LuffyNet：面向硬件感知的边缘智能

林海[1]，王和钰[1]，曹越[1]，王丽园[2]，王世杰[1]
（1.武汉大学国家网络安全学院，湖北 武汉 430072；2.中交第二公路勘察设计研究院有限公司，湖北 武汉 430056）

摘　要：边缘智能面临计算实时性、资源受限等挑战，且不同设备间差异较大。多数研究通过模型压缩设计轻量化网络，以满足边缘场景下的快速推理需求。然而过度压缩导致精度下降的同时并不能缩短推理延时，从而影响边缘智能的性能。为了满足实时需求与计算资源约束条件下提高模型精度，提出面向硬件感知的边缘智能框架：LuffyNet。该框架通过查找表估算模型推理性能，并以计算时延和设备内存资源为约束，实现对边缘设备的硬件感知。为得出符合时延约束且匹配边缘设备计算资源的高精度网络，LuffyNet 框架以模型精度、推理时延和网络大小为优化目标，通过梯度下降完成目标网络模型的构建。为缩减架构搜索的时间，LuffyNet 框架基于 Best Optimize 策略与 Worst Optimize 策略，减少搜索过程中的无效计算，降低搜索的时间成本和计算开销。在 3 个 LuffyNet 网络和 4 个先进模型上的实验结果表明，LuffyNet-A 以 1.69 ms 延迟实现 66.50%的 Top-1 精度，比 ResNet50 快近 5 倍，且大小仅为 6.58 MB。LuffyNet-B 与 LuffyNet-C 在 2.65 ms 延迟内实现超过 73%的 Top-1 精度，优于 ResNet18、ResNet50、DenseNet121 与 DenseNet169 等先进模型的精度与推理速度表现。消融实验进一步验证了基于 Best Optimize 策略与 Worst Optimize 策略的 LuffyNet 框架，不仅能找到匹配边缘设备的网络，而且能将搜索时间缩短接近 25%。

关键词：边缘智能；硬件感知；网络架构搜索；推理时延；网络大小
源代码链接：https://github.com/YMEN6/LuffyNetFramework
中图分类号：TP391　　**文献标志码：**A　　**DOI：**10.19678/j.issn.1000-3428.0070165

LuffyNet: Toward Hardware-Aware Edge Intelligence

LIN Hai[1], WANG Heyu[1], CAO Yue[1], WANG Liyuan[2], WANG Shijie[1]
(1.School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, Hubei, China;
2.CCCC Second Highway Consultants Co., Ltd., Wuhan 430056, Hubei, China)

【Abstract】Edge intelligence faces challenges such as real-time computation, limited resources, and device variations. Typically, models are compressed to create lightweight networks for fast inference in edge environments. However, excessive compression reduces accuracy and does not always shorten the inference time, thereby affecting the performance of edge intelligence. To address these issues, this paper proposes a hardware-aware edge intelligence framework called LuffyNet. The framework uses a lookup table to estimate the inference performance. It applies constraints on the computational latency and device memory to make the model hardware-aware. LuffyNet aims to create high-accuracy networks that fit edge devices while satisfying latency limits. The framework optimizes the model accuracy, inference latency, and network size through gradient descent. To reduce the search time, LuffyNet uses best optimization and worst optimization strategies. This approach reduces unnecessary computation and saves time and resources. Comparison experiments evaluate LuffyNet networks against four advanced models. LuffyNet-A achieves 66.50% Top-1 accuracy with a 1.69 ms delay, approximately five times faster than ResNet50, and is only 6.58 MB in size. LuffyNet-B and LuffyNet-C exceed 73% Top-1 accuracy with a 2.65 ms delay. They outperform ResNet18, ResNet50, DenseNet121, and DenseNet169 in terms of accuracy and speed. Ablation experiments confirm that networks formed using the LuffyNet framework are not only suitable for edge devices but also reduce the search time by approximately 25%.

【Key words】edge intelligence; hardware-aware; network architecture search; inference latency; network size

0　引言

近年来，随着人工智能和边缘计算的发展，边缘智能受到广泛关注。边缘智能[1-3]是指以边缘计算为基础，将人工智能模型部署到边缘设备上，实现边缘智能化的技术方案。传统的人工智能算

基金项目：湖北省重点研发计划项目(2023BAB022)；湖北省国际科技合作项目(2023EHA033)；国家重点研发计划(2022YFB3102100)。
作者简介：林海(CCF 会员)，男，副教授、博士，主研方向为车联网、边缘计算、强化学习、数据融合；王和钰(通信作者)，硕士研究生；曹越(CCF 会员)，教授、博士；王丽园，教授级高级工程师、硕士；王世杰，硕士研究生。
收稿日期：2024-07-23　　**修回日期：**2024-09-12　**E-mail：**ymen1112@foxmail.com

法通常以模型精度为主要优化目标，而对推理速度、模型大小等其他性能指标关注较少。然而，在边缘场景中实时任务占据重要地位，因此对模型的推理速度有更高的要求。同时，大多数边缘设备的计算资源有限，难以支持复杂的网络模型，而且不同边缘设备在计算能力、组成结构等方面存在较大差异，导致相同网络模型在不同设备上的性能表现存在显著差异。

鉴于边缘设备资源有限，难以运行复杂的神经网络模型，计算卸载[4-5]和边云协同[5-6]技术将计算任务卸载到云端或其他服务器，共同完成神经网络的推理计算。然而，这些技术对通信网络以及其他设备存在依赖关系，难以提供稳定的高速推理服务。模型压缩[7-8]是另一种应对边缘设备资源短缺的主流方法，具体包括量化[9-11]、剪枝[12-14]，知识蒸馏[15-16]等技术。模型压缩通过减小神经网络的深度与规模，得到轻量化模型。然而过度轻量化可能导致精度的降低，且网络模型的大小与推理速度并不直接相关，大小相近的模型可能具有不同的推理速度[17-18]。所以，模型压缩难以满足边缘智能的高速推理需求。

上述两类方法都存在局限性，且都忽略了硬件资源对神经网络性能的影响，因此无法保证网络模型与边缘设备的计算能力以及边缘场景的实际目标相契合。为此，通过硬件感知将边缘设备的硬件能力抽象为与模型性能相关的关键指标，以此设计匹配的网络模型。为了将硬件资源抽象为指标，一种常见的方法是直接量化硬件资源，从而可以直接表达网络模型对资源的需求。在硬件资源相对一致的情况下，这种方法计算方便，且较为直观。但是不同边缘设备的框架结构、资源种类等存在较大差异，直接量化资源会使问题变得更为复杂。另一种方法是使用网络模型在对应设备上的性能表现作为衡量指标，这种方法实现简单。本文综合运用这两种方法，对于一致的关键资源，内存直接量化并用于计算；而对于差异较大的资源，处理器则使用推理时延作为衡量指标。这样可以更好地感知硬件资源对模型性能的影响。

为了设计适用于边缘领域的人工智能算法，一个直观可行的方案是将硬件感知与现有的人工智能算法相结合，为边缘设备量身定制专用模型。然而，目前针对边缘领域的人工智能算法研究相对较少，相关经验和参考有限。神经网络的设计通常高度依赖专业知识，为每个边缘设备单独训练专用模型不仅难度大，而且存在较高的不确定性，需要较多的计算资源与时间投入，同时也难以保证模型与设备的完全匹配。因此，亟需一种能够自动适应边缘场景的智能算法。本文通过神经网络架构搜索（NAS）技术[19-20]，设计一种具有硬件感知的边缘智能算法。NAS 以自动化方式搜索合适的网络架构，将网络结构的设计过程从人工设计转移到计算算法上，从而减少对专家经验的依赖。与传统的专用模型相比，NAS 不仅有效地解决了专家经验依赖的问题，还确保了网络模型与目标设备的良好适配，提供了一种更加高效且灵活的解决方案。当设备切换时，NAS 的优势更加明显。此外，NAS 可与其他手段相结合，从而定制满足多种需求的网络架构。近年来，部分研究[21-24]将硬件感知与 NAS 相结合，以寻求匹配边缘场景的高速推理模型。然而，这些研究主要关注模型精度和推理速度，忽视了模型大小的潜在影响。高精度的复杂模型由于参数过多，难以在内存有限的设备上实现高速推理；而轻量化模型虽然在内存有限的设备上更易保持推理速度，但通常需要牺牲模型精度。因此，网络的大小对边缘智能至关重要。为了设计匹配设备的高精度、低时延网络模型，需同时考虑推理时延和网络大小。需要指出的是，网络大小与推理速度之间并不存在直接关系，如 ShuffleNetv2[18] 中所提到的 MobileNetv2[25] 和 NASNET-A[26] 大小与浮点运算次数（FLOPs）相近，但前者的推理速度却明显优于后者。本文实验同样验证了类似场景：轻量化模块的推理效率较低，而复杂网络结构的推理效率更高。这表明在架构搜索时，需要综合考虑推理速度和网络大小的关系。本文将模型精度、推理时延以及网络大小作为直接的搜索目标，通过构建多目标损失函数，寻找在目标边缘设备上高精度、低时延、高适应性的网络。

架构搜索的性能取决于搜索空间与搜索算法，许多 NAS 方案则将架构搜索分为两个阶段：第一阶段构建超级网络，并对超级网络进行均匀训练；第二阶段使用搜索算法从超级网络中搜索目标网络。目前大多数 NAS 方案主要优化第二阶段，对第一阶段的研究并未引起足够重视。超级网络类似于一个生物集群，包含多种网络结构。然而，最终的目标网络结构仅由其中的小部分组成。因此，使用公平算法对整个超级网络进行训练可能导致不必要的资源浪费和时间开销。对此，ProxylessNas[22] 采用动态剪枝对超级网络进行压缩，尽可能地将资源集中到潜在网络的训练上。然而改进表现较差的网络结构也同样重要，在挖掘搜索空间性能潜力的同时，避免搜索算法出现

假阴和假阳情况。

受进化算法中精英策略与快速收敛[27-28]的启发,本文结合全局搜索与局部搜索策略以实现快速收敛,并采用 BO(Best Optimize)策略和 WO(Worst Optimize)策略,有针对性地分配计算资源和训练时间,聚焦于候选对象中表现较优与较差的部分。BO 策略将表现较好的候选块视作可能构成最终网络的候选对象,并集中计算资源对其进一步地优化。而 WO 策略则将表现较差的候选块视作潜在的优秀对象,同样集中计算资源对其进行改善,以期找出潜在的最优候选对象。通过优化表现较优的候选块,旨在提升最终网络的性能和表现。同时,通过改善表现较差的部分,防止搜索算法的偏差,进而避免引发假阳结果。

基于上述分析,本文提出了 LuffyNet 框架,一种具有硬件感知的边缘智能算法。LuffyNet 框架通过 Gumbel Softmax[29-30]构建可微神经网络架构搜索(DNAS),结合硬件感知,构建推理时延估计与计算资源约束,以获得高精度、低时延且匹配边缘设备的神经网络。LuffyNet 的搜索过程与边缘设备高度绑定,当设备切换时需重新搜索匹配的网络模型,以确保精度、推理时延和网络大小的平衡。

本文的贡献主要包括以下 3 个方面:

1)通过硬件感知,构建模型精度、推理时延和网络大小的多目标损失函数,搜索匹配边缘设备的神经网络。

2)全局搜索与局部搜索策略相结合以实现快速收敛,并基于 BO 策略与 WO 策略,在提高架构搜索性能的同时,减少搜索所需的时长与计算开销。

3)进行了对比实验和消融实验。在对比实验中,通过模型精度、推理时延和网络大小等方面的分析,结果表明,LuffyNet 框架能得到比现有方法更适配于边缘硬件的网络模型。在消融实验中,通过算法性能和搜索时间的分析,表明所提出算法的搜索性能有所提高,并缩短了 25%的搜索时间。

1 相关工作

本节首先对边缘智能领域的研究现状进行了概述,总结和探讨该领域面临的问题及其解决方法。鉴于边缘硬件的感知和抽象,以及网络架构搜索的重要性,将深入讨论这几个方面的相关研究成果。

1.1 边缘智能

边缘智能作为人工智能与边缘计算融合的前沿领域,致力于提升边缘系统的计算效率,并提供具有低时延的服务,以满足物联网和各种边缘场景任务的多样需求。文献[31]指出,大多数边缘设备计算能力有限,传统的神经网络过于复杂,难以在边缘环境部署或高速推理。针对这一问题,边缘计算领域的许多研究探索了部署与调度层面的优化方法。鉴于单一边缘设备计算资源有限,难以独自完成智能模型推理,一些研究[5-6]提出以协同推理方式,借助其他设备并行完成推理,缩短整体响应时间。CloneCloud[5]将部分计算任务卸载到云端,克服复杂模型无法在边缘设备部署的问题。文献[32]构建并发架构,利用多个处理器同时计算,以降低推理时延。然而,协同推理对通信网络以及其他设备存在依赖关系,因此难以确保推理时延的稳定性。

另外一些研究从模型本身的角度出发,旨在通过简化模型以克服其过度复杂的问题,并提高推理速度。这些研究主要基于模型压缩技术,以优化现有的人工智能模型,力图通过压缩和简化模型的规模,打造适用于边缘领域轻量、高效的推理模型。

模型压缩领域涉及多个技术方向,包括量化、剪枝和蒸馏等广泛采用的技术。量化技术通过二值量化、定点量化等手段,将繁琐的浮点数替换为位数更少的数值,从而简化模型参数,降低计算和存储开销。文献[9]提出动态定点量化方法,根据不同层级和特征的重要性,动态调整量化的精度;XNORNet[10]通过二值量化对 ImageNet 上的分类网络进行简化;文献[11]基于敏感度分析,对各层进行合适的量化以压缩模型。剪枝技术通过删除网络中的一些连接或神经元来实现对网络的压缩,文献[12]基于目标驱动,分析不同层间的依赖与关系,对模型进行全局剪枝;QSFM[13]基于特征图之间的量化相似度,对模型进行剪枝;文献[14]则提出通道剪枝方法,基于通道的重要性,对网络进行剪枝。蒸馏学习先训练大型教师模型,再训练小型学生模型,让学生模型从教师模型中学习,从而实现模型的简化和压缩。NeuRes[15]利用稀疏激活图表示高度激活的神经元,提高学生模型的性能与泛化能力。

总而言之,这些技术致力于对复杂的网络模型进行简化,以获得适用于边缘设备的轻量化模型。然而,上述的模型压缩方案在考虑边缘设备硬件环境方面存在不足,忽略了边缘场景下不同设备之间的架构和资源差异。首先,相同的量化方法在不同硬件架构上表现出差异,导致推理速度的不稳定性。其次,剪枝和蒸馏能得到轻量化网络,但由于结构过度碎片化,网络的推理速度反而会降低。因此,在边缘智能领域,应当根据硬件平台和任务需求,针对性

地设计智能模型,将边缘设备的硬件环境视为重要指标,从而规避上述问题。

1.2 硬件感知

边缘设备的硬件资源有限,难以运行复杂的神经网络,在设计面向边缘场景的网络模型时,需要对边缘设备的硬件资源进行充分的考虑。但是不同边缘设备的计算资源在种类、结构和数量上存在差异,难以用统一指标进行抽象,增加了感知边缘设备硬件环境的复杂性。

早期的研究主要集中于设计轻量化模型,以提供通用的边缘智能解决方案,典型代表如 SqueezeNet[33] 和 ShiftNet[34]。然而,简单地压缩模型未必是合适的策略。因此,部分研究将注意力转向了其他指标,从而间接评估硬件性能。其中,浮点运算次数(FLOPs)得到广泛应用,其可代表模型的计算复杂度,用于推算在不同设备上的推理速度,代表性的工作是 ShuffleNet[18] 和 MobileNetV2[25]。然而,最新研究表明,FLOPs 并不能直接反映模型的硬件效率。碎片化的网络架构设计虽然能降低 FLOPs,但在许多硬件平台上并行性能较差,导致推理时延增加。这些研究指出了通用算法在边缘场景中的局限性,也表明了硬件感知手段在减少硬件对模型性能影响方面的必要性。

近期,一些研究开始朝着个性化方案的方向发展,结合硬件感知手段为不同的边缘设备设计最佳的网络架构。MnasNet[20] 通过直接测量每个网络架构在目标硬件上的推理时延进行评估;ChamNet[21] 通过高斯过程(GP)回归与贝叶斯优化构建预测模块,计算在不同设备上模型的性能表现;FBNet[23] 在设备上测量不同网络结构的推理速度,构建查找表,用于推测不同网络结构组合的推理时延。总体而言,这些方案均使用推理时延作为硬件环境的衡量指标,这种方法有其合理性,一方面推理时延直接反映了模型与设备的匹配程度,避免了对不同资源量化、计算引入的复杂计算。另一方面,推理时延更为直观且容易测量。

然而,仅使用推理时延作为硬件感知指标过于单一,忽略了其他硬件资源可能对模型性能表现造成的影响。在神经网络计算中,处理器资源与内存资源尤为重要,前者在计算中扮演关键角色,后者涉及数据与模型的载入,直接影响运行的批次、并行度以及部署调度,同样对模型性能表现有着关键影响。同时,文献[21,23]只关注模型在边缘设备上的推理速度,忽略了边缘设备的内存资源是否足够支撑模型以最高速度运行。

因此,为弥补上述方案的不足,本文提出的 LuffyNet 框架通过对边缘设备内存资源的感知,试图以综合的视角评估模型与设备的匹配程度。鉴于内存资源易于抽象为统一指标且便于计算,本文根据边缘设备的内存容量,计算出在目标设备上维持高速推理的神经网络最大规模,并将其构建为约束条件,结合推理时延指标,从而指导网络架构匹配边缘设备。

1.3 架构搜索

NAS 技术主要由搜索空间和搜索算法构成。搜索空间涵盖了构建网络架构所需的所有因素,通常可划分为网络结构和超参数两类,它们共同塑造了神经网络模型的结构和行为。网络结构描述了神经网络的拓扑结构,即各个网络模块之间的连接方式和行为特征。而超参数则定义了每个网络模块的相关参数和具体构成,例如卷积通道数量、卷积核大小等。早期的 NAS 工作[19] 尝试搜索卷积网络的结构和超参数,巨大的搜索空间带来了极其复杂的计算代价。对此,文献[26, 35]做出改进,使用单一的卷积细胞结构堆叠成完整网络,并将搜索对象侧重于卷积细胞结构,以此降低搜索的计算开销。但这一做法限制了网络结构的多样性和网络的表现能力。文献[36]以模块封装方式(Block-wise)将网络结构模块化,从而将搜索完整网络架构问题转化为搜索模块堆叠问题,既简化了搜索空间,还避免了单一卷积细胞结构堆叠带来的影响。MnasNet[20] 构建分层搜索空间,逐渐演化高效的神经网络结构,不仅实现了整个网络中结构的多样性,还有助于遍历搜索空间,使得搜索过程更加高效。此外,Proxylessnas[22] 使用动态的搜索空间,在搜索期间动态剪枝非目标的结构来压缩搜索空间,从而加速搜索过程。

搜索算法是指用于在搜索空间中寻找目标网络架构的算法,目前主要有基于强化学习、基于进化算法以及基于梯度优化 3 种。基于强化学习的搜索算法[19-20,26,34] 将搜索过程看作一个序列决策问题,再通过强化学习来学习一个策略网络,该网络生成网络架构的序列。最新的 UCB-ENAS[37] 将 UCB 算法与强化学习相结合,通过减少不必要的计算来加速搜索。此类方法可以通过学习调整策略来适应复杂的搜索空间,灵活性强,但计算开销高。基于进化算法的搜索算法[21,38] 则通过遗传和变异来采样候选网络架构,将其中表现好的网络架构用于种群更新,最终得到目标网络架构。最新的 ENASA[39] 结合进化算法与 CIM 技术,利用硬件加速器实现加

速。此类方法具备较高的并行性，适用于不可求导或复杂问题，并且能够跳出局部最优解，但也面临收敛速度慢和局部搜索能力较弱的挑战。基于梯度优化的搜索算法[22-23,35]则将搜索空间中的网络结构看作可微分，进而直接通过梯度下降来调整网络结构参数。文献[40]采用分离策略提高架构搜索的性能和效率。此类方法更新效率高、局部搜索能力强，但容易陷入局部最优，且不适用于不可微空间或高度离散空间。

NAS 搜索算法的流程大致都可归结为采样、训练和评估。其中，在采样环节，搜索算法从搜索空间中采样得到一个网络结构，该网络在训练环节被优化以达到更出色的性能和表现，并在最终的评估环节反过来对采样和训练算法进行评估和改进。在这 3 个环节中，最耗时的是训练环节，因为在搜索空间中，必然存在大量的无用或低效的非目标网络，对这些网络进行训练会造成巨大的额外计算开销和时间成本。

为了解决这一问题，可以采用前文提到的针对搜索空间的优化算法。此外，通过改进采样算法和训练算法，也能够有效降低计算开销。在采样算法方面，一些方法采用权值共享的方式[21,24,38,40]，从超级网络直接继承参数，避免对采样得到的子网进行额外训练，从而降低训练时间和计算负担。另外一些方法则采用逐步搜索策略[41-42]，OFA[41]采用逐步缩小的采样策略，先训练超级网络，然后通过蒸馏、剪枝等技术获取各个子网；文献[42]则从小的子空间开始搜索，逐渐扩展搜索空间。文献[43-45]采用 Single-Path 替代传统的离散采样方法，通过定义单一路径的网络结构，在该结构上进行采样，结合权值共享方法来降低计算负担。在训练算法方面，文献[23,30]使用 Gumbel-Softmax，通过梯度下降和反向传播完成训练和搜索，显著缩短了架构搜索的时间。

2 硬件感知的架构搜索方法

面向硬件感知的边缘智能要求在与边缘设备硬件能力相匹配的条件下，实现低时延、高精度神经网络架构。为了实现此目标，架构搜索的目标为：

$$\min_{a\in A}\min_{W}\mathrm{Loss}(a,W) \tag{1}$$

式中：A 为给定的搜索空间；a 为采样得到的网络架构；$\mathrm{Loss}(a,W)$ 为网络架构的整体损失；W 为 a 的权值参数。对这一目标的求解，取决于三方面：搜索空间，损失函数和搜索算法。其中，搜索空间涵盖了构建网络架构所需的全部信息，损失函数 $\mathrm{Loss}(a,W)$ 决定了搜索的方向和策略，搜索算法决定着搜索的性能。

2.1 搜索空间

本文受文献[20,23,36]的启发，将 Block-wise 和分层搜索空间相结合，构建一个固定的分层超级网络，如图 1 所示。超级网络由固定层和搜索层构成，固定层不参与搜索，主要分布在超级网络的开始和结尾。搜索层间相互独立，且每个搜索层都由若干个不同的块构成。在同一层内，每个块都是一个独立的神经网络复合结构，而且不同块间相互独立。搜索算法需要在每一层唯一确定一个块，最终构建出符合目标的网络架构。通过这种方式，可以同时在每一层对相关的超参数以及网络结构进行搜索，避免了单一网络结构对网络表达能力的限制。

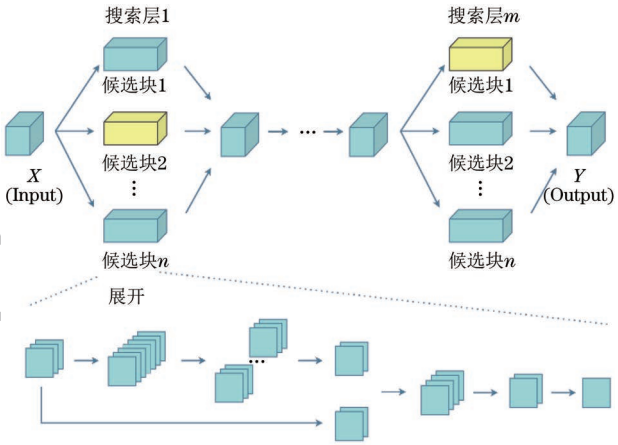

图 1 超级网络示意图

Fig. 1 Schematic diagram of the super network

本文以 ResNet18[46]作为基准网络，将卷积通道宽度、残差块堆叠深度、卷积核大小为目标进行搜索，如表 1 所示。同时，以 ResNet18 的残差块作为基础结构，探索轻量结构以寻求推理快、参数少的高性能网络结构，并探索复合结构以寻求高精度结构。如表 2 所示，本文用恒等映射、池化、普通卷积等轻量化结构替代相对复杂的残差块，在确保精度相近的前提下，寻找推理更快的轻量化结构。如表 3 所示，本文基于 Inception、BottleNeck、MBConv 等构建复合网络结构，寻找匹配设备计算资源的高精度结构。

表 1 超参数及其值域

Table 1 Hyperparameters and their value ranges

超参数	搜索值域
残差块堆叠深度	{1, 2, 3, 4}
卷积通道宽度	{8, 16, 32, 64}
卷积核大小	{1, 3, 5, 7}

表 2 NAS 中的轻量化结构

Table 2 Lightweight structures in NAS

网络结构	模块大小/MB	推理时延/ms
ResNet18 残差块	0.581	0.885
恒等映射	—	0.007
普通卷积层	0.141	0.102
深度可分离卷积[47]	0.018	0.333
最大池化层	—	0.027
平均池化层	—	0.024

表 3 NAS 中的复合结构

Table 3 Composite structures in NAS

网络结构	模块大小/MB	推理时延/ms
ResNet18 残差块	0.581	0.885
Inception[46]	0.029	0.448
BottleNeck[44]	0.034	0.467
MBConv[25]	0.019	0.166
CBR	0.141	0.148

基于上述超参数与结构，本文构建出分层超级网络，包含 2 个固定层与 4 个搜索层，其中第一层与最后一层为固定层，其余均为搜索层。每个搜索层由 14 个块组成，因此超级网络内包含 $14^4=38\,416$ 种潜在的网络，逐一采样并训练将耗费大量的时间和资源。对此，LuffyNet 框架通过梯度下降完成搜索。

2.2 损失函数

为了找到适配边缘设备硬件资源的智能模型，本文感知边缘设备的计算能力和内存资源，并将推理时延和网络大小作为评估模型性能的关键指标。由于计算资源难以抽象量化，引入推理时延作为能直接反映计算资源与模型匹配度的替代性能指标。同时，内存资源易于量化，网络模型大小可以直接计算，因此网络大小被用来评估模型与设备内存资源的匹配程度。至此，本文将网络架构的精度、推理时延以及大小作为直接目标，从而构建多目标损失函数，并将式(1)中的 $\mathrm{Loss}(a,W)$ 定义如下：

$$\mathrm{Loss}(a,W)=C_{\mathrm{CE}}(a,W)+L_{\mathrm{LAT}}(a)+H_{\mathrm{HC}}(a) \tag{2}$$

式中：$C_{\mathrm{CE}}(a,W)$ 表示权值参数为 W 的网络架构 a 的交叉熵损失，代表网络精度；$L_{\mathrm{LAT}}(a)$ 表示网络架构 a 的推理时延；$H_{\mathrm{HC}}(a)$ 为网络架构 a 的大小约束，代表与硬件的匹配度。当网络架构 a 变化时，推理时延 $L_{\mathrm{LAT}}(a)$ 也会改变。直接为每个网络架构 a 测量推理时延过于繁琐。鉴于网络模型的总推理时延是由每层网络结构的推理时延累积而成，因此可以通过每个块的推理时延来计算网络架构 a 的推理时延。本文先在目标硬件平台上测量每个块的推理时延，构建相应的查找表，再基于查找表来计算每个网络架构 a 的推理时延。由于本文的超级网络按层顺序执行，因此网络架构 a 的推理时延由每层的推理时延组成：

$$L_{\mathrm{LAT}}(a)=\sum_{l\in a}L_{\mathrm{LAT}}(b_l) \tag{3}$$

式中：b_l 表示网络架构 a 中第 l 层的块；$L_{\mathrm{LAT}}(b_l)$ 表示在目标硬件平台上的推理时延。网络架构的大小 $S_{\mathrm{Size}}(a)$ 可以表述为：

$$S_{\mathrm{Size}}(a)=\sum_{l\in a}S_{\mathrm{Size}}(b_l) \tag{4}$$

式中：$S_{\mathrm{Size}}(b_l)$ 代表了第 l 层的大小，单位为 MB。为了进一步限制网络架构的大小，本文设置网络大小的阈值 Scale，单位为 MB，并通过约束函数 $H_{\mathrm{HC}}(a)$ 实现这一目标：

$$H_{\mathrm{HC}}(a)=\left[\frac{S_{\mathrm{Size}}(a)}{S_{\mathrm{Scale}}}\right]^{\beta} \tag{5}$$

式中：指数系数 β 为一较大的正实数，对搜索的策略起控制作用。由于分子与分母单位相同，因此式(5)反映网络架构 a 与阈值的匹配情况。当网络架构 a 的大小服从硬件平台的内存资源约束，即 $S_{\mathrm{Size}}(a)\leqslant S_{\mathrm{Scale}}$ 时，$H_{\mathrm{HC}}(a)$ 趋于 0，此时式(2)仅由 $C_{\mathrm{CE}}(a,W)$ 和 $L_{\mathrm{LAT}}(a)$ 组成，搜索的目标是高精度、低时延的网络架构。同时，本文在超级网络中使用 BN、ReLU 等技术，以避免 $H_{\mathrm{HC}}(a)$ 梯度消失对搜索的影响。反之，当 $S_{\mathrm{Size}}(a)>S_{\mathrm{Scale}}$ 时，$H_{\mathrm{HC}}(a)$ 将远大于 $C_{\mathrm{CE}}(a,W)$ 和 $L_{\mathrm{LAT}}(a)$，式(2)可视为仅由 $H_{\mathrm{HC}}(a)$ 组成，此时搜索的唯一目标是降低网络架构的大小，迫使优先搜索满足资源约束的网络架构。值得注意的是，式(5)的形式不仅对网络架构 a 的网络大小起约束作用，也确保了损失函数的可微性，从而确保架构搜索可以通过梯度下降进行。

2.3 搜索算法

2.3.1 搜索算法原理

为了在超级网络的每一层选择最佳的候选块，受 FBNet 的启发，算法在每层对候选块进行抽样，且各层之间的采样独立进行。第 l 层第 i 块 $b_{l,i}$ 的选中概率可以表述为：

$$P_{\theta_l}(b_l=b_{l,i})=\mathrm{Softmax}(\theta_{l,i};\theta_l)=\frac{\exp(\theta_{l,i})}{\sum_i\exp(\theta_{l,i})} \tag{6}$$

式中：θ_l 代表第 l 层的采样概率的参数，由每块对应的 $\theta_{l,i}$ 构成。每层的输出由被选中的块表示，因此第 l 层的输出可以表述为：

$$x_{l+1}=\sum_i m_{l,i}\cdot b_{l,i}(x_l) \tag{7}$$

式中：$m_{l,i}$ 为{0，1}中的数，块 $b_{l,i}$ 被选中时为 1，否则为 0；$b_{l,i}(x_l)$则代表第 l 层在给定输入 x_l 时，块 $b_{l,i}$ 的输出。同时，第 l 层的推理时延 $L_{\mathrm{LAT}}(b_l)$以及大小 $S_{\mathrm{Size}}(b_l)$也可以表述为：

$$L_{\mathrm{LAT}}(b_l)=\sum_i m_{l,i}\cdot L_{\mathrm{LAT}}(b_{l,i}) \tag{8}$$

$$S_{\mathrm{Size}}(b_l)=\sum_i m_{l,i}\cdot S_{\mathrm{Size}}(b_{l,i}) \tag{9}$$

式中：$L_{\mathrm{LAT}}(b_{l,i})$表示块 $b_{l,i}$ 的推理时延；$S_{\mathrm{Size}}(b_{l,i})$表示块 $b_{l,i}$ 的大小。这两者对 $m_{l,i}$ 而言都为常系数。进而可以将网络架构 a 对应的推理时延和网络架构大小重新表述为：

$$L_{\mathrm{LAT}}(a)=\sum_l\sum_i m_{l,i}\cdot L_{\mathrm{LAT}}(b_{l,i}) \tag{10}$$

$$S_{\mathrm{Size}}(a)=\sum_l\sum_i m_{l,i}\cdot S_{\mathrm{Size}}(b_{l,i}) \tag{11}$$

由于各层间采样相互独立，网络架构 a 由各层采样的块构成，因此采样得到网络架构 a 的概率可以通过各层的块采样概率表示，基于式(6)，网络架构 a 的采样概率表述为：

$$P_\theta(a)=\prod_l P_{\theta_l}(b_l=b_{l,i}) \tag{12}$$

式中：θ 为超级网络的采样概率参数，由每层中每块对应的 $\theta_{l,i}$ 构成。给定 θ，在超级网络中采样到网络架构 a 的概率为 P_ϑ。基于式(2)，可以计算网络架构的损失值并用于后续的优化。通过这种方式，将最佳网络架构的搜索问题放宽为对采样概率的优化问题，通过优化超级网络的采样概率 P_ϑ 以实现式(1)的目标。从而架构搜索目标式(1)重新表述为以下形式：

$$\min_\theta \min_W E_{a\sim P_\theta}\{\mathrm{Loss}(a,W)\} \tag{13}$$

基于式(6)和式(12)可知，$P_\vartheta(a)$是关于 θ 的函数，网络架构 a 的损失 $\mathrm{Loss}(a,W)$是权值参数 W 与采样概率参数 θ 的函数，因此式(2)改写为：

$$L_{\mathrm{Loss}}(\theta,W)=C_{\mathrm{CE}}(\theta,W)+L_{\mathrm{LAT}}(\theta)+H_{\mathrm{HC}}(\theta) \tag{14}$$

通过以上方式，将超级网络描述为权值参数 W 和采样概率参数 θ 的函数。对损失函数式(14)，W 可微，但 $m_{l,i}$ 为离散随机变量，且与 θ 相关，因此不能直接通过离散随机变量 $m_{l,i}$ 构建对 θ 的梯度。为此，通过 Gumbel Softmax 函数将 $m_{l,i}$ 松弛为一个连续随机变量，即：

$$m_{l,i}=\mathrm{GumbelSoftmax}(\theta_{l,i}|\theta_l)=\frac{\exp[(\theta_{l,i}+g_{l,i})/\tau]}{\sum_i \exp[(\theta_{l,i}+g_{l,i})/\tau]} \tag{15}$$

式中：$g_{l,i}\sim\mathrm{Gumbel}(0,1)$是服从于 Gumbel 分布的随机噪声，Gumbel Softmax 由温度参数 τ 控制，当 τ 趋于 0 时，近似于离散分类抽样，当 τ 变大时，$m_{l,i}$ 变为连续随机变量，且 $m_{l,i}$ 对参数 $\theta_{l,i}$ 直接可微。式(14)中的 $C_{\mathrm{CE}}(\theta,W)$关于 θ 也是直接可微的。由于 $L_{\mathrm{LAT}}(b_{l,i})$和 $S_{\mathrm{Size}}(b_{l,i})$都为常系数，因此损失函数式(14)整体对 θ 可导。综上所述，可以通过梯度下降实现快速解决问题，并通过训练常规的神经网络的方法来实现架构搜索。

为了实现目标函数式(13)，需轮流训练参数 W 和 P_ϑ。权值参数 W 的训练可提高每个候选块在模型精度上的贡献，而 P_ϑ 的优化可提升高精度、低时延的模块被选中的概率，并降低贡献度较低模块的选中概率，从而实现最小的损失期望。

与基于强化学习和基于进化算法的搜索算法相比，通过梯度下降和反向传播实现了快速求解，减少了大量的无效计算。但是，多个候选块并行的超级网络仍然非常复杂。因此，本文通过改进搜索流程降低计算量，进一步提高搜索效率。

2.3.2 搜索算法流程

本文将架构搜索算法分解成 3 个阶段，如图 2 所示。在阶段 1 中，搜索空间被构建成一个超级网络，并采用均匀采样策略，公平地对超级网络的任一结构进行优化和训练，从而避免随机初始化带来的不良影响。在阶段 2 中，对超级网络进行搜索和评估，并基于 BO 和 WO 策略，选取出超级网络中表现最好以及最差的结构，并单独对这些结构进行训练和优化。不断迭代阶段 2，直至超级网络的表现趋于稳定。在阶段 3 中，从超级网络中采样得到最佳目标网络架构，对其重新进行训练，从而得到最终的目标网络。

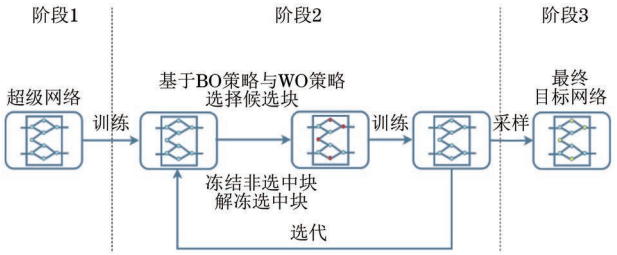

图 2 搜索算法流程

Fig. 2 Search algorithm procedure

为了搜索高精度、低时延且符合计算资源约束的目标网络架构，算法需要从模型精度、推理时延和网络大小 3 个方面对候选块进行全面评估，这与本文提出的目标损失函数式(14)保持一致。由于难以直接计算超级网络中各个块的精度，而且块对超级网络的贡献度是上述 3 个关键指标的间接反映，因

此算法使用贡献度对块进行评估。由式(7)和式(15)可知,超级网络的输出可以表述为各个块的期望,对于每一层的输出,块 $b_{l,i}$ 对应的 $m_{l,i}$ 越高,其贡献越大。又因为 Gumbel Softmax 是定义域上的增函数,基于式(15),对于第 l 层上的每个 $\theta_{l,i} \in \theta_l$ 而言,$\theta_{l,i}$ 越大,$m_{l,i}$ 越高,即块 $b_{l,i}$ 的 $\theta_{l,i}$ 越大,其贡献越高。因此算法使用 θ 对块进行评估,并构建 BO 策略与 WO 策略以优化架构搜索过程。BO 策略可描述为在超级网络训练过程中,选择每层中 $\theta_{l,i}$ 最高的若干个候选块;而 WO 策略则是选择 $\theta_{l,i}$ 最低的若干个块。在阶段 2 中,仅对这些被选中块进行训练和优化。至此,本文提出了完整的搜索算法,如算法 1 所示。

算法 1 LuffyNet 搜索算法

输入 搜索空间 A,BO 策略,WO 策略,预训练周期 n,搜索周期 m,局部训练周期 s,筛选数 k,重新训练周期 r

输出 目标网络架构 Net

Stage 1:

1. SuperNet←buildSuperNet(A);
2. for i=0 to n do
3. train θ and W for SuperNet;
4. end for

Stage 2:

5. for i = 0 to m do
6. freezeW(SuperNet);
7. B1←select(SuperNet, BO, k);
8. B2←select(SuperNet, WO, k);
9. unfreezeW(SuperNet, B1, B2);
10. for i=0 to s do
11. train θ and W for SuperNet;
12. end for
13. end for

Stage 3:

14. Net←sampleTargetNetwork(SuperNet);
15. for i=0 to r do
16. train W for Net;
17. end for

通过算法 1,LuffyNet 框架能够搜索出适配目标设备的高精度、低时延网络架构,其工作流程如图 3 所示。然而,推理速度和精度不仅取决于网络架构,还与输入数据的尺寸相关。当数据尺寸变化较小时,可通过预处理使其符合模型要求,避免额外开销。但如果数据格式发生较大变化,为了提高目标网络性能,则需要重新训练超级网络,以生成更适合新环境的网络模型。

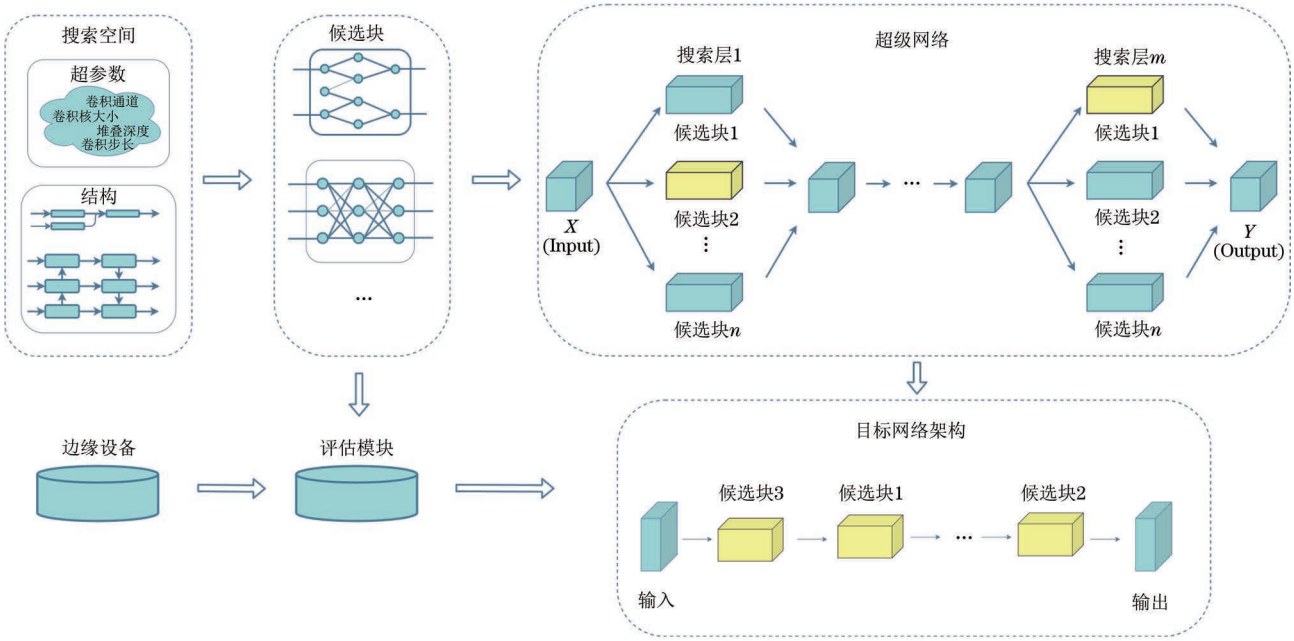

图 3 LuffyNet 框架工作流程

Fig.3 LuffyNet framework workflow procedure

3 实验

3.1 实验设置

本文所有的实验均在一张 RTX 4090 24 GB 上进行,实验平台为 PyTorch 2.0.1,Python 3.10.9。本文选择 ResNet18 作为基准网,在每个训练周期内,使用 80%的数据对权值参数 W 进行训练,并使用剩余的 20%对采样概率参数 θ 进行训练。实验分别在 CIFAR-100 与 ImageNet-100 上进行,其中 ImageNet-100 是从 ImageNet 中随机抽取的 100 类数据构成的子集。

为了保持超级网络中各个候选块的公平性,权

值参数 W 被随机初始化。类似地，零向量被用于初始化采样概率参数 θ。考虑到 W 和 θ 在训练中的差异，本文保持了 FBNet 的实验方法，同样使用 SGD 对 W 进行训练，其初始学习率为 0.025，动量为 0.9，权重衰减设置为 3×10^{-4}。对于采样概率参数 θ，使用 Adam 进行训练，初始学习率为 3×10^{-4}，权重衰减为 10^{3}。

对于采样得到的最终网络架构，本文使用完整的数据集对其训练 200 个周期以确保最终网络架构达到其性能的上限。最终网络架构的权值参数 W 通过 SGD 进行训练，初始学习率为 0.1，并在第 60、120、160 次训练时将学习率衰减为原本的 1/5。

3.2 CIFAR-100 对比实验

本文选取 ResNet18[44]、ResNet50、DenseNet121[47] 与 DenseNet169 模型作为主要对比对象，并按照原文中的方法在 CIFAR-100 上进行训练和优化。

LuffyNet 按算法 1 所示流程，在 CIFAR-100 上展开架构搜索。首先对超级网络训练 20 个周期以完成第 1 阶段。在该阶段，超级网络中每个候选块的权值参数 W 和采样概率参数 θ 都得到了优化和训练。在第 2 阶段中对超级网络进行 10 轮迭代，在每轮迭代中根据 BO 策略和 WO 策略，分别在每层选中表现最好的 2 个块和最差的 2 个块，解冻选中块的权值参数 W，并冻结剩余未选中块的权值参数 W，然后再对超级网络的权值参数 W 和采样概率参数 θ 训练 5 个周期。当阶段 2 结束时，总共对超级网络进行了 70 个周期的训练。由于超级网络的训练需要大量的计算资源和时间，而最终网络架构的训练所需的开销相对较小，因此 LuffyNet 框架仅对超级网络进行 70 轮的训练，并通过 BO 策略与 WO 策略来加速和优化搜索过程。最后，选择每层中采样概率参数 θ 最高的块，构建为新的网络，并对其重新进行训练 200 个周期，得到最终网络架构。此外，本文根据不同的边缘设备，将网络大小阈值 Scale 分别设置为 10、45、90 MB，并将各自搜索得到的网络架构称为 LuffyNet-A、LuffyNet-B 和 LuffyNet-C。

此外，为了进一步探讨超级网络与子网构建方法，实验引入了 OFA 方案中的 OFA_ResNet50 超级网络生成的子网 OFA-ResNet50-C 作为补充对照组，与 LuffyNet 网络一同训练。

3.2.1 超级网络训练过程分析

本文以 Top-1 精度、Top-5 精度、推理时延以及模型大小作为主要评估指标，将 FLOPs 作为次要的度量标准。在架构搜索期间，模型的硬件性能比模型的精度更重要。LuffyNet 框架会对最终网络架构进行重新训练，以达到更高的模型精度。然而网络架构确定后，模型的推理速度和大小就不再变化，因此在超级网络训练期间，需要将重点放在网络架构的推理速度和大小上。

图 4 显示超级网络训练过程中推理时延变化（彩色效果见《计算机工程》官网 HTML 版，下同），随着训练轮次的增加，3 个 LuffyNet 网络的推理时延呈现较大幅度的波动，这反映了推理时延对网络结构的敏感性。图 5 显示了在超级网络训练期间网络大小的变化情况。与图 4 相比，图 5 中网络大小的波动幅度较小，且波动频率较低，整体而言更加稳定，即相对于推理时延，网络大小对网络结构的敏感度更低。通过图 4 和图 5 的观察可以发现，尽管 LuffyNet-B 的网络大小始终小于 LuffyNet-C，但是 LuffyNet-B 的推理时延却存在高于 LuffyNet-C 的情况。这表明了网络的大小与推理速度之间并无必然关系，同时也说明了设置多目标损失函数式(14)的合理性。

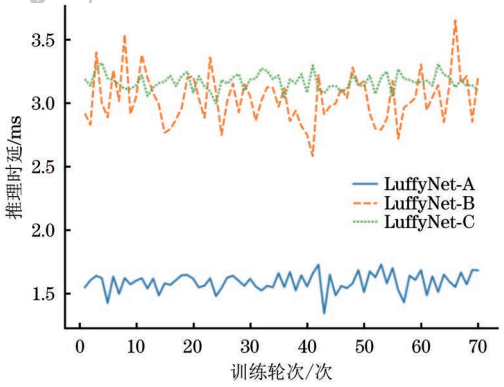

图 4 推理时延变化

Fig. 4 Variation of inference latency

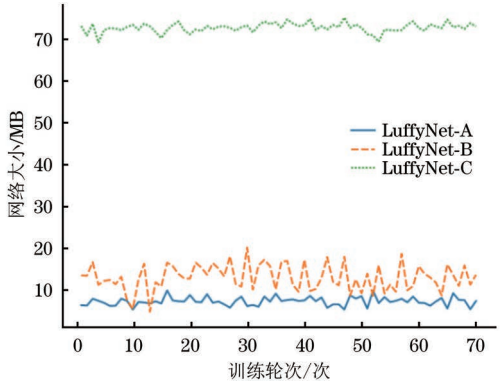

图 5 网络大小变化

Fig. 5 Variation of network size

图 6 显示随着训练次数的增加，网络的精度逐渐提升，但仍然保持在较低水平，因此权值共享方法在 LuffyNet 框架中不可行。此外，结合图 5 可以发

现，LuffyNet-C 的网络结构比 LuffyNet-B 更复杂，然而在超级网络训练期间，前者的精度却低于后者。这些现象可能是由于超级网络的权值参数训练不充分所致。然而，考虑到超级网络训练的高昂成本，对其进行充分训练并不明智。因此，重新训练搜索得到的目标网络架构是必要的。

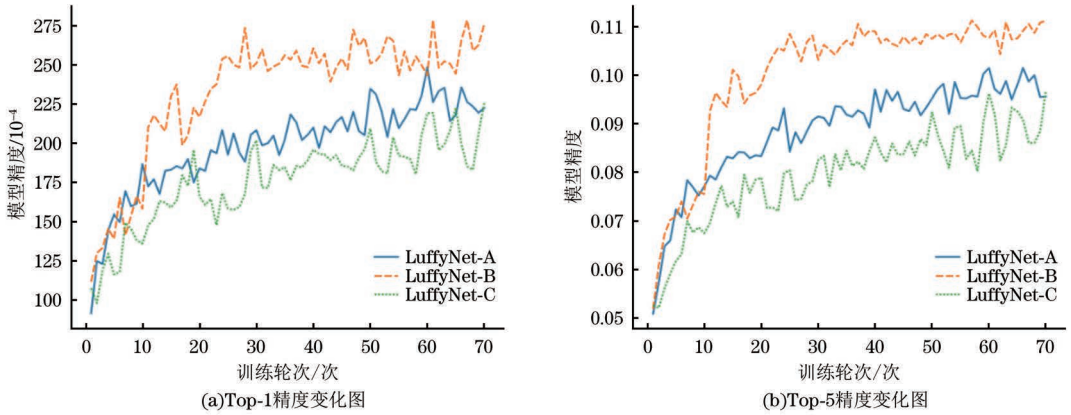

图 6 模型精度变化

Fig. 6 Variation of model accuracy

3.2.2 对比实验结果分析

本文将上述训练后的网络模型部署在相同环境下进行测试，实验结果如表 4 所示。可以观察到推理时延与网络架构大小、FLOPs 之间不存在必然的关联，这与图 4 和图 5 中观察到的现象一致。以 ResNet50 和 DenseNet169 为例，尽管两者的 FLOPs 非常接近，但推理延迟相差近一倍。同样，ResNet18 和 DenseNet169 的模型大小相近，但 ResNet18 的推理时延仅为 DenseNet169 的 1/10，LuffyNet 的结果也呈现相同的趋势。根据表 4，在相近的 Top-1 准确率下，ResNet18 的推理时延仅为 1.92 ms，远低于 ResNet50 的 5.32 ms。这表明在网络架构设计中，可以牺牲一些模型精度来换取推理和部署方面更大的性能优势。反之，如果模型精度是主要关注点，也可在其他性能方面做出一些牺牲以提升模型精度。此外，根据表 4 的数据，可以观察到 3 个 LuffyNet 网络的精度均处于正常水平，这表明对最终的网络架构进行重新训练是可行的。因此，可以有选择地减少对超级网络的训练，从而加速搜索过程。

表 4 CIFAR-100 对比实验结果

Table 4 Comparative experiment results of CIFAR-100

模型	Top-1 精度/%	Top-5 精度/%	推理时延/ms	网络大小/MB	FLOPs
ResNet18	54.69	82.03	1.912	44.592	4.829
ResNet50	55.73	82.94	5.320	97.492	11.055
DenseNet121	68.45	90.34	12.729	30.437	7.696
DenseNet169	69.57	90.63	14.894	53.976	9.184
LuffyNet-A	66.50	88.48	1.697	6.585	14.626
LuffyNet-B	73.44	90.62	2.650	14.603	14.602
LuffyNet-C	75.00	92.97	2.659	72.015	13.264
OFA-ResNet50-C	64.83	83.60	2.990	43.600	13.102

从表 4 中观察到，LuffyNet-A 是所有网络中最小的，仅为 6.585 MB。相较于基准 ResNet18，LuffyNet-A 模型大小仅为 ResNet18 的 1/9，但在 Top-1 精度上提升了 10 百分点以上，且推理速度也比 ResNet18 快了 15%。与其他 3 个代表性模型相比，LuffyNet-A 在牺牲小部分模型精度的情况下，取得了最小的推理时延和网络规模。这表明，在资源受限的边缘计算平台上，LuffyNet-A 更加适合。

从表 4 中可以观察到，LuffyNet-B 在 Top-1 准确率上比 LuffyNet-A 提高了近 7 百分点，但推理时延仅多出 1 ms，且规模仅为 ResNet18 的 1/3，也远小于另外几个代表性模型。也就是 LuffyNet-B 在大大降低网络规模的条件下，在其他指标上也取得了良好的性能。观察 LuffyNet-C 的性能可知，通过增大网络规模，可取得最好的 Top-1 和 Top-5 精度。所以在边缘计算平台，平衡精度、延时和资源需求是决定网络综合性能的关键。

图 7 展示的是 LuffyNet-A、LuffyNet-B 和

LuffyNet-C 的网络架构。现通过对 3 个 LuffyNet 网络进行逐层分析来解释上面的现象。LuffyNet-A 在搜索层 3 和搜索层 4 直接采用了 MBConv，替代了相对复杂的残差块结构，从而在网络架构大小和推理速度方面实现了卓越的性能。然而，这种选择在一定程度上牺牲了 LuffyNet-A 的精度性能。值得注意的是，在搜索层 2 中，LuffyNet-A 采用了 5×5 卷积核的残差结构，进一步提升了网络架构的学习能力，这也解释了为何 LuffyNet-A 在模型精度方面超过了 ResNet18。相较之下，LuffyNet-B 在搜索层 2 和搜索层 3 都采用了更深的残差块堆叠结构，以增强网络架构的学习能力，仅在搜索层 4 时使用了简单的 CBR 结构代替了复杂的残差块结构。LuffyNet-B 虽然在精度上表现较好，但与 LuffyNet-A 相比，在推理速度和网络架构大小方面稍显逊色。LuffyNet-C 则专注于内存资源较为充裕的场景，主要目标是提高模型的精度和推理速度。它直接采用了复杂的多残差块堆叠结构，并添加了许多下采样操作以增强网络架构的学习能力，但整体结构与 LuffyNet-B 较为相似，因此两者的性能表现较为接近。

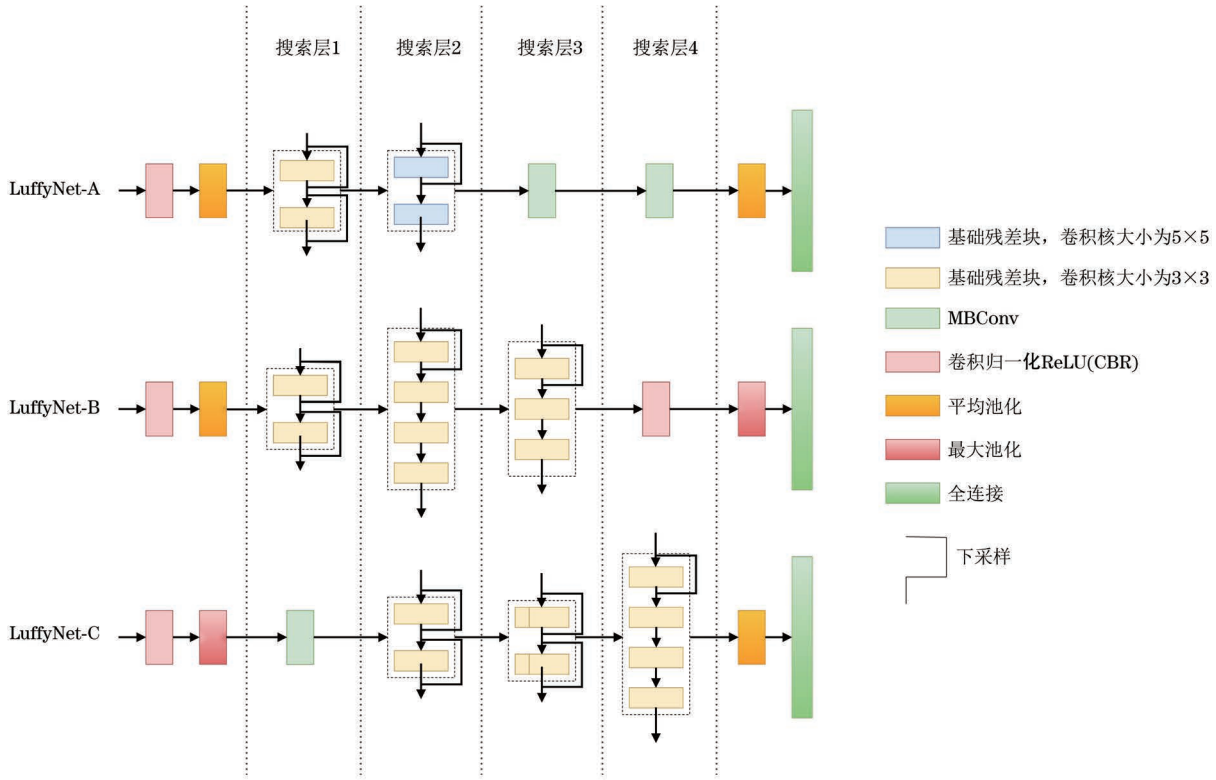

图 7 LuffyNet 架构

Fig. 7 LuffyNet architecture

与 ResNet18、ResNet50、DenseNet121 和 DenseNet169 等由专家手工设计的神经网络不同，OFA-ResNet50-C 是通过架构搜索从超级网络中生成的模型，这类模型的性能受到超级网络的限制。从表 4 中可以看出，LuffyNet-A 与 OFA-ResNet50-C 在模型精度上差距不大，但在推理时延和网络规模上存在显著差异。OFA-ResNet50-C 的推理时延是 LuffyNet-A 的 2 倍，网络规模甚至达到了前者的 7 倍。

OFA 与 LuffyNet 在超级网络构建过程中均对网络架构的超参数进行了搜索，这些超参数对卷积神经网络的精度影响较大。由于两者使用相同的训练方法进行优化，精度表现相近。然而，与 OFA 不同的是，LuffyNet 不仅搜索网络的超参数，还对整体网络结构进行优化。网络结构直接影响推理时延和网络规模，这也是 LuffyNet 在这两项指标上表现更优的原因。考虑到边缘智能场景对推理时延和网络规模的严格要求，LuffyNet 在超级网络构建与子网生成过程中综合考虑了这些性能指标，因此表现出更显著的优势。

总体而言，3 个 LuffyNet 网络满足了各自的工作场景需求。在低内存资源平台上，LuffyNet-A 实现了高精度和高推理速度的卓越平衡。LuffyNet-B 在此基础上进一步提升了模型精度，仅在推理速度和网络架构大小方面稍作牺牲，同时确保网络架构仍然符合其平台资源约束。而在资源充足的情况下，LuffyNet-C 研究了模型精度与推理速度之间的良好平衡，实现了高精度和高推理速度的双赢。与 OFA 相比，LuffyNet 框架的超级网络构建与子网生成方法更符

合边缘智能场景的需求，能够生成在精度、推理速度和网络规模上表现更为优异的网络架构。

上述对比实验充分证明了本文提出的方法能够有效实现搜索精度高、推理速度快且匹配边缘设备的网络架构。

3.3 CIFAR-100 消融实验

本文将 LuffyNet 框架的实验组命名为 LSG（Luffy Search Group），并设立标准对照组 SCG（Standard Control Group）以验证本文提出的搜索流程以及 BO 策略、WO 策略的有效性。SCG 基于均匀采样方式对超级网络中所有候选块的权值参数进行训练，LSG 则基于算法 1，按照 BO 策略与 WO 策略对超级网络进行训练。在其他方面，两个实验组保持一致。实验结果如图 8 和图 9 所示，分别从硬件性能表现和模型精度两个角度进行分析与评价。硬件性能通过推理时延、网络大小和 FLOPs 进行衡量，模型精度则通过 Top-1 和 Top-5 进行评估。所有实验均在 CIFAR-100 数据集上进行。

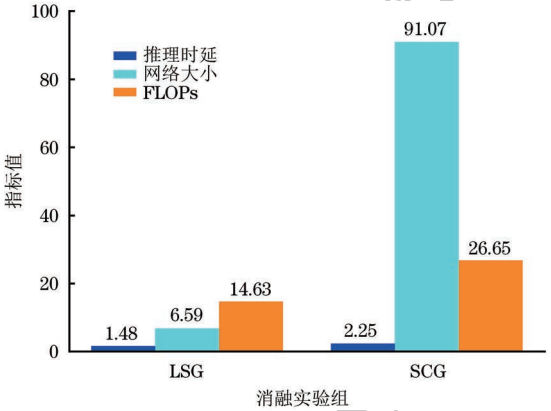

图 8 硬件性能对比

Fig. 8 Comparison of hardware performance

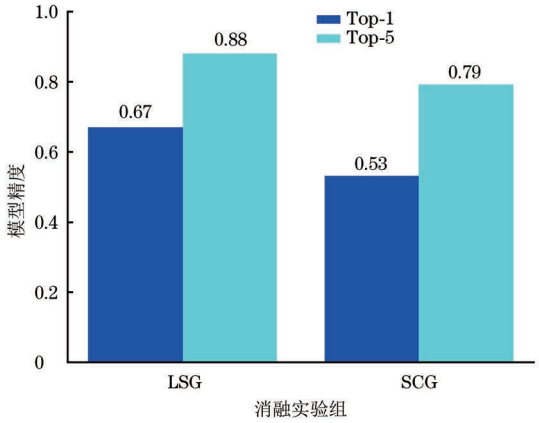

图 9 模型精度对比

Fig. 9 Comparison of model accuracy

通过图 8 可以看到，LSG 推理速度比 SCG 快了 34.2%，但网络大小仅有 6.59 MB，不到 SCG 的 1/10。图 9 显示在模型精度方面，LSG 的 Top-1 精度为 67%，比 SCG 高出 26.4%，Top-5 精度同样是 LSG 的更高。这表明相较于基于均匀策略的 NAS，算法 1 能够有效提升架构搜索的性能表现。与 SCG 相比，LSG 采用 BO 策略，有针对性地对可能构成目标网络架构的高贡献块进行优化，逐步提升了最终网络架构的性能。同时，LSG 还通过 WO 策略，对每个阶段表现较差的块进行优化。这一策略既能够避免潜在的最优块由于表现较差而引发的假阴结果，也能确保在搜索算法出现错误导致假阳时，最终网络架构依然能够表现出优异的性能。值得注意的是，图 8 和图 9 说明，BO 策略与 WO 策略并不会因为只训练部分权值参数而导致搜索结果的恶化。

最后，本文将搜索时长作为指标，对两个实验组的搜索流程进行衡量，结果如表 5 所示。可知，LSG 的搜索时长仅为 61 h，比 SCG 的搜索时长缩短了 25%。正如算法 1 所示，LSG 只需对部分候选块的权值参数进行训练，而权值参数占据了超级网络训练的绝大部分计算开销，因此与 SCG 相比，LSG 减少了大量的计算开销，从而降低搜索的时长。综上所述，消融实验证实了算法 1 不仅能够提高搜索的性能，还能够降低搜索的时间成本。

表 5 架构搜索时间

Table 5 Architecture search time 单位：h

实验组	搜索时长
LSG	61
SCG	80

3.4 ImageNet-100 对比实验

为了验证 LuffyNet 框架在不同数据集上的效果，实验以 OFA 方案中的 OFA_ResNet50 超级网络生成的子网 OFA-ResNet50-I 为对照实验组；同时将 LuffyNet 框架的网络大小阈值 Scale 放宽到 50 MB，以 LuffyNet-A 的超级网络生成的网络 LuffyNet-A-I 为实验组。两个实验组均在 ImageNet-100 上以相同方式进行训练和推理验证，分析指标为模型精度、推理时延和网络大小，结果如表 6 所示。

表 6 不同方案对比实验结果

Table 6 Comparative experimental results of different schemes

模型	Top-1 精度/%	Top-5 精度/%	推理时延/ms	网络大小/MB
OFA-ResNet50-I	52.52	79.90	8.64	183.51
LuffyNet-A-I	51.27	78.37	3.83	44.59

结合表 4 与表 6 可见，LuffyNet 框架与 OFA 框架在 CIFAR-100 与 ImageNet-100 上表现相似。虽然 LuffyNet-A-I 的模型精度略低于 OFA-

ResNet50-I,但其推理速度是 OFA-ResNet50-I 的 2 倍,且大小仅为 OFA-ResNet50-I 的 1/4。相比 CIFAR-100 上的对比实验,网络大小的差距在 ImageNet-100 上有所缩小。一方面,LuffyNet-A-I 在架构搜索期间放宽了对网络大小的限制,另一方面,ImageNet 的尺寸远大于 CIFAR-100,为了平衡精度与推理速度,网络结构引入了诸如 1×1 卷积、残差连接等复杂模块。

这一结果表明,LuffyNet 框架能有效调整网络结构以适应不同场景,确保所搜索的网络能够匹配目标边缘设备。这不仅得益于硬件感知方法,还源于对网络超参数和结构的联合搜索。

综上所述,在边缘智能场景中,LuffyNet 框架相较于 OFA 展现出更大的优势,能够根据不同场景自我调节,确保搜索到的网络适配目标设备。

4 结束语

本文基于硬件感知并结合神经网络架构搜索,提出了面向边缘智能的 LuffyNet 框架。通过硬件感知,将边缘设备的处理器资源与内存资源抽象为推理时延与网络大小,并结合多目标损失函数,搜索适配于边缘设备的高精度、低时延网络架构。此外,LuffyNet 框架采用 BO 策略与 WO 策略,进一步提升了架构搜索的性能,同时降低了搜索所需的时长与计算开销。对比实验结果验证了 LuffyNet 框架能够找到与边缘设备匹配的高精度、低时延的网络模型。消融实验进一步验证了本文提出的算法在提升架构搜索性能的同时,减少了大量无效计算,从而降低了搜索的时间成本。下一步将研究轻量化搜索方法,以减少框架的计算资源需求和时间开销。同时,结合代理模型技术进一步优化搜索算法和策略,提升在实际工程中的应用价值。

参考文献

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

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

[3] 谭郁松, 李恬, 张钰森. 面向边缘智能的神经网络模型生成与部署研究[J]. 计算机工程, 2024, 50(8): 1-12.
TAN Y S, LI T, ZHANG Y S. Research on generation and deployment of neural network model for edge intelligence[J]. Computer Engineering, 2024, 50(8): 1-12. (in Chinese)

[4] MACH P, BECVAR Z. Mobile edge computing: a survey on architecture and computation offloading [J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1628-1656.

[5] CHUN B G, IHM S, MANIATIS P, et al. CloneCloud: elastic execution between mobile device and cloud[C]// Proceedings of the 6th Conference on Computer Systems. New York, USA: ACM Press, 2011: 301-314.

[6] 宋艳蕊, 庄雷, 徐泽汐, 等. 基于云边协同的可靠服务功能链部署算法[J]. 计算机工程, 2024, 50(12): 184-193.
SONG Y R, ZHUANG L, XU Z X, et al. Reliable service function chain deployment algorithm based on edge-cloud collaboration[J]. Computer Engineering, 2024, 50(12): 184-193. (in Chinese)

[7] HAN S, MAO H, DALLY W J. Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding[EB/OL]. [2024-06-20]. https://arxiv.org/abs/1510.00149.

[8] 魏铭康, 李嘉楠, 韩林, 等. 面向深度学习编译器的多粒度量化框架支持与优化[J]. 计算机工程, 2025, 51(5): 62-72.
WEI M K, LI J N, HAN L, et al. Support and optimization of multi-granularity quantization framework for deep learning compiler[J]. Computer Engineering, 2025, 51(5): 62-72. (in Chinese)

[9] CHEN P Y, LIN H C, GUO J I. Multi-scale dynamic fixed-point quantization and training for deep neural networks[C]// Proceedings of the IEEE International Symposium on Circuits and Systems. Washington D. C., USA: IEEE Press, 2023: 1-5.

[10] RASTEGARI M, ORDONEZ V, REDMON J, et al. XNOR-Net: ImageNet classification using binary convolutional neural networks [C] // Proceedings of the European Conference on Computer Vision. Berlin, Germany: Springer, 2016: 525-542.

[11] VARDAR A, ZHANG L, HU S S, et al. Layer sensitivity aware CNN quantization for resource constrained edge devices[C]//Proceedings of the 9th International Conference on Soft Computing & Machine Intelligence. New York, USA: ACM Press, 2022: 26-30.

[12] AKPOLAT M Z, BULBUL A. A global approach for goal-driven pruning of object recognition networks[C]//Proceedings of the 30th Signal Processing and Communications Applications Conference. Washington D. C., USA: IEEE Press, 2022: 1-4.

[13] WANG Z D, LIU X X, HUANG L, et al. QSFM: model pruning based on quantified similarity between feature maps for AI on edge[J]. IEEE Internet of Things Journal, 2022, 9(23): 24506-24515.

[14] CHEN Y L, CHEN J, WANG Y, et al. A model compression method for power edge intelligent inspection via channel pruning[C]//Proceedings of the 3rd Power System and Green Energy Conference. New York, USA: ACM Press, 2023: 1169-1173.

[15] AKHTER S, HOSSAIN M I, HOSSAIN M D, et al. NeuRes: highly activated neurons responses transfer via distilling sparse activation maps[J]. IEEE Access, 2022, 10: 131555-131566.

[16] 林烁彬, 蔡捷仪, 方晓城, 等. 基于强度相关正则化学习的对抗鲁棒蒸馏方法[J]. 计算机工程, 2025, 51(1): 42-50.
LIN S B, CAI J Y, FANG X C, et al. Adversarial robust distillation method based on intensity correlation regularization learning [J]. Computer Engineering, 2025, 51(1): 42-50. (in Chinese)

[17] RISSO M, BURRELLO A, CONTI F, et al. Lightweight neural architecture search for temporal convolutional networks at the edge[J]. IEEE Transactions on Computers, 2022, 72(3): 744-758.

[18] MA N N, ZHANG X Y, ZHENG H T, et al. ShuffleNet V2: practical guidelines for efficient CNN architecture design[C]// Proceedings of the European Conference on Computer Vision. Berlin, Germany: Springer, 2018: 122-138.

[19] ZOPH B, LE Q. Neural architecture search with reinforcement learning[EB/OL]. [2024-06-20]. https://openreview.net/forum?id=r1Ue8Hcxg.

[20] TAN M X, CHEN B, PANG R M, et al. MnasNet: platform-aware neural architecture search for mobile[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Washington D. C., USA: IEEE Press, 2019: 2820-2828.
[21] DAI X L, ZHANG P Z, WU B C, et al. ChamNet: towards efficient network design through platform-aware model adaptation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Washington D. C., USA: IEEE Press, 2019: 11398-11407.
[22] CAI H, ZHU L G, HAN S. ProxylessNAS: direct neural architecture search on target task and hardware[EB/OL]. [2024-06-20]. https://arxiv.org/abs/1812.00332.
[23] WU B C, KEUTZER K, DAI X L, et al. FBNet: hardware-aware efficient ConvNet design via differentiable neural architecture search[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Washington D. C., USA: IEEE Press, 2019: 10734-10742.
[24] PHAM H, GUAN M, ZOPH B, et al. Efficient neural architecture search via parameters sharing[C]//Proceedings of the International Conference on Machine Learning. New York, USA: ACM Press, 2018: 4095-4104.
[25] SANDLER M, HOWARD A, ZHU M L, et al. MobileNetV2: inverted residuals and linear bottlenecks[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Washington D. C., USA: IEEE Press, 2018: 4510-4520.
[26] ZOPH B, VASUDEVAN V, SHLENS J, et al. Learning transferable architectures for scalable image recognition[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Washington D. C., USA: IEEE Press, 2018: 8697-8710.
[27] ESHELMAN L J. Genetic algorithms[M]. New York, USA: CRC Press, 2018.
[28] HU X L, HUANG Z, WANG Z F. Hybridization of the multi-objective evolutionary algorithms and the gradient-based algorithms[C]//Proceedings of the 2003 Congress on Evolutionary Computation. Washington D. C., USA: IEEE Press, 2003: 870-877.
[29] JANG E, GU S, POOLE B. Categorical reparameterization with gumbel-softmax[EB/OL]. [2024-06-20]. https://arxiv.org/abs/1611.01144.
[30] XIE S, ZHENG H, LIU C, et al. SNAS: stochastic neural architecture search[EB/OL]. [2024-06-20]. https://arxiv.org/abs/1812.09926.
[31] GIBBS M, KANJO E. Realising the power of edge intelligence: addressing the challenges in AI and tinyML applications for edge computing[C]//Proceedings of the IEEE International Conference on Edge Computing and Communications. Chicago, USA: IEEE Press, 2023: 337-343.
[32] HADIDI R, CAO J S, RYOO M S, et al. Reducing inference latency with concurrent architectures for image recognition at edge[C]//Proceedings of the IEEE International Conference on Edge Computing and Communications. Chicago, USA: IEEE Press, 2023: 245-254.
[33] IANDOLA F N, HAN S, MOSKEWICZ M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size[EB/OL]. [2024-06-20]. https://arxiv.org/abs/1602.07360.
[34] YAN Z Y, LI X M, LI M, et al. Shift-net: image inpainting via deep feature rearrangement[C]// Proceedings of the European Conference on Computer Vision and Pattern Recognition. Berlin, Germany: Springer, 2018: 3-19.
[35] LIU H, SIMONYAN K, YANG Y. Darts: differentiable architecture search[EB/OL]. [2024-06-20]. https://arxiv.org/abs/1806.09055.
[36] ZHONG Z, YAN J J, WU W, et al. Practical block-wise neural network architecture generation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Washington D. C., USA: IEEE Press, 2018: 2423-2432.
[37] XUE S, ZHAO B, CHEN H L, et al. UCB-ENAS based on reinforcement learning[C]//Proceedings of the 16th IEEE Conference on Industrial Electronics and Applications. Washington D. C., USA: IEEE Press, 2021: 2008-2013.
[38] WANG D L, LI M, GONG C Y, et al. AttentiveNAS: improving neural architecture search via attentive sampling[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Nashville, USA: IEEE Press, 2021: 6418-6427.
[39] ZHAO S X, QU S Y, WANG Y, et al. ENASA: towards edge neural architecture search based on CIM acceleration[C]// Proceedings of the 2023 Design, Automation & Test in European Conference & Exhibition. Berlin, Germany: Springer, 2023: 321-332.
[40] XU K P, HE G. DNAS: a decoupled global neural architecture search method[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. New Orleans, USA: IEEE Press, 2022: 1979-1985.
[41] CAI H, GAN C, HAN S. Once for all: train one network and specialize it for efficient deployment[EB/OL]. [2024-06-20]. https://arxiv.org/abs/1908.09791.
[42] LIU C X, ZOPH B, NEUMANN M, et al. Progressive neural architecture search[C]//Proceedings of the European Conference on Computer Vision and Pattern Recognition. Berlin, Germany: Springer, 2018: 19-35.
[43] STAMOULIS D, DING R Z, WANG D, et al. Single-path NAS: designing hardware-efficient ConvNets in less than 4 hours[C]//Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Berlin, Germany: Springer, 2020: 481-497.
[44] STAMOULIS D, DING R Z, WANG D, et al. Single-path mobile AutoML: efficient ConvNet design and NAS hyperparameter optimization[J]. IEEE Journal of Selected Topics in Signal Processing, 2020, 14(4): 609-622.
[45] GUO Z C, ZHANG X Y, MU H Y, et al. Single path one-shot neural architecture search with uniform sampling[C]// Proceedings of the 16th European Conference on Computer Vision. Berlin, Germany: Springer, 2020: 544-560.
[46] HE K M, ZHANG X Y, REN S Q, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, USA: IEEE Press, 2016: 770-778.
[47] CHOLLET F. Xception: deep learning with depthwise separable convolutions[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Honolulu, USA: IEEE Press, 2017: 1251-1258.
[48] SZEGEDY C, LIU W, JIA Y Q, et al. Going deeper with convolutions[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Boston, USA: IEEE Press, 2015: 1-9.
[49] HUANG G, LIU Z, VAN DER MAATEN L, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Washington D. C., USA: IEEE Press, 2017: 4700-4708.

文字编辑 索书志
栏目编辑 赖玉玲