神经架构搜索的高效评估方法：综述

Xiaotian Song、Xiangning Xie、Zeqiong Lv、Gary G. Yen，*IEEE* 会士、Weiping Ding，*IEEE* 高级会员、Jiancheng Lv，*IEEE* 高级会员以及 Yanan Sun，*IEEE* 高级会员

arXiv:2301.05919v2 [cs.NE] 9 Oct 2024

摘要——**神经架构搜索（NAS）凭借其自动化设计深度神经网络（DNN）架构的卓越优势，日益受到关注。然而，性能评估过程作为 NAS 的关键环节，通常需要训练大量 DNN。这不可避免地导致 NAS 的计算开销高昂。过去几年，为解决这一关键问题，人们提出了许多高效评估方法（EEMs）。本文全面综述了截至目前已发表的这些 EEM，并详细分析相关工作，以推动这一研究方向的进一步发展。具体而言，我们根据构建这些 EEM 所训练的 DNN 数量，将现有 EEM 划分为四类。原则上，这种分类能够反映效率水平，进而帮助读者快速掌握各方法的特点。在综述各类别时，我们进一步探讨其设计原则，并分析其优势与不足，以厘清现有 EEM 的整体格局，从而便于理解 EEM 的研究趋势。此外，我们还讨论当前面临的挑战与问题，以确定这一新兴主题未来的研究方向。总之，本综述为感兴趣的读者提供了 EEM 的便捷概览，帮助他们轻松选择适用于当前任务的 EEM 方法。此外，NAS 领域的研究人员也可以继续探索本文提出的未来研究方向。**

影响声明—**神经架构搜索（NAS）是自动化机器学习（AutoML）的关键子领域，可帮助用户（例如股票分析师和临床医生）在实践中轻松获得有前景的深度神经网络架构，而无需具备广泛的深度神经网络领域知识。然而，NAS 的性能评估过程通常耗时较长，阻碍了其广泛应用。为解决这一问题，研究人员开发了许多 EEM，以加速性能评估过程。本文系统回顾了最先进的 EEM，包括一种针对 EEM 的新颖分类方法、对 EEM 的全面比较分析，以及对现有 EEM 中问题的总结。此外，本文还详细讨论了未来的研究方向与挑战。通过提供 EEM 的鸟瞰式综述，本文推动了高效神经架构搜索的发展及其在现实世界中的更广泛应用。**

索引词—**深度神经网络（DNN）、神经架构搜索（NAS）、性能预测器、权重共享、高效评估**

Xiaotian Song、Xiangning Xie、Zeqiong Lv、Jiancheng Lv 和 Yanan Sun 任职于中国成都 610065 的四川大学计算机科学学院（电子邮箱：songxt@stu.scu.edu.cn；xnxie@stu.scu.edu.cn；zq lv@stu.scu.edu.cn；lvjiancheng@scu.edu.cn；ysun@scu.edu.cn）。（通信作者：*Yanan Sun*。）

Gary G. Yen 任职于俄克拉何马州立大学电气与计算机工程学院，斯蒂尔沃特，俄克拉何马州 74078，美国（电子邮箱：gyen@okstate.edu）。

Weiping Ding 任职于中国南通市南通大学人工智能与计算机科学学院（邮编：226019），同时任职于中国澳门城市大学数据科学学院（邮编：999078）（电子邮箱：dwp9988@163.com）。

I. 引言

NEURAL 神经架构搜索（NAS）旨在自动发现高性能深度神经网络（DNN）架构，使缺乏相关专业知识或专业知识有限的研究人员也能轻松受益于深度神经网络的成功。研究表明，在某些任务中，NAS 设计的架构甚至优于人工设计的架构 [1],[2], ，并且在深度学习领域日益受到欢迎 [3]。从数学角度看，NAS 将深度神经网络架构的设计过程建模为一个优化问题 [4]。具体而言，NAS 首先定义一个包含所有候选架构的搜索空间，然后采用精心设计的搜索策略来寻找最优架构。在搜索过程中，NAS 必须评估每个搜索得到的架构的性能，以有效指导搜索策略的运行。通常，NAS 问题难以求解，因为它面临着多项优化挑战，例如高昂得令人难以承受的计算成本，以及多个相互冲突的目标 [5]。

进化计算（EC） [6], [7],强化学习（RL） [2], [8],和梯度算法 [9]目前是解决神经架构搜索（NAS）的三种主流优化技术。基于 EC 的 NAS 将深度神经网络架构视为个体，通过应用遗传算子迭代生成种群，并通过选择算子淘汰性能较差的个体。当满足停止条件时，从存活的种群中选出性能最佳的架构以供使用。在基于 RL 的 NAS 中，训练一个控制器来引导搜索过程。它利用架构的性能作为奖励来更新自身，以便在下一次迭代中搜索更优的架构。基于梯度的 NAS 通常将离散搜索空间松弛为连续空间，并使用梯度下降算法搜索有潜力的架构。需要注意的是，基于梯度的 NAS 被不恰当地宣称为比其他方法更高效。这种误解主要源于该类别中的代表性方法 DARTS [10],，该方法与一种名为权重共享的高效评估方法（EEM）共同设计（将在第三节 C 部分讨论）。原则上，如果所有神经架构搜索算法采用相同的性能评估技术，那么它们的计算复杂度相近。

一般而言，无论采用何种优化算法，在搜索过程中都需要评估许多深度神经网络架构。这是因为这些优化算法具有迭代性，必须预先了解当前性能，才能有效指导下一步的迭代搜索 {v1}。

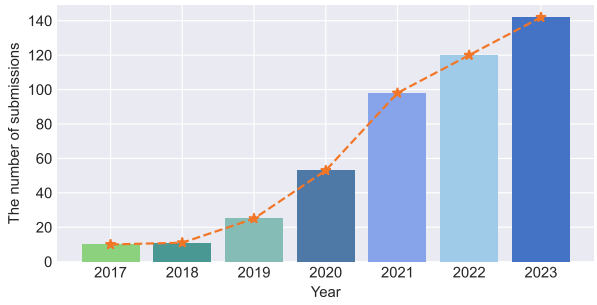

图 1。“投稿”数量指 EEM。统计结果通过谷歌学术按以下步骤检索得出：(1) 选择特定年份；(2) 使用以下关键词进行检索：“提前停止” OR “学习曲线” OR “ 网络形态变换 ” OR “ 权重共享 ” OR “ 单样本 ” OR “DARTS” OR “ 可微 ” OR “ 预测器 ” OR “ 种群记忆 ” OR “ 零成本 ” OR “ 零样本 ” OR “ 免训练 ” AND “架构搜索” OR “架构设计” OR “卷积神经网络” OR “深度学习” OR “深度神经网络”；(3) 详细检查筛选出的论文，确认它们是否属于 EEM。

对于传统评估方法（TEM），神经架构搜索中的性能是通过对每次迭代搜索得到的相应架构进行完整训练来评估的。通常，在小规模数据集（如 CIFAR-10 [11] ）上从头开始训练一个深度神经网络直至收敛，根据深度神经网络的规模，可能需要数小时甚至数天。因此，由于神经架构搜索中通常需要训练数千个深度神经网络，整个神经架构搜索算法会变得计算密集且耗时，难以承受。例如，在 CIFAR-10 上，LargeEvo 算法 [6] 使用了 250 个图形处理器，耗时 11 天。NAS-RL 算法 [8] 使用了 800 个图形处理器，耗时 28 天。更有甚者，RegularizedEvo 算法 [1] 使用了 450 个图形处理器，运行了 7 天。在实践中，对大多数研究人员而言，购买或租用如此规模的图形处理器资源通常难以负担 [12]。因此，如何加速传统评估方法以降低难以承受的计算开销至关重要，这催生了神经架构搜索领域对 EEM 的研究 [3]。具体而言，EEM 指的是在神经架构搜索的整个性能评估过程中，耗时少于传统评估方法的性能评估方法。

据我们所知，最早的 EEM 研究是通过 LargeEvo 算法实现的，该算法于 2016年 在 arXiv 上以预印本形式发布 [6] ，不过也有一些更早的研究可能实现了同样的目标 [13]。随着神经架构搜索的发展，EEM 研究受到越来越多的关注，并逐渐成为当前人工智能领域最热门的研究主题之一。如图 1 所示，2017年 至 2023年 EEM 的投稿数量逐年增加。尽管 EEM 广受关注且至关重要，但目前尚缺乏对相关研究进行系统梳理的综述。这使得相关知识较少的研究人员难以迅速了解这一研究主题的现状和代表性成果。

相关综述 [3]–[5], [14] 主要关注神经架构搜索的发展，对 EEM 的讨论较少。然而，高效评估是神经架构搜索方法的重要组成部分；正如我们上文所述，过去几年间，约有 300 篇论文致力于研究如何通过 EEM 实现高效的神经架构搜索。据我们所知，只有 Liu 等人 [15]对高效神经架构搜索进行了综述。然而，他们的综述缺乏对 EEM 的系统研究。首先，他们遗漏了一些重要的 EEM，例如第三节-D 中讨论的零样本方法，也没有讨论 EEM 的评估标准。其次，他们的分类方法仅考虑神经架构搜索运行过程中 EEM 所耗费的时间，却忽视了预计算时间，而这对于构建 EEM 同样至关重要。本综述旨在系统涵盖所有现有 EEM，并采用一种新颖的分类方法将其划分为四个不同类别，从而为读者提供一种更清晰、更有效的方式来理解现有 EEM 的多样性。总而言之，本文的贡献如下：

- 本综述全面、系统地探讨了 EEM。通过本综述，感兴趣的研究人员可以轻松了解 EMM 的发展历程和分类体系，而 EMM 是人工智能领域最热门的研究主题之一。
- 我们通过全面考察近年来近 200 篇参考文献，进行了深入分析，总结了各种 EEM 的优缺点。本研究可为神经架构搜索研究人员提供宝贵见解，帮助他们快速选择合适的 EEM，并推动高效神经架构搜索的发展。
- 我们分析了各类 EEM 的现状和未来发展方向，指出了 EEM 领域未来需要应对的挑战。这将为研究人员探索具有重要影响力的 EEM 提供指导，并促进自动化机器学习的发展。

下文将介绍第二节中的分类与组织详情。随后，第三节至第五节将综述这些 EEM 的详情，包括方法设计、评估指标和未来方向。最后，我们将在第六节作出总结。

II. 分类与组织

本节首先在II-A小节中给出神经架构搜索的形式化定义。然后，在II-B小节中介绍本文使用的符号与术语。随后，在II-C小节中详细说明EEM的分类。最后，在II-D小节中讨论本文的组织结构。

A. 神经架构搜索的形式化定义

如第I节所述，神经架构搜索的优化问题可表示为公式（1）：

$$\arg\max_{A\in\mathcal{A}} \mathcal{L}(A, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{valid}}) \tag{1}$$

其中，$\mathcal{A}$ 表示由一组神经架构组成的搜索空间，$\mathcal{L}(\cdot)$表示神经架构 A 在训练数据集 $\mathcal{D}_{\text{train}}$上训练后，在验证数据集 $\mathcal{D}_{\text{valid}}$ 上的性能。例如，对于图像分类任务中的深度神经网络，$\mathcal{L}(\cdot)$表示分类准确率。

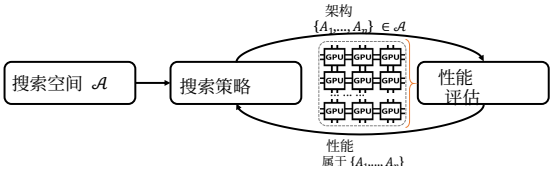

图 2. 神经架构搜索算法的工作流程。

公式（1）中的优化问题可分为三个阶段求解，如图2所示：搜索空间、搜索策略和性能评估。具体而言，搜索过程在预定义搜索空间中进行，搜索策略用于从搜索空间中选出有潜力的架构。然而，搜索策略选出的架构通常没有标签，即其性能未知。因此，我们无法判断该架构是否优秀。性能评估旨在解决这一问题，它可以获取每个架构的 $\mathcal{L}(\cdot)$，从而指导搜索过程。通常，搜索评估过程以循环方式迭代，需要评估大量架构（数百个以上）。对于传统评估方法，这一过程需要在图形处理器上评估每个架构，计算开销大且耗时。因此，EEM 试图减少甚至消除对图形处理器的需求，从而显著加速神经架构搜索算法。

B. 符号与术语

如第 II-A 小节所述，神经架构搜索旨在找到具有良好性能的架构 A 。因此，神经架构搜索的评估过程需要获取每个搜索得到的架构的性能，如公式 (1) 所示。请注意，我们使用 $\mathcal{A}^s$ 表示神经架构搜索算法所有迭代中搜索到的架构。

根据评估过程所消耗的时间，现有评估方法可分为 TEM 和 EEM。为了更好地描述它们，我们首先定义初始化时间、评估时间和运行时间。具体而言，初始化时间是指在图形处理器上评估架构、以预先构建评估方法所需的时间；评估时间则是指直接在图形处理器上评估架构性能所需的总时间。运行时间是初始化时间与评估时间之和，可表示为公式（2）：

$$T = \mathcal{T}_{train}(\mathcal{A}^{in}) + \mathcal{T}^*_{train}(\mathcal{A}^{ev}) \tag{2}$$

其中，$\mathcal{T}_{train}(\cdot)$ 表示训练架构并构建评估方法所需的时间，而 $\mathcal{T}^*_{train}(\cdot)$ 表示仅在图形处理器上评估架构所需的时间。此外，$\mathcal{A}^{in}$ 表示用于构建评估方法的架构，$\mathcal{A}^{ev}$ 则表示需要直接在图形处理器上获取其性能的架构。

TEM 需要在图形处理器上对每个搜索得到的架构进行完整训练；由于无需预计算，其初始化时间为零。具体而言，$\mathcal{A}^{TEM}$ 表示需要进行 TEM 评估的架构数量，等于 $\mathcal{A}^s$。因此，TEM 的运行时间为 $T_{TEM} = \mathcal{T}^*_{train}(\mathcal{A}^{ev})$，取决于 $\mathcal{A}^{TEM} = \mathcal{A}^{ev}$。这表明每个搜索得到的架构都需要经过训练，因此 TEM 耗时较长。

EEM 对于相同的神经架构搜索算法而言，其运行时间比 TEM 更短。这是通过不同的工作原理实现的。一种方式是采用训练加速方法直接缩短评估时间。另一部分则涉及评估少量搜索得到的架构的性能，并以此为基础构建评估方法，这需要消耗初始化时间。具体而言，$\mathcal{A}^{EEM}$ 表示 EEM 中使用的架构。因此，EEM 的运行时间可表示为$T_{EEM} = \mathcal{T}_{train}(\mathcal{A}^{in}) + \mathcal{T}^*_{train}(\mathcal{A}^{ev})$，且满足 $\mathcal{A}^{EEM} = \mathcal{A}^{in} + \mathcal{A}^{ev}$。请注意，$\mathcal{A}^{EEM}$ 与 $\mathcal{A}^s$ 不同，并且可根据不同的 $\mathcal{A}^{EEM}$对 EEM 进行分类。

C. 分类

现有的分类方法存在一些局限性。具体而言，White 等人 [16]将现有的 EEM 分为四类：基于模型、基于学习曲线、零成本代理和权重共享。然而，他们是根据现有 EEM 的工作原理进行分类的。这意味着，一旦出现采用新工作原理的 EEM，这种分类方法可能就不再适用。事实上，第三节 A 中提到的缩小数据集方法和缩小模型方法并不属于这四类。Liu 等人 [15] 提出将现有 EEM 分为基于代理的方法和代理辅助方法。基于代理的方法在评估过程中需要神经网络的权重，而代理辅助方法则恰好相反。这种分类的原因在于，基于代理的方法在神经架构搜索算法运行期间需要额外的计算资源，而代理辅助方法则不需要。尽管这种方法能够稳定地对 EEM 进行分类，但它只考虑了评估时间，而忽略了同样至关重要的初始化时间。因此，现有的分类方法无法准确衡量 EEM 的真实时间复杂度。

在本综述中，我们根据$|\mathcal{A}^{EEM}|$对现有的 EEM 进行分类。本综述的分类方法受到机器学习领域“K-shot learning”概念的启发 [17] 。具体而言，“K-shot learning”指一种机器学习范式，即模型使用每个类别“K”个带标注的样本进行训练。这个术语是一个概括性表述，涵盖零样本学习、单样本学习和少样本学习等多个研究主题。鉴于这一概念已被机器学习社区接受为一种惯例，本综述也采用“shot”一词来描述神经架构搜索过程中带标注的深度神经网络数量，并将 EEM 分为零样本、单样本、少样本和 N-shot 方法。通常，训练过程中使用的带标注样本越多，相应的机器学习算法就越复杂。鉴于此，这种分类方法还可以直观地推断目标 EEM 的时间复杂度，进而帮助相关研究人员轻松选择复杂度适当的 EEM。具体而言，由此得到以下四种类别：

- *N-shot* 评估方法。对于 N-shot 评估方法，训练过的架构数量大于或等于搜索到的架构数量

，即 $|\mathcal{A}^{EEM}| \geq |\mathcal{A}^{s}|$。 N-shot 评估方法仍需训练每个搜索到的架构，主要通过加速这些架构的训练来缩短运行时间，使其少于 TEM。

- 少样本评估方法。对于少样本评估方法，训练的架构数量少于搜索到的架构数量且大于一，即 $|\mathcal{A}^{s}| > |\mathcal{A}^{EEM}| > 1$。由于少样本评估方法训练的架构数量少于搜索到的架构数量，其运行时间自然短于 T_{TEM}。

- 单样本评估方法。对于单样本评估方法，训练的架构数量等于一，即 $|\mathcal{A}^{EEM}| = 1$。由于只需训练一个架构，单样本评估方法的运行时间可能短于 TEM。

- 零样本评估方法。对于零样本评估方法，训练的架构数量等于零，即 $\mathcal{A}^{EEM} = \emptyset$。这种方法不涉及训练，因此成本极低。

我们在此承认，所提出的分类方法存在一定的局限性和潜在偏差。具体而言，EEM 的效率并非仅由在 $\mathcal{A}^{EEM}$ 中训练深度神经网络所花费的时间决定。对于无需训练神经网络的零样本方法，仍需将一个或多个批次的数据输入每个网络，并进行多次前向和反向传播，这需要耗费时间。如果未来的 EEM 中，非训练过程占据了运行时间的大部分，那么这种分类方法的可解释性可能会受到影响。

D. 组织结构

第三节讨论 N-shot 评估方法、少样本评估方法、单样本评估方法和零样本评估方法，重点介绍这些方法的设计原则，并分析其优缺点。第四节详细介绍这些 EEM 的评估，包括评估指标、基准数据集以及在基准数据集上的比较结果。第五节讨论挑战与未来方向。为便于快速浏览感兴趣的部分，图4展示了上述组织结构。

III. 方法

本节讨论四类 EEM 的工作原理和研究现状。对于每种方法，我们首先介绍其主要操作或工作流程。

随后，本文给出了详细的文献实例。最后，我们讨论了该方法的优缺点。

A. N-样本评估方法

如图3所示，对于所有搜索到的架构， N-shot评估方法仍需使用图形处理器在数据集上进行训练和验证。 N-shot方法的不同子类别在不同阶段发挥作用，以加速评估过程。

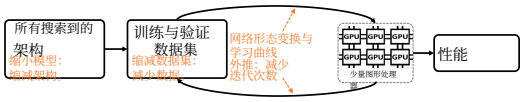

图3. N-样本评估方法的流程图及其与子类别方法的关系。

为了便于更好地讨论，我们首先描述深度神经网络的训练过程。在训练过程中，数据集中的所有样本都会输入模型（即架构）中，完成一次前向传播与反向传播，该过程重复 N_{epoch} 次。训练时间主要取决于三个因素：数据集规模、模型大小和训练轮数。具体而言，完成一次前向传播与反向传播所需的时间取决于数据集规模和模型大小。如果数据集规模或模型大小较大，计算复杂度就会更高，从而需要耗费更多时间进行计算。此外，训练时间还与训练轮数（N_{epoch}）正相关。 N_{epoch}越大，训练时间就越长。

N-样本评估方法主要包括缩小数据集方法、缩小模型方法、网络形态变换和学习曲线外推。请注意，除学习曲线外推外，所有 N-样本评估方法中的$\mathcal{A}^{EEM}$ 均等于 $\mathcal{A}^{s}$（即 $\mathcal{A}^{EEM} = \mathcal{A}^{s}$）。顾名思义，缩小数据集方法旨在缩小数据集规模，以加快训练速度。缩小模型方法则着重于减小模型规模。此外，网络形态变换和学习曲线外推旨在减少训练轮数。

1）缩小数据集方法：与 TEM 类似，缩小数据集方法会训练每个搜索得到的架构，以获取其性能并指导搜索过程。不同之处在于，它们使用缩小后的数据集替代原始数据集，以加快架构训练。自然，缩小数据集方法的运行时间短于 TEM。缩小数据集方法可分为三类：（1）从原始数据集中抽取一个子集；（2）对原始数据集中的每个样本进行下采样；（3）使用具有与原始数据集相似属性的不同且较小的代理数据集。这些方法均不依赖预构建评估方法，其运行时间仅包含评估时间。请注意，“代理数据”、“代理数据集”和“数据代理”是用于筛选特定方法的关键词。

类别（1）： 此类别从原始数据集中采样一个子集，作为缩小后的数据集。由于数据集中的样本数量减少，数据集的整体规模也随之缩小。获取子集最简单的方法是从原始数据集中随机选取一些样本。文献示例：Liu 等人 [18] 从整个数据集中随机选取少量医学图像，用于训练搜索得到的架构。然而，随机采样可能会导致一些具有代表性的样本被移除，从而大幅削弱在子集上训练的架构的泛化能力 [19], [20]。这会导致性能评估不准确。为缓解这一问题，一些研究设计了新的采样方法。例如，Park等人 [19]

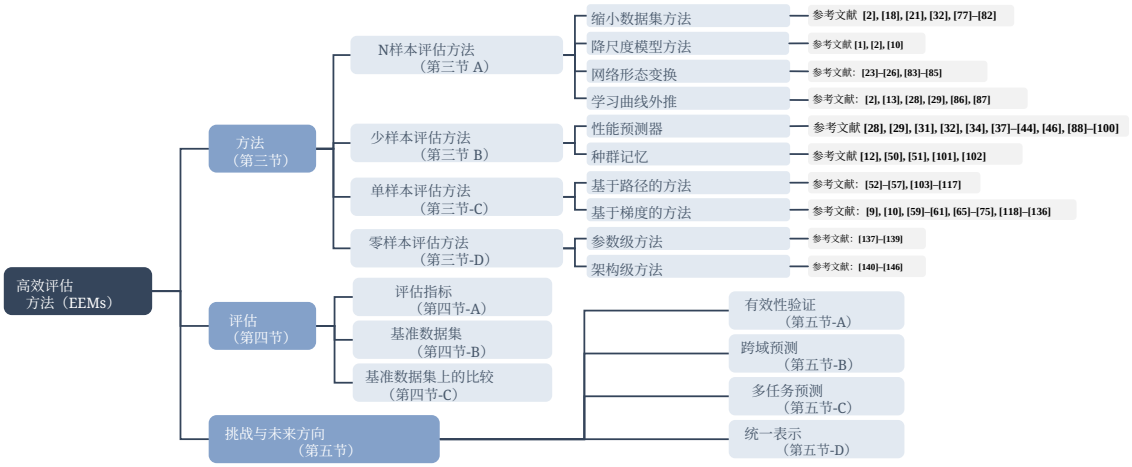

图4。EEM描述的组织结构。

设计了一种探测网络，用于衡量每个样本对架构性能排名的影响。随后，他们移除了对架构性能影响较小的样本。Na等人 [20] 利用数据熵分析了现有的采样方法，并提出了一种新的采样方法，优先选择数据熵分布两端的样本。

类别（2）：该类别会对原始数据集中的每个样本进行下采样，以缩小数据集规模。由于下采样后每个样本的尺寸都会变小，整个数据集规模自然也会缩小。一种典型做法是降低图像分类数据集中图像的分辨率。具体而言，随着图像分辨率降低，图像中的像素数量减少，图像尺寸自然也会变小。因此，降低数据集中图像的分辨率可以缩小数据集规模，从而加快训练。文献示例：Chrabaszcz 等人 [21]提出了 ImageNet 的变体 [22], , 即 ImageNet16x16、ImageNet32x32 和 ImageNet64x64。具体而言，ImageNet16x16、ImageNet32x32 和 ImageNet64x64 使用箱式技术分别将 ImageNet 中的所有图像下采样至 16x16、32x32 和 64x64。

类别（3）：该类别使用一个属性与原始数据集相似、规模更小的代理数据集来训练搜索得到的架构。与原始数据集相比，代理数据集的样本数量和/或样本规模可能更小。文献示例：Zoph 等人 [2] 使用 CIFAR-10 作为 ImageNet 的代理数据集，因为 CIFAR-10 的样本数量和规模都小于 ImageNet，而且目标任务与 ImageNet 相同，都是图像分类。

优点：缩小数据集方法简单直接，易于实现。根据缩小后的数据集与原始数据集的规模，该方法可将计算成本降低一个至数个数量级。

缺点：实际上，在原始数据集上训练的深度神经网络的性能排名，可能与在缩小后的数据集上训练时的性能排名并不一致。这种不一致主要由两个因素造成。(1) 信息丢失：缩小后的数据集不可避免地会丢失原始数据集中的部分信息。这些丢失的信息可能包含影响性能的重要特征和模式。因此，在原始数据集上表现良好的某些模型，其能力可能无法在缩小后的数据集中得到充分体现。(2) 类别不平衡：缩小数据集可能导致各类别中的样本比例失衡。如果缩小过程中未能保留各类别的相对比例，某些类别的样本可能会被过度削减，从而影响模型从这些类别中学习的能力。这会导致模型在缩小后的数据集上训练时性能下降，尤其是在少数类别上，与其在原始数据集上的性能有所偏离。为尽可能避免这些问题，应谨慎设计缩小方法，以维持类别平衡，并避免丢失对性能有重大影响的信息。

2）缩小模型方法：缩小模型方法会在搜索过程中缩小模型大小（例如，搜索一个单元而非整个架构），并在搜索过程结束后重复使用找到的最优单元。与缩小数据集方法类似，缩小模型方法只需评估运行时间。

文献示例：许多研究使用这种方法来加速性能评估过程，我们通过关键词“单元重复次数”和“重复模块”来筛选下文提及的论文。例如，Zoph 等人 [2]提出了一种基于单元的搜索空间（即 NASNet 搜索空间），通过重复堆叠相同的单元结构来构建架构，因此搜索过程中只需找到单元结构。为了缩小模型大小，他们在搜索过程中将每个模型变换为单元重复次数更少、初始卷积单元中的滤波器更少的模型。由于这种方法效率较高，许多研究 [1], [10] 使用了 NASNet 搜索空间或其变体，并采用其缩小模型方法来加速训练。

优点：缩小模型方法也易于实现，并且可以节省 GPU 内存。此外，训练时间会缩短，性能评估所需的运行时间也会少于 T_{TEM}。

缺点：缩小方法通常假设，相同的缩小操作对不同模型的

性能影响相同。只有当所有模型的性能变化程度相同时，缩小后的模型与原始模型的性能排名才会保持一致。然而，这一假设并非总是成立。例如，从高复杂度模型中移除一个卷积操作，可能会提升其性能，因为该操作可能是冗余的。而从低复杂度模型中移除相同的卷积操作，则可能会降低其性能，因为这会移除一个可学习操作。因此，这些方法可能会降低缩小后的模型与原始模型性能之间的相关性。未来，这种方法应从理论上证明原始模型与缩小后的模型之间性能排名的相关性。这将有助于开发更严谨的缩小模型方法，从而避免在缩小原始模型时仅依赖直觉。

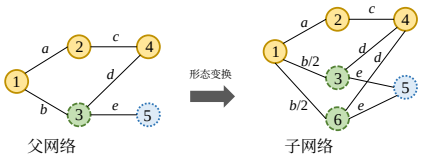

图5. 宽度形变示例，不同类型的节点表示不同的操作（例如卷积层、池化层）。边上的标签表示相应的权重值。父网络是一个训练完成的网络，子网络则是通过宽度形变操作从父网络形变而来。具体而言，在子网络中，节点#6是节点#3的副本，节点#3的权重也被复制到节点#6。为保持两个网络的函数不变，权重 b 的值除以二，而权重 a、 c、 d和 e 的值保持不变。这样，分别向父网络和子网络输入相同的输入时，两个网络的输出将相同，这意味着父网络的函数在子网络中得以保留。

3）网络形态变换：网络形态变换通过减少训练轮数，可以加快对大多数搜索到的架构的评估过程，因此，网络形态变换的运行时间成本仅为评估时间。请注意，并非搜索得到的架构的每个训练过程都能加速，原因将在本小节第三段中说明。由于大多数搜索到的架构的训练时间都缩短了，网络形态变换的运行时间显著短于 TEM。网络形态变换最初是在迁移学习的背景下提出的，目的是将知识从一个训练完成的网络（即父网络）快速迁移到另一个网络（即子网络）。由于能有效加快子网络的训练，它很快便被应用于神经架构搜索领域。为便于理解，我们先解释网络形态变换的加速原理，然后讨论其如何作为一种 EEM 应用于神经架构搜索。

术语形态变换在数学上指从一种结构到另一种同类型结构的保结构映射。网络形态变换是指一种函数保持操作，它能将训练完成的父网络变换为一个能够完全保留父网络功能的子网络 [23]。具体来说，首先通过预定义的形变操作将父网络变换为不同的子网络。然后，子网络直接继承其权重

与父网络具有相同的功能和输出。之后，通常会在数据集上训练子网络。由于子网络无需从头开始训练，因此训练至收敛所需的训练轮数自然会减少。由于继承过程不需要大量资源，网络形态变换可以缩短子网络的训练时间。形变操作（即函数保持操作）主要包括宽度形变、深度形态变换、卷积核尺寸形变等。不同类型的形变会改变父网络的不同部分，以生成新的子网络。为便于理解，图5展示了宽度形变的示例。顾名思义，子网络比父网络更宽。不过，子网络和父网络执行的功能仍然相同。

通常，网络形态变换无法用于加速神经架构搜索中的评估，因为使用网络形态变换必须存在经过充分训练的父网络。为此，搜索策略第一次迭代中搜索到的架构通常会经过充分训练，作为下一次迭代中搜索到的架构的父网络。具体而言，神经架构搜索第一次迭代中搜索到的架构会经过充分训练，作为父网络。然后，对这些父网络应用各种形变操作，以生成子网络。这些子网络从父网络继承权重，并通过较少的训练轮数进行训练，以获得其性能。之后，这些子网络会被视为父网络，并迭代执行上述过程。由于除第一次迭代搜索到的架构外，所有搜索到的架构所需的训练轮数都少于 TEM，因此网络形态变换比 T_{TEM} 消耗的运行时间更少。

文献示例：许多研究采用网络形态变换来加速性能评估过程，其关键词是“网络形态变换”和“网络变换”。例如，Cai 等人 [24] 采用基于强化学习的搜索策略，并利用控制器生成形变操作，将其应用于当前网络以进行搜索。设计的形变操作包括一种称为 Net2Wider 的宽度形变，以及一种称为 Net2Deeper 的深度形态变换。Net2Wider 可以用更宽的层替换某一层，例如为卷积层增加更多滤波器；Net2Deeper 则在两层之间插入一个初始化为恒等映射的新层。随后，新搜索得到的架构继承当前网络的权重，并通过更少的训练轮数进行训练，以获得其性能。Cai 等人 [25] 提出了路径级形态变换操作，因为此前的工作采用层级形态变换，只能增加滤波器或添加层。路径级形态变换可以修改父网络的路径拓扑结构，同时保持其功能。需要指出的是，此前的形变操作只能增大网络规模，因为缩小网络规模时无法保证功能保持属性。这并不适用于某些计算资源受限的场景。为解决这一问题，Elsken 等人 [26] 设计了近似网络形态变换，能够缩小网络规模并大致保持其功能。随后，他们采用基于进化计算的搜索策略，并将近似网络

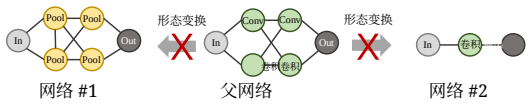

图6. 展示网络形态变换局限性的示例。网络 #1、网络 #2 和父网络来自同一搜索空间，且父网络是一个训练完成的网络。网络 #1 和网络 #2 无法由父网络通过形态变换得到。具体而言，网络 #1 无法由具有不同操作的父网络通过形态变换得到。这是因为父网络与网络 #1 之间的操作类型（输入节点和输出节点除外）不同。网络形态变换只会修改神经网络的拓扑结构，而不会改变现有神经元的操作类型。网络 #2 无法由父网络通过形态变换得到，因为具有四个卷积层的父网络的函数不可能与仅具有一个卷积层的网络 #2 相同。

形态变换作为变异算子来搜索架构。

优势： 网络形态变换方法可以充分利用父网络中的大量知识，并且具有更多理论基础。

劣势： 搜索空间中的并非所有网络都可以由训练完成的父网络通过变换得到。在这种情况下，网络形态变换方法将会失效。具体而言，现有形变操作（例如深度形变、宽度形变、卷积核尺寸形变和子网形变）所能生成的神经架构是有限的。如图6所示，并非所有潜在架构都能在搜索过程中得到探索，从而限制了搜索到的架构的多样性和创新性。此外，网络形态变换可能导致搜索过程陷入局部最优。这是因为通过对父网络进行形变可能无法获得全局最优解，从而使算法陷入局部最优。具体而言，假设图6中的网络 #2 表示全局最优解，而父网络是当前解，那么显然，仅依赖形变操作无法达到最终的全局最优解。这些局限性可能限制对搜索空间的探索，并阻碍 NAS 找到最优网络架构。总体而言，网络形态变换的局限性源于现有形变操作缺乏多样性。未来的一个方向是探索更多类型的形变操作，以提升形变网络的多样性。通过这种方式，可以在一定程度上缓解网络形态变换的不灵活性。

4）学习曲线外推：不同于其他 N- shot 评估方法（其中 $\mathcal{A}^{EEM}$ is equal to $\mathcal{A}^{s}$），在由 $\mathcal{A}^{s}$ 和 $\mathcal{A}^{f}$ 组成的学习曲线外推方法中，即 $\mathcal{A}^{EEM}=\mathcal{A}^{s}\cup\mathcal{A}^{f}$，$\mathcal{A}^{f}$ 是一个架构集合，其中包含一些经过完整训练、用于学习曲线外推的架构。学习曲线外推方法在 $\mathcal{A}^{f}$ 上训练一个模型，以预测仅经过少量训练轮数后训练的架构的实际性能。通过这种方式，由于训练轮数减少，搜索到的架构的训练时间大幅缩短。由于在 $\mathcal{A}^{s}$ 中训练架构所节省的时间通常大于在 $\mathcal{A}^{f}$ 中完整训练架构所花费的时间，因此，与 TEM 相比，该方法在性能评估上仍然可以耗费更少的时间。

为说明学习曲线外推的原理，我们首先介绍三个术语：学习曲线、部分学习曲线和最终学习曲线。具体而言，学习曲线通常指迭代机器学习算法的性能随迭代次数增加而变化的函数。我们用“最终学习曲线”表示算法从训练开始到结束的完整学习曲线 $f_t=(p_1,\ p_2,\ \ldots,\ p_t)$，其中 p_i表示迭代 i 时的性能，p_t 则是最终性能。此外，我们用“部分学习曲线”指截至训练轮次 l时观察到的学习曲线，即 $f_l=(p_1,\ p_2,\ \ldots,\ p_l)(l<t)$。

学习曲线外推方法的工作流程如下。首先，从预定义搜索空间中采样一些架构，然后对其进行完整训练，以构建架构集合 $\mathcal{A}^{f}=\{(x^1,\ f_t^1)、(x^2,\ f_t^2)\ldots\ldots、(x^k,\ f_t^k)\}$，其中 x^i 表示第 i 个架构，f_t^i 表示其最终学习曲线。随后，将该架构集合变换为另一种形式$\mathcal{A}^{f}=\{(x^1,\ f_l^1,\ p_t^1)、(x^2,\ f_l^2,\ p_t^2)\ldots\ldots、(x^k,\ f_l^k,\ p_t^k)\}$，其中 f_l^i 表示部分学习曲线，p_t^i 表示架构 x^i 的最终性能。学习曲线外推方法通过在 $\mathcal{A}^{f}$ 上训练模型 $\mathcal{P}'(x^i,\ f_l^i)$ 来构建该模型。模型 $\mathcal{P}'(\cdot)$ 可以通过输入搜索到的架构及其部分学习曲线，预测这些架构的最终性能。最后，在获得搜索到的架构的部分学习曲线后，使用该模型预测其最终性能。学习曲线外推方法的运行时间包括初始化时间（用于构建预测模型）和评估时间（用于获取部分学习曲线）。

值得一提的是，提前停止策略是一种常见的 EEM，被视为学习曲线外推的特例。具体而言，提前停止策略直接将仅经过几个训练轮次后获得的性能视为最终性能。我们可以认为，提前停止策略针对部分学习曲线 $f_l=(p_1,\ p_2,\ \ldots\ldots,\ p_l)\ (l<t)$ 构建了一个函数 $p_t=f_l(t)=p_l$，用于预测最终性能 p_t。由于无需训练架构来构建 p_t，对于提前停止策略而言，$\mathcal{A}^{EEM}$ 等于 $\mathcal{A}^{s}$。

文献示例：我们用于筛选学习曲线外推研究的关键词是“部分学习曲线”“部分训练”和“提前停止”。学习曲线外推最初被提出，是为了自动为神经网络找到性能良好的超参数配置 [27]。由于超参数优化和神经架构搜索都面临网络评估成本高的问题，学习曲线外推很快便被应用于神经架构搜索。例如，Rawal 等人 [28] 开发了一种由编码器循环神经网络和解码器循环神经网络组成的长短期记忆网络（LSTM），用于预测部分训练模型的最终性能。他们将前 10 个训练轮次的验证损失输入编码器，并通过解码器获得最终验证损失。Baker 等人 [29]使用一组支持向量机，通过输入模型架构的特征、训练超参数和部分学习曲线来预测架构性能。此外，他们将估计过程建模为高斯扰动，以计算概率 $p(\hat{y}\leq y_{best})$，其中 $\hat{y}$ 表示估计的最终性能，y_{best} 表示最佳模型的性能

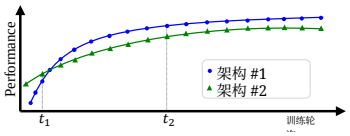

图 7. 提前停止策略导致预测不准确的示例。图中显示了架构 #1 和架构 #2 在每个训练轮次的性能值。在 t_1 时，架构 #1 和架构 #2 的性能排名与 t_2 时的最终性能排名不一致。这是因为架构 #2 在 t_1 时已接近收敛，而架构 #1 仍需进一步训练。因此，当两个架构都收敛时，架构 #1 的性能高于架构 #2。

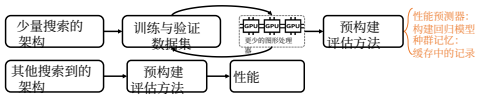

图 8. 少样本评估方法的流程图及其与子类别方法的关系。“Eval.”是“评估”的缩写。

架构。当使用 $p(\hat{y} \leq y_{best}) \geq \Delta$，其中 Δ 用于权衡加速幅度的提升与过早终止优良架构的风险时，该架构的训练便会终止。Wistuba 等人 [30] 构建了一个神经网络，以架构、部分学习曲线、数据集和超参数为输入，对架构进行排序。通过这种方式，他们可以搜索最优架构。至于提前停止策略，它很容易嵌入神经架构搜索算法中。例如，Zoph等人 [2] 只对每种架构训练 20 个训练轮次，以降低搜索成本。

优点： 学习曲线外推可以减少训练轮数，从而加快训练过程。它易于与不同的神经架构搜索算法结合，无需修改数据或模型。因此，它具有很高的灵活性。

缺点： 学习曲线外推需要对一些架构进行完整训练，以此构建预测模型的数据集。数据量不能太小，否则预测模型的泛化能力可能较差。因此，收集数据集仍然需要耗费大量时间。例如，Baker 等人 [29]分别为残差网络、MetaQNN 和长短期记忆网络构建了三个架构数据集，数据量分别为 1,000、1,000 和 300。训练这些架构耗时较长。此外，提前停止策略假设通过部分训练的架构得到的性能排名与实际性能排名一致。然而，这一假设并非总是成立，因为不同网络的收敛速度并不一致，如图 7 所示。因此，提前停止策略可能会导致性能估计不准确。事实上，Zoph 等人 [2]也在 CIFAR-10 上将性能最高的前 250 个架构重新训练至收敛，以便在搜索后找出最佳架构。这证实了提前停止策略存在不准确性。

B. 少样本评估方法

如图 8 所示，少样本评估方法首先需要训练少量架构（即 $|\mathcal{A}^{EEM}| < |\mathcal{A}^{s}|$），以预先构建评估方法。然后，通过该评估方法获得其他架构的性能。这种方法的运行时间可能比 TEM 更短。根据所采用的评估方法，现有的少样本方法可分为性能预测器 和 种群记忆。所有这些方法都需要先构建评估方法，再用它评估搜索得到的架构的性能。因此，它们的运行时间包括初始化时间和评估时间。

1) 性能预测器：性能预测器是一种回归模型，经过在架构集合 $\mathcal{A}^{EEM} = \{$ 上训练后，可以直接预测 $\mathcal{A}^{s}$ 中架构的性能（A_i，P_i），其中 A_i 表示第 i 个架构，P_i 表示其实际性能。由于 $|\mathcal{A}^{EEM}|$ 通常小于 $|\mathcal{A}^{s}|$，且预测 $\mathcal{A}^{s}$ 中架构性能所需的时间可以忽略不计，因此性能预测器消耗的运行时间会少于 TEM。构建性能预测器的主要步骤如下：

步骤 1 对预定义搜索空间中的若干架构进行采样和训练，将其作为架构集合；**步骤2** 对架构集合中的架构进行编码；**步骤 3** 训练性能预测器，将架构的编码映射为相应的性能值；

各步骤的文献示例： 本部分将分别讨论现有研究中有关步骤 1-3 的工作，所选论文的关键词为“性能预测器”或“神经预测器”。

步骤 1： 研究人员通常从搜索空间中随机采样大量架构，并从头开始训练这些架构，以构建架构集合。然而，通过随机采样获得的集合可能无法涵盖所有具有代表性的架构，从而导致性能预测器的泛化能力较弱。因此，一些研究采用其他采样方法来解决这一问题。例如，Dai 等。[31]使用准蒙特卡洛从搜索空间中采样架构。Hassantabar 等 [32]选择 Sobol 序列从搜索空间中采样架构，因为这种序列能够生成分布更加均匀的样本。这些研究的目标都是生成一个能够充分代表整个搜索空间的小型架构集合，从而最大限度地减少需要训练的架构数量。

步骤 2： 步骤 1 中获得的架构集合无法直接输入回归模型，必须将其编码为回归模型能够处理的形式。实际上，编码架构有两种常用方式：基于序列的方案和基于图的方案 [33]。

基于序列的方案对架构的特定序列化信息进行编码，并将整个架构展平为字符串。这种编码方案广泛应用于各种搜索空间。例如，Deng 等 [34]采用基于层的搜索空间，提出了统一层编码，将每一层编码为数值向量，并使用擅长处理序列数据的长短期记忆网络（LSTM），将层序列中的信息整合为一个最终字符串。Sun 等 [35]提出了一种由 ResNet 块、DenseNet 块和池化块组成的基于块的搜索空间，并根据每个块的卷积核大小、输入通道和输出通道按序进行编码，

最终得到一个字符串。对于基于单元的搜索空间，Luo 等 [36] 使用输入层的标识符和所应用操作的名称，将每一层编码为字符串标记。随后，使用离散标记序列来描述给定的架构。基于序列的方案易于实现且简单直接。然而，它只能隐式地对给定架构的拓扑信息建模，因此性能预测器难以从编码中捕获拓扑信息。结果，对于拓扑信息丰富的某些架构，这种方案可能会降低性能预测器的预测能力。

基于图的方案通常将架构视为有向无环图（DAG），并显式表示架构的拓扑信息。例如，Wen等人 [37] 使用邻接矩阵描述操作之间的连接，并使用独热编码表示操作类型。为编码更多信息，Xu等人 [38]获取了每种操作的向量、浮点运算次数（FLOPs）以及参数大小。随后，这些向量分别广播到邻接矩阵中，以生成类型矩阵、FLOP 矩阵和参数矩阵。最后，将这些矩阵拼接为最终编码。Ning等人 [33] 将操作建模为传播信息的变换，以模拟架构的数据处理。尽管基于图的方案能够更好地编码拓扑信息，但它不适用于层式和基于块的搜索空间，因为这类搜索空间几乎都是线性结构。引入拓扑信息可能会为其带来冗余信息 [39]。

步骤 3： 性能预测器本质上可视为回归模型，其训练过程可表示为公式 (3)：

$$\min_{T_p} \mathcal{L}\left(R\left(T_p, Encoder(X)\right), y\right) \tag{3}$$

其中， T_p 是回归模型R的可训练参数， $\mathcal{L}(\cdot)$表示损失函数。 $Encoder(\cdot)$表示编码方法。 X 和 y 分别对应架构集合中的架构及其性能。接下来，我们从回归模型和损失函数两个方面介绍性能预测器的训练过程。

对于回归模型，我们根据编码方案进行说明，因为不同模型擅长处理不同类型的数据。首先，对于基于序列的方案，通常会广泛使用适合处理序列数据的模型，例如长短期记忆网络 [34], [40], 循环神经网络 [41], Transformer。此外，由于具备区分特征重要性的能力 [39]，也会使用一些非神经模型，例如随机森林和梯度提升决策树。对于基于图的方案，通常使用图神经网络，因为它们在处理图数据方面具有优势 [37]。

关于损失函数，主要可分为两种类型。第一种是逐元素损失函数，用于计算每个样本的预测标签与真实标签之间的距离。最常用的是均方误差（MSE），如公式（4）所示：

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2 \tag{4}$$

其中， x_i 和 y_i 分别表示架构及其对应的性能， N 表示样本数量。 $f(\cdot)$ 表示性能预测器。Huber损失也是一种常用的逐元素损失函数，如公式（5）所示。它可以防止模型受到离群值的主导 [31]。

$$\mathcal{L}_{Huber} = \begin{cases} \frac{1}{2}(y_i - f(x_i))^2 & \text{If } |y_i - f(x_i)| \le \delta \\ \delta|y_i - f(x_i)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{5}$$

在公式（5）中， δ 可以根据具体情况设定。当预测偏差小于 δ时， $\mathcal{L}_{Huber}$ 使用平方误差；否则，使用线性误差。第二类损失函数是成对排序损失函数。它最初用于 [38]中的性能预测器。这是因为对于神经架构搜索而言，架构之间的性能排名比每个架构的绝对性能更重要。成对排序损失函数更关注性能排名。具体而言，典型的铰链成对排序损失如公式（6）所示：

$$\mathcal{L}_{pairwise} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|T_i|}\sum_{j\in T_i}\mathrm{Max}\left(0, m - (y_i - y_j)\right) \tag{6}$$

其中， $T_i = \{j|y_i > y_j\}$。 m 是比较间隔。

基于分类的文献示例： 根据神经架构搜索过程中是否会重新训练性能预测器以进行更新，性能预测器可分为在线预测器和离线预测器。

对于离线预测器，架构-性能对在训练预测器之前就已全部收集完毕。请注意，离线性能预测器的架构集合 $\mathcal{A}^{EEM}$ 并非从搜索得到的架构 $\mathcal{A}^s$中采样而来。预测器开始工作后，训练集中不会再加入新样本。许多研究采用这种方式训练预测器，因为它简单易行。例如，Wen 等人 [37] 直接使用少量架构训练图卷积网络（GCN）模型，将其用作性能预测器。然而，离线预测器的性能很大程度上取决于所收集数据的质量。如果数据质量较差，预测器的性能也可能不佳。此外，一旦数据集中加入新样本，就必须重新训练离线预测器，这使其缺乏灵活性。

与离线预测器不同，在线预测器在经过训练并投入使用后，可以添加新样本来提升性能。因此，在线预测器比离线预测器更灵活、实用。大多数在线预测器会先训练少量架构来构建预测器。然后，选择一些未经训练的架构，并由预测器预测其性能。最后，训练预测性能排名前-k 的架构，并将其添加到架构集合中，以重新训练预测器。因此， $\mathcal{A}^{EEM}$ 中的架构通常部分取样自 $\mathcal{A}^s$ ，因为新添加的架构通常由神经架构搜索算法找到。请注意，在线预测器的运行时间还包括更新时间，此外还包括第II-B小节中提到的初始化时间和评估时间

。具体而言，更新时间指重新训练离线预测器所需的时间。许多研究采用了在线性能预测器。例如，Liu 等人 [41] 从简单架构开始训练预测器，并持续使用该预测器评估更复杂的架构。随后，他们挑选并训练最有潜力的架构，将其作为新样本来微调预测器。Wei等人 [42] 提出了一种基于进化计算的搜索策略，并使用初始种群的架构-性能对训练预测器。随后，他们训练下一代种群中的架构，并将其加入架构集合，以重新训练性能预测器。

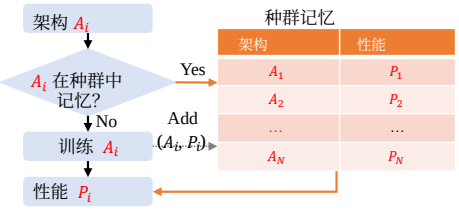

图 9. 种群记忆的工作流程。具体而言，对于每个搜索得到的架构 A_i，如果它已存储在种群记忆中（其中保存了架构及其对应的性能），我们就可以直接获取其性能 P_i。否则，将对其进行训练以获得性能 P_i，随后将 (A_i, P_i) 存储在种群记忆中。

根据架构是否都有完整标签，现有研究也可分为监督式和半监督式性能预测器。

监督式性能预测器完全使用带标签的架构来训练预测器。由于这种方法简单，大多数预测器都属于监督式。然而，带标签的架构数量通常有限，因为架构标注的成本很高。因此，监督式性能预测器通常会尝试从有限的架构中提取更有意义的特征，以提升其性能。例如，Chen 等人 [43] 在预测器中提出了一种操作自适应注意力模块，用于捕捉不同操作之间的相对重要性。Ning 等人 [33]设计了一种编码方法，可对神经架构的计算过程建模，从而从架构中提取更有意义的信息。

与监督式预测器不同，半监督式预测器不仅使用带标签的架构，也会利用无标签架构。这是因为大量无标签架构能够提供宝贵的信息，以优化性能预测器。例如，Tang等人 [44] 使用自动编码器从带标签和无标签的架构中学习有意义的特征，并构建关联图来捕捉内部相似性。随后，通过图卷积网络预测器将关联图和提取的特征映射为性能值。Luo等人 [45] 使用一小组已标注架构训练性能预测器。随后，他们使用该预测器来预测

无标签架构。随后使用这些预测得到的架构重新训练性能预测器。

优点：近年来，性能预测器已成为 EEM 研究的热点之一 [34], [41], [46]。由于无需在图形处理器上评估搜索得到的架构，因此其评估速度很快。

缺点： 性能预测器面临的一个关键问题是，构建性能预测器需要大量完全训练的架构。这一问题的矛盾在于，训练大量架构可能与 EEM 的设计意图相冲突，而架构数量过少则可能导致过拟合问题 [47]。在实践中，性能预测器始终面临数据不足的问题。现有工作主要从三个方面解决这一问题：编码方法、模型和数据。在编码方法方面，研究人员致力于通过编码方法表示架构中更多有用的信息，例如，Xu等 [38] 提出了一种基于图的编码方法，用于提取架构中更有价值的信息。具体而言，他们不仅对每个节点的拓扑信息和操作类型进行了编码，还对每个节点的浮点运算次数和参数进行了编码。在模型方面，许多工作关注如何从编码中提取更有意义的特征。例如，Wang等 [48] 提出了一种多阶段多层感知机模型，使用不同的模型来预测性能值处于不同范围内的架构。在数据方面，Liu等 [49] 提出了一种针对架构的同质增强方法，以可负担的方式生成大量带标签数据。尽管已经提出了许多解决方案，但作为机器学习中的一个普遍问题，带标签架构的缺乏无法从根本上得到解决。随着大语言模型不断发展，性能预测器的研究可以将重点转向利用自监督学习方法，为神经网络架构创建大型预测模型。这项研究将通过独立的数据收集，大幅提升性能预测器的泛化能力，并消除针对每个搜索空间分别构建性能预测器的必要性。

2）种群记忆：种群记忆常用于加速进化计算中的适应度评估。随后，它被用于基于进化计算的神经架构搜索，以避免重复评估相同的架构。这是因为前几代种群中的个体可能会出现在后续种群中，重新评估这些个体会浪费计算资源。种群记忆可以看作一种缓存系统，主要通过重复利用已有的架构信息来缩短耗时。对于种群记忆而言，经过训练的架构集合 $\mathcal{A}^{EEM}$ 是搜索得到的架构集合 $\mathcal{A}^{s}$ 的子集（即 $\mathcal{A}^{EEM} \in \mathcal{A}^{s}$）。种群记忆的工作流程如图 9 所示。这样，当遇到与先前相同的架构时，我们就可以避免重复评估。许多基于进化计算的神经架构搜索都采用了这种方法，以避免不必要的开销。

文献示例：我们使用“种群记忆”和“适应度记忆”作为关键词来筛选特定论文。Fujino 等人 [50] 使用种群记忆来存储个体的性能值，并在出现与先前架构具有相同编码的架构时从内存中检索这些值。同样，Sun 等人 [51]设计了一个全局缓存系统，用于记录架构的哈希码和性能。他们获得了

当遇到编码已存储在缓存中的架构时所对应的性能。

优势：与其他 EEM 相比，种群记忆能够更准确地评估性能，因为通过种群记忆获得的性能等于实际性能。

劣势：种群记忆方法的缺点是缺乏灵活性。具体而言，它无法用于评估此前未训练过的架构。不过在实践中，重复架构仅占搜索到的所有架构中的一小部分 [51]。

C. 单次评估方法

如图 10 所示，单次方法在整个搜索过程中只需训练一个神经网络架构（即 $|\mathcal{A}^{EEM}| = 1$）。在神经架构搜索领域，这种方法也称为权重共享方法。由于只需训练一个网络（也称为“超网络”），因此这种方法能够节省成本。根据架构搜索与超网络训练是否耦合，现有的单次评估方法可分为基于路径的方法和基于梯度的方法。这些方法都不依赖预构建模型，运行时间仅包含评估时间。

1）基于路径的方法：基于路径的方法会训练一个包含所有候选架构（即子网）的超网络。子网的权重可直接从超网络中提取。每条路径对应超网络中的一个子网。评估子网的工作流程如图 11 所示。基于路径的方法的主要工作流程如下：
步骤1 设计一个涵盖所有候选架构的超网络；**步骤2** 通过路径采样策略从头开始训练超网络；**步骤3** 通过继承超网络的权重来预测搜索得到的架构，并在验证数据集上推断其性能。

在这些步骤中，超网络的训练（即步骤 2）至关重要。这是因为超网络的权重是通过训练获得的，而子网的权重则从超网络中提取。在实践中，超网络的权重几乎不会被直接优化。原因是

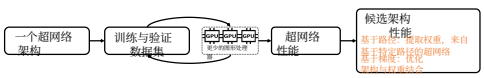

图 10. 单样本评估方法的流程图及其与子类别方法的关系。

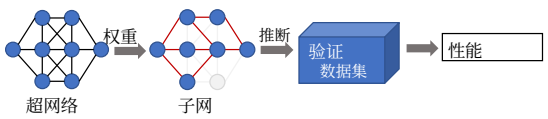

图 11. 基于路径的方法评估子网的工作流程。子网中的灰色连线和节点分别表示非活动连接和非活动节点。只有超网络的权重是通过训练获得的。子网的权重从超网络中提取。随后，可通过在验证数据集上进行推断，获得这些子网的性能。

超网络中的权重与超网络架构深度耦合。因此，超网络对架构 [52], [53] 的变化并不鲁棒。这可能导致继承权重所得的性能与实际性能之间的相关性较低。

文献示例：研究者提出了多种路径采样策略，用于在超网络训练的每个训练轮次中决定训练哪些路径，从而将超网络的权重与架构解耦；我们用于筛选研究的关键词是“one-shot”和“single path”。例如，Bender 等人 [52]提出在每次迭代中随机丢弃一部分操作。丢弃概率由一个称为丢弃率的参数决定。然而， [52]中的实验表明，训练对丢弃率很敏感，这使得超网络训练变得复杂。Guo 等人 [53]提出在超网络中随机采样一个子网，并且每次迭代只更新该子网的权重。这种方法无超参数，因此避免了路径丢弃策略对丢弃率敏感这一缺陷。Chu 等人 [54] 指出，Guo 等人所提方法固有的不公平性可能导致评估偏差。因此，他们提出了严格公平采样策略。该策略在每个训练轮次采样 m （m 是每层选择块的数量）。在该策略中，所有选择块仅激活一次，并且只使用一个批次的数据进行优化。Zhang 等人 [55]设计了新颖性驱动采样策略。该策略只优化通过新颖性搜索采样得到的架构的权重。You 等人 [56] 提出了一种带拒绝机制的多路径采样策略，通过在部分验证数据集上进行评估来筛除较弱的路径。在超网络训练过程中，只训练那些有潜力的路径。Zhang 等人 [57]认识到，计算资源不同的子网可能需要不同的训练策略。为此，他们提出了概率偏移，并根据每个子网的训练充分性对其进行学习。该概率偏移能够自动调整采样概率，从而优化超网络的训练过程。

优点：基于路径的方法只需训练一个架构（即 $|\mathcal{A}^{EEM}| = 1$），且在验证数据集上的推理速度很快，因此能够大幅加快性能评估。

缺点： 尽管借助路径采样策略，基于路径的方法提高了预测准确率，但仍面临多模型遗忘问题 [58]。具体而言，多模型遗忘问题是指在训练后续子网时，先前训练的子网性能下降。这是因为逐个训练具有部分共享权重的后续子网时，共享权重会被覆盖。图 12 展示了该问题的一个示例。多模型遗忘问题会降低预测能力。因此，尽管基于路径的方法能够大幅减少计算时间，却可能导致性能排名不可靠。已有许多方法被提出以缓解多模型遗忘问题。例如，Benyahia 等人 [58] 提出了一种具有统计学依据的权重可塑性损失，用于对共享参数的优化进行正则化。通过这种方式，他们能够

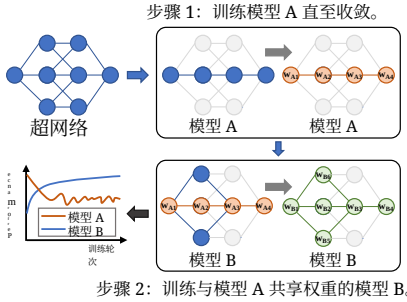

图 12. 多模型遗忘问题示例。模型 A 和模型 B 是两个不同的子网络，它们共享超网络中的部分权重。在超网络训练过程中，首先在步骤 1 训练模型 A，然后在步骤 2 训练模型 B。这会导致多模型遗忘问题。具体而言，步骤 1 中，模型 A 的权重在超网络中得到优化。随后，在步骤 2 训练与模型 A 共享部分权重的模型 B。训练模型 B 时，模型 A 的性能会下降。这种现象称为多模型遗忘。

保留对先前模型至关重要的共享参数。Zhang 等人 [59] 将超网络训练表述为一个约束持续学习优化问题。在当前子网的训练过程中，他们约束所有先前子网的训练损失，使其低于上一步的损失。然而，随着先前子网数量增加，约束的数量也会线性增长。因此，他们提出了一种贪心新颖性搜索方法，从先前子网对应的现有约束中选取一个子集，以解决多模型遗忘问题。尽管上述工作能够缓解多模型遗忘现象，但无法从根本上解决这一问题。这是因为该问题源于基于路径的方法训练超网络以评估子网的机制。因此，基于路径的方法所得性能与实际性能之间的排序相关性可能较差。

2）基于梯度的方法：与基于路径的方法不同，基于梯度的方法将超网络训练与架构搜索结合起来。具体而言，这些方法将离散搜索空间放宽为连续空间，并通过基于梯度的搜索策略联合优化架构和超网络权重。此类方法中最著名的工作是 DARTS [10]，于 2019年提出。此后，DARTS 凭借其有效性成为基于梯度的方法中的主流方法。具体而言，DARTS 使用基于单元的搜索空间，并将超网络视为具有N 个节点的有向无环图。每个节点 $x^{(i)}$ 都是一种潜在表示，例如特征图；每条边（i，j）表示一种操作 $o^{(i,j)}$，该操作可以变换 x^{i}。如公式（7）所示，每个中间节点都基于其所有前驱节点计算得出：

$$x^{(j)}=\sum_{i<j} o^{(i,j)}\left(x^{(i)}\right) \tag{7}$$

为了将离散搜索空间放宽为连续空间，如公式 (8) 所示，通过对所有可能的操作应用 softmax 函数，可以对特定操作的类别选择进行连续放宽：

$$\bar{o}^{(i,j)}(x)=\sum_{o\in\mathcal{O}}\frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o'\in\mathcal{O}}\exp\left(\alpha_{o'}^{(i,j)}\right)}o(x) \tag{8}$$

其中，$\mathcal{O}$ 是一组候选操作。$\alpha^{(i,j)}$ 表示节点对（i，j）对应的操作混合权重（即架构参数）。通过将每个混合操作 $o^(i,j)$ 替换为最可能的操作，即 $o^{(i,j)}=\text{argmax}_{o\in\mathcal{O}}\alpha_o^{(i,j)}$，即可得到一个架构。随后，DARTS 旨在联合学习架构参数 α 和权重ω，它们决定训练损失 $\mathcal{L}_{train}$ 和验证损失$\mathcal{L}_{val}$。这可以表述为一个双层优化问题，其中 α是上层变量，而 ω 是下层变量，如公式 (9) 所示：

$$\begin{cases}\min_{\alpha}\mathcal{L}_{val}\left(w^*(\alpha),\alpha\right)\\ \text{s.t. } w^*(\alpha)=\text{argmin}_w\,\mathcal{L}_{train}(w,\alpha)\end{cases} \tag{9}$$

通过这种方式，我们可以找到使 $\mathcal{L}_{val}(w^*(\alpha)$ $\alpha)$ 最小化的 α^*，其中权重 ω^* 则使 $\mathcal{L}_{train}(w,$ $\alpha)$ 最小化。

文献示例：许多基于梯度的方法被提出以提高有效性。我们使用“可微神经架构搜索”“可微架构搜索”和“可微神经架构搜索”作为关键词来筛选论文。这些论文主要聚焦于以下三个研究方向：

- **降低内存消耗：**许多研究试图降低基于梯度的方法的内存消耗。例如，Cai 等人 [60]利用二值化架构参数，确保运行时内存中仅有一条激活路径处于激活状态。Xu 等人 [61] 随机采样一部分通道，作为所有通道的代理数据集，以降低内存消耗。此外，他们采用了边归一化，通过添加一组新的边级参数来提高搜索的稳定性。同样，Xue 等人 [62]通过采用通道子集来节省内存。不同之处在于，他们提出了一种注意力机制，并选择注意力权重较高的通道。该方法能够更好地将重要特征信息传递到搜索空间中，并防止搜索过程不稳定。

•增强泛化能力：基于梯度的方法在搜索得到的架构的各种数据集上通常泛化能力较差。为了解决这一问题，Li等人 [63] 引入了领域自适应的思路。他们通过缩小不同领域之间的泛化差距，提升了架构的泛化能力。Liu等人 [64] 提出了一种新方法，将多个任务和领域的数据混合到一个合成数据集中。Ye 等人 [65] 提出了一种正则化方法，可使搜索得到的架构保持较小的 Lipschitz 常数。这是因为 Lipschitz 常数较小时，架构具有良好的泛化能力。

- **提高稳定性：**基于梯度的方法因性能崩溃而缺乏稳定性。现有的缓解方法可分为五类。第一类通常使用丢弃或停止操作来限制跳跃连接的数量，例如 P-Darts [66] 和 DARTS+ [67]。然而，这类方法会排除一些具有多个跳跃连接、且可能性能较高的架构。第二类是对相关指标进行正则化，例如 Hessian 特征值 [68], [69]和层对齐 [70], , 以

表明网络发生了性能崩溃。这些方法的主要局限在于依赖指标的质量。第三类是修改单元，通常涉及辅助跳跃连接 [71]和额外的 Sigmoid 函数 [72]。因此，这类方法的计算开销较大。第四类是在训练过程中根据组丢弃不重要的操作 [73],[74]。遗憾的是，这类方法高度依赖人工设定的超参数，例如组数。第五类基于正则化方法，例如 Beta 衰减 [65] 和超网络的批归一化 [75]。然而，这类方法可能导致更新过于保守，从而限制探索搜索空间的能力。

优点：基于梯度的方法将超网络训练与架构搜索相结合，相较于基于路径的方法更为有效，并且在大型数据集上取得了出色的性能。

缺点：基于梯度的方法有三个缺点：(1) 内存占用巨大：由于联合优化不仅要训练超网络，还要搜索架构，因此基于梯度的方法总是面临内存占用过大的问题。(2) 泛化能力较弱：基于梯度的方法搜索得到的架构，在其他数据集上的表现不如其在搜索所用数据集上的表现。(3) 性能崩溃：基于梯度的方法（例如 DARTS）往往会累积无参数操作（即跳跃连接），导致梯度迅速下降 [68], [71]。然而，卷积等可学习操作更有利于提升深度神经网络的表达能力。如上所述，现有工作只能缓解这些问题，却无法彻底解决。

总而言之，单次方法只需训练一个超网络，便能大幅缩短计算时间。然而，基于路径的方法会出现多路径遗忘现象，而基于梯度的方法则面临性能崩溃问题。这些问题可能导致排序不准确。因此，单次方法可能无法反映架构的真实排序。这也会影响所发现架构的性能。证据在于，单次方法搜索得到的架构有时不如随机搜索发现的架构 [76]。

D. 零样本评估方法

对于所有搜索到的架构，无需在图形处理器上对其进行训练（即 $\mathcal{A}^{EEM} = \emptyset$），而是如图 13 所示，使用评分函数对其进行评估。根据评分函数的类型，零样本方法主要分为参数级方法和架构级方法。此外，评分函数通常是一个数学公式，不依赖于预构建的评估



图 13. 零样本评估方法及其与子类别方法关系的流程图。

方法。因此，零样本方法的运行时间就是获取评分所需的评估时间。由于获取架构的评分通常耗时较少，因此成本极低。请注意，零样本方法无法直接评估给定架构的绝对性能，只能根据性能评分对架构进行排序。

1) 参数级方法：参数级方法首先使用指标来衡量某些参数的显著性（即重要性）。然后，通过汇总某些参数的显著性，为整个架构评分。这些方法仅需一个数据小批量和一次前向/反向传播过程，即可计算某些参数的指标。由于不涉及训练，其运行时间远短于 TEM。一般而言，这些显著性指标通常源自网络剪枝领域的文献。它们最初用于评估参数的重要性，并据此移除不重要的参数。

文献示例：已有一些研究提出了此类别的方法，其关键词为“零样本”和“参数显著性”。例如，Abdelfattah 等人 [137]采用了一系列初始化时剪枝指标，包括梯度范数、SNIP [147], GraSP [148], fisher [149],[150], SynFlow [151]，以衡量参数的显著性。随后，他们将所有参数的指标值相加，以评估架构性能。具体而言，梯度范数使用训练数据小批量经过一次前向/反向传播过程后所得梯度的欧几里得范数来衡量参数。SNIP 通过近似参数被剪枝时交叉熵损失的变化，计算特定参数的显著性。SynFlow 是 SNIP 的一个修改版本，区别在于 SynFlow 中的损失是网络中所有参数的乘积。GraSP 通过衡量移除参数时梯度范数的近似变化来计算显著性。Fisher 衡量移除激活通道时损失的近似变化。Li等人 [139] 发现，高性能网络往往具有较高的梯度绝对均值和较低的标准差。因此，他们提出了零样本逆变异系数，该指标同时考虑每一层的绝对均值和标准差。

优势：参数级方法的计算复杂度较低，因为只需计算少量简单参数。

劣势：尽管参数级方法能够节省成本，但已被证明准确性不足。简单地将各个参数的显著性相加来衡量架构的性能存在问题，因为显著性仅反映参数对架构的影响。事实上，近期的一项研究 [152] 也指出，现有的参数级方法不适合对架构进行排序。其排序质量甚至不如参数大小或浮点运算次数。这也证明了参数级方法的不可靠性。

2）架构级方法：架构级方法通过测量与架构性能正相关的属性来对架构进行评分。由于无需训练任何架构，因此可以大幅节省时间

表 I 神经架构搜索中不同 EEM 的总结，包括其原理和相关参考文献。第一列表示 EEM 的类别。第二列展示各类别的子类别及其运行时间构成，其中“IT”和“ET”分别表示初始化时间和评估时间。第四列和第五列分别列出各方法的优点和缺点。第六列描述各子类别对应的工作原理。第五列标明各子类别对应的相关参考文献。

类别	子类别	工作原理	优点	缺点	参考文献
N-shot	缩小后的数据集 (ET)	通过训练减少训练时间 在下采样数据集上。	易于实施	排序 不匹配 在缩小后的数据集	[2], [18], [21], [32], [77]–[82]
	缩小后的模型 (ET)	训练时间缩短了，缩小后的缩放架构。	节省 GPU 内存	Low 相关性 之间原始模型和代理数据集模型	[1], [2], [10]
	网络形态变换 (ET)	通过继承 父模型的权重，缩短训练时间。	利用 现有的 知识	子网络失败时 在父网络中不存在	[23]–[26], [83]–[85]
	学习曲线外推（IT、ET）	性能被外推 仅训练几个训练轮次后。	轻松 组合 with 各种神经架构搜索算法	Need 训练好的 网络 to 构建预测模型	[2], [13], [28], [29], [86], [87]
少样本	性能预测器 （IT，ET）	使用回归模型直接 预测深度神经网络的性能	快速、无需图形处理器的评估——速度	需要完全训练好的架构来构建回归模型	[28], [29], [31], [32], [34], [37]–[44], [46], [88]–[100]
	种群记忆 （IT，ET）	存储训练好的深度神经网络，并在使用时直接匹配它们。	无损评估性能	Lack 灵活性 无法评估未见过的架构	[12], [50], [51], [101], [102]
单样本	基于路径 (ET)	只需对超网络进行 经过训练，其中候选架构的权重来自它，候选架构的权重来自它。	快速推理以获取 搜索到的网络的性能网络	多模型遗忘问题 问题会削弱预测能力	[52]–[57], [103]–[117]
	基于梯度 (ET)	联合优化架构 参数和超网络权重 通过双层优化。	有效地，适用于大规模数据集	（1）占用大量内存；（2）较低的泛化能力；（3）性能-能崩溃	[9], [10], [59]–[61], [65]–[75], [118]–[136]
零样本	参数级 (ET)	汇总所有显著性 用于衡量整个 未经训练的深度神经网络。	Low 计算 复杂度	不可靠性和不准确	[137]–[139]
	架构级 (ET)	积极地测量属性 与架构性能相关 无需训练。	考虑 the 全局 特征 of the 架构	性能排名不准确	[140]–[146]

用于性能评估。

文献示例： 已有许多架构级方法被提出，用于高效评估架构的性能，我们使用“零样本”“初始化时评分”和“初始化时剪枝”等关键词来筛选这些方法。例如，Mellor 等人 [140], [141] 提出量化网络对不同输入的激活重叠，以评估未训练网络。对于不同输入，激活分离得越好，架构的可学习性就越强。他们在 [140] 中提出了一种基于输入雅可比相关性的指标。此外，他们还在本文的另一版本中提出使用汉明距离来判断两个输入之间的差异 [141]。Xu 等人 [145] 假设梯度可用于评估随机初始化网络。这一方法基于这样的观察：梯度能够直接决定收敛和泛化结果。他们提出了梯度核，将每一层视为一个基本单元。随后，计算每层格拉姆矩阵的均值，以此为网络评分。Lin 等人 [142] 对每个线性区域中的线性函数高斯复杂度取平均，以衡量网络表达能力。Zhou 等人 [146]观察到，视觉 Transformer（ViT）中多头自注意力的突触多样性会显著影响性能。因此，他们提出了一种指标，从突触多样性和突触显著性的角度对 ViT 架构进行排序。Chen 等人 [143] 通过分析神经切线核的谱以及输入空间中的线性区域数量来衡量架构性能。Shu等人 [144] 提出了一种对 NTK 迹范数进行高效近似的方法，以缓解计算成本高昂的问题

NTK，并估计神经架构的性能。

优点： 架构级方法能够全面评估架构的全局特征，例如层次结构和连接性。这有助于捕捉架构对性能的影响。

缺点： 架构级方法通常由针对神经网络的理论研究所启发。此类方法提出指标，用于判断架构的可训练性、可学习性、泛化能力或表达能力，并对架构进行排序。尽管效率很高，架构级方法得出的性能排名却往往不准确。这是因为它们仅粗略衡量与架构性能正相关的架构属性。

总而言之，与其他方法相比，零样本方法能够进一步缩短计算时间。然而，零样本方法在实践中的性能通常不够理想。此外，零样本方法的鲁棒性无法得到保证。其性能在不同任务之间波动很大。

迄今为止，我们已详细介绍了所有 EEM 的分类，并在表 I 中对这些 EEM 进行了总结。值得注意的是，有些方法包含不止一种 EEM，也称为混合方法。例如，Zoph 等人 [2]结合下采样数据集和下采样模型，以缩短评估时间。在此基础上，Zoph 等人 [2]甚至加入了学习曲线外推方法，以进一步提高评估效率。此外，Rawal 等人 [28]引入了一种结合提前停止策略的性能预测器，以加快评估过程。此外，EEM 之间也有相互结合的潜力，例如，

性能预测器与种群记忆，用户可以根据实际需求使用这些方法。

IV. 评估

本节首先在第四节-A中讨论评估指标。然后，在第四节-B中介绍常用的基准数据集。最后，在第四节-C中报告各种 EEM 在基准数据集上的性能。

A. 评估指标

根据免费午餐定理，与 TEM 相比，EEM 的速度虽然快得多，但评估结果可能不够准确。因此，我们主要从两个方面衡量 EEM：运行时间和预测能力。EEM 的运行时间通常以 GPU 天数衡量（GPU 天数 = 图形处理器数量 × 天数) [35]。对于预测能力的衡量，常用指标主要可分为基于相关性的指标、基于排序的指标和基于 Top 的指标。为便于讨论，我们将架构的真实性能和预测性能 $\{A_i\}_{i=1}^N$ 分别记为 $\{y_i\}_i^N$ 和 $\{\hat{y}_i\}_i^N$。架构 A_i 的真实性能排名（即 y_i 在 $\{y_i\}_i^N$ 中的排名）为 r_i，而 A_i 的预测性能排名（即 $\hat{y}_i$ 在 $\{\hat{y}_i\}_i^N$ 中的排名）为 $\hat{r}_i$。

基于相关性的指标主要计算真实性能与预测性能之间的相关性，主要包括皮尔逊系数和决定系数（即 $\mathcal{R}^2$）。皮尔逊系数的公式如下，见公式（10）：

$$\mathcal{M}_{pearson} = \frac{\sum_{i=1}^{N}(y_i-\bar{y})\left(\hat{y}_i-\bar{\hat{y}}\right)}{\sqrt{\sum_{i=1}^{N}(y_i-\bar{y})^2\sum_{i=1}^{N}\left(\hat{y}_i-\bar{\hat{y}}\right)^2}} \tag{10}$$

其中，y_i 和 $\hat{y}_i$ 分别表示真实性能和预测性能。$\bar{y}$ 和 $\hat{y}$ 分别表示所有真实性能和预测性能的平均值。$\mathcal{R}^2$ 的公式如下，见公式（11）：

$$\mathcal{M}_{R^2} = 1 - \frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{N}(y_i-\bar{y}_i)^2} \tag{11}$$

对于这两种指标，数值越接近 1，EEM 就越准确。

基于排序的指标用于计算真实性能排名与预测性能排名之间的相关性。肯德尔秩相关系数（KTau） [153] 和斯皮尔曼排序相关性（SpearmanR） [154] 是常用的基于排序的指标。KTau 表示一致对与不一致对之间的相对差异，如公式（12）所示:

$$\mathcal{M}_{KTau} = 2 \times \frac{\text{number of concordant pairs}}{N(N-1)/2} - 1 \tag{12}$$

其中，一致对表示给定的一对对象中，预测排序与真实排序相同。另一方面，SpearmanR 是根据预测排序与真实排序之间的差异计算得出的，如公式（13）所示:

$$\mathcal{M}_{SpearmanR} = 1 - \frac{6\sum_{i=1}^{N}(r_i-\hat{r}_i)^2}{N\left(N^2-1\right)} \tag{13}$$

对于 KTau 和 SpearmanR，数值越接近 1，表示 EEM 越准确。

基于 Top 的指标衡量 EEM 发现最佳架构的能力，这对神经架构搜索至关重要。基于 Top 的指标主要由 N@K 和 Precision@K 组成。具体而言，N@K 指预测排名前 K 的架构中，真实排名最佳的架构所处的位置。N@K 的值为大于或等于 1 的正整数。N@K 的值越低，表示 EEM 越有能力找到最佳架构。Precision@K 表示预测排名前 K 的架构中，真实排名前 K 的架构所占的比例。与 N@K 不同，Precision@K 的值越接近 1，表示 EEM 发现最佳架构的能力越强。

B. 基准数据集

EEM 通常在公开的基准数据集上进行评估，以便开展公平比较。具体而言，这些基准数据集包含特定搜索空间中的大量架构-性能对。这些基准数据集中的架构-性能对可用于评估 EEM 的预测准确率。常用的基准数据集包括用于图像分类的 NAS-Bench-101 [155], NAS-Bench-201 [155], NAS-Bench-301 [156]、用于自然语言处理的 NAS-Bench-NLP [157] ，以及用于自动语音识别的 NAS-Bench-ASR [158]。

这些基准数据集都采用基于单元的搜索空间。具体而言，它们将单元堆叠起来构成架构，且只能搜索单元的结构。每个单元都可以视为一个有向无环图（DAG）。根据操作在有向无环图中的位置，基于单元的搜索空间可分为节点操作（OON）和边操作（OOE）搜索空间。具体来说，在 OON 中，节点被视为操作，而边被视为操作之间的连接；在 OOE 中，边表示操作，而节点被视为连接。我们在表 II 中总结了上述基准。

C. 基准数据集上的比较

为了让读者更直观地比较各种 EEM 的性能，我们汇总了它们在 NAS-Bench-101、NAS-Bench-201 和 NAS-Bench-301 上的 KTau。请注意，这些结果取自原始论文或报告了这些 EEM 结果的其他论文。我们未报告缩小模型方法、网络形态变换、基于梯度的方法和种群记忆的结果。对于缩小模型方法，我们没有足够的计算资源，无法在这些基准数据集上重新评估缩小后的模型。网络形态变换、基于梯度的方法和种群记忆则无法用于加速这些基准数据集中的所有架构。具体而言，网络形态变换只能用于加速从父网络形态变换得到的网络。基于梯度的方法也是一种搜索策略，只能用于加速搜索得到的架构。性能记忆只能用于查询已评估架构的性能。因此，我们未报告这些方法的实验结果。

表 II NAS 基准的特征。$|V|$ 表示节点数量，$|E|$ 表示边的数量。请注意，NAS-Bench-301 提供了 60,000 个已标注架构以及一个性能预测器，该预测器可以预测 DARTS 搜索空间中的所有 10^{18} 架构。

基准	Year	Task	数据集	搜索空间	OON/OOE	架构数量
NAS-Bench-101	2019	图像分类	CIFAR-10	$\|V\| \leq 7$,#运算=3,$\|E\| \leq 9$	OON	423,624
NAS-Bench-201	2020	图像分类	CIFAR-10、CIFAR-100, ImageNet-16-120	$\|V\| = 4$,#ops=5	OOE	15,625
NAS-Bench-301	2020	图像分类	CIFAR-10	$\|V\| \leq 7$,#ops=7	OOE	60,000/10^{18}
NAS-Bench-NLP	2020	自然语言处理	宾夕法尼亚树库	$\|V\| \leq 24$,#ops=7	OOE	14,322
NAS-Bench-ASR	2021	自动语音识别	TIMIT 音频数据集	$\|V\| \leq 4$,#运算=7,$\|E\| \leq 9$	OOE	8,242

表 III

N-样本、少样本、单样本和零样本方法在 NAS-Bench-101（NB101）、NAS-Bench-201（NB201）和 NAS-Bench-301（NB301）上的肯德尔秩相关系数（KTau）值。具体而言，第一列显示 EEM 的类别，第二列显示各类别的子类别。第三列显示具体方法，第五、六和七列分别显示各具体方法在 NB101、NB201 和 NB301 上的 KTau。第八列为获取 KTau 值的实验说明。† 表示基准未提供足够的数据来开展实验。‡ 表示该方法的核心代码未开源，且原始论文中未提供结果。* 表示 NB301 未提供真实准确率，因此性能评估社区无法提供 KTau 值。

类别	子类别	方法	NB101	NB201	NB301	Note
N 次	缩小后的数据集	代理数据集	-†	0.827	-†	CIFAR-10 是代理数据集，而 ImageNet 是原始数据集。† NB101 和 NB201 仅有 CIFAR10 的结果。
		数据集子集	-†	0.867	-†	使用 CIFAR-10 作为 CIFAR-100 的数据集子集。† NB101 和 NB201 仅提供 CIFAR10 的结果。
	学习曲线外推	提前停止（1/4）	-†	0.602	0.614	当训练轮次达到完整训练轮次的四分之一时，终止训练。† NB101 缺少 1/4 训练轮次的结果。
		提前停止（1/2）	0.438	0.663	0.662	当训练轮次达到完整训练轮次。
少样本	性能预测器	NeuralPredictor [37]	0.679	0.646	-*	使用 424 和 1,564 个样本作为训练数据，分别为 NAS-Bench-101 和 NAS-Bench-201。
		Peephole [34]	0.4556	-‡	-*	使用 424 个样本作为训练数据。
		E2EPP [46]	0.5038	0.7669	-*	使用 424 个样本作为训练数据。
		ReNAS [38]	0.657	-‡	-*	使用 424 个样本作为训练数据。
		HOP [43]	0.813	0.897	-*	在 NAS- 中，分别使用 381 个和 781 个样本作为训练数据 Bench-101 和 NAS-Bench-201。
		TNASP [97]	0.722	0.726	-*	在以下数据集中，分别使用 424 个和 1,564 个样本作为训练数据分别为 NAS-Bench-101 和 NAS-Bench-201。
单样本	基于路径	OSNAS [152]	0.446	0.744	0.548	在超网络训练中使用蒙特卡洛样本。
		FairNAS [54]	-‡	0.706	0.527	-
零样本	参数级	SynFlow [137]	-0.063	0.573	0.201	-
		梯度范数 [137]	-0.276	0.401	0.070	-
		SNIP [137]	-0.206	0.402	0.050	-
		GraSP [137]	-0.266	0.348	0.365	-
		fisher [137]	-0.202	0.362	-0.158	-
		普通 [137]	0.240	0.311	0.394	-
	架构级	雅可比协方差 [140]	0.066	0.608	0.230	-
		ReLU_对数行列式 [141]	0.290	0.611	0.539	-

实验结果见表 III，我们将依次讨论各类方法。(1) N 少样本方法：这些方法仍需评估所有搜索架构，与其他方法相比通常耗时更长。具体而言，学习曲线外推方法的结果不如缩小数据集方法。这是因为提前停止时的学习曲线无法恰当地反映最终性能，而 CIFAR10 与 ImageNet 具有相似特征，因此被广泛用作代理数据集对。此外，提前训练至 1/2 时的性能优于训练至 1/4 时的性能。这意味着更多训练迭代次数能够更好地匹配最终学习曲线。(2)少样本方法：性能预测器需要额外的初始化时间，但其性能在所有方法中最佳。具体而言，HOP [43]达到了 0。813 和 0。891 的 KTau 值，分别在 NB101 和 NB201 上取得最优结果。(3) 单次-镜像方法：这些方法需要评估大型超网络的性能，因此需要一定的评估时间；路径批处理方法可以获得与早停策略相近的结果。(4) 零样本方法：这类方法被认为是最省时的，因为它们不依赖任何基于 GPU 的评估过程。然而，这类方法的表现不如其他类别，因为它们评估的是一些无法反映架构实际准确率的统计指标。值得注意的是，参数级方法在 NB101 和 NB301 上甚至会得到负的 KTau 值。

各类方法的性能会受到多种因素影响。具体而言，代理数据集与原始数据集之间的相似度，以及子数据集的代表性，都会显著影响缩小数据集方法的性能。此外，性能预测器的能力可能受到架构的编码策略以及

回归模型的影响。此外，路径采样策略也会影响基于路径的方法的性能。最后，零样本方法的性能取决于所设计的评分函数。如果评分函数能够准确反映架构的实际准确率，零样本方法就能取得出色的性能。

V. 挑战与未来方向

总而言之，目前的研究主要致力于设计在 NAS-Bench-101、NAS-Bench-201 和 NAS-Bench-301 等基准数据集上具有较高预测准确率和效率的 EEM，并已取得了显著成果。首先，现有架构基准数据集中的神经架构性能是在 CIFAR10 等小规模数据集上测得的。这些基准数据集无法反映 EEM 在真实场景中的性能。其次，当前的 EEM 难以实现跨多个不同搜索空间的神经架构跨域预测。由于必须针对每个特定搜索空间重新设计 EEM，因此时间复杂度随之增加。此外，当前的 EEM 缺乏在单个搜索空间内评估神经架构在多个数据集上性能的能力。因此，即使是同一搜索空间中的架构，在多个数据集上构建 EEM 也需要更多时间。最后，现有的 EEM 方法也难以统一表示多种类型的神经架构，这影响了它们有效挖掘有价值架构数据的能力。本节将详细讨论这些挑战。

A. 有效性验证

目前，大多数研究人员认为 EEM 的有效性验证缺乏说服力。具体而言，基准数据集被用于有效性验证和公平比较。然而，由于规模较小，现有架构数据集无法反映实际场景。例如，NAS-Bench-101 仅包含三种操作（即，卷积 1×1、卷积 3×3 和最大池化）。其节点和边的数量上限分别为七个和九条。NAS-Bench-101 搜索空间非常小，仅包含 423,624 个架构。NAS-Bench-201 仅包含四种不同的操作，且每个架构只有四个顶点。NAS-Bench-201 搜索空间仅包含 15,625 个架构，甚至比 NAS-Bench-101 还小。相比之下，实际应用场景中采用的搜索空间，例如 NASNet、MobileNet 或 Transformer 的搜索空间，要复杂得多。它们的规模通常比 NAS-Bench-101 和 NAS-Bench-201 大几个数量级。例如，NASNet 搜索空间设计了一个包含 15 种操作的操作集，以及五个不包括输入节点和输出节点的节点。作为一种基于块的搜索空间，MobileNet 搜索空间为每个块提供多种选择。块大小为五时，搜索空间规模约为 10^{39}。因此，现有基准数据集上的结果无法反映 EEM 在实际场景中的效果。我们需要构建更大的架构数据集，以帮助验证所提出的 EEM。

B. 跨域预测

跨域预测主要指预测不同搜索空间中架构的性能。在现有的 EEM 中，只有缩小数据集方法、缩小模型方法和种群记忆具备跨域预测能力。这是因为这些方法不关注架构所属的领域。其他 EEM 则针对特定的目标搜索空间，不具备跨域能力。例如，网络形态变换中设计的保函数操作只能用于特定结构。如果架构中不包含该结构，就无法使用网络形态变换方法。性能预测器通常会针对特定搜索空间设计编码方法和回归模型。因此，它们无法应用于其他搜索空间，或者在其他搜索空间中的效果较差。单次方法只能预测超网络所包含子网络的性能。零样本方法的跨域能力也无法得到保证，因为不同类型架构的属性测量方式各不相同。这意味着，一旦搜索空间发生变化，我们就必须重新构建 EEM。这既耗费人力，也需要大量计算资源。已有研究探索跨域预测方法。例如，Han 等人 [159] 将候选卷积神经网络表示为仅由基本算子构成的计算图。随后，他们提出了一种半监督图表示学习流程，用于预测来自多个系列的架构。然而，该方法仅适用于基于单元的搜索空间。如何开发跨域 EEM 仍是一项具有挑战性的课题。

C. 多任务预测

多任务预测是指 EEM 需要预测同一架构在多个任务中的多个性能值。此外，同一架构的多个标签之间存在内在相关性。多任务场景常见于多任务学习。具体而言，多任务学习利用不同但相关的任务之间的有用信息，以提升网络的泛化能力 [160]。多任务学习之所以成为热门研究方向，是因为它能够通过将一个网络应用于多个任务来降低计算开销。此外，在多个任务上训练一个网络也能提升泛化能力。在多任务学习中，我们需要估计架构在多个任务上的性能。一些研究人员已经注意到这一问题并开展了相关研究。例如，Huang 等人 [89] 将任务嵌入为输入的一部分，以实现对多个标签的预测。Shala 等人 [161] 提出了一种基于深度核高斯过程的可迁移性能预测器，用于预测架构在不同数据集上的性能。为推动多任务预测的发展，Duan 等人 [162] 提出了一个基准数据集（称为 TransNAS-Bench-101）。具体而言，TransNAS-Bench-101 涵盖了 51,464 个架构在七项任务上的性能，这些任务包括分类、回归、像素级预测和自监督任务。然而，相关研究仍相当有限。随着多任务学习在现实

部署场景中的成功，我们认为，多任务 EEM 是一个很有前景的未来研究方向。

D. 统一表示

架构的性能在很大程度上取决于架构的设计。因此，如何从架构中提取有意义的信息，对于神经架构搜索的预测至关重要。为了挖掘架构信息，我们首先需要提供一种表示方法来描述架构。尽管已有许多强大的表示方法被提出，但它们只能表示特定类型搜索空间中的架构。例如，常用的邻接矩阵编码 [163]只能用于表示基于单元的架构。这使研究人员无法同时学习各种架构的知识。此外，这不利于提升 EEM 的可迁移性。为解决这一问题，Sun 等人 [164] 设计了一种统一的文本方法来描述卷积神经网络。具体而言，该方法设计了四种单元，用于描述卷积神经网络中每一层的详细信息。此外，它还规定了层的唯一顺序，使拓扑信息保持不变。然而，除卷积神经网络外，该方法无法描述其他类型的架构。对于神经架构搜索领域而言，面向各种架构的统一表示研究仍处于早期阶段。

VI. 结论

本文全面综述了 EEM。具体而言，根据训练过的架构数量，我们将 EEM 分为 N-shot 方法、少样本方法、单次评估方法和零样本方法。 N-shot 方法需要训练每个搜索得到的架构，主要包括缩小数据集方法、缩小模型方法、学习曲线外推方法和网络形变。少样本方法只需训练比 TEM 更少数量的架构，主要包括性能预测器和种群记忆。单次评估方法只需训练一个架构，主要包括基于路径的方法和梯度方法。零样本方法无需训练，因此能进一步缩短运行时间。它们可分为参数级方法和架构级方法。此外，我们回顾了 EEM 的评估指标和基准数据集，并报告了在这些基准数据集上的结果，以直观展示各种 EEM 的性能。最后，我们总结了现有 EEM 面临的挑战和未来研究方向。具体而言，主要挑战包括有效性验证、跨域预测、多任务预测和统一表示。

参考文献

[1] E. Real、A. Aggarwal、Y. Huang 和 Q. V. Le，《用于图像分类器架构搜索的正则化进化》，载于*AAAI*人工智能会议论文集，第 33 卷，第 01 期，2019年，第 4780–4789 页。[2] B. Zoph、V. Vasudevan、J. Shlens 和 Q. V. Le，《学习可迁移架构以实现可扩展图像识别》，载于*IEEE/CVF*计算机视觉与模式识别会议论文集，2018年，第 8697–8710 页。

[3] P. Ren、Y. Xiao、X. Chang、P.-Y. Huang、Z. Li、X. Chen 和 X. Wang，《神经架构搜索综合综述：挑战与解决方案》，ACM计算综述，第54卷，第4期，页码1–34，2021年。[4] T. Elsken、J. H. Metzen 和 F. Hutter，《神经架构搜索：综述》，机器学习研究杂志，第20卷，第1期，页码1997–2017，2019年。[5] Y. Liu、Y. Sun、B. Xue、M. Zhang、G. G. Yen 和 K. C. Tan，《进化神经架构搜索综述》，IEEE神经网络与学习系统汇刊，第34卷，第2期，页码550–570，2023年。[6] E. Real、S. Moore、A. Selle、S. Saxena、Y. L. Suematsu、J. Tan、Q. V. Le 和 A. Kurakin，《图像分类器的大规模进化》，载于《国际机器学习会议论文集》，第70卷，2017年，页码2902–2911。[7] L. Xie 和 A. Yuille，《遗传CNN》，载于《IEEE/CVF国际计算机视觉会议论文集》，2017年，页码1379–1388。[8] B. Zoph 和 Q. Le，《基于强化学习的神经架构搜索》，载于《国际学习表征会议论文集》， 2016, [在线]。可访问：https://openreview.net/pdf?id=r1Ue8Hcxg。[9] M. Fil、B. Ru、C. Lyle 和 Y. Gal，《无需验证集的DARTS：优化边际似然》，arXiv预印本 arXiv:2112 13023，2021年。。[10] H. Liu、K. Simonyan 和 Y. Yang，《DARTS：可微架构搜索》，载于《国际学习表征会议论文集》， 2018, [在线]。可访问：https://openreview.net/pdf?id=S1eYHoC5FX。[11] A. Krizhevsky 和 G. Hinton，《从微小图像中学习多层特征》，《系统与自身免疫疾病》，第1卷，第4期，2009年。[12] X. Xie、Y. Liu、Y. Sun、G. G. Yen、B. Xue 和 M. Zhang，《BenchENAS：进化神经架构搜索基准测试平台》，IEEE进化计算汇刊，第26卷，第6期，页码1473–1485，2022年。[13] T. Domhan、J. T. Springenberg 和 F. Hutter，《通过外推学习曲线加速深度神经网络的自动超参数优化》，载于《国际人工智能联合会议论文集》，第9期，2015年，页码3460–3468。[14] D. Baymurzina、E. Golikov 和 M. Burtsev，《神经架构搜索综述》，《神经计算》，第474卷，页码82–93，2022年。[15] S. Liu、H. Zhang 和 Y. Jin，《计算高效神经架构搜索综述》，《自动化与智能》，第1卷，100002，2022年1，第100002页。。[16] C. White、A. Zela、R. Ru、Y. Liu 和 F. Hutter，《神经架构搜索中的性能预测器有多强大？》，载于《神经信息处理系统进展论文集》，第34卷，2021年，页码28 454–28 469。[17] Y. Wang、Q. Yao、J. T. Kwok 和 L. M. Ni，《从少量示例中泛化：少样本学习综述》，ACM计算综述，第53卷，第3期，页码1–34，2020年。[18] P. Liu、M. D. El Basha、Y. Li、Y. Xiao、P. C. Sanelli 和 R. Fang，《采用加速遗传算法的深度进化网络用于医学图像去噪》，《医学图像分析》，第54卷，页码306–315，2019年。[19] M. Park，《用于快速高效神经架构搜索的数据代理生成》，arXiv预印本 arXiv:1911.09322，2019年。[20] B. Na、J. Mok、H. Choe 和 S. Yoon，《通过代理数据加速神经架构搜索》，载于《国际人工智能联合会议论文集》，2021年，页码2848–2854。[21] P. Chrabaszcz、I. Loshchilov 和 F. Hutter，《作为CIFAR数据集替代方案的ImageNet下采样变体》，arXiv预印本 arXiv:1707.08819，2017年。[22] J. Deng、W. Dong、R. Socher、L.-J. Li、K. Li 和 L. Fei-Fei，《ImageNet：大型分层图像数据库》，载于《IEEE/CVF计算机视觉与模式识别会议论文集》，2009年，页码248–255。[23] T. Wei、C. Wang、Y. Rui 和 C. W. Chen，《网络形态变换》，载于《国际机器学习会议论文集》，2016年，页码564–572。[24] H. Cai、T. Chen、W. Zhang、Y. Yu 和 J. Wang，《通过网络变换实现高效架构搜索》，载于《AAAI人工智能会议论文集》，第32卷，第1期，2018年。[25] H. Cai、J. Yang、W. Zhang、S. Han 和 Y. Yu，《通过路径级网络变换实现高效架构搜索》，载于《国际机器学习会议论文集》，2018年，页码678–687。[26] T. Elsken、J. H. Metzen 和 F. Hutter，《通过拉马克进化实现高效的多目标神经架构搜索》，arXiv预印本 arXiv:1804.09081，2018年。[27] J. N. v. Rijn、S. M. Abdulrahman、P. Brazdil 和 J. Vanschoren，《使用学习曲线进行快速算法选择》，载于《智能数据分析国际研讨会论文集》，2015年，页码298–309。[28] A. Rawal 和 R. Miikkulainen，《从节点到网络：循环神经网络的进化》，arXiv预印本 arXiv:1803.04439，2018年。[29] B. Baker、O. Gupta、R. Raskar 和 N. Naik，《利用性能预测加速神经架构搜索》，arXiv预印本 arXiv:1705.10823，2017年。

[30] M. Wistuba 和 T. Pedapati，《学习曲线排序》，载于《国际机器学习会议论文集》，2020年，页码10 303–10 312。[31] X. Dai、A. Wan、P. Zhang、B. Wu、Z. He、Z. Wei、K. Chen、Y. Tian、M. Yu、P. Vajda 等，《FBNetV3：利用预测器预训练联合搜索架构与配方》，载于《IEEE/CVF计算机视觉与模式识别会议论文集》，2021年，页码16 276–16 285。[32] S. Hassantabar、X. Dai 和 N. K. Jha，《CURIOUS：基于性能预测器和进化搜索的高效神经架构搜索》，IEEE计算机辅助设计与集成电路系统汇刊，第41卷，第11期，页码4975–4990，2022年。[33] X. Ning、Y. Zheng、T. Zhao、Y. Wang 和 H. Yang，《用于基于预测器的神经架构搜索的通用图神经架构编码方案》，载于《欧洲计算机视觉会议论文集》，2020年，页码189–204。[34] B. Deng、J. Yan 和 D. Lin，《Peephole：训练前预测网络性能》，arXiv预印本 arXiv:1712.03351，2017年。[35] Y. Sun、B. Xue、M. Zhang 和 G. G. Yen，《基于模块的全自动卷积神经网络架构设计》，IEEE神经网络与学习系统汇刊，第31卷，第4期，页码1242–1254，2019年。[36] R. Luo、F. Tian、T. Qin、E. Chen 和 T.-Y. Liu，《神经架构优化》，载于《神经信息处理系统进展论文集》，第31卷，2018年，页码7816–7827。[37] W. Wen、H. Liu、Y. Chen、H. Li、G. Bender 和 P.-J. Kindermans，《用于神经架构搜索的神经预测器》，载于《欧洲计算机视觉会议论文集》，2020年，页码660–676。[38] Y. Xu、Y. Wang、K. Han、Y. Tang、S. Jui、C. Xu 和 C. Xu，《ReNAS：神经架构搜索的相对论式评估》，载于《IEEE/CVF计算机视觉与模式识别会议论文集》，2021年，页码4411–4420。[39] R. Luo、X. Tan、R. Wang、T. Qin、E. Chen 和 T.-Y. Liu，《使用非神经模型预测神经架构搜索的准确率》，arXiv预印本 arXiv:2007.04785，2020年。[40] R. Istrate、F. Scheidegger、G. Mariani、D. Nikolopoulos、C. Bekas 和 A. C. I. Malossi，《TAPAS：用于架构搜索的免训练准确率预测器》，载于《AAAI人工智能会议论文集》，第33卷，第01期，2019年，页码3927–3934。[41] C. Liu、B. Zoph、M. Neumann、J. Shlens、W. Hua、L.-J. Li、L. Fei-Fei、A. Yuille、J. Huang 和 K. Murphy，《渐进式神经架构搜索》，载于《欧洲计算机视觉会议论文集》，2018年，页码19–34。[42] C. Wei、C. Niu、Y. Tang、Y. Wang、H. Hu 和 J. Liang，《NPENAS：神经预测器引导的神经架构搜索进化》，IEEE神经网络与学习系统汇刊，第34卷，8455，2022年11，页码8441–。。[43] Z. Chen、Y. Zhan、B. Yu、M. Gong 和 B. Du，《并非所有操作的贡献都相同：用于神经架构搜索的层次化操作自适应预测器》，载于《IEEE/CVF国际计算机视觉会议论文集》，2021年，页码10 508–10 517。[44] Y. Tang、Y. Wang、Y. Xu、H. Chen、B. Shi、C. Xu、C. Xu、Q. Tian 和 C. Xu，《神经架构的半监督评估器》，载于《IEEE/CVF计算机视觉与模式识别会议论文集》，2020年，页码1810–1819。[45] R. Luo、X. Tan、R. Wang、T. Qin、E. Chen 和 T.-Y. Liu，《半监督神经架构搜索》，载于《神经信息处理系统进展论文集》，第33卷，2020年，页码10 547–10 557。[46] Y. Sun、H. Wang、B. Xue、Y. Jin、G. G. Yen 和 M. Zhang，《使用端到端随机森林性能预测器的代理辅助进化式深度学习》，IEEE进化计算汇刊，第24卷，第2期，页码350–364，2019年。[47] C. Shorten 和 T. M. Khoshgoftaar，《深度学习图像数据增强综述》，《大数据杂志》，第6卷，第1期，页码1–48，2019年。[48] L. Wang、Y. Zhao、Y. Jinnai、Y. Tian 和 R. Fonseca，《AlphaX：利用深度神经网络和蒙特卡洛树搜索探索神经架构》，arXiv预印本 arXiv:1903.11059，2019年。[49] Y. Liu、Y. Tang 和 Y. Sun，《用于神经预测器的同质架构增强》，载于《IEEE/CVF国际计算机视觉会议论文集》，2021年，页码12 249–12 258。[50] S. Fujino、N. Mori 和 K. Matsumoto，《利用进化式深度学习构建人类草图深度卷积网络》，载于《国际模糊系统协会国际软计算与智能系统会议论文集》，2017年，页码1–5。[51] Y. Sun、B. Xue、M. Zhang、G. G. Yen 和 J. Lv，《使用遗传算法自动设计用于图像分类的卷积神经网络架构》，IEEE控制论汇刊，第50卷，3854，2020年9，页码3840–。。[52] G. Bender、P.-J. Kindermans、B. Zoph、V. Vasudevan 和 Q. Le，《理解并简化单次架构搜索》，载于《国际机器学习会议论文集》，2018年，页码550–559。[53] Z. Guo、X. Zhang、H. Mu、W. Heng、Z. Liu、Y. Wei 和 J. Sun，《采用均匀采样的单路径单次神经架构搜索》，载于《欧洲计算机视觉会议论文集》，2020年，页码544–560。

[54] X. Chu、B. Zhang 和 R. Xu，“FairNAS：重新思考权重共享神经架构搜索的评估公平性”，载于 IEEE/CVF国际计算机视觉会议论文集，2021年，第 12 239–12 248 页。[55] M. Zhang、H. Li、S. Pan、T. Liu 和 S. W. Su，“通过新颖性驱动采样进行单次神经架构搜索”，载于国际人工智能联合会议论文集，2020年，第 3188–3194 页。[56] S. You、T. Huang、M. Yang、F. Wang、C. Qian 和 C. Zhang，“GreedyNAS：借助贪心超网络实现快速单次神经架构搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2020年，第 1999–2008 页。[57] M. Zhang、X. Yu、H. Zhao 和 L. Ou，“Shiftnas：通过概率偏移改进单次神经架构搜索”，载于 IEEE/CVF国际计算机视觉会议论文集，2023年，第 5919–5928 页。[58] Y. Benyahia、K. Yu、K. B. Smires、M. Jaggi、A. C. Davison、M. Salzmann 和 C. Musat，“克服多模型遗忘”，载于国际机器学习会议论文集，2019年，第 594–603 页。[59] M. Zhang、H. Li、S. Pan、X. Chang、C. Zhou、Z. Ge 和 S. Su，“单次神经架构搜索：最大化多样性以克服灾难性遗忘”，IEEE模式分析与机器智能汇刊，第 43 卷，第 9 期，第 2921–2935 页，2020年。[60] H. Cai、L. Zhu 和 S. Han，“ProxylessNAS：直接针对目标任务和硬件进行神经架构搜索”，载于国际学习表征会议论文集，2018，[在线]。可访问：https://openreview.net/pdf?id=HylVB3AqYm。[61] Y. Xu、L. Xie、X. Zhang、X. Chen、G.-J. Qi、Q. Tian 和 H. Xiong，“PC-DARTS：用于节省内存的架构搜索的部分通道连接”，载于国际学习表征会议论文集，2019, [在线]。可访问：https://openreview.net/forum?id=BJlS634tPr。[62] Y. Xue 和 J. Qin，“基于通道注意力的部分连接可微神经架构搜索”，IEEE工业信息学汇刊，第 19 卷，第 5 期，第 6804–6813 页，2022年。[63] Y. Li、Z. Yang、Y. Wang 和 C. Xu，“跨领域适配神经架构”，载于神经信息处理系统进展论文集，第 33 卷，2020年，第 789–798 页。[64] L. Liu、Z. Wen、S. Liu、H.-Y. Zhou、H. Zhu、W. Xie、L. Shen、K. Ma 和 Y. Zheng，“Mixsearch：搜索领域泛化医学图像分割架构”，arXiv 预印本 arXiv:2102 13280，2021年。[65] P. Ye、B. Li、Y. Li、T. Chen、J. Fan 和 W. Ouyang，“Beta-DARTS：用于可微架构搜索的 Beta 衰减正则化”，arXiv 预印本 arXiv:2203.01665，2022年。[66] X. Chen、L. Xie、J. Wu 和 Q. Tian，“渐进式可微架构搜索：弥合搜索与评估之间的深度差距”，载于 IEEE/CVF国际计算机视觉会议论文集，2019年，第 1294–1303 页。[67] H. Liang、S. Zhang、J. Sun、X. He、W. Huang、K. Zhuang 和 Z. Li，“DARTS+：通过提前停止改进可微架构搜索”，arXiv 预印本 arXiv:1909.06035，2019年。[68] T. E. Arber Zela、T. Saikia、Y. Marrakchi、T. Brox 和 F. Hutter，“理解并增强可微架构搜索的鲁棒性”，载于国际学习表征会议论文集，2020，[在线]。可访问：https://openreview.net/forum?id=H1gDNyrKDS。[69] X. Chen 和 C.-J. Hsieh，“通过基于扰动的正则化稳定可微架构搜索”，载于国际机器学习会议论文集，2020年，第 1554–1565 页。[70] S. Movahedi、M. Adabinejad、A. Imani、A. Keshavarz、M. Dehghani、A. Shakery 和 B. N. Araabi，“Λ-DARTS：通过协调单元间的操作选择来缓解性能崩溃”，载于国际学习表征会议论文集，2023，[在线]。可访问：https://openreview.net/forum?id=oztkQizr3kk。[71] X. Chu、X. Wang、B. Zhang、S. Lu、X. Wei 和 J. Yan，“DARTS-：在没有指标的情况下稳健地摆脱性能崩溃”，arXiv 预印本 arXiv:2009.01027，2020年。[72] X. Chu、T. Zhou、B. Zhang 和 J. Li，“Fair DARTS：消除可微架构搜索中的不公平优势”，载于欧洲计算机视觉会议论文集，2020年，第 465–480 页。[73] W. Hong、G. Li、W. Zhang、R. Tang、Y. Wang、Z. Li 和 Y. Yu，“DropNAS：用于可微架构搜索的分组操作丢弃”，arXiv 预印本 arXiv:2201.11679，2022年。[74] Y.-C. Gu、L.-J. Wang、Y. Liu、Y. Yang、Y.-H. Wu、S.-P. Lu 和 M.-M. Cheng，“Dots：在可微架构搜索中解耦操作与拓扑”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2021年，第 12 311–12 320 页。[75] X. Zhang、Y. Li、X. Zhang、Y. Wang 和 J. Sun，“基于随机特征的可微架构搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2023年，第 16 060–16 069 页。

[76] A. Yang、P. M. Esperanc¸a 和 F. M. Carlucci，“神经架构搜索评估难得令人沮丧”，载于国际学习表征会议论文集，2019，[ ] = 在线。可访问：https://openreview.net/forum?id HygrdpVKvr。[77] A. Klein、S. Falkner、S. Bartels、P. Hennig 和 F. Hutter，“大型数据集上机器学习超参数的快速贝叶斯优化”，载于人工智能与统计会议论文集，2017年，第528–536 页。[78] H. Shu 和 Y. Wang，“自动搜索 U-Net 图像转换器架构”，arXiv 预印本 arXiv:2002.11581，2020年。[79] D. Sapra 和 A. D. Pimentel，“用于设计卷积神经网络的约束进化式分段训练”，载于工业、工程及其他应用智能系统国际会议论文集，2020年，第 709–721 页。[80] B. Wang、B. Xue 和 M. Zhang，“通过进化和堆叠可迁移模块，采用粒子群优化进化用于图像分类的深度神经网络”，载于 IEEE进化计算会议论文集，2020年，第 1–8 页。[81] Y. Xu、L. Xie、W. Dai、X. Zhang、X. Chen、G.-J. Qi、H. Xiong 和 Q. Tian，“用于减少计算冗余的部分连接神经架构搜索”，IEEE模式分析与机器智能汇刊，第 43 卷，第 9 期，第 2953–2970 页，2021年。[82] J. Liu、M. Gong、Q. Miao、X. Wang 和 H. Li，“基于多目标优化的深度神经网络结构学习”，IEEE神经网络与学习系统汇刊，第 29 卷，第 6 期，第 2450–2463 页，2017年。[83] T. Chen、I. Goodfellow 和 J. Shlens，“Net2net：通过知识迁移加速学习 ”，arXiv 预印本 arXiv:1511.05641，2015年。[84] T. Elsken、J.-H. Metzen 和 F. Hutter，“用于卷积神经网络的简单高效架构搜索”，arXiv 预印本 arXiv:1711.04528，2017年。[85] Z. Zhang、Y. Chen 和 C. Zhou，“自生长二值激活网络：一种具有动态架构的新型深度学习模型”，IEEE神经网络与学习系统汇刊，第 35 卷，第 1 期，第 624–633 页，2022年。[86]A. Klein、S. Falkner、J. T. Springenberg 和 F. Hutter，“使用贝叶斯神经网络预测学习曲线”，载于国际学习表征会议论文集， 2016，[在线]。可访问：https://openreview.net/forum?id=S11KBYclx。[87] Y. Kim、W. J. Yun、Y. K. Lee 和 J. Kim，“在图像分类中利用提前停止进行神经架构搜索的两阶段架构微调”，arXiv 预印本 arXiv:2202.08604，2022年。[88] Y. Sun、X. Sun、Y. Fang、G. G. Yen 和 Y. Liu，“用于进化神经架构搜索算法性能预测器的新型训练方案”，IEEE进化计算汇刊，第 25 卷，第 3 期，第 524–536 页，2021年。[89] M. Huang、Z. Huang、C. Li、X. Chen、H. Xu、Z. Li 和 X. Liang，“Arch-graph：用于任务迁移神经架构搜索的无环架构关系预测器”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2022年，第 11 881–11 891 页。[90] C. White、W. Neiswanger 和 Y. Savani，“Bananas：通过用于神经架构搜索的神经架构进行贝叶斯优化”，载于 AAAI人工智能会议论文集，第 35 卷，第 12 期，2021年，第 10 293–10 301 页。[91] L. Dudziak、T. Chau、M. Abdelfattah、R. Lee、H. Kim 和 N. Lane，“BRP-NAS：使用图卷积网络进行基于预测的神经架构搜索 ”，载于神经信息处理系统进展论文集，第 33 卷，2020年，第 10 480–10 490 页。[92] Y. Chen、Y. Guo、Q. Chen、M. Li、W. Zeng、Y. Wang 和 M. Tan，“基于神经架构比较器的对比式神经架构搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2021年，第 9502–9511 页。[93] L. Mauch、S. Tiedemann、J. A. Garcia、B. N. Cong、K. Yoshiyama、F. Cardinaux 和 T. Kemp，“用于基于预测器的神经架构搜索的高效采样”，arXiv 预印本 arXiv:2011.12043，2020 年。[94] C.-H. Liu 、 Y.-S. Han 、 Y.-Y. Sung 、 Y. Lee、H.-Y. Chiang 和 K.-C. Wu，“FOX-NAS：快速、端侧且可解释的神经架构搜索”，载于 IEEE/CVF国际计算机视觉会议论文集，2021年，第 789–797 页。[95] Y. Li、C. Hao、P. Li、J. Xiong 和 D. Chen，“通过回归实现通用神经架构搜索”，载于神经信息处理系统进展论文集，第 34 卷，2021年，第 20 476–20 490 页。[96] Z. Li、T. Xi、J. Deng、G. Zhang、S. Wen 和 R. He，“GP-NAS：基于高斯过程的神经架构搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2020年，第 11 933–11 942 页。[97] S. Lu、J. Li、J. Tan、S. Yang 和 J. Liu，“TNASP：采用自进化框架的基于Transformer的神经架构搜索预测器”，载于神经信息处理系统进展论文集，第 34 卷，2021年，第 15 125–15 137 页。[98] Y. Peng、A. Song、V. Ciesielski、H. M. Fayek 和 X. Chang，“Pre-nas：预测器辅助的进化神经架构搜索”，arXiv 预印本 *arXiv:2204.12726*，2022年。

[99] R. Wang、X. Chen、M. Cheng、X. Tang 和 C.-J. Hsieh，“Rank-nosh：通过非均匀连续减半实现高效的基于预测器的架构搜索”，载于 IEEE/CVF国际计算机视觉会议论文集，2021年，第 10 377–10 386 页。[100] J. Wu、X. Dai、D. Chen、Y. Chen、M. Liu、Y. Yu、Z. Wang、Z. Liu、M. Chen 和 L. Yuan，“以更弱的预测器实现更强的神经架构搜索”，载于 神经信息处理系统进展，第 34 卷，2021年，第 28 904–28 918 页。[101] F. M. Johner 和 J. Wassner，“用于 GTSRB 上卷积神经网络优化的高效进化式架构搜索”，载于 IEEE 国际机器学习与应用会议论文集，2019年，第 56–61 页。[102] E. Miahi、S. A. Mirroshandel 和 A. Nasr，“用于人类精子图像自动评估的遗传神经架构搜索”，《专家系统与应用》，第 188 卷，第 115937 页，2022年。[103] Y. Chen、T. Yang、X. Zhang、G. Meng、X. Xiao 和 J. Sun，“DetNAS：用于目标检测的骨干网络搜索”，载于 神经信息处理系统进展，第 32 卷，2019年，第 6642–6652 页。[104] H. Peng、H. Du、H. Yu、Q. Li、J. Liao 和 J. Fu，“优中选优：为单次神经架构搜索提炼优先路径”，载于 神经信息处理系统进展，第 33 卷，2020年，第 17 955–17 964 页。[105] S.-Y. Huang 和 W.-T. Chu，“通过生成进行搜索：采用架构生成器的灵活高效单次神经架构搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2021年，第 983–992 页。[106] X. Xia、X. Xiao、X. Wang 和 M. Zheng，“为单次神经架构搜索逐步自动设计搜索空间”，载于 IEEE/CVF冬季计算机视觉应用会议论文集，2022年，第 2455–2464 页。[107] R. Guo、C. Lin、C. Li、K. Tian、M. Sun、L. Sheng 和 J. Yan，“利用稳定的共享参数代理为单次拓扑神经架构搜索赋能”，载于 欧洲计算机视觉会议论文集，2020年，第 625–641 页。[108] T. Liang、Y. Wang、Z. Tang、G. Hu 和 H. Ling，“OpaNAS：用于目标检测的单次路径聚合网络架构搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2021年，第 10 195–10 203 页。[109] M. Chen、J. Fu 和 H. Ling，“通过多样性引导的搜索空间缩减进行单次神经集成架构搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2021年，第 16 530–16 539 页。[110] X. Dong 和 Y. Yang，“通过自评估模板网络进行单次神经架构搜索”，载于 IEEE/CVF国际计算机视觉会议论文集，2019年，第 3681–3690 页。[111] H. Cai、C. Gan、T. Wang、Z. Zhang 和 S. Han，“Once for all：训练一个网络并使其专用于高效部署”，载于 国际学习表征会议论文集， 2020，[在线]。获取地址：https://openreview.net/forum?id=HylxE1HKwS。[112] B. Yan、H. Peng、K. Wu、D. Wang、J. Fu 和 H. Lu，“LightTrack：通过单次架构搜索寻找用于目标跟踪的轻量级神经网络”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2021年，第 15 180–15 189 页。[113] O. T.-C. Chen、Y. C. Zhang、Y.-X. Chang 和 Y.-L. Chang，“面向优化性能的迭代式多路径单次神经架构搜索”，载于 IEEE 国际电路与系统研讨会，2021年，第 1–5 页。[114] X. Li、C. Lin、C. Li、M. Sun、W. Wu、J. Yan 和 W. Ouyang，“通过抑制后验衰减改进单次神经架构搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2020年，第 13 836–13 845 页。[115] H. Shi、R. Pi、H. Xu、Z. Li、J. Kwok 和 T. Zhang，“通过 BONAS 弥合基于样本的神经架构搜索与单次神经架构搜索之间的差距”，载于 神经信息处理系统进展，第 33 卷，2020年，第 1808–1819 页。[116] D. Wang、M. Li、C. Gong 和 V. Chandra，“AttentiveNAS：通过注意力采样改进神经架构搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2021年，第 6418–6427 页。[117] K. Liu、R. Ding、Z. Zou、L. Wang 和 W. Tang，“图网络中权重共享在三维人体姿态估计方面的全面研究”，载于 欧洲计算机视觉会议论文集，2020年，第 318–334 页。[118] X. Wang、J. Lin、J. Zhao、X. Yang 和 J. Yan，“EAutoDet：用于目标检测的高效架构搜索”，载于 欧洲计算机视觉会议论文集，2022年，第 668–684 页。[119] K. Xu 和 G. He，“DNAS：一种解耦式全局神经架构搜索方法”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2022年，第 1979–1985 页。[120] H. Yu、H. Peng、Y. Huang、J. Fu、H. Du、L. Wang 和 H. Ling，“循环式可微架构搜索”，《IEEE模式分析与机器智能汇刊》，第 45 卷，第 1 期，第 211–228 页，2022年。[121] M. Zhang、S. Pan、X. Chang、S. Su、J. Hu、G. R. Haffari 和 B. Yang，“BaLeNAS：通过贝叶斯学习规则进行可微架构搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2022年，第 11 871–11 880 页。

[122] T. Huang、S. You、F. Wang、C. Qian、C. Zhang、X. Wang 和 C. Xu，“GreedyNASv2：采用贪心路径筛选器进行更积极的搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2022年，第 11 902–11 911 页。[123] X. Dong 和 Y. Yang，“在四个图形处理器小时内搜索稳健的神经架构”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2019年，第 1761–1770 页。[124] M. Zhang、H. Li、S. Pan、X. Chang 和 S. Su，“通过多样性最大化克服单次神经架构搜索中的多模型遗忘”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2020年，第 7809–7818 页。[125] J. Pan、C. Sun、Y. Zhou、Y. Zhang 和 C. Li，“分布一致性神经架构搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2022年，第 10 884–10 893 页。[126] Y. Akimoto、S. Shirakawa、N. Yoshinari、K. Uchida、S. Saito 和 K. Nishida，“用于单次神经架构搜索的自适应随机自然梯度法”，载于 国际机器学习会议论文集，2019年，第 171–180 页。[127] Q. Zhou、X. Zheng、L. Cao、B. Zhong、T. Xi、G. Zhang、E. Ding、M. Xu 和 R. Ji，“EC-DARTS：将均衡且一致的优化引入 DARTS”，载于 IEEE/CVF国际计算机视觉会议论文集，2021年，第 11 986–11 995 页。[128] B. Wu、X. Dai、P. Zhang、Y. Wang、F. Sun、Y. Wu、Y. Tian、P. Vajda、Y. Jia 和 K. Keutzer，“FBNet：通过可微神经架构搜索实现硬件感知高效卷积网络设计”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2019年，第 10 734–10 742 页。[129] A. Wan、X. Dai、P. Zhang、Z. He、Y. Tian、S. Xie、B. Wu、M. Yu、T. Xu、K. Chen 等人，“FBNetV2：针对空间和通道维度的可微神经架构搜索”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2020年，第 12 965–12 974 页。[130] N. Nayman、Y. Aflalo、A. Noy 和 L. Zelnik，“HardCoRe-NAS：硬约束可微神经架构搜索”，载于 国际机器学习会议论文集，2021年，第 7979–7990 页。[131] M. Zhang、S. W. Su、S. Pan、X. Chang、E. M. Abbasnejad 和 R. Haffari，“iDARTS：采用随机隐式梯度的可微架构搜索”，载于 国际机器学习会议论文集，2021年，第 12 557–12 566 页。[132] S. Xue、R. Wang、B. Zhang、T. Wang、G. Guo 和 D. Doermann，“IDARTS：交互式可微架构搜索”，载于 IEEE/CVF国际计算机视觉会议论文集，2021年，第 1163–1172 页。[133] S. Tang、M. S. Hosseini、L. Chen、S. Varma、C. Rowsell、S. Damaskinos、K. N. Plataniotis 和 Z. Wang，“可探测DARTS及其在计算病理学中的应用”，载于 IEEE/CVF国际计算机视觉会议论文集，2021年，第 572–581 页。[134] C. Xue、X. Wang、J. Yan、Y. Hu、X. Yang 和 K. Sun，“重新思考神经架构搜索中的双层优化：吉布斯采样视角”，载于 AAAI人工智能会议论文集，第 35 卷，第 12 期，2021年，第 10 551–10 559 页。[135] P. Hou、Y. Jin 和 Y. Chen，“Single-DARTS：迈向稳定的架构搜索”，载于 IEEE/CVF国际计算机视觉会议论文集，2021年，第 373–382 页。[136] Y. Li、Z. Wen、Y. Wang 和 C. Xu，“采用动态搜索空间的单次图神经架构搜索”，载于 AAAI人工智能会议论文集，第 35 卷，第 10 期，2021年，第 8510–8517 页。[137] M. S. Abdelfattah、A. Mehrotra、Ł. Dudziak 和 N. D. Lane，“用于轻量级神经架构搜索的零成本代理”，载于 国际学习表征会议论文集，2020，[在线]。获取地址：https://openreview.net/forum?id=0cmMMy8J5q。[138] Y. Shu、Z. Dai、Z. Wu 和 B. K. H. Low，“统一并增强基于梯度的免训练神经架构搜索”，arXiv 预印本 arXiv:2201.09785，2022年。[139] G. Li、Y. Yang、K. Bhardwaj 和 R. Marculescu，“Zico：通过梯度变异系数的倒数进行零样本神经架构搜索”，载于 国际学习表征会议论文集，2023，[在线]。获取地址：https://openreview.net/forum?id=rwo-ls5GqGn。[140] J. Mellor、J. Turner、A. Storkey 和 E. J. Crowley，“无训练神经架构搜索”，arXiv 预印本 arXiv:2006.04647v1，2020年。[141] ——，“无训练神经架构搜索”，载于 国际机器学习会议论文集，2021年，第 7588–7598 页。[142] M. Lin、P. Wang、Z. Sun、H. Chen、X. Sun、Q. Qian、H. Li 和 R. Jin，“Zen-NAS：用于高性能图像识别的零样本神经架构搜索”，载于 IEEE/CVF国际计算机视觉会议论文集，2021年，第 347–356 页。[143] W. Chen、X. Gong 和 Z. Wang，“在四个图形处理器小时内于 ImageNet 上进行神经架构搜索：一种理论启发视角”，载于 国际学习表征会议论文集，2021，[在线]。获取地址：https://openreview.net/forum?id=Cnon5ezMHtu。[144] Y. Shu、S. Cai、Z. Dai、B. C. Ooi 和 B. K. H. Low，“NASI：一种在初始化时进行的与标签和数据无关的神经架构搜索”，载于《国际学习表征会议论文集》，2021，[线上]。可访问：https://openreview.net/forum?id=v-v1cpNNK v。[145] J. Xu、L. Zhao、J. Lin、R. Gao、X. Sun 和 H. Yang，“KNAS：绿色神经架构搜索”，载于《国际机器学习会议论文集》，2021年，第 11 613–11 625页。[146] Q. Zhou、K. Sheng、X. Zheng、K. Li、X. Sun、Y. Tian、J. Chen 和 R. Ji，“免训练变换器架构搜索”，载于《IEEE/CVF计算机视觉与模式识别会议论文集》，2022年，第10 894–10 903页。[147] N. Lee、T. Ajanthan 和 P. H. Torr，“SNIP：基于连接敏感度的单次网络剪枝”，载于《国际学习表征会议论文集》，2019，[线上]。可访问：https://openreview.net/forum?id=B1VZqjAcYX。[148] C. Wang、G. Zhang 和 R. Grosse，“通过保持梯度流，在训练前挑选中奖彩票”，载于《国际学习表征会议论文集》，2019，[线上]。可访问：https://openreview.net/forum?id=SkgsACVKPH。[149] L. Theis、I. Korshunova、A. Tejani 和 F. Husz´ar，“ 利用稠密网络和费 Fisher 剪枝加速注视预测 ”，arXiv 预印本 arXiv:1801.05787，2018年。[150] J. Turner、E. J. Crowley、M. O’Boyle、A. Storkey 和 G. Gray，“Blockswap：面向预算有限的网络压缩的 Fisher 引导块替换”，载于《国际学习表征会议论文集》，2020, [线上]。可访问：https://openreview.net/forum?id=SklkDkSFPB。[151] H. Tanaka、D. Kunin、D. L. Yamins 和 S. Ganguli，“通过迭代保持突触流，在不使用任何数据的情况下剪枝神经网络”，载于《神经信息处理系统进展》，第33卷，2020年，第6377–6389页。[152] X. Ning、C. Tang、W. Li、Z. Zhou、S. Liang、H. Yang 和 Y. Wang，“评估神经架构的高效性能估计器”，载于《神经信息处理系统进展》，第34卷，2021年，第12 265–12 277页。[153] P. K. Sen，“基于肯德尔秩相关系数的回归系数估计”，《美国统计学会杂志》，第63卷，第324期，第1379–1389页，1968年。[154] J. Hauke 和 T. Kossowski，“在相同数据集上比较 Pearson 相关系数与斯皮尔曼相关系数的值”，《地理学问题》，第30卷，第2期，第87–93页，2011年。[155] C. Ying、A. Klein、E. Christiansen、E. Real、K. Murphy 和 F. Hutter，“NAS-Bench-101：迈向可复现的神经架构搜索”，载于《国际机器学习会议论文集》，2019年，第7105–7114页。[156] J. Siems、L. Zimmer、A. Zela、J. Lukasik、M. Keuper 和 F. Hutter，“NAS-Bench-301：论神经架构搜索代理基准的必要性”，arXiv预印本 arXiv:2008.09777，2020年。[157] N. Klyuchnikov、I. Trofimov、E. Artemova、M. Salnikov、M. Fedorov、A. Filippov 和 E. Burnaev，“NAS-Bench-NLP：自然语言处理神经架构搜索基准”，《IEEE Access》，第10卷，第45 736–45 747页，2022年。[158] A. Mehrotra、A. G. C. Ramos、S. Bhattacharya、Ł. Dudziak、R. Vip- perla、T. Chau、M. S. Abdelfattah、S. Ishtiaq 和 N. D. Lane，“NAS-Bench-ASR：可复现的语音识别神经架构搜索”，载于《国际学习表征会议论文集》，2020, [线上]。可访问：https://openreview.net/forum?id=CU0APx9LMaL。[159] F. X. Han、K. G. Mills、F. Chudak、P. Riahi、M. Salameh、J. Zhang、W. Lu、S. Jui 和 D. Niu，“面向神经架构搜索的通用可迁移预测器”，载于《SIAM国际数据挖掘会议论文集》，2023年，第721–729页。[160] Y. Zhang 和 Q. Yang，“多任务学习综述”，《IEEE知识与数据工程汇刊》，第34卷，第12期，第5586–5609页，2022年。[161] G. Shala、T. Elsken、F. Hutter 和 J. Grabocka，“基于元学习贝叶斯代理模型的迁移 NAS”，载于《国际学习表征会议论文集》，2022，[线上]。可访问：https://openreview.net/forum?id=paGvsrl4Ntr。[162] Y. Duan、X. Chen、H. Xu、Z. Chen、X. Liang、T. Zhang 和 Z. Li，“TransNAS-Bench-101：提升跨任务神经架构搜索的可迁移性与泛化能力”，载于《IEEE/CVF计算机视觉与模式识别会议论文集》，2021年，第5251–5260页。[163] C. White、W. Neiswanger、S. Nolen 和 Y. Savani，“神经架构搜索编码研究”，载于《神经信息处理系统进展》，第33卷，2020年，第20 309–20 319 页。[164] Y. Sun、G. G. Yen、B. Xue、M. Zhang 和 J. Lv，“ArcText：描述卷积神经网络架构的统一文本方法”，《IEEE人工智能汇刊》，第3卷，第4期，第526–540页，2022年。