core CPU 上的速度 (FPS), 批大小 = 1

在 Jetson Nano 上的速度 (FPS), 批大小 = 4

在 Huawei Nova6 SE 上的速度 (FPS), 批大小 = 1

图 3.8 **与当前最先进的 SOD 方法在不同设备的速度对比。**

3.3.2.3 评估指标

遵循通用设置[1,74,86]，本文使用 MAE[10]，Max F-measure (F_β)[1] 和 S-measure (S_m)[25] 作为结果的评估指标。由于本文旨在设计低延迟显著性检测模型，因此在部署设备上的推理时延也被用作评估指标。

3.3.3 量化性能对比

表 3.3和 表 3.4 展示了 iNAS 搜索的模型与现有的手工设计的 SOD 方法之间的性能比较。首先，iNAS(GPU)-L，即 GPU 上搜索到的大模型，需要与 CSNet 相似的 FLOPs，但是在 GPU 设备上减少了 47% 的推理时延，并在 ECSSD 数据集上提高了的 3.1% 的 F_β。该结果表明与设备无关的计算量指标 FLOPs 与在真实部署设备上的推理延迟没有太多相关性。

接着，本文在图 3.8展示了在不同设备上本文搜索的模型的时延-性能比较。相对于手工设计的大网络，本文的方法实现了与最先进方法相似的性能，但在 GPU，CPU，嵌入式设备和手机上分别减少了 1.9 倍，3.3 倍，2.6 倍，3.8 倍的延

迟。与之前手工设计的轻量化网络相比，iNAS 搜索的最快模型在这些设备上的速度提高了 2.1 倍，3.7 倍，2.5 倍和 4.8 倍，并大幅推高性能。

最后，本文可以看出，现有的 SOD 模型主要是为 GPU 设计的，而忽略了其他设备。由于内存不足错误，某些基于 ResNet 和基于 VGG 的方法甚至无法应用于嵌入式设备。相比之下，本文的设备感知搜索模型可在所有设备上实现一致的时延减少。

3.3.4 消融研究

3.3.4.1 整体性搜索空间

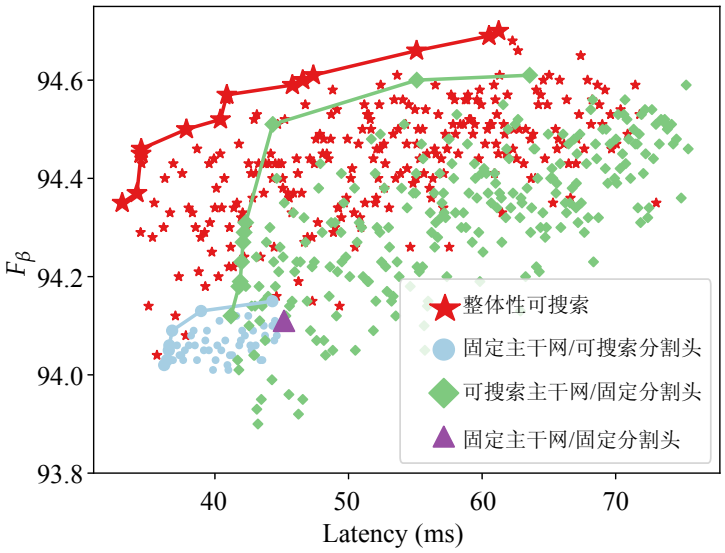


图 3.9 整体性搜索空间和部分可搜索空间的可视化对比。

iNAS 构建了整体性 SOD 搜索空间。本文在图 3.9和表 3.5中验证了整体性搜索空间的重要性。基准网络为全手工设计的轻量化 SOD 网络，包含 MobileNetV2 主干网，Amulet 传输器[130] 和 DSS 解码器[41] 结合构成的显著性分割头。该模型不可搜索，在验证集上以 45.17 毫秒的推理时延得到 94.1% 的性能。

本文进一步放开不同的可搜索维度，来探索部分搜索空间带来的收益。首先，本文将固定的主干网络换成 iNAS 的可搜索主干并进行搜索，在该情况下，本文仅需要 41.20 毫秒就可以得到和全固定结构一致的性能，减少了 8.8% 的推理时延。如果放宽时延限制，该搜索空间最高可取得 94.6% 的性能。

接着，本文采用固定的主干网，并使用可搜索的显著性分割头。该空间仅需

表 3.5 整体性搜索空间和部分可搜索空间的数值对比。

可搜索维度		低时延架构		高性能架构	
主干网	分割头	时延 (毫秒)	F_β	时延 (毫秒)	F_β
✘	✘	45.17	0.941	45.17	0.941
✔	✘	41.20	0.941	63.56	0.946
✘	✔	36.20	0.940	44.30	0.942
✔	✔	**33.06**	**0.944**	**61.24**	**0.947**

要 36.20 毫秒就可以得到和全固定结构一致的性能，减少了 19.85% 的推理时延。

最后，本文采用提出的整体性搜索空间，整体考虑不同组件间的联系。该空间仅用 33.06 毫秒的推理时延达到更好的性能。相对于全固定的结构，推理时延减少了 26.81%，性能提高了 0.4%。如果放宽了时延要求，最好模型的性能相对于全固定网络提升了 0.6%。从图 3.9中可以看出整体性搜索空间的帕累托前沿始终优于部分可搜索空间，并且在推理时延和性能方面都相对于手工设计的结构有了大幅的改进。

3.3.4.2 设备感知搜索

为了验证设备感知搜索的有效性，本文在表 3.6比较了在 GPU 和部署设备上搜索的模型。首先本文仅在 GPU 上进行模型搜索，将搜索到的模型部署到不同设备上。该模型分别在其他三个设备，CPU，手机和嵌入式设备上需要 48.90 毫秒，397.17 毫秒，71.70 毫秒的推理时延。接着本文通过提出的设备感知搜索，在对应的部署设备上直接搜索获得专用的模型。在保持性能对齐的情况下，在 CPU，手机和嵌入式设备上的推理时延分别减少了 12.1%，14.5%，10.9%。该结果验证了提出的设备感知搜索可以搜索到部署设备上的专用模型以减少推理时延。

表 3.6 在 GPU 上搜索和设备感知搜索的对比。

搜索方案	推理时延 (毫秒)			
	GPU	CPU	手机	嵌入式设备
GPU	1.94	48.90	397.17	71.70
设备感知搜索	1.94	42.99	339.61	63.39
推理时延缩减	0%	12.1%	14.5%	10.9%

3.3.4.3 时延组采样

图 3.10 比较了基于均匀采样和本文提出的时延组采样 (LGS) 的进化搜索。如 图 3.10所示，搜索空间的时延范围分别在 32.12 毫秒和 74.14 毫秒之间。均匀采样得到的时延范围在 38.55 毫秒和 59.62 毫秒之间，仅占整个时延的 50.2%。而 LGS 确保每个时延组都有平衡的样本和后代，从而可以探索 99% 的搜索空间。因此，本文提出的基于时延组采样的进化搜索相对于均匀采样可以获得更优的帕累托前沿。

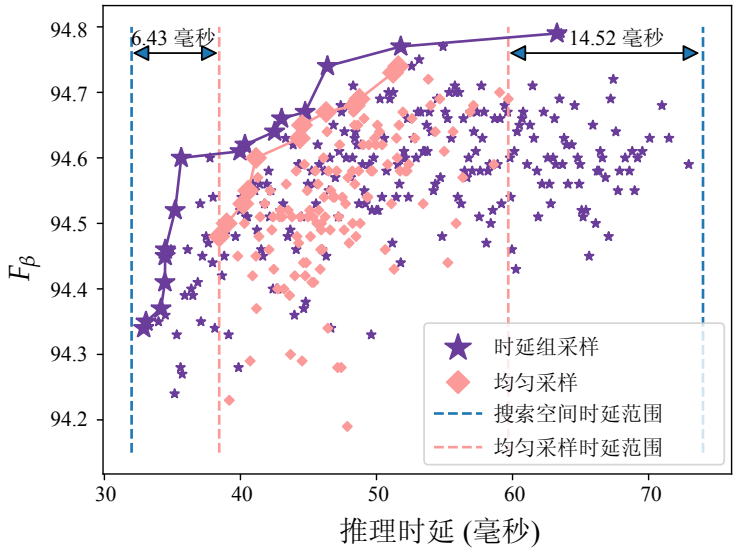


图 3.10　均匀采样和提出的时延组采样的对比。

3.3.4.4　多尺度基本搜索单元。

本文在图 3.11 中验证了所提出的多尺度基本搜索单元 (SMSU) 的有效性。本文将 SMSU 构造的搜索空间与 MobileNet[90] 搜索空间进行了比较。SMSU 构造的搜索空间增强了多尺度能力，与 MobileNet 的搜索空间相比，其得到更好的时延-性能的帕累托前沿。从结果可以观察到较高时延模型的改进更为显著，这是因为较大时延模型放宽了时延约束，可以搜索更多的多分支的卷积核组合。

多尺度基本搜索单元采用了重参数技巧来减少推理时延。本文在表 3.7验证了重参数技巧带来的提升。与最先进的手工方法 ITSD-R 相比，没有重新参数化的 iNAS 在 GPU 和 CPU 上减少了 1.5 倍和 2.7 倍时延。而经过重参数化的网络在 GPU 和 CPU 上减少了 1.9 倍和 3.5 倍的时延。

3.3.4.5　与随机搜索的比较

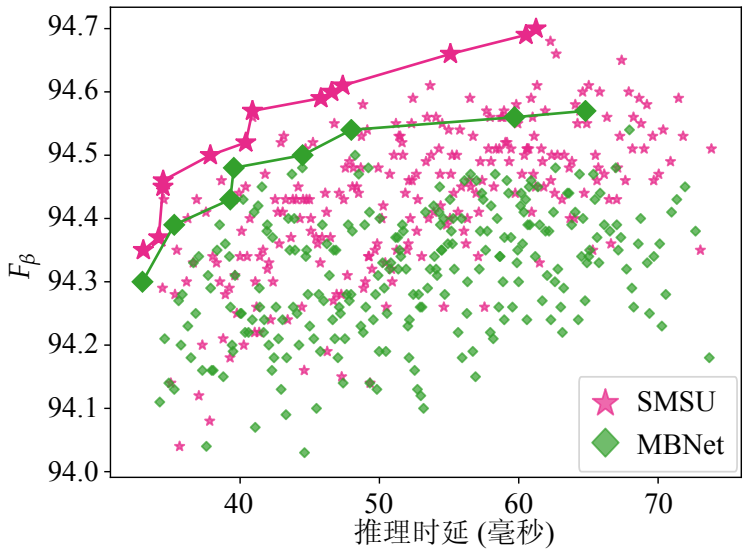


图 3.11 MobileNets(MBNet) 搜索单元[90] 和提出的多尺度搜索单元 (SMSU) 的对比。

表 3.7 **重参数化 (Rep.) 带来的性能提升。**

方法	ITSD-R	iNAS w/ Rep.	iNAS w/o Rep.
CPU 推理时延 (毫秒)	151.10	43.68	56.92
GPU 推理时延 (毫秒)	3.57	1.90	2.41

本文提出了基于时延组采样 (LGS) 的进化搜索来探索整体性 SOD 搜索空间。本文将基于 LGS 的进化搜索与随机搜索基线进行比较。图 3.12表明基于 LGS 的进化搜索相对于随即搜索有明显的优势，可以在整体性搜索空间中找到更高性能和更低时延的模型。

3.3.4.6 设备时延测量

本文在部署设备上构建时延查找表 (LUT)，用于搜索中获得采样模型在部署设备上的时延。LUT 有助于将本文的方法推广到各种专用硬件。本文在图 3.13 中验证了实际时延和估计时延的相关性。从图中，本文可以发现估计时延和实际时延的相关性接近 $y = x$，这验证了本文的时延测量的有效性。

3.3.4.7 与基于其他任务的 NAS 方法的比较

本文将提出的 iNAS 与 Auto-Deeplab[67] 进行比较，后者专为语义分割任务而设计。表 3.8和表 3.9表明 iNAS 在性能和推理延迟方面都比 Auto-Deeplab 具有明显优势。本文总结了三种不同的设计理念，使得 iNAS 优于 Auto-Deeplab：

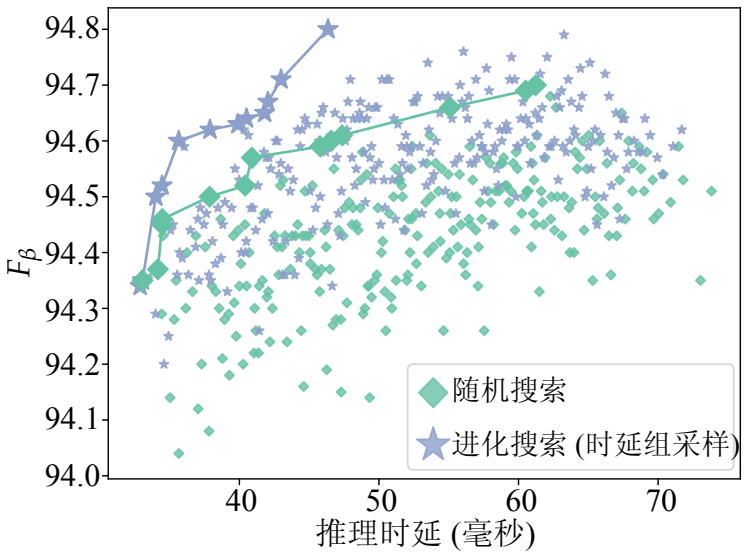


图 3.12 随机搜索和基于时延组采样的进化搜索的对比。

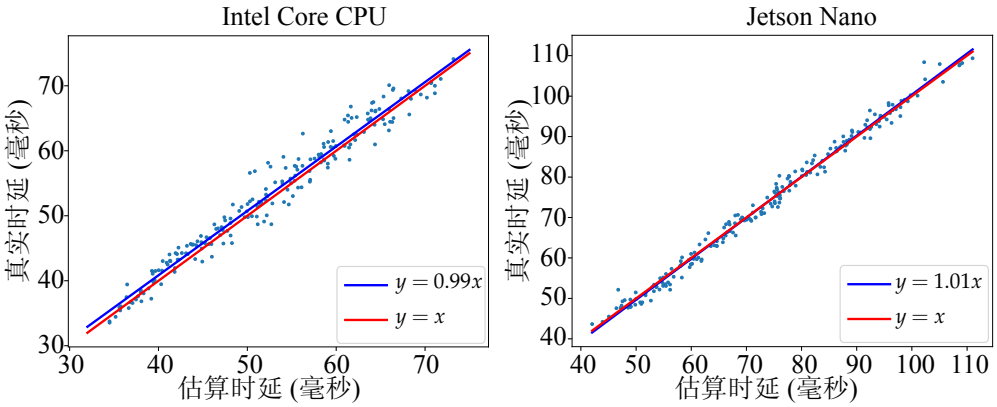


图 3.13 估算时延和实际延迟之间的相关性。

- Auto-Deeplab 搜索空间建立在基于单元的搜索空间之上。但是，搜索到的单元结构包含复杂的分支连接，这对硬件不友好。而 iNAS 得多尺度基本搜索单元在训练时保持多分支结构，但是在部署推理时可以重参数化为单分支结构来确保推理速度。
- Auto-Deeplab 支持搜索尺度变换的宏观结构。宏观结构仅支持单路径尺度变换，但缺乏在主干中编码多尺度表示的能力。然而，iNAS 支持搜索不同层级间的特征融合。
- Auto-Deeplab 使用固定的解码器结构。但是，iNAS 会自动为不同复杂度的主干网搜索合适的显著性分割头，从而减少整体推理时延。

除了有更好的性能和更低的推理时延，iNAS 的搜索成本也比 Auto-Deeplab 小。

表 3.8 **iNAS 与基于语义分割的搜索方法 Auto-Deeplab 的性能对比。**

搜索方法	推理时延 (毫秒)				ECSSD		
	GPU	CPU	手机	嵌入式设备	F_β	MAE	S_m
Auto-Deeplab[67]	15.72	446.43	4825.14	N/A	0.894	0.082	0.854
iNAS	1.94	48.90	397.17	71.70	0.947	0.036	0.924

表 3.9 **iNAS 与基于语义分割的搜索方法 Auto-Deeplab 的搜索耗时。**

搜索方法	搜索耗时 (GPU/天)	重训耗时 (GPU/天)	总耗时 (GPU/天)
Auto-Deeplab[67]	$3N$	$2N$	$5N$
iNAS	2.83+0.8N	0	2.83+0.8N

给定一个新的部署场景，Auto-Deeplab 需要重新搜索合适的模型以满足约束。搜索到的模型需要在部署之前从头开始重新训练。Auto-Deeplab 的总成本为 $5N$，其中 N 是部署场景的数量。iNAS 需要 2.83 个 GPU/天来训练一次超网。对于每个部署场景，iNAS 只需要 0.8 个 GPU/天即可获取一系列帕累托前沿的模型。iNAS 搜索到的模型无需中重新训练即可部署。当部署场景扩展到 $N = 100$ 时，本文提出的 iNAS 的搜索成本仅占 Auto-Deeplab 的 16.6%。

3.3.5 观察

为了探索模型性能与主干时延消耗和分割头时延消耗之间的关系，本文将主干和头部时延分为 10 组，并在每个网格中采样 20 个模型，得出 2000 个样本。观察 图 3.14，本文发现 (1) 更复杂的主干网能够不断提高模型性能；(2) 更复杂的显著性分割头并非总是最佳选择。这些观察结果表明，本文并不能简单地使用最复杂的显著性分割头来适配所有主干网络，而应该动态地从搜索空间中进行选取。这也就可以表明在轻量化显著性检测模型设计上，本文提出的基于整体性设备感知的网络搜索相对于手工设计，是更有效的解决方案。

第四节 本章小结

本章观察到现有的 SOD 模型设计通常着重于单独设计主干特征提取器或显著性分割头，而忽略了它们之间的关系。并且现有的模型通常基于 GPU 进行设计，当迁移至其他硬件设备上会导致性能评估带来偏差。为了解决上述问题，本章设计了基于设备感知的整体性搜索框架 (iNAS)。iNAS 首先构建了整体性 SOD 搜素空间，涵盖了现有的 SOD 模型的设计模式，并采用多尺度基本搜

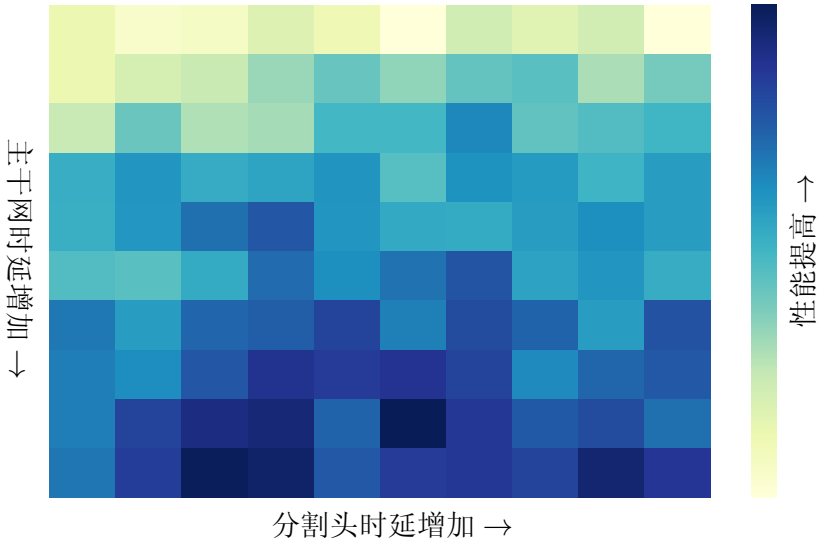


图 3.14 **骨干网/分割头时延和性能之间的对应关系可视化。**

索单元提高其建模多尺度信息的能力。接着，iNAS 将该搜索空间组织成参数共享的超网，仅需要训练一次模型，就能迅速在多个部署设备上找到高性能低延迟的专用模型。为了解决不平衡采样的问题，本文提出了时延组采样的进化搜索。实验结果表明，iNAS 可为多种设备自动设计轻量化显著性目标检测模型。

第四章 总结展望

第一节 本文工作总结

深度学习的快速发展推动神经网络模型在计算机视觉任务上的应用。随着性能的提升，神经网络模型的复杂度也在日益增加。当部署到真实应用场景的端侧设备时，虽然有良好的性能，但是这些模型需要消耗大量的计算资源，并在端侧有较大的推理时延。因此，视觉模型的轻量化在引起了广泛地关注。由于不同的部署设备，不同的时延限制和不同的视觉任务，视觉模型的探索面临着巨大的设计空间。手工为不同场景设计专用模型需要消耗大量的人力资源。受到神经网络架构搜索的启发，本文探索了自动搜索轻量化视觉模型。该课题的难点主要在于搜索算法的模型性能评估和模型真实性能存在偏差。针对该难点，本文探索了两个潜在的原因以及提出了对应的解决方案。

首先，本文发现了在可微网络搜索中，拓扑搜索耦合于操作搜索中，导致其拓扑搜索存在偏差。通过相关性实验分析，拓扑和操作共享的可微搜索中，其拓扑重要性表示和独立模型性能不存在排序相关性，这意味着这类方法对应的拓扑搜索与随机搜索相比不存在优势。为了解决该问题，本文提出了解耦操作和拓扑共享的可微搜索框架 (DOTS)。DOTS 引入了边组合权重，解耦了拓扑和操作的重要性表示。同时，DOTS 解耦了操作和拓扑的搜索过程。为了考虑到一些无参数化操作会影响到拓扑结构，DOTS 设计了基于组策略的操作搜索来保留尽可能多的拓扑结构。之前的可微搜索方法可以作为 DOTS 的操作搜索部分，通过拓扑搜索得到进一步提升。DOTS 以较低的搜索时长得到当前性能最好的网络结构，同时当迁移到下游任务构建轻量化目标检测和轻量化语义分割，该方法大幅减少了手工设计大模型的运算消耗，并取得类似的性能。

其次，本文发现了在 one-shot 网络搜索中，由于搜索设备和真实部署设备存在差距，其搜索到的结构应用于真实部署设备上存在性能偏差。为了解决该问题，本文提出了基于设备感知的整体性搜索框架 (iNAS)。iNAS 首先构建整体性的显著性检测搜索空间，涵盖了手工模型的设计模式。接着，iNAS 将该搜索空间构建为权重共享的超网，并通过采样训练同时优化该搜索空间内的所有模

型。基于这个方式，iNAS 只需要一次训练，搜索空间中的所有子网络都可以取得与单独训练类似的性能。这极大减少了模型性能评估的时间并减少了评估偏差。最后，iNAS 在待部署设备上构建时延查找表。基于该表，iNAS 可以准确度量采样模型在部署设备的真实时延，减少了搜索误差。相对于手工设计的模型，iNAS 在不同硬件设备的显著性物体检测任务上均大幅减少推理时延，为轻量化视觉显著性检测铺平了道路。

第二节　未来工作展望

本文探索了自动搜索轻量化视觉网络。虽然探讨了神经网络结构搜索存在的评估偏差的两点重要原因，但是还有未知的造成偏差的原因值得挖掘。除此之外，如何搜索多任务适配的通用模型仍待发掘。本文探索的是用有监督的数据进行评估网络性能，当利用更复杂的未标注数据进行无监督地神经网络搜索仍缺乏探索。因此，如何进一步扩展神经网络搜索在不同的视觉任务上的应用以及如何进一步通过无监督方式进行准确的模型性能评估具有很高的研究意义和挑战。

参考文献

[1] Achanta R, Hemami S, Estrada F, et al. Frequency-tuned salient region detection. In: CVPR, 2009: 1597–1604.

[2] Borji A, Cheng M.-M, Hou Q, et al. Salient object detection: A survey. Computational visual media, 2019, 5(2): 117–150.

[3] Cai H, Zhu L and Han S. ProxylessNAS: Dsearchirect Neural Architecture Search on Target Task and Hardware. In: ICLR, 2019.

[4] Cai H, Gan C, Wang T, et al. Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791, 2019.

[5] Chen K, Wang J, Pang J, et al. Mmdetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155, 2019.

[6] Chen S, Tan X, Wang B, et al. Reverse attention for salient object detection. In: ECCV, 2018: 234–250.

[7] Chen X, Hsieh C.-J. Stabilizing Differentiable Architecture Search via Perturbation-based Regularization. In: ICML, 2020.

[8] Chen X, Xie L, Wu J, et al. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In: ICCV, 2019: 1294–1303.

[9] Chen Y, Fan H, Xu B, et al. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019: 3435–3444.

[10] Cheng M.-M, Warrell J, Lin W.-Y, et al. Efficient Salient Region Detection with Soft Image Abstraction. In: ICCV, 2013: 1529–1536.

[11] Cheng M.-M, Mitra N J, Huang X, et al. Global contrast based salient region detection. IEEE TPAMI, 2015, 37(3): 569–582.

[12] Cheng M.-M, Hou Q.-B, Zhang S.-H, et al. Intelligent Visual Media Processing: When Graphics Meets Vision. Journal of Computer Science and Technology, 2017, 32(1): 110–121.

[13] Cheng M.-M, Zhang F.-L, Mitra N J, et al. Repfinder: finding approximately repeated scene elements for image editing. ACM Transactions on Graphics (TOG), 2010, 29(4): 1–8.

[14] Chu X, Wang X, Zhang B, et al. DARTS-: Robustly Stepping out of Performance Collapse Without Indicators. arXiv preprint arXiv:2009.01027, 2020.

[15] Chu X, Zhou T, Zhang B, et al. Fair darts: Eliminating unfair advantages in differentiable architecture search. arXiv preprint arXiv:1911.12126, 2019.

[16] Chu X, Zhang B, Xu R, et al. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. arXiv preprint arXiv:1907.01845, 2019.

[17] Cordts M, Omran M, Ramos S, et al. The Cityscapes dataset for semantic urban scene understanding. In: CVPR, 2016: 3213–3223.

[18] Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database. In: CVPR, 2009: 248–255.

[19] Deng Z, Hu X, Zhu L, et al. R3net: Recurrent residual refinement network for saliency detection. In: IJCAI, 2018: 684–690.

[20] DeVries T, Taylor G W. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552, 2017.

[21] Ding X, Guo Y, Ding G, et al. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In: ICCV, 2019: 1911–1920.

[22] Ding X, Zhang X, Ma N, et al. RepVGG: Making VGG-style ConvNets Great Again. arXiv preprint arXiv:2101.03697, 2021.

[23] Dong X, Yang Y. One-shot neural architecture search via self-evaluated template network. In: ICCV, 2019: 3681–3690.

[24] Dong X, Yang Y. Searching for a robust neural architecture in four gpu hours. In: CVPR, 2019: 1761–1770.

[25] Fan D.-P, Cheng M.-M, Liu Y, et al. Structure-measure: A new way to evaluate foreground maps. In: ICCV, 2017: 4548–4557.

[26] Fang J, Sun Y, Zhang Q, et al. Densely connected search space for more flexible neural architecture search. In: CVPR, 2020: 10628–10637.

[27] Feng M, Lu H and Ding E. Attentive feedback network for boundary-aware salient object detection. In: CVPR, 2019: 1623–1632.

[28] Gao D, Mahadevan V and Vasconcelos N. The discriminant center-surround hypothesis for bottom-up saliency. NeurIPS, 2007, 20: 497–504.

[29] Gao S.-H, Tan Y.-Q, Cheng M.-M, et al. Highly Efficient Salient Object Detection with 100K Parameters. In: ECCV, 2020.

[30] Gao S, Cheng M.-M, Zhao K, et al. Res2net: A new multi-scale backbone architecture. IEEE TPAMI, 2019.

[31] Ghiasi G, Lin T.-Y and Le Q V. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In: CVPR, 2019: 7036–7045.

[32] Gu Y, Wang L, Wang Z, et al. Pyramid constrained self-attention network for fast video salient object detection. In: AAAI, 2020: 10869–10876.

[33] Guo Z, Zhang X, Mu H, et al. Single path one-shot neural architecture search with uniform sampling. arXiv preprint arXiv:1904.00420, 2019.

[34] Guo Z, Zhang X, Mu H, et al. Single path one-shot neural architecture search with uniform sampling. In: European Conference on Computer Vision, 2020: 544–560.

[35] He C, Ye H, Shen L, et al. Milenas: Efficient neural architecture search via mixed-level reformulation. In: CVPR, 2020: 11993–12002.

[36] He J, Feng J, Liu X, et al. Mobile product search with Bag of Hash Bits and boundary reranking. In: CVPR, 2012.

[37] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: CVPR, 2016: 770–778.

[38] Hinton G, Vinyals O, Dean J, et al. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015, 2(7).

[39] Hong S, You T, Kwak S, et al. Online tracking by learning discriminative saliency map with convolutional neural network. In: ICML, 2015: 597–606.

[40] Hong W, Li G, Zhang W, et al. DropNAS: Grouped Operation Dropout for Differentiable Architecture Search. In: IJCAI, 2020: 2326–2332.

[41] Hou Q, Cheng M.-M, Hu X, et al. Deeply supervised salient object detection with short connections. IEEE TPAMI, 2019, 41(4): 815–828.

[42] Howard A, Sandler M, Chu G, et al. Searching for mobilenetv3. In: ICCV, 2019: 1314–1324.

[43] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

[44] Hu S.-M, Liang D, Yang G.-Y, et al. Jittor: a novel deep learning framework with meta-operators and unified graph execution. Science China Information Sciences, 2020, 63(222103): 1–21.

[45] Hu S, Xie S, Zheng H, et al. DSNAS: Direct Neural Architecture Search without Parameter Retraining. In: CVPR, 2020: 12084–12092.

[46] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks. In: CVPR, 2017: 4700–4708.

[47] Kendall M G. A new measure of rank correlation. Biometrika, 1938, 30(1/2): 81–93.

[48] Klein D A, Frintrop S. Center-surround divergence of feature statistics for salient object detection. In: ICCV, 2011: 2214–2219.

[49] Krizhevsky A, Sutskever I and Hinton G E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2012, 25.

[50] Kurniawan A. Introduction to NVIDIA Jetson Nano. Springer, 2021: 1–6.

[51] Larsson G, Maire M and Shakhnarovich G. Fractalnet: Ultra-deep neural networks without residuals. arXiv preprint arXiv:1605.07648, 2016.

[52] Li C, Peng J, Yuan L, et al. Block-wisely Supervised Neural Architecture Search with Knowledge Distillation. In: CVPR, 2020: 1989–1998.

[53] Li G, Yu Y. Deep contrast learning for salient object detection. In: CVPR, 2016: 478–487.

[54] Li G, Yu Y. Visual saliency based on multiscale deep features. In: CVPR, 2015: 5455–5463.

[55] Li G, Xie Y, Lin L, et al. Instance-level salient object segmentation. In: CVPR, 2017: 2386–2395.

[56] Li G, Zhang X, Wang Z, et al. StacNAS: Towards stable and consistent optimization for differentiable Neural Architecture Search. arXiv preprint arXiv:1909.11926, 2019.

[57] Li G, Qian G, Delgadillo I C, et al. Sgas: Sequential greedy architecture search. In: CVPR, 2020: 1620–1630.

[58] Li H, Kadav A, Durdanovic I, et al. Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710, 2016.

[59] Li X, Lin C, Li C, et al. Improving one-shot nas by suppressing the posterior fading. In: CVPR, 2020: 13836–13845.

[60] Li X, Yang F, Cheng H, et al. Contour knowledge transfer for salient object detection. In: ECCV, 2018: 355–370.

[61] Li Y, Song L, Chen Y, et al. Learning dynamic routing for semantic segmentation. In: CVPR, 2020: 8553–8562.

[62] Li Y, Hou X, Koch C, et al. The secrets of salient object segmentation. In: CVPR, 2014: 280–287.

[63] Li Y, Jin X, Mei J, et al. Neural architecture search for lightweight non-local networks. In: CVPR, 2020: 10297–10306.

[64] Liang H, Zhang S, Sun J, et al. Darts+: Improved differentiable architecture search with early stopping. arXiv preprint arXiv:1909.06035, 2019.

[65] Lin T.-Y, Goyal P, Girshick R, et al. Focal Loss for Dense Object Detection. In: ICCV, Oct. 2017.

[66] Lin T.-Y, Maire M, Belongie S, et al. Microsoft COCO: Common objects in context. In: ECCV, 2014: 740–755.

[67] Liu C, Chen L.-C, Schroff F, et al. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: CVPR, 2019: 82–92.

[68] Liu C, Zoph B, Neumann M, et al. Progressive neural architecture search. In: ECCV, 2018: 19–34.

[69] Liu H, Simonyan K and Yang Y. DARTS: Differentiable Architecture Search. In: ICLR, 2019.

[70] Liu H, Simonyan K, Vinyals O, et al. Hierarchical Representations for Efficient Architecture Search. In: ICLR, 2018.

[71] Liu J.-J, Liu Z.-A and Cheng M.-M. Centralized Information Interaction for Salient Object Detection. arXiv preprint arXiv:2012.11294, 2020.

[72] Liu J.-J, Hou Q, Cheng M.-M, et al. A simple pooling-based design for real-time salient object detection. In: CVPR, 2019: 3917–3926.

[73] Liu N, Han J. DHSNet: Deep hierarchical saliency network for salient object detection. In: CVPR, 2016: 678–686.

[74] Liu N, Han J and Yang M.-H. PiCANet: Learning pixel-wise contextual attention for saliency detection. In: CVPR, 2018: 3089–3098.

[75] Liu Y, Gu Y.-C, Zhang X.-Y, et al. Lightweight Salient Object Detection via Hierarchical Visual Perception Learning. IEEE TCYB, 2020.

[76] Long J, Shelhamer E and Darrell T. Fully convolutional networks for semantic segmentation. In: CVPR, 2015: 3431–3440.

[77] Luo Z, Mishra A K, Achkar A, et al. Non-local Deep Features for Salient Object Detection. In: CVPR, 2017: 6609–6617.

[78] Ma N, Zhang X, Zheng H.-T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: ECCV, 2018: 116–131.

[79] Mindspore, 2020.

[80] Noy A, Nayman N, Ridnik T, et al. Asap: Architecture search, anneal and prune. In: AISTATS, 2020: 493–503.

[81] NVIDIA, Vingelmann P and Fitzek F H. CUDA, release: 10.2.89, 2020. `https://developer.nvidia.com/cuda-toolkit`.

[82] Pang Y, Zhao X, Zhang L, et al. Multi-Scale Interactive Network for Salient Object Detection. In: CVPR, 2020: 9413–9422.

[83] Paszke A, Gross S, Massa F, et al. Pytorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703, 2019.

[84] Pham H, Guan M Y, Zoph B, et al. Efficient Neural Architecture Search via Parameter Sharing. In: ICML, 2018.

[85] Pham H, Guan M, Zoph B, et al. Efficient Neural Architecture Search via Parameters Sharing. In: ICML. PMLR, 2018: 4095–4104.

[86] Qin X, Zhang Z, Huang C, et al. BASNet: Boundary-Aware Salient Object Detection. In: CVPR, 2019: 7479–7489.

[87] Qin X, Zhang Z, Huang C, et al. U2-Net: Going deeper with nested U-structure for salient object detection. Pattern Recognition, 2020, 106: 107404.

[88] Real E, Aggarwal A, Huang Y, et al. Regularized evolution for image classifier architecture search. In: AAAI, 2019: 4780–4789.

[89] Saikia T, Marrakchi Y, Zela A, et al. AutoDispNet: Improving Disparity Estimation With AutoML. In: ICCV, Oct. 2019.

[90] Sandler M, Howard A, Zhu M, et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: CVPR, June 2018.

[91] Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In: CVPR, 2018: 4510–4520.

[92] Shu Y, Wang W and Cai S. Understanding Architectures Learnt by Cell-based Neural Architecture Search. In: ICLR, 2020.

[93] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[94] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting. JMLR, 2014, 15(1): 1929–1958.

[95] Steiner B, DeVito Z, Chintala S, et al. Pytorch: An imperative style, high-performance deep learning library. NeurIPS, 2019, 32: 8026–8037.

[96] Su J, Li J, Zhang Y, et al. Selectivity or invariance: Boundary-aware salient object detection. In: ICCV, 2019: 3799–3808.

[97] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. In: CVPR, 2015: 1–9.

[98] Tan M, Chen B, Pang R, et al. Mnasnet: Platform-aware neural architecture search for mobile. In: CVPR, 2019: 2820–2828.

[99] Wang J, Jiang H, Yuan Z, et al. Salient Object Detection: A Discriminative Regional Feature Integration Approach. IJCV, 2017, 123(2): 251–268.

[100] Wang L, Lu H, Wang Y, et al. Learning to detect salient objects with image-level supervision. In: CVPR, 2017: 136–145.

[101] Wang L, Wang L, Lu H, et al. Saliency detection with recurrent fully convolutional networks. In: ECCV, 2016: 825–841.

[102] Wang Q, Yuan Y and Yan P. Visual saliency by selective contrast. IEEE TCSVT, 2012, 23(7): 1150–1155.

[103] Wang T, Borji A, Zhang L, et al. A stagewise refinement model for detecting salient objects in images. In: ICCV, 2017: 4019–4028.

[104] Wang T, Zhang L, Wang S, et al. Detect globally, refine locally: A novel approach to saliency detection. In: CVPR, 2018: 3127–3135.

[105] Wang W, Shen J, Cheng M.-M, et al. An iterative and cooperative top-down and bottom-up inference network for salient object detection. In: CVPR, 2019: 5968–5977.

[106] Wang W, Lai Q, Fu H, et al. Salient object detection in the deep learning era: An in-depth survey. IEEE TPAMI, 2021.

[107] Wang W, Zhao S, Shen J, et al. Salient object detection with pyramid attention and salient edges. In: CVPR, 2019: 1448–1457.

[108] Wang X, Liang X, Yang B, et al. No-reference synthetic image quality assessment with convolutional neural network and local image saliency. Computational Visual Media, 2019, 5(2): 193–208.

[109] Wang X, Xue C, Yan J, et al. MergeNAS: Merge Operations into One for Differentiable Architecture Search. In: IJCAI, 2020: 3065–3072.

[110] Wu B, Dai X, Zhang P, et al. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: CVPR, 2019: 10734–10742.

[111] Wu J, Leng C, Wang Y, et al. Quantized convolutional neural networks for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016: 4820–4828.

[112] Wu R, Feng M, Guan W, et al. A mutual learning method for salient object detection with intertwined multi-supervision. In: CVPR, 2019: 8150–8159.

[113] Wu Z, Su L and Huang Q. Cascaded Partial Decoder for Fast and Accurate Salient Object Detection. In: CVPR, 2019: 3907–3916.

[114] Wu Z, Su L and Huang Q. Stacked cross refinement network for edge-aware salient object detection. In: ICCV, 2019: 7264–7273.

[115] Xie L, Yuille A. Genetic cnn. In: CVPR, 2017: 1379–1388.

[116] Xie S, Girshick R, Dollár P, et al. Aggregated residual transformations for deep neural networks. In: CVPR, 2017: 1492–1500.

[117] Xie S, Kirillov A, Girshick R, et al. Exploring randomly wired neural networks for image recognition. In: CVPR, 2019: 1284–1293.

[118] Xie S, Zheng H, Liu C, et al. SNAS: stochastic neural architecture search. In: ICLR, 2019.

[119] Xu Y, Xu D, Hong X, et al. Structured modeling of joint deep feature and prediction refinement for salient object detection. In: ICCV, 2019: 3789–3798.

[120] Xu Y, Xie L, Zhang X, et al. PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. In: ICLR, 2020.

[121] Yan Q, Xu L, Shi J, et al. Hierarchical saliency detection. In: CVPR, 2013: 1155–1162.

[122] Yang C, Zhang L, Lu H, et al. Saliency detection via graph-based manifold ranking. In: CVPR, 2013: 3166–3173.

[123] Yu C, Wang J, Peng C, et al. BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation. In: ECCV, Sept. 2018.

[124] Yu J, Huang T S. Universally slimmable networks and improved training techniques. In: ICCV, 2019: 1803–1811.

[125] Yu J, Jin P, Liu H, et al. Bignas: Scaling up neural architecture search with big single-stage models. In: ECCV, 2020: 702–717.

[126] Yu K, Sciuto C, Jaggi M, et al. Evaluating The Search Phase of Neural Architecture Search. In: ICLR, 2020.

[127] Yu Q, Yang D, Roth H, et al. C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation. In: CVPR, 2020: 4126–4135.

[128] Zela A, Elsken T, Saikia T, et al. Understanding and Robustifying Differentiable Architecture Search. In: ICLR, 2020.

[129] Zhang L, Dai J, Lu H, et al. A bi-directional message passing model for salient object detection. In: CVPR, 2018: 1741–1750.

[130] Zhang P, Wang D, Lu H, et al. Amulet: Aggregating multi-level convolutional features for salient object detection. In: ICCV, 2017: 202–211.

[131] Zhang P, Wang D, Lu H, et al. Learning uncertain convolutional features for accurate saliency detection. In: ICCV, 2017: 212–221.

[132] Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: CVPR, 2018: 6848–6856.

[133] Zhao J.-X, Liu J, Fan D.-P, et al. EGNet: Edge Guidance Network for Salient Object Detection. In: ICCV, 2019: 8779–8788.

[134] Zhao R, Ouyang W, Li H, et al. Saliency detection by multi-context deep learning. In: CVPR, 2015: 1265–1274.

[135] Zhao X, Pang Y, Zhang L, et al. Suppress and balance: A simple gated network for salient object detection. In: ECCV, 2020: 35–51.

[136] Zheng X, Ji R, Wang Q, et al. Rethinking Performance Estimation in Neural Architecture Search. In: CVPR, 2020: 11356–11365.

[137] Zhou D, Zhou X, Zhang W, et al. EcoNAS: Finding Proxies for Economical Neural Architecture Search. In: CVPR, 2020: 11396–11404.

[138] Zhou H, Xie X, Lai J.-H, et al. Interactive Two-Stream Decoder for Accurate and Fast Saliency Detection. In: CVPR, June 2020.

[139] Zhu W, Liang S, Wei Y, et al. Saliency optimization from robust background detection. In: CVPR, 2014: 2814–2821.

[140] Zoph B, Le Q V. Neural architecture search with reinforcement learning. In: ICLR, 2017.

[141] Zoph B, Vasudevan V, Shlens J, et al. Learning transferable architectures for scalable image recognition. In: CVPR, 2018: 8697–8710.

致谢

岁月如梭。转眼间，三年的研究生求学生活即将结束，站在毕业的门槛上，回首往昔，奋斗和辛劳成为丝丝的记忆。值此毕业论文完成之际，我谨向所有支持我的人们表示最诚挚的感谢与最美好的祝愿。

感谢我的导师程明明教授。程老师的谆谆教导引领我进入了科研的大门。三年来，导师渊博的专业知识，严谨的治学态度，精益求精的工作作风，诲人不倦的高尚师德，平易近人的人格魅力对我影响深远。在此我向我的导师程明明教授表示深切的谢意与祝福。

感谢刘云师兄在研究生阶段对我的帮助和引领。感谢实验室的所有同学们，大家一起在实验室共同努力的日子我将永远铭记。还要感谢父母和王丽娟同学在我求学生涯中给与我无微不至的关怀，一如既往地支持我、鼓励我。

希望在未来的日子里还能与大家并肩前行。

个人简历

教育背景:

- 南开大学计算机技术专业，硕士，2019/09-2022/06
- 北京化工大学计算机科学与技术，学士，2015/09-2019/06

学术成果:

[1] **Yu-Chao Gu**, Zi-Qin Wang, Yun Liu, Ming-Ming Cheng, Shao-Ping Lu. Pyramid Constrained Self-Attention Network for Fast Video Salient Object Detection. Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). 2020. **(第一作者, CCF A 类)**

[2] **Yu-Chao Gu**, Yun Liu, Yi Yang, Yu-Huan Wu, Shao-Ping Lu, Ming-Ming Cheng. DOTS: Decoupling operation and topology in differentiable architecture search. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021. **(第一作者, CCF A 类)**

[3] **Yu-Chao Gu**, Shang-Hua Gao, Xu-Sheng Cao, Peng Du, Shao-Ping Lu, Ming-Ming Cheng. iNAS: Integral NAS for Device-Aware Salient Object Detection. IProceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 2021. **(第一作者, CCF A 类)**

专利：

[1] 程明明，**顾宇超**，一种基于设备感知的显著性物体检测方法及系统，公开号：CN113222934A，授权日：2021-08-06

[2] 程明明，**顾宇超**，卢少平，一种基于多尺度受约束自注意机制的视频显著性物体检测的方法，公开号：CN111242003A，授权日：2020-06-05

所获荣誉:

- 南开大学-SK 人工智能奖学金，2021 年