中国科学: 信息科学 2023 年 第 53 卷 第 5 期: 899–917
SCIENTIA SINICA Informationis
论文

硬件感知的神经架构搜索

王鑫[1], 姚洋[1], 蒋昱航[2], 关超宇[1], 朱文武[1*]

1. 清华大学计算机科学与技术系, 北京 100084
2. 清华大学清华 - 伯克利深圳学院, 深圳 518055
* 通信作者. E-mail: wwzhu@tsinghua.edu.cn

收稿日期: 2021–12–31; 修回日期: 2022–04–22; 接受日期: 2022–06–10; 网络出版日期: 2023–05–05

科技创新 2030—"新一代人工智能" 重大项目 (批准号: 2020AAA0106300)、国家优秀青年科学基金 (批准号: 62222209) 和国家自然科学基金青年基金 (批准号: 62102222) 资助项目

摘要 深度神经网络 (deep neural networks, DNNs) 能否取得令人满意的性能很大程度上依赖于其神经网络架构. 研究人员提出神经网络架构搜索 (neural architecture search, NAS) 来自动搜索神经网络的最优架构, 现有的工作大多使用每秒浮点运算次数 (floating point operations per second, FLOPs) 来评价神经网络架构的实际效率, 但是 FLOPs 和实际延迟并不是完全一致的. 随着任务变得越来越复杂以及越来越多的硬件平台开始运行基于深度神经网络的算法, 为硬件平台搜索高效的神经网络架构已成为亟待解决的难题. 为了解决这一问题, 本文提出了硬件感知的搜索空间构造方法, 并借助考虑架构推断延迟的搜索策略, 来搜索最优的神经网络架构. 本文在可变换神经网络架构搜索方法 (transformable architecture search, TAS) 和图神经网络架构搜索方法 (graph neural architecture search, GraphNAS) 上应用了该方法, 提出了硬件可感知的可变换神经网络架构搜索方法 (hardware-aware transformable architecture search, HTAS) 和硬件感知的图神经网络架构搜索方法 (hardware-aware graph neural architecture search, HGNAS). 相比于现有方法, 本文所提出的这两种算法在多种数据集上均针对不同类型目标硬件搜索出了更加高效的深度神经网络架构, 从而证明了该方法的有效性.

关键词 深度学习, 神经网络架构搜索, 可变换神经网络架构搜索, 图神经网络架构搜索, 硬件感知

1 介绍

深度神经网络 (deep neural networks, DNNs) 已广泛应用于图像分类、目标识别等领域, 并且在改善模型性能上取得了很好的效果. 神经网络的架构对其性能有着重要的影响, 而到目前为止它的架构主要是人们基于专业知识来手动设计的. 因此, 自动搜索使模型性能最优的架构对深度学习算法来说是很重要的. 然而, 深度神经网络的的设计空间是多种可选择参数组合起来的巨大空间, 这使得人

引用格式: 王鑫, 姚洋, 蒋昱航, 等. 硬件感知的神经架构搜索. 中国科学: 信息科学, 2023, 53: 899–917, doi: 10.1360/SSI-2021-0446
Wang X, Yao Y, Jiang Y H, et al. Hardware-aware neural architecture search (in Chinese). Sci Sin Inform, 2023, 53: 899–917, doi: 10.1360/SSI-2021-0446

工从大量的候选架构中选出最优的架构是不具可行性的. 神经架构搜索 (neural architecture search, NAS) [1~4] 就是人们为了自动搜索出神经网络的最优架构而提出的方法. 同时, 将深度神经网络应用于大量不同的硬件平台的需求与日俱增, 这对寻找不同硬件平台上的最优神经架构提出了迫切的挑战.

此外, 在实际应用中, 由于不同类型的硬件设备的特性不同, 在它们上面运行的模型的实际效率 (如推断延迟) 也会有很大的差异. 随着当前越来越多种类的硬件被用于运行神经网络, 学术界和工业界都迫切需要研究更具挑战性的硬件感知的神经架构搜索问题. 然而, 现有的神经架构搜索方面的研究大多采用硬件无关的度量标准, 例如, 每秒浮点运算次数 (floating point operations per second, FLOPs) 来评价模型效率. 这种做法忽略了硬件无关的度量标准和在真实硬件上测量出的实际效率之间的不一致. 例如, 更低的 FLOPs 并不意味着更低的推断延迟或能量消耗. 其他的关于硬件感知的神经架构搜索的工作 [5] 主要通过改进搜索策略来降低推断延迟, 它们在损失函数中加入了硬件感知的度量标准 (如平均推断延迟). 然而, 这种做法忽略了搜索空间对架构搜索过程的实际效率的重要影响 [1,6].

为了应对这些挑战, 本文研究了硬件感知的神经架构搜索方法, 并通过考虑硬件感知的搜索空间来搜索最优的网络架构. 本文提出了一种硬件感知的神经架构搜索方法来找出不同硬件平台的最优架构. 更具体地来说, 对于 M 维搜索空间, 我们从中选择 N 维硬件敏感的子空间, 然后使用合适的函数去拟合该子空间与神经网络架构在目标硬件上推断延迟的关系, 并结合拟合结果与实际推断延迟从 N 维子空间中选择出在目标硬件上性能高效的子集, 构造硬件感知的搜索空间. 最后, 我们使用加入推断延迟的搜索策略来寻找最优神经网络架构. 第 3.2 小节以 HTAS 为例讨论了 $N = 1$ 的情况, 第 4.1 小节以 HGNAS 为例讨论了 $N = 2$ 的情况, 对于 $N > 2$ 的情况, 我们的方法同样适用.

本文在可变换神经网络架构搜索方法 (transformable architecture search, TAS) 和图神经网络架构搜索方法 (graph neural architecture search, GraphNAS) 上应用了上述方法, 提出了硬件感知的可变换神经网络架构搜索方法 (hardware-aware transformable architecture search, HTAS) 和硬件感知的图神经网络架构搜索方法 (hardware-aware graph neural architecture search, HGNAS). 我们在 CIFAR-10 和 CIFAR-100 数据集上评估了本文提出的方法 HTAS, 并且进行实验来证明 HTAS 能找出比 TAS 更高效, 且准确率相近或者更高的网络架构. 对 CIFAR-100 数据集上的 ResNet110, 我们的方法在 GPU 和 CPU 上都比 TAS [7] 快 2.2 倍, 并且准确率提高了 1.3% 以上. 对 CIFAR-10 数据集上的 ResNet110, HTAS 也快了 1.9 倍, 并且准确率更高. 同样地, 我们在 Cora, Citeseer 以及 Pubmed 数据集上评估了本文提出的方法 HGNAS, 它在 GPU 和 CPU 上相比 GraphNAS 找出的架构都更为高效.

本文的贡献总结如下:

- 提出了一种硬件感知的神经架构搜索方法, 该方法构建了一个硬件感知的搜索空间, 并且使用加入了推断延迟的搜索策略, 在不需要硬件详细信息的情况下即可为目标硬件找出最高效的神经网络架构.
- 提出了硬件感知的可变换神经网络架构搜索方法 (HTAS). 在两个数据集上的实验表明, HTAS 在不同的硬件上的表现都比 TAS 要好, 这证明了我们提出的 HTAS 方法的有效性. 此外, 我们对在不同的硬件上搜索出的模型进行了比较, 其结果说明了设计适合不同目标硬件的神经网络的必要性.
- 提出了硬件感知的图神经网络架构搜索方法 (HGNAS). 在 3 个数据集上的实验表明, HGNAS 在不同硬件上的表现都比 GraphNAS 要好, 这同样证明了 HGNAS 方法的有效性.

本文的其他部分组织如下, 第 2 节回顾了相关工作, 第 3 节详细介绍了本文提出的 HTAS 模型, 第 4 节详细介绍了本文提出的 HGNAS 模型, 第 5 节展示了我们进行的大量的实验. 最后, 第 6 节对全文进行了总结, 并对未来的研究方向进行了展望.

2 相关工作

设计高效的神经网络是一个活跃的研究领域. 本部分对相关工作进行了总结.

2.1 高效的神经网络

深度神经网络在机器翻译、图像和语音识别、强化学习等领域有着广泛的应用. 深度学习在感知任务上取得成功很大程度上应该归功于它的自动化特征工程: 深度学习使用端到端的方式从数据中学习分级特征提取器, 而不是使用手动设计的特征. 然而, 伴随着这一成功, 人们对架构工程的需求也在不断增长, 这意味着需要手动设计的神经架构越来越复杂.

此外, 由于需要将神经架构部署到资源受限的硬件上, 人们迫切地需要高效的架构. 因此, 涌现了许多高效的 ConvNet 模型, 其中 SqueezeNet [8] 是早期致力于减小 ConvNet 模型参数数量的工作之一. 紧随 SqueezeNet 之后, SqueezeNext [9] 和 ShiftNet [10] 进一步实现了参数数量缩减. 最近的工作将关注点从参数数量转移到了 FLOPs. MobileNetV1 [11] 和 MobileNetV2 [12] 使用深度卷积来代替更昂贵的空间卷积. ShuffleNet [13] 使用分组卷积和通道重组操作来进一步减少 FLOP 数. 但之后研究发现 FLOP 数并不总是反映实际的硬件效率. 为了降低实际延迟, ShuffleNet V2 [14] 提出了一系列的设计高效的 ConvNet 的实用准则. 然而, 手动设计架构是一个乏味的过程, 而且需要多年的经验. 对深度学习领域的新手来说, 架构设计通常不太直观, 从而难以手动进行.

2.2 通道剪枝

通道剪枝研究的是通过剪枝滤波器来减小神经网络宽度的问题. 大多数的通道剪枝方法 [15~18] 都使用启发式的重要性度量标准来决定应该修剪哪些通道. 例如, Liu 等 [15] 使用批量归一化层的比例因子, Luo 等 [16] 提出了使用下一层的输入特征的 ThiNet. 然而, 人类设计的规则仍然存在产生次优架构的风险. 本文提出的方法使用显式学习最优宽度和高度的搜索方法来找到高效的神经网络架构.

2.3 神经架构搜索

良好的神经网络架构设计可以大大提高神经网络的性能. 然而, 从大量的候选架构中找出最优架构对人类而言过于困难, 这导致了神经架构搜索 (NAS) [1~4] 的兴起. NAS 能自动搜索具有最佳架构的神经网络. NAS 的方法可以根据 3 个维度进行分类: 搜索空间、优化方法和性能评估策略. 搜索空间定义了理论上可以表示的架构. 性能评估是指评估模型性能的过程, 通常来说 NAS 的目标是找到在新的数据上能得到良好的预测结果的架构. 神经架构搜索 (NAS) [1~4,19,20] 主要是搜索神经网络的拓扑结构. 一些已有的工作 [19,20] 提出使用进化算法来搜索神经架构, 然而一些其他的工作 [2,21] 则使用强化学习. 特别地, NASNet [22] 首先提出使用强化学习来搜索具有低 FLOPs 的架构. 这些方法通常需要付出很高的计算代价来进行搜索, 从而激发了使用可微方法来探索搜索空间的想法, 这样就能大大降低搜索代价 [1,3,4].

最近的方法 [3,6,19,23] 采用了固定宏观架构下的层级分级搜索空间, 它允许在网络的不同分辨率的块中使用不同的层架构. 在这些方法中, 神经架构搜索的目标变成了搜索每一层的算子, 从而使得神经架构在给定的约束条件下能达到较高的准确率. 为了评估模型的高效程度, 许多 NAS 方法 [22] 使用了硬件无关的度量标准 FLOPs. 然而, 具有较低 FLOPs 的架构并不一定更快 [24]. 近来, 基于梯度的方法 [3,4,25] 采用了直接的度量标准, 比如为移动 CPU 测量的延迟. 这些方法测量出每个算子的延迟然后建立预测模型, 并把延迟视为可微正则化损失加入搜索过程中. 然而, 正如 Howard 等 [26] 指出

的那样, 多目标损失函数仍有不足之处, 因为对小模型来说延迟的变化会导致准确率的显著变化. 此外, 一些算法也引入了硬件感知的约束 [27] 来寻找特定硬件上的高效的架构.

2.4 可变换神经网络架构搜索

除了拓扑结构外, 宽度和深度对神经网络的性能也有重要的影响. 因此, 自动寻找神经网络的最优宽度和深度也是一个十分值得研究的领域. 可变换神经网络架构搜索 (TAS) [5, 7, 28, 29] 通过调整网络的宽度和深度来取得更好的性能. TAS [7] 首先通过应用可微 NAS 来降低模型的宽度和深度, 从而获得更为精简的模型, 然后对搜索出的架构进行知识迁移. MorphNet [28] 通过交替地压缩和扩张神经网络来找出给定资源限制下的最优宽度. 然而, 这些工作都使用了 FLOPs 等间接度量标准来指导搜索过程, 而这些度量标准和真实硬件上的实际性能是不直接相关的. NetAdapt [5] 通过在损失函数中加入硬件感知的资源限制来使得搜索出的网络能适应不同的硬件. 这些方法大多着眼于通过改进搜索策略来找到在给定资源限制下的高效的网络, 然而对搜索空间并没有投以较多关注. 和这些工作相比, 本文更加关注搜索空间的设计. 本文构建了一个硬件感知的搜索空间, 它在为特定硬件搜索高效架构的过程中能够提供更好的选择.

2.5 图神经网络架构搜索

图神经网络的架构搜索近年来引起了越来越多的研究 [30~40]. 其中, GraphNAS [41] 首次使用强化学习的方法来搜索图神经网络架构, 它综合了以往图神经网络方面的优秀的架构作为搜索空间, 如 GCN, GAT 等, 并把搜索空间分为了以下 6 个部分: 邻居采样算子, 信息计算函数, 聚合函数, 激活函数, 多头机制 (multi-head mechanism) 的头数量, 层的输出维度, 其中邻居采样算子默认为 1 层. Auto-GNN [42] 在图神经网络架构的搜索过程中引入了参数共享机制来更加稳定地训练架构. SNAG [43] 为图神经网络架构搜索设计了新的基于节点操作和层操作的搜索空间. DSS [44] 提出了一次性 (one-shot) 图神经网络架构搜索方法, 即共享最优几种架构的参数. GNAS [45] 设计了一种树状的三层搜索空间, 并使用可微搜索策略来搜索图神经网络的最佳架构. 与已有工作不同的是, 本文致力于为图神经网络搜索出高效的架构.

3 HTAS: 硬件感知的可变换神经网络架构搜索

本节将介绍我们提出的硬件感知的可变换神经网络架构搜索 (HTAS) 方法. HTAS 由图 1 所示的 3 个主要步骤组成. 首先, 我们扩大了全局搜索空间. 然后, 我们在其中选择那些硬件友好的通道数量组成硬件感知的搜索空间. 最后, 我们使用加入了延迟限制的可微搜索策略来搜索神经网络的最优宽度和深度.

3.1 扩大全局搜索空间

全局搜索空间包含搜索可能的输出通道数量的候选值. 已有的可变换神经网络架构搜索方法 [7, 46] 使用的是大小被基础模型限制的搜索空间, 候选网络的每一层都不能比基础模型的相应层更宽. 然而, 尽管一个更高效的模型通常具有更小的尺寸, 但它的某些层的宽度可能会比基础模型的更宽. 可是在原来限制下的搜索空间中我们不能找出这样的模型, 这可能导致我们找出的是次优架构. 因此, 我们扩大了全局搜索空间, 以便可能找到这些更优的神经架构. 受 MorphNet [28] 启发, 我们将基础网络的宽度扩大到 e 倍作为全局搜索空间的上界, 使得全局搜索空间更大, 更灵活. 表 1 列举了在 CIFAR 数

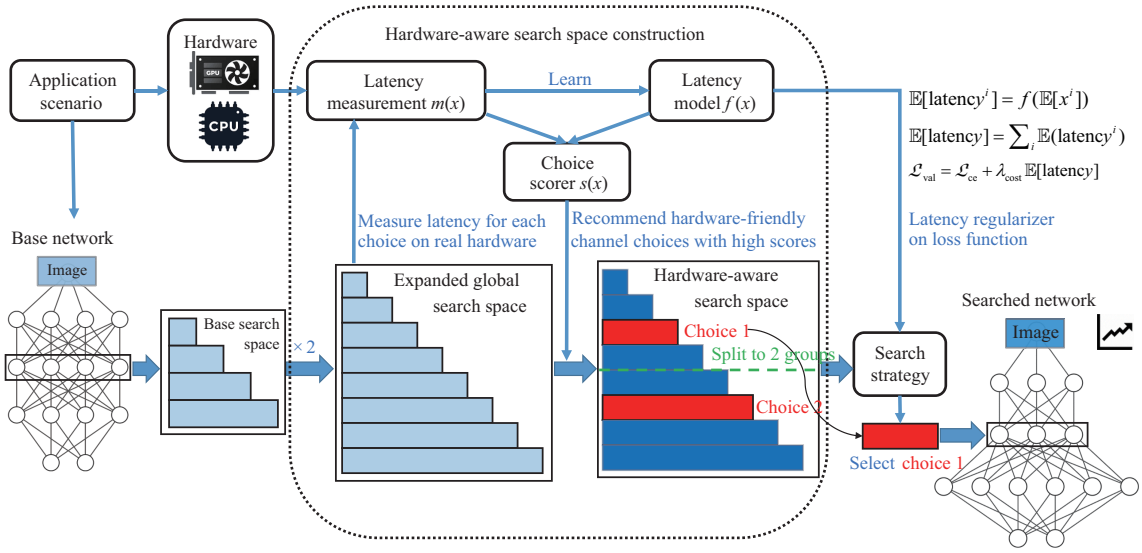

图 1 (网络版彩图) HTAS 的框架, 用于搜索神经网络的宽度和深度. 我们首先扩大全局搜索空间到 $e = 2$ 倍, 然后根据测量出的通道数的延迟, 选出硬件友好的通道数来构建硬件可感知的搜索空间. 最终的架构由加入了延迟正则项的搜索策略给出. 图中虚线框中的部分为本文提出的硬件感知的搜索空间的构建过程

Figure 1 (Color online) The framework of hardware-aware TAS searching for width and depth of a neural network. We first expand the global search space by $e = 2$ times. Then based on the latency measurements over the channel numbers, the choice scorer recommends the hardware-friendly channel choices to construct the hardware-aware search space. The final architecture is chosen by the search strategy with the latency regularizer. The part in the dotted box in the figure is the construction of hardware-aware search space proposed in this paper

表 1 CIFAR 数据集上 ResNets 的扩大后的全局搜索空间. "Output size" 和 "Width" 表示层的输出特征的分辨率和通道数

Table 1 Global search space expansion for ResNets on CIFAR datasets. "Output size" and "Width" denote the resolution and number of output feature maps in the layer

Stage	Output size	Width	Global search space
1	32×32	16	$\{n \mid 1 \leqslant n \leqslant 16 \times e\}$
2	16×16	32	$\{n \mid 1 \leqslant n \leqslant 32 \times e\}$
3	8×8	64	$\{n \mid 1 \leqslant n \leqslant 64 \times e\}$

据集上 ResNets [47] 的不同阶段的扩大后的全局搜索空间. 例如, 我们设 e 为 2, 则 3 个阶段的全局搜索空间应分别为 $1 \sim 32$, 64 和 128.

3.2 构造硬件感知的搜索空间

由于全局搜索空间通常很大, 搜索策略使用的搜索空间仅由从全局搜索空间中选定的一部分组成. 已有的 TAS 方法 [7] 的搜索空间大多由一组等差比率和基础模型的层宽相乘得到. 例如, Resnet32 第一阶段的通道数量是 16, 比率可以从 $(0.25, 0.50, 0.75, 1.0)$ 中选择, 那么它的搜索空间应为 $(4, 8, 12, 16)$. 如果有 N 个通道数量, 最高 L 层可供搜索, 那么搜索空间的大小为 $O(N^L)$.

然而, 这种搜索空间是平凡而且硬件无关的, 它忽略了在真实硬件上不同通道数量的实际高效程度的差异. 更具体地说, 使用 FLOPs 作为计算代价的度量标准来估计实际的推理延迟是一种常见的做法, 但是某些通道数量可能比它们的 FLOPs 体现得要更加高效, 延迟更低. 此外, 由于硬件设计的

多样性, 不同硬件上的最高效的通道数量通常是不同的. 受此启发, 我们选择不同硬件上的高效的通道数量来构建硬件感知的搜索空间.

这里我们使用实际延迟和估计延迟之间的差作为评价通道数量高效程度的指标. 为了得到估计延迟, 我们测量了全局搜索空间中不同通道数量的推断延迟, 然后建立一个延迟模型 $f(x)$ 用于学习实际延迟和通道数量之间的关系.

我们使用以下模型来通过输出通道数量估计推断延迟:

$$f(x) = kx^{\alpha} + b, \tag{1}$$

其中 x 是输出通道数量, 参数 α 是根据任务设置的常量. 图 2 展示了当输出特征维度为 32×32 并把 α 设为 1 时神经网络在 CPU 上的推断延迟和估计延迟随输出通道数量变化而变化的折线图.

借助延迟模型, 我们可以通过评估输出通道数量的高效程度, 即它的实际延迟和估计延迟之间的差, 来对输出通道数量进行评分. 评分函数 $s(x)$ 如下:

$$s(x) = f(x) - m(x), \tag{2}$$

其中 $m(x)$ 为真实硬件上测量出的实际推断延迟.

不同任务的复杂程度通常是不一样的, 因此它们对模型的学习能力也提出了不同的要求. 例如, 适用于 ImageNet 的模型通常比适用于 CIFAR-10 的模型需要更强的学习能力. 另一方面, 在输入通道数量固定的情况下, 神经网络中某一层的学习能力通常随着输出通道数量的增长而增长. 为了适应不同的任务, 我们将每一层中的全局搜索空间分为 g 组, 每一组具有不同学习能力的宽度值. 在每一组中, 给分函数给分最高的宽度值就是最高效的通道数量.

$$\text{choice}_j = \arg\max_{x \in X_j} s(x), \tag{3}$$

其中 choice_j 是全局搜索空间 X 的第 j 组选出的最高效的通道数量. 图 3 展示了不同输出通道数量和其中被选出的最高效的通道数量的得分.

最后, 我们将这些被选出的最高效的通道数量组合为硬件感知的搜索空间 HS.

$$\text{HS} := \cup_{j=1}^{g} \{\text{choice}_j\}, \tag{4}$$

表 2 列举了 CPU 和 GPU 上 ResNets 的每个阶段的硬件感知的搜索空间.

3.3 搜索策略

我们应用文献 [7] 中的可微搜索策略来探索 3.2 小节中的硬件感知的搜索空间. 可变换神经网络架构搜索的目的是找到具有最优宽度和深度的神经网络架构 $\mathcal{A}$. 在这里最优指的是当神经网络在训练集上进行优化后能使得在测试集上的损失函数 $\mathcal{L}_{\text{val}}$ 达到最小. 上述问题可以表述为

$$\min_{\mathcal{A}} \mathcal{L}_{\text{val}}(\omega^*_{\mathcal{A}}, \mathcal{A}) \quad \text{s.t.} \quad \omega^*_{\mathcal{A}} = \arg\min_{\omega} \mathcal{L}_{\text{train}}(\omega, \mathcal{A}), \tag{5}$$

其中 $\omega^*_{\mathcal{A}}$ 表示 $\mathcal{A}$ 的优化后的权值. 训练损失 $\mathcal{L}_{\text{train}}$ 是网络的交叉熵分类损失. $\mathcal{L}_{\text{val}}$ 由交叉熵损失 $\mathcal{L}_{\text{ce}}$ 和计算代价损失两部分组成. 在本文中后者使用的是网络的期望延迟 $\mathbb{E}[\text{latency}]$, 不过我们也可以使用能量等其他度量标准来评估计算代价.

$$\mathcal{L}_{\text{val}} = \mathcal{L}_{\text{ce}} + \lambda_{\text{cost}} \mathbb{E}[\text{latency}]. \tag{6}$$

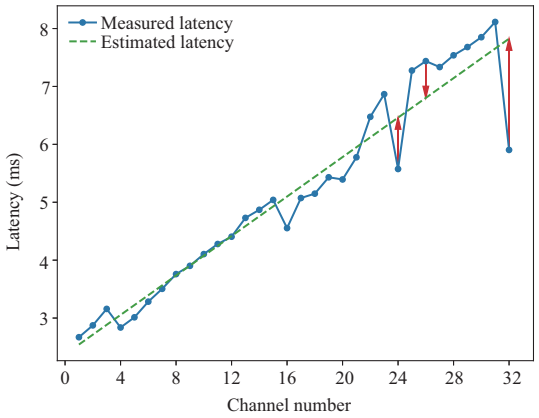

图 2 (网络版彩图) 在全局搜索空间上测量延迟并构建延迟模型. 实际延迟 (实线) 低于估计延迟 (虚线) 的通道数更高效. 实际延迟和估计延迟之间的差是对通道数的评分 (箭头)

Figure 2 (Color online) Measure the latency over the global search space and construct the latency model. Some channel choices are more efficient, as their actual latency (solid) is lower than the estimated latency (dashed). The difference between the actual latency and the estimated latency is the choice score (arrow)

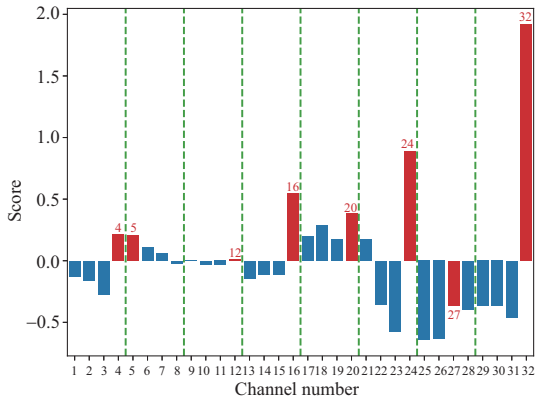

图 3 (网络版彩图) 选择硬件友好的输出通道数来为 CPU 构建硬件感知的搜索空间. 虚线将全局搜索空间切分成 $g = 8$ 组. 红色条上方的数字表示每组中评分最高的通道数, 这些高效的通道数将用于构建硬件感知的搜索空间

Figure 3 (Color online) Select hardware-friendly output channel choices to construct the hardware-aware search space for CPU. The dashed lines split the global search space into $g = 8$ groups. The numbers above the red bars indicate the selected efficient channel choices with the highest score in each group

表 2 CPU 和 GPU 的硬件感知的搜索空间

Table 2 Hardware-aware search space for CPU and GPU

Hardware	Output size	Hardware-aware search space
CPU	32×32	4, 5, 12, 16, 20, 24, 27, 32
	16×16	8, 13, 20, 27, 35, 45, 52, 64
	8×8	15, 20, 46, 50, 70, 92, 107, 128
GPU	32×32	4, 6, 12, 16, 20, 24, 28, 32
	16×16	6, 16, 24, 32, 40, 48, 56, 64
	8×8	16, 32, 47, 64, 80, 96, 112, 128

$\mathbb{E}[\text{latency}]$ 是通过 3.2 小节中的延迟模型 $f(x)$ 来进行评估的.

$$\mathbb{E}[\text{latency}^i] = f(\mathbb{E}[x^i]), \tag{7}$$

$$\mathbb{E}[\text{latency}] = \sum_i \mathbb{E}[\text{latency}^i], \tag{8}$$

其中 $\mathbb{E}[\text{latency}^i]$ 和 $\mathbb{E}[x^i]$ 分别是第 i 层的期望延迟和期望宽度.

TAS [7] 在深度和宽度的搜索过程中为每个候选值引入参数 α, 并由此为候选值生成了概率 p. 神经网络的输出是候选输出的加权和, 这样就能使用基于梯度的方法来学习架构参数 α 和网络的权值 ω. 最终的架构由具有最大 α 的宽度值和深度值组成. 本文为了降低搜索代价选择了可微搜索策略, 不过我们提出的搜索空间也能和进化算法等许多其他的搜索策略方便地结合起来.

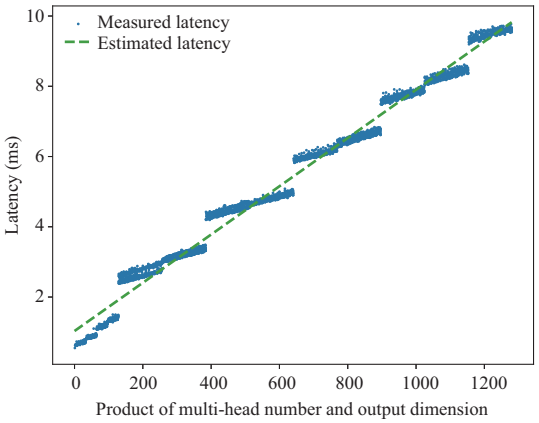

图 4 (网络版彩图) 在全局搜索空间上测量延迟并构建延迟模型. 从多头数和输出维度的选择组合可以看出, 实际延迟低于估计延迟的组合更高效

Figure 4 (Color online) Measure the latency over the global search space and construct the latency model. The choices of multi-head number and output dimension where the actual latency is lower than the estimated latency are more efficient

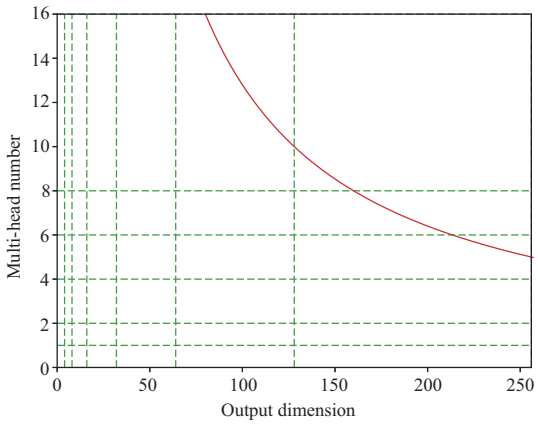

图 5 (网络版彩图) 多头数和输出维度分块图

Figure 5 (Color online) Partitioning of multi-head number and output dimension search space

4 HGNAS: 硬件感知的图神经网络架构搜索

本节将介绍我们提出的硬件感知的图神经网络架构搜索 (HGNAS) 方法. 与 HTAS 类似, 我们为图神经网络构建了硬件感知的搜索空间, 并使用加入了延迟限制的强化学习方法来搜索神经网络的最优架构.

4.1 构造硬件感知的搜索空间

出于与 3.2 小节相似的理由, 我们在保持 GraphNAS 原有搜索空间其他部分不变的情况下, 为其输出维度和多头数选出更加高效的候选值, 来构造硬件感知的搜索空间.

在 GraphNAS 方法中, 搜索空间中的信息计算函数、聚合函数和激活函数是作者总结出的当前图神经网络领域中具有代表性的函数种类, 我们不对其进行修改, 而输出维度和多头数是人工选出的若干值, 其选择过程无特定准则, 我们决定选取若干个多头数和输出维度的二元组来替代原来的搜索空间中两者的选择. 如果有 N_1 个信息计算函数, N_2 个聚合函数, N_3 个激活函数, N_4 个多头数, N_5 个输出维度, L 层可供搜索, 则搜索空间的大小为 $O((N_1N_2N_3N_4N_5)^L)$.

与 3.2 小节相同, 我们使用延迟模型 $f(x)$ 用于学习实际延迟与多头数和输出维度二元组之间的关系, 并使用 $s(x)$ 对二元组进行评分, 其中 x 是多头数和输出维度的乘积. 图 4 展示了把 α 设为 1 时图神经网络在 GPU 上的推断延迟和估计延迟随多头数和输出维度乘积变化而变化的散点图. 可以看出延迟随着该乘积的增加呈阶梯状上升趋势, 我们在 CPU 上也进行了大量的实验, 结果发现 CPU 上没有这种规律, 而是呈不规则上升趋势, 这种情况应该与硬件的特性有关.

为了和 GraphNAS 原有搜索空间中的多头数和输出维度的选择范围和间隔保持基本一致, 我们根据 GraphNAS 中原本使用的这两种参数的候选值, 将这两种参数的可能选择切割成图 5 中绿色线围成的 42 个块. 考虑到这两种参数取值很大的架构会具有很高的推断延迟, 为了减少构造搜索空间

中测量推断延迟的代价, 我们将两种参数乘积大于 1280 的二元组 (即图中红色曲线的右上方) 直接排除选择范围. 我们在每个块中选取评分 $s(x)$ 最高的二元组作为这一块中的多头数和输出维度的候选值. 这 42 个二元组和原搜索空间中的三类函数即为我们为目标硬件构建的硬件感知的搜索空间.

4.2 搜索策略

我们使用文献 [41] 中的强化学习方法在硬件感知的搜索空间中为目标硬件搜索高效的图神经网络架构. 该问题可以如下表述.

给定一个图神经网络的搜索空间 M, 一个训练集 D_{train}, 一个验证集 D_{val}, 硬件平台 H, 以及时间限制 T, 找到架构 $m^* \in M$, 使得其对应图神经网络在训练集 D_{train} 上优化后, 在验证集 D_{val} 上的准确率最高, 而且在硬件 H 上的耗时不大于 T, 即

$$m^* = \arg\max_{m \in \mathcal{M}'} \text{accuracy}_{\text{val}}(m, \theta(m)), \tag{9}$$

其中

$$\mathcal{M}' = \{m \mid m \in \mathcal{M} \wedge \text{latency}_H(m) \leqslant T\}, \tag{10}$$

$$\theta(m) = \arg\min_{\theta} \mathcal{L}_{\text{train}}(m, \theta), \tag{11}$$

$\text{accuracy}_{\text{val}}$ 是图神经网络架构在验证集上的准确率, $\text{latency}_H(m)$ 是 m 对应的图神经网络在硬件 H 上的推断延迟, $\mathcal{L}_{\text{train}}$ 是网络的交叉熵分类损失.

考虑到图神经网络架构的搜索空间比较大而且相对复杂, 本文使用一个三层 MLP 模型来估计图神经网络架构的推断延迟. 强化学习过程中的奖励值 reward 由如下方式算得:

$$\text{reward} = \text{accuracy} \times \left(\frac{\text{latency}}{T}\right)^{-\omega}, \tag{12}$$

其中 T, ω 均为给定的常量, accuracy 是图神经网络架构在验证集 D_{val} 上的准确率, latency 是架构的推断延迟.

5 实验和讨论

5.1 数据集和参数设置

5.1.1 数据集

为了证明我们提出的方法的有效性, 本文在 CIFAR-10 和 CIFAR-100 数据集上对 HTAS 进行了对比实验. CIFAR-10 是一个 10 分类的图像数据集, 其中包含 5 万张训练图像和 1 万张测试图像, 它们的分辨率为 32×32. CIFAR-100 是一个 100 分类的图像数据集, 其他方面与 CIFAR-10 一致.

同时, 我们也在 Cora, Citeseer 和 Pubmed 数据集上对 HGNAS 进行了对比实验. 这 3 个数据集都是论文引用关系图, 其详细信息在表 3 中给出.

5.1.2 搜索和训练参数设置

在 HTAS 的实验中, 为了构建硬件感知的搜索空间, 我们扩大全局搜索空间到 $e = 2$ 倍, 并把全局搜索空间分为 $g = 8$ 组. 我们设 α 为 1 来学习延迟模型, 并给通道数评分. 在搜索神经架构的过程

表 3 论文引用关系图数据集信息

Table 3 Information of citation network datasets

Dataset	Node count	Edge count	Feature length	Classes
Cora	2708	5278	1433	7
Citeseer	3327	4552	3703	6
Pubmed	19717	44324	500	3

中, 以及搜索结束后对搜索出的神经架构的训练中, 我们使用和文献 [7] 相同的搜索和训练参数 (包括批尺寸、学习率调度方法等). 为了进行公平比较, 我们使用与 TAS [7] 中相同的知识蒸馏方法 [48] 来训练搜索出的神经架构. 在知识蒸馏方法中, 我们将预训练的基础模型作为教师模型, 然后把知识迁移到搜索出的架构上.

在 HGNAS 的实验中, 我们设 α 为 1 来选出 42 个输出维度和多头数二元组, 以构建硬件感知的搜索空间, 并将奖励值函数中的 ω 设为 0.07. 具体搜索和训练过程与 GraphNAS [41] 保持一致.

5.1.3 延迟评估

我们在 GPU 和 CPU 上分别进行了实验. 在 HTAS 的实验中, GPU 的型号为 TITAN X(Pascal) GPU, CPU 的型号为 2.30 GHz Intel(R) Xeon(R) CPU E5-2630 0. 在 HGNAS 的实验中, GPU 的型号为 GeForce GTX TITAN X GPU, CPU 的型号为 Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60 GHz.

在 HTAS 的实验中, 由于 CIFAR 数据集的图像分辨率为 32×32, 相对较小, 我们使用一批图像而非单个图像作为输入来测量推断延迟. 此外, 由于在使用较小批尺寸时 GPU 的利用率不足, 我们在 GPU 上实验时使用比 CPU 上更大的批尺寸. 具体来说, 我们在 GPU 上将批尺寸设置为 1024 来测量 GPU 延迟, 在带有两个 CPU 的服务器上将批尺寸设置为 4 来测量 CPU 延迟.

5.1.4 基线

目前主流的硬件感知的 TAS 方法主要着眼于改进搜索策略, 然而本文主要关注的是搜索空间. 因此, 我们不将 HTAS 模型和那些研究搜索策略的模型进行比较. 为了探究本文提出的硬件感知的搜索空间的有效性, 我们选择了一个当前最先进的方法 TAS [7], 并比较了其在原有搜索空间和本文提出的搜索空间下的搜索结果. 同样地, 我们在 HGNAS 和 GraphNAS 间进行比较.

5.2 HTAS 和 TAS 的实验结果比较

在 CIFAR-10 上的结果. 我们在 GPU 上使用 HTAS 方法去搜索 CIFAR-10 数据集上的多个 ResNets 神经网络架构 (包含 ResNets32, ResNets56, ResNets110), 结果如表 4 所示. 相比于 TAS [7], 我们的方法在不同深度的 ResNet 上正确率相近或者更高, 且推断延迟更低. 对于 ResNet32, 尽管 HTAS 的 FLOPs 比 TAS 的更高, 但在 HTAS 的准确率比 TAS 更高的同时, 它的推断延迟 (18.4 ms) 仍比 TAS (35.0 ms) 快了 1.9 倍. 搜索 ResNet110 的实验也得到了相似的结果, 这证明了神经网络的 FLOPs 和实际的效率并不一致. 硬件感知的搜索空间提供了高效的通道数量选择, 它们的延迟比 FLOPs 体现得要更低, 这是 HTAS 的推断延迟更低的主要原因.

在 CIFAR-100 上的结果. 我们在不同的硬件上搜索了 CIFAR-100 数据集上的 ResNets, 在 GPU 和 CPU 上的大多数情况下 HTAS 都达到了更好的性能, 结果如表 5 所示. 在 GPU 上搜索 ResNet100 时, TAS [7] 的推断延迟是 90.9 ms, 而 HTAS 的推断延迟是 41.0 ms, 比 TAS 快了 2.2 倍, 并且 FLOPs 更低, 准确率更高. 除了 GPU 外, 我们观察到在 CPU 上我们的方法也达到了更好的性能. 值得注意

表 4 在 CIFAR-10 上对 ResNets 与当前最先进的方法进行比较. "Dep." 表示 ResNet 的深度. "CIFAR10" 表示在 CIFAR-10 上的 top-1 准确率. "GPU" 表示在 GPU 上测量出的推断延迟. "FLOPs" 表示乘加法操作的次数. "Param" 表示网络参数的数量. 对不同深度的 ResNet, 我们提出的 HTAS 和 TAS 相比, 正确率近似或更高, 且延迟更低

Table 4 Comparison with the state-of-the-art method for ResNets on CIFAR-10. "Dep." denotes the depth of ResNet. "CIFAR10" denotes the top-1 accuracy on CIFAR-10. "GPU" denotes the inference latency measured on GPU. "FLOPs" denotes the number of multiply-adds. "Param" means the number of network parameters. Our HTAS achieves lower inference latency with competitive or higher accuracy for different depths of ResNet

Dep.	Method	CIFAR10 (%)	GPU (ms)	FLOPs (M)	Param (M)
110	TAS [7]	94.33	81.6	199.3	0.80
	Ours	**94.33**	**44.8**	124.0	1.57
56	TAS [7]	**93.69**	42.4	59.5	0.45
	Ours	93.42	**31.5**	58.5	0.73
32	TAS [7]	93.16	35.0	35.0	0.32
	Ours	**93.34**	**18.4**	37.7	0.48

表 5 在 CIFAR-100 上对 ResNets 与当前最先进的方法进行比较. "Manual" 表示搜索前的人工设计的 ResNets. "GPU" 和 "CPU" 分别表示在 GPU 和 CPU 上搜索和测量的结果. "CIFAR100" 表示在 CIFAR-100 上的 top-1 准确率. "Latency" 表示模型在对应硬件上的推断延迟. 我们提出的 HTAS 在 GPU 和 CPU 上都能找出更高效且准确率相当或更高的架构

Table 5 Comparison with the existing method for ResNets on CIFAR-100. "Manual" denotes the manually designed ResNets without searching. "GPU" and "CPU" represent that models are searched and measured on GPU and CPU respectively. "CIFAR100" denotes the top-1 accuracy on CIFAR-100, "Latency" denotes the inference latency measured on this hardware. Our HTAS finds more efficient architecture with comparable or improved accuracy on both GPU and CPU

Depth	Method	GPU				CPU			
		CIFAR100 (%)	Latency (ms)	FLOPs (M)	Param (M)	CIFAR100 (%)	Latency (ms)	FLOPs (M)	Param (M)
110	Manual	74.51	144.0	253.2	1.73	74.51	144.0	253.2	1.74
	TAS [7]	73.16	90.9	119.6	0.84	73.16	113.6	119.6	0.84
	Ours	**74.51**	**41.0**	119.1	1.71	**74.93**	**51.5**	126.0	0.84
56	Manual	73.18	73.0	125.8	0.86	**73.18**	100.0	125.8	0.86
	TAS [7]	72.25	44.1	61.2	0.47	72.25	59.1	61.2	0.47
	Ours	**74.22**	**29.7**	72.9	0.99	73.01	**34.9**	66.8	0.47
32	Manual	70.44	41.8	69.1	0.47	70.44	56.6	69.1	0.47
	TAS [7]	**72.41**	33.7	42.5	0.43	**72.41**	42.8	42.5	0.43
	Ours	72.34	**21.6**	47.0	0.58	72.39	**24.4**	47.0	0.32

的是, HTAS 在 ResNet110 上的准确率为 74.93%, 比 TAS [7] 高出 1.77%, 并且 HTAS 搜索出的架构的推断延迟 (51.5 ms) 比 TAS [7] (113.6 ms) 快了 2.2 倍. HTAS 在 ResNet32 上的准确率比 TAS 略低, 但推断延迟仍然更快. 这可能是因为在硬件感知的搜索空间中被选出的宽度值之间的间隔不同, 而比起原有的间隔相同的搜索空间, 搜索策略可能更难以探索我们提出的搜索空间.

本文通过 HTAS 的这些强有力的实验结果证明了在搜索空间的设计中考虑硬件的好处. 此外, 由于硬件感知的搜索空间的大小和文献 [7] 中原有搜索空间的大小相等, 所以我们的方法不需要花费额外的搜索或者训练代价就能达到更好的性能.

表 6 HTAS 能在 CIFAR-100 上为不同的硬件搜索高效的模型. "ResNet32-G" 表示为 GPU 搜索 ResNet32 得到的模型, "ResNet32-C" 表示为 CPU 搜得的模型. "(t)" 表示搜出的模型在其目标硬件上测量的结果. 搜出的模型在其目标硬件上都比另一个模型快

Table 6 HTAS searches efficient models for different hardware on CIFAR-100. "ResNet32-G" denotes the searched ResNet32 for GPU, "ResNet32-C" denotes the searched ResNet32 for CPU. "(t)" denotes the searched model is measured on its target hardware. Both of the searched models are faster on their target hardware

Model	GPU (ms)	CPU (ms)	Top-1 (%)	FLOPs (M)
ResNet32-G	**21.6(t)**	26.8	72.34	47.0
ResNet32-C	23.5	**24.4(t)**	72.39	47.0

5.2.1 HTAS 在不同硬件上搜索架构的结果

以往的硬件无关的 TAS 方法在不同的硬件上部署相同的神经架构. 然而, 不同硬件设备的不同特性可能会导致这些方法只能搜索出神经网络的次优架构.

为了证明上述观点, 我们比较了在 CIFAR-100 数据集上 HTAS 在 GPU 和 CPU 上搜索出的两个模型, 结果如表 6 所示. 我们测量了这两个模型在 GPU 和 CPU 上的推断延迟. 在两种硬件上搜索出的模型达到了相似的准确率, 分别为 72.34% 和 72.39%. 对于 ResNet32-G, 在其目标硬件 (GPU) 上的推断延迟为 21.6 ms, 而它在部署到 CPU 上时的推断延迟为 26.8 ms. ResNet32-C 在 GPU 上的推断延迟为 23.5 ms, 比 ResNet32-G 在 GPU 上的高 1.9 ms. 然而, ResNet32-C 在部署到它的目标硬件 (CPU) 上时推断延迟为 24.4 ms, 比 ResNet32-G 在 CPU 上的低 2.4 ms. 这些结论说明为不同的目标硬件搜索专门的架构是十分有必要的.

5.2.2 HTAS 在不同搜索空间中的结果

之前的实验已经证明了新的搜索空间的有效性. 从第 3 节我们可以看出 HTAS 对 TAS 原有的搜索空间的改进主要分为两部分: 扩大全局搜索空间, 构建硬件感知的搜索空间. 在这里, 我们用两组 CIFAR-10 数据集上的实验来说明两种改进的有效性.

首先, 我们对原有的和扩大后的全局搜索空间分别使用 TAS 的搜索空间构造方法和第 3 节中提出的硬件感知的搜索空间构造方法, 得到 4 种不同的搜索空间进行实验, 结果如表 7 所示. 然后, 我们进行实验来探究式 (1) 中参数 α 的选择对硬件感知的搜索空间以及搜索结果性能的影响, 结果如表 8 所示.

扩大全局搜索空间的效果. 从表 7 中我们可以看出, 当全局搜索空间扩大了 $e = 2$ 倍时, TAS 和 HTAS 的准确率和 FLOPs 与之前几乎相等, 但是延迟显著降低了. 经过对搜索出架构的进一步观察, 我们发现这一现象主要是因为当搜索空间的通道数扩到 $e = 2$ 倍时, 搜索方法往往会优先选择较大的通道数与较低的层数, 而相比于通道数, 层数对架构的延迟影响更大, 因此搜索出的架构延迟明显降低.

构建硬件感知的搜索空间的效果. 从表 7 中我们也可以发现, 在原有的和扩大后的搜索空间 (如 $e = 2$) 中, 使用硬件感知的搜索空间构造方法搜索出的神经架构, 和不使用该方法搜索出的架构相比, 推断延迟总体更低. 此外, 表 8 展示了不同 α 的取值下 (例如, 0.5, 1 和 1.5) 搜索出的模型性能. 我们可以明显地看出 $\alpha = 1$ 时, 模型的准确率和延迟总体上比其他两种情况要好.

5.2.3 HTAS 在 CPU 和 GPU 上搜索出的架构间的比较

我们进一步比较了在 GPU 和 CPU 上搜索 ResNet32 得到的架构. 如表 9 所示, CPU 偏好窄而

表 7 全局搜索空间扩张和搜索空间构造方法的比较结果. “e” 表示搜索空间扩大倍数. “TAS” 表示使用文献 [7] 中搜索空间构造方法搜得的模型. “Ours” 表示使用硬件感知的搜索空间构造方法搜得的模型

Table 7 Comparison on global search space expansions and on the search space construction methods. e is the factor of search space expansions. “TAS” indicates a model with the search space construction in [7], while “Ours” indicates a model with the search space construction in Subsection 3.2

Dep.	Method	e	CIFAR10 (%)	GPU (ms)	FLOPs (M)
110	TAS [7]	1	**94.33**	81.6	119.3
	Ours	1	91.17	110.8	119.6
	TAS [7]	2	93.81	49.5	128.3
	Ours	2	**94.33**	**44.8**	124.0
56	TAS [7]	1	**93.69**	42.4	59.5
	Ours	1	91.95	36.8	65.8
	TAS [7]	2	93.29	38.5	65.5
	Ours	2	93.42	**31.5**	58.5
32	TAS [7]	1	93.16	35.0	35.0
	Ours	1	93.26	32.0	37.5
	TAS [7]	2	91.51	**18.0**	36.3
	Ours	2	**93.34**	18.4	37.7

表 8 在硬件感知的搜索空间构造方法中, 不同 α 取值的比较结果. α 是式 (1) 中的参数. 实验中的全局搜索空间都扩大了 $e = 2$ 倍

Table 8 Comparison on different α in the Hardware-aware search space construction. α is the parameter in Eq. (1). We chose $e = 2$ in all of the experiments

Dep.	α	CIFAR-10 (%)	GPU (ms)	FLOPs (M)
110	0.5	94.00	49.7	127.9
	1	**94.33**	**44.8**	124.0
	1.5	94.27	50.1	132.0
56	0.5	**93.76**	**30.8**	69.4
	1	93.42	31.5	58.5
	1.5	91.48	39.1	65.4
32	0.5	93.16	22.0	40.8
	1	**93.34**	**18.4**	37.7
	1.5	93.10	22.4	41.1

深的模型, 而 GPU 偏好宽而浅的模型. 不过, CPU 和 GPU 都偏好在最后的阶段层数较多的模型. 这些结论可以为我们在不同硬件平台上设计神经网络提供一些有用的见解.

5.3 HGNAS 和 GraphNAS 的实验结果比较

在本实验中, 我们对 HGNAS 和 GraphNAS 均使用文献 [41] 中的方式选择最终架构, 即从搜索过程中产生的 1000 个架构中选出准确率最高的架构. 结果如表 10 所示, 可以看出, HGNAS 搜出的架构与 GraphNAS 的相比, 准确率近似而且延迟更低. 因此, 可以认为, HGNAS 的搜索结果是符合预期的, 即能够维持架构具有较高准确率的同时达到较低的推断延迟.

表 9 在 CPU 和 GPU 上搜索 ResNet32 得到的架构的比较. "ResNet32-G" 表示为 GPU 搜得的模型, "ResNet32-C" 表示为 CPU 搜得的模型

Table 9 Comparisons of the architectures searched for ResNet32 on CPU (i.e., ResNet32-C) and GPU (i.e., ResNet32-G)

Model	Stage	Layer	Block	Input channels	Output channels	Stride
ResNet32-C	stage = 0	ilayer = 00/05	block = 001	iCs = [20, 4, 4]	oC = 4	stride = 1
	stage = 1	ilayer = 00/05	block = 002	iCs = [4, 8, 35]	oC = 35	stride = 2
	stage = 2	ilayer = 00/05	block = 003	iCs = [35, 107, 107]	oC = 107	stride = 2
	stage = 2	ilayer = 01/05	block = 004	iCs = [107, 128, 92]	oC = 92	stride = 1
	stage = 2	ilayer = 02/05	block = 005	iCs = [92, 128, 107]	oC = 107	stride = 1
	stage = 2	ilayer = 03/05	block = 006	iCs = [107, 15, 128]	oC = 128	stride = 1
	stage = 2	ilayer = 04/05	block = 007	iCs = [128, 15, 128]	oC = 128	stride = 1
ResNet32-G	stage = 0	ilayer = 00/05	block = 001	iCs = [32, 6, 6]	oC = 6	stride = 1
	stage = 1	ilayer = 00/05	block = 002	iCs = [6, 64, 64]	oC = 64	stride = 2
	stage = 2	ilayer = 00/05	block = 003	iCs = [64, 128, 128]	oC = 128	stride = 2
	stage = 2	ilayer = 01/05	block = 004	iCs = [128, 32, 64]	oC = 64	stride = 1
	stage = 2	ilayer = 02/05	block = 005	iCs = [64, 128, 128]	oC = 128	stride = 1

表 10 在 Cora, Citeseer, Pubmed 三种数据集和 CPU, GPU 两种硬件上与 GraphNAS [41] 进行比较. "GPU" 和 "CPU" 分别表示在 GPU 和 CPU 上搜索和测量的结果. "Accuracy" 表示搜索出的模型的准确率. "Latency" 表示模型在对应硬件上的推断延迟. 我们提出的 HGNAS 在 GPU 和 CPU 上都能找出更高效且准确率相当或更高的架构

Table 10 Comparison with GraphNAS [41] on three datasets (Cora, Citeseer and Pubmed) and two hardware platforms (GPU and CPU). "GPU" and "CPU" represent that models are searched and measured on GPU and CPU respectively. "Accuracy" denotes the accuracy of the searched model, "Latency" denotes the inference latency measured on this hardware. Our HGNAS finds more efficient architecture with comparable or improved accuracy on both GPU and CPU

Dataset	Method	GPU		CPU	
		Latency (ms)	Accuracy (%)	Latency (s)	Accuracy (%)
Cora	GraphNAS	2.4	$\mathbf{83.7 \pm 0.4}$	0.13	83.7 ± 0.4
	HGNAS	**2.1**	83.3 ± 0.1	**0.03**	$\mathbf{84.0 \pm 0.2}$
Citeseer	GraphNAS	7.0	73.5 ± 0.3	0.66	73.5 ± 0.3
	HGNAS	**2.9**	$\mathbf{73.8 \pm 0.1}$	**0.13**	$\mathbf{73.9 \pm 0.2}$
Pubmed	GraphNAS	13.5	$\mathbf{80.5 \pm 0.3}$	2.15	$\mathbf{80.5 \pm 0.3}$
	HGNAS	**4.6**	80.0 ± 0.2	**0.18**	80.1 ± 0.1

5.3.1 HGNAS 和 GraphNAS 的带延迟限制的实验结果比较

除了关注 HGNAS 作为一种神经架构搜索方法搜出的最终架构的性能外, 我们还想要讨论我们使用的硬件感知的搜索空间和搜索策略本身的效果, 即最终得到的 RNN 控制器产生高效架构的能力. 因此在本实验中, 我们对 GraphNAS 和 HGNAS 均从训练结束后的 RNN 控制器中取样 100 个架构, 然后对这些架构进行比较和分析.

根据第 4.2 小节中对研究问题的定义, 我们对每种硬件和数据集上搜出的架构给定延迟限制, 然后选出这些架构中在目标硬件上的推断延迟低于延迟限制的架构, 并重新多次训练这些架构, 计算其准确率取平均值, 最后得到准确率最高的架构作为延迟低于限制的最优架构. 我们对每种硬件和数据

表 11 HGNAS 和 GraphNAS 在 GPU 上的带延迟限制的结果

Table 11 Search results of HGNAS and GraphNAS with latency limits on GPU

Dataset	Latency limit (ms)	Method	Latency (ms)	Accuracy (%)
Cora	2	GraphNAS	**1.9**	83.2 ± 0.2
		HGNAS	**1.9**	$\mathbf{83.5 \pm 0.1}$
	4	GraphNAS	**2.1**	$\mathbf{83.6 \pm 0.2}$
		HGNAS	2.3	$\mathbf{83.6 \pm 0.1}$
Citeseer	2	GraphNAS	2.0	73.3 ± 0.1
		HGNAS	**1.7**	$\mathbf{73.5 \pm 0.1}$
	4	GraphNAS	3.0	73.5 ± 0.1
		HGNAS	**2.4**	$\mathbf{73.7 \pm 0.2}$
Pubmed	2	GraphNAS	**1.3**	79.0 ± 0.1
		HGNAS	1.6	$\mathbf{79.6 \pm 0.1}$
	4	GraphNAS	**2.1**	$\mathbf{80.2 \pm 0.1}$
		HGNAS	3.6	80.1 ± 0.1

表 12 HGNAS 和 GraphNAS 在 CPU 上的带延迟限制的结果

Table 12 Search results of HGNAS and GraphNAS with latency limits on CPU

Dataset	Latency limit (ms)	Method	Latency (ms)	Accuracy (%)
Cora	40	GraphNAS	31.7	83.2 ± 0.2
		HGNAS	**27.5**	$\mathbf{83.5 \pm 0.2}$
	80	GraphNAS	45.4	83.3 ± 0.1
		HGNAS	**40.3**	$\mathbf{83.6 \pm 0.1}$
Citeseer	40	GraphNAS	**32.6**	72.2 ± 0.2
		HGNAS	39.7	$\mathbf{73.2 \pm 0.2}$
	80	GraphNAS	68.9	73.3 ± 0.1
		HGNAS	**46.2**	$\mathbf{73.6 \pm 0.1}$
Pubmed	100[a)]	GraphNAS	**89.7**	77.4 ± 0.3
		HGNAS	90.5	$\mathbf{79.2 \pm 0.2}$
	200	GraphNAS	**115.0**	79.6 ± 0.1
		HGNAS	191.6	$\mathbf{79.7 \pm 0.1}$

a) We choose different latency limits on Pubmed, because it is a larger dataset, and CPU has low parallel computing capacity, leading to higher inference latencies on this dataset.

集选择了两种不同的延迟限制. 实验结果如表 11 和 12 所示. 可以看出, 在两种延迟限制下, HGNAS 的搜索结果的准确率总体上均优于 GraphNAS, 且在较低的延迟限制下, HGNAS 搜出的模型的准确率明显更高. 这证明 HGNAS 的 RNN 控制器更容易产生低延迟且高准确率的架构.

5.3.2 HGNAS 使用的奖励值函数对搜索结果的影响

为了探究奖励值函数中的参数取值对搜索结果的影响, 我们在 GPU 和 Citeseer 数据集上对参数 ω 取不同的值来进行实验并比较结果. 为了与 GraphNAS 进行比较, 本实验中搜索过程均未使用硬件感知的搜索空间.

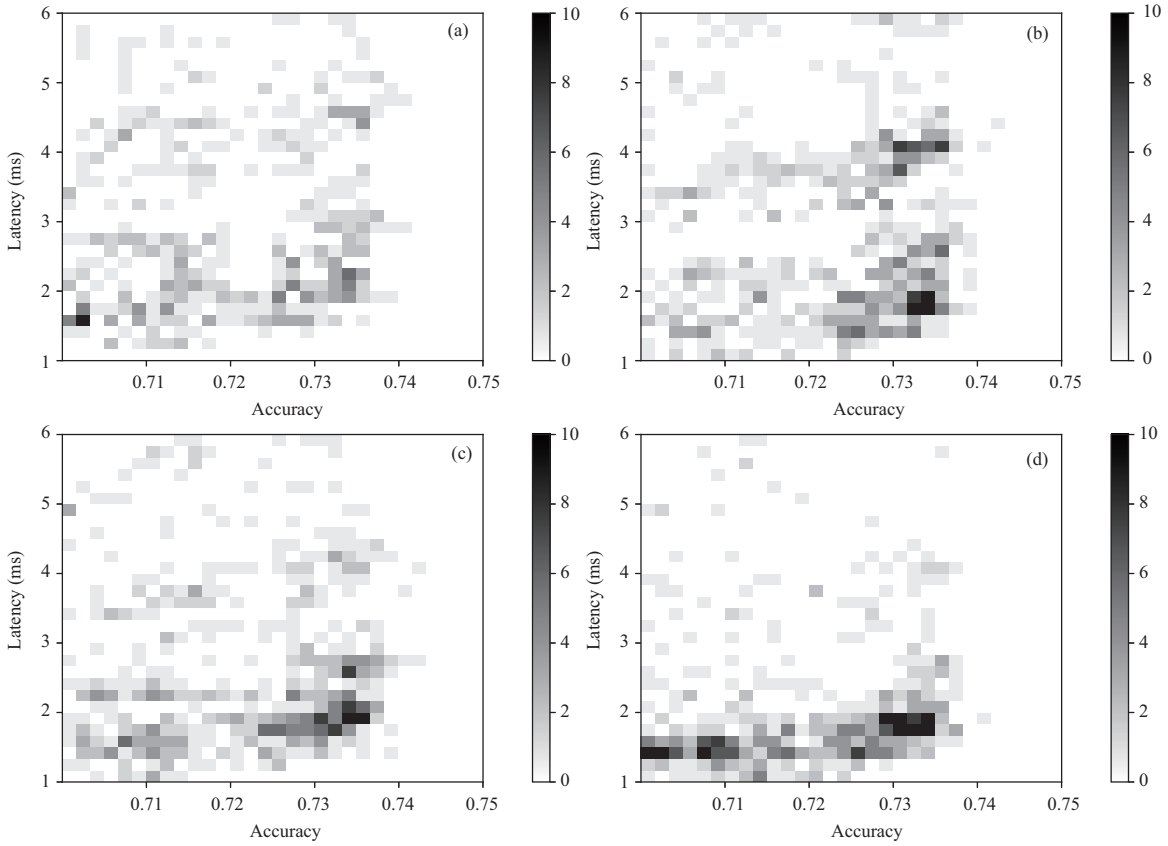

图 6　ω 的不同取值对搜索结果的影响

Figure 6　Effect of different ω values on search results. (a) $\omega = 0$ (GraphNAS); (b) $\omega = 0.05$; (c) $\omega = 0.07$; (d) $\omega = 0.1$

图 6 比较了使用不同 ω 的奖励值函数时, 训练得到的 RNN 控制器取样出的 1000 个架构的推断延迟与准确率的分布. 该图为架构推断延迟和准确率的二维直方图, 其中只对准确率在 $0.7 \sim 0.75$ 之间, 推断延迟在 $1 \sim 6$ ms 之间的架构进行统计. 准确率和推理延迟被均匀分为 30 个区间, 图中小块的颜色越深, 说明落入该区间的架构数越多. 其中, 当 $\alpha = 0$ 时, 搜索过程即等价于 GraphNAS.

可以看出, GraphNAS 搜索出的架构分布整体比较均匀, 这主要是因为延迟并非它的搜索目标. ω 值越大, RNN 控制器取样出的架构的延迟整体越低, 即更偏向于低延迟而非高准确率.

6　结论

本文提出了一种硬件感知的神经架构搜索框架. 它构建了一个新的硬件感知的搜索空间, 并使用加入了推断延迟的搜索策略为不同硬件找到最优的架构. 据我们所知, 本文是首个致力于研究在 TAS 和 GraphNAS 中使用硬件感知搜索空间的工作. 我们的实验表明 HTAS 模型在两个 CIFAR 数据集和不同的硬件上都比当前最先进的方法要好, 这证明了我们的方法的优越性. 同时, HGNAS 在相关图数据上的实验也取得了很好的效果, 这也说明我们提出的方法是具有普适性的, 即能够与各种神经架构搜索方法相结合.

在未来的研究中, 进一步改进硬件感知的搜索空间的构造方法以及设计更好的搜索策略是一个可

行的研究方向. 同时, 在当前互联网的大环境下, 边缘智能得到越来越广泛的应用, 如何设计能够适用于边缘智能计算的硬件感知的 (图) 架构搜索算法也是一个很有趣的研究方向.

参考文献

1 Liu H, Simonyan K, Yang Y. DARTS: differentiable architecture search. In: Proceedings of the 7th International Conference on Learning Representations, New Orleans, 2019

2 Zoph B, Le Q V. Neural architecture search with reinforcement learning. In: Proceedings of the 5th International Conference on Learning Representations, Toulon, 2017

3 Wu B, Dai X, Zhang P, et al. FBNet: hardware-aware efficient ConvNet design via differentiable neural architecture search. In: Proceedings of 2019 IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, 2019. 10734–10742

4 Cai H, Zhu L, Han S. ProxylessNAS: direct neural architecture search on target task and hardware. In: Proceedings of the 7th International Conference on Learning Representations, New Orleans, 2019

5 Yang T, Howard A G, Chen B, et al. NetAdapt: platform-aware neural network adaptation for mobile applications. In: Proceedings of the 15th European Conference on Computer Vision, Munich, 2018. 289–304

6 Tan M, Chen B, Pang R, et al. MnasNet: platform-aware neural architecture search for mobile. In: Proceedings of 2019 IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, 2019. 2820–2828

7 Dong X, Yang Y. Network pruning via transformable architecture search. In: Proceedings of Annual Conference on Neural Information Processing Systems 2019, Vancouver, 2019. 759–770

8 Iandola F N, Moskewicz M W, Ashraf K, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1 MB model size. 2016. ArXiv:1602.07360

9 Gholami A, Kwon K, Wu B, et al. SqueezeNext: hardware-aware neural network design. In: Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, 2018. 1638–1647

10 Wu B, Wan A, Yue X, et al. Shift: a zero flop, zero parameter alternative to spatial convolutions. In: Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, 2018. 9127–9135

11 Howard A G, Zhu M, Chen B, et al. MobileNets: efficient convolutional neural networks for mobile vision applications. 2017. ArXiv:1704.04861

12 Sandler M, Howard A G, Zhu M, et al. MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, 2018. 4510–4520

13 Zhang X, Zhou X, Lin M, et al. ShuffleNet: an extremely efficient convolutional neural network for mobile devices. In: Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, 2018. 6848–6856

14 Ma N, Zhang X, Zheng H, et al. ShuffleNet V2: practical guidelines for efficient CNN architecture design. In: Proceedings of the 15th European Conference on Computer Vision, Munich, 2018. 122–138

15 Liu Z, Li J, Shen Z, et al. Learning efficient convolutional networks through network slimming. In: Proceedings of IEEE International Conference on Computer Vision, Venice, 2017. 2755–2763

16 Luo J, Wu J, Lin W. ThiNet: a filter level pruning method for deep neural network compression. In: Proceedings of IEEE International Conference on Computer Vision, Venice, 2017. 5068–5076

17 Li H, Kadav A, Durdanovic I, et al. Pruning filters for efficient ConvNets. In: Proceedings of the 5th International Conference on Learning Representations, Toulon, 2017

18 He Y, Liu P, Wang Z, et al. Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of 2019 IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, 2019. 4340–4349

19 Guo Z, Zhang X, Mu H, et al. Single path one-shot neural architecture search with uniform sampling. In: Proceedings of the 16th European Conference on Computer Vision, Glasgow, 2020. 544–560

20 Real E, Aggarwal A, Huang Y, et al. Regularized evolution for image classifier architecture search. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, Honolulu, 2019. 4780–4789

21 Pham H, Guan M, Zoph B, et al. Efficient neural architecture search via parameters sharing. In: Proceedings of the 35th International Conference on Machine Learning, 2018. 4095–4104

22 Zoph B, Vasudevan V, Shlens J, et al. Learning transferable architectures for scalable image recognition. In: Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, 2018. 8697–

8710
23 Bender G, Kindermans P, Zoph B, et al. Understanding and simplifying one-shot architecture search. In: Proceedings of the 35th International Conference on Machine Learning, Stockholm, 2018. 549–558
24 Stamoulis D, Cai E, Juan D, et al. HyperPower: power- and memory-constrained hyper-parameter optimization for neural networks. In: Proceedings of 2018 Design, Automation & Test in Europe Conference & Exhibition, Dresden, 2018. 19–24
25 Stamoulis D, Ding R, Wang D, et al. Single-path NAS: designing hardware-efficient ConvNets in less than 4 hours. In: Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases, Würzburg, 2019. 481–497
26 Howard A, Pang R, Adam H, et al. Searching for MobileNetV3. In: Proceedings of 2019 IEEE/CVF International Conference on Computer Vision, Seoul, 2019. 1314–1324
27 Zhang L L, Yang Y, Jiang Y, et al. Fast hardware-aware neural architecture search. In: Proceedings of 2020 IEEE Conference on Computer Vision and Pattern Recognition, Seattle, 2020. 2959–2967
28 Gordon A, Eban E, Nachum O, et al. MorphNet: fast & simple resource-constrained structure learning of deep networks. In: Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, 2018. 1586–1595
29 He Y, Lin J, Liu Z, et al. AMC: automl for model compression and acceleration on mobile devices. In: Proceedings of the 15th European Conference on Computer Vision, Munich, 2018. 815–832
30 Guan C, Wang X, Chen H, et al. Large-scale graph neural architecture search. In: Proceedings of International Conference on Machine Learning, 2022
31 Qin Y, Wang X, Zhang Z, et al. Graph neural architecture search under distribution shifts. In: Proceedings of International Conference on Machine Learning, 2022
32 Guan C, Zhang Z, Li H, et al. AutoGL: a library for automated graph learning. In: Proceedings of ICLR 2021 Workshop on Geometrical and Topological Representation Learning, 2021
33 Wang X, Fan S, Kuang K, et al. Explainable automated graph representation learning with hyperparameter importance. In: Proceedings of the 38th International Conference on Machine Learning, 2021. 10727–10737
34 Guan C, Wang X, Zhu W. AutoAttend: automated attention representation search. In: Proceedings of the 38th International Conference on Machine Learning, 2021. 3864–3874
35 Wang X, Zhu W. Automated machine learning on graph. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Singapore, 2021. 4082–4083
36 Zhang Z, Wang X, Zhu W. Automated machine learning on graphs: a survey. In: Proceedings of the 30th International Joint Conference on Artificial Intelligence, Montreal, 2021. 4704–4712
37 Qin Y, Wang X, Zhang Z, et al. Graph differentiable architecture search with structure learning. In: Proceedings of Annual Conference on Neural Information Processing Systems, 2021. 16860–16872
38 Qin Y, Wang X, Cui P, et al. GQNAS: graph Q network for neural architecture search. In: Proceedings of IEEE International Conference on Data Mining, Auckland, 2021. 1288–1293
39 Wang X, Zhang Z, Zhu W. Automated graph machine learning: approaches, libraries and directions. 2022. ArXiv:2201.01288
40 Cai J, Wang X, Guan C, et al. Multimodal continual graph learning with neural architecture search. In: Proceedings of ACM Web Conference, Lyon, 2022. 1292–1300
41 Gao Y, Yang H, Zhang P, et al. Graph neural architecture search. In: Proceedings of the 29th International Joint Conference on Artificial Intelligence, 2020. 1403–1409
42 Zhou K, Song Q, Huang X, et al. Auto-GNN: neural architecture search of graph neural networks. 2019. ArXiv:1909.03184
43 Zhao H, Wei L, Yao Q. Simplifying architecture search for graph neural network. In: Proceedings of Workshops Co-located with 29th ACM International Conference on Information and Knowledge Management, Galway, 2020
44 Li Y, Wen Z, Wang Y, et al. One-shot graph neural architecture search with dynamic search space. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence, 2021. 8510–8517
45 Cai S, Li L, Deng J, et al. Rethinking graph neural architecture search from message-passing. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2021. 6657–6666

46 Yu J, Yang L, Xu N, et al. Slimmable neural networks. In: Proceedings of the 7th International Conference on Learning Representations, New Orleans, 2019
47 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, 2016. 770–778
48 Hinton G E, Vinyals O, Dean J. Distilling the knowledge in a neural network. 2015. ArXiv:1503.02531

Hardware-aware neural architecture search

Xin WANG[1], Yang YAO[1], Yuhang JIANG[2], Chaoyu GUAN[1] & Wenwu ZHU[1*]

1. *Department of Computer Science and Technology, Tsinghua University, Beijing* 100084*, China*;
2. *Tsinghua-Berkeley Shenzhen Institute, Tsinghua University, Shenzhen* 518055*, China*
* Corresponding author. E-mail: wwzhu@tsinghua.edu.cn

Abstract Deep neural networks (DNNs) heavily rely on their architectures to achieve satisfactory performance. Neural network architecture search (NAS) was proposed to automatically search for the optimal architecture for neural networks. Most existing studies use FLOPs to evaluate the efficiency of neural network architecture, but FLOPs are inconsistent with the actual latency. As tasks become increasingly complex and hardware platforms begin to run neural networks, it becomes increasingly necessary to discover efficient neural network architecture for the target hardware platform. To solve this problem, this paper proposes a hardware-aware search space construction method. It uses a novel search strategy considering architecture inference latency to search for the optimal neural network architecture. In particular, we propose two variants, i.e., hardware-aware transformable architecture search (HTAS) and hardware-aware graph neural architecture search (HGNAS), to tackle problems in transformable architecture search (TAS) and graph neural architecture search, respectively. Compared with the existing state-of-the-art literature, the proposed HTAS and HGNAS models can discover more efficient neural architectures for different target hardware on various datasets, validating the effectiveness of the proposed methods.

Keywords deep learning, neural architecture search, transformable architecture search, graph neural architecture search, hardware-aware