基于Transformer结构增强的神经网络架构搜索性能预测器

王继禾[1)] 吴 颖[1),2)] 迟恒喆[1),2)] 王党辉[1),3)] 梅魁志[4)]

[1)](西北工业大学计算机学院 西安 710129)

[2)](嵌入式系统集成教育部工程中心 西安 710129)

[3)](空天地海一体化大数据应用技术国家工程实验室 西安 710129)

[4)](西安交通大学人工智能与机器人研究所 西安 710049)

摘 要 神经网络架构搜索(Neural Architecture Search,NAS)作为一种通过搜索算法设计神经网络架构的方法,在计算机视觉和自然语言处理等领域得到广泛应用,相较于人工设计网络,NAS方法可以减少设计成本并提高模型性能.但是NAS的性能评估需要对候选架构进行大量训练,由此带来的计算量占整个NAS的80%以上.为降低计算开销和时间成本,近年来已提出许多基于Transformer的NAS预测器,由于Transformer出色的结构编码能力可以更好地表示拓扑信息,因而得到广泛应用.但是,现有基于Transformer的NAS预测器依然存在三个问题:其一是在预处理阶段,传统的One-hot编码方式描述节点特征的能力较弱,只能区分不同操作节点类型,而难以表达操作的细节特征,如卷积核尺寸等.其二是在编码阶段,Transformer的自注意力机制导致模型结构信息缺失;其三是在评估阶段,现有的Transformer预测器仅使用多层感知机(Multilayer Perceptron,MLP)对前向传播图进行精度预测,忽略了反向传播梯度流对预测精度的影响,因此难以真正拟合NAS评估中的正、反向交替信息流图,导致预测器精度与实际运行精度误差波动极大(10%~90%).为解决上述问题,本文提出了一种基于Transformer结构增强的NAS性能预测方法.首先,在预处理阶段,本文提出了一种超维嵌入方法增加输入数据维度以强化节点操作的参数描述能力,其次,在编码阶段将Transformer编码后的信息与图结构信息共同输入一个图卷积网络(Graph Convolutional Network,GCN),弥补由自注意力机制引起的结构缺失.最后,在性能评估阶段,本文构建了同时包含前向传播和反向传播的全训练图,并将数据集信息、图结构编码与梯度编码共同输入到GCN网络预测器中,使预测结果更贴近模型真实性能.实验结果表明,本方法与目前最先进方法相比,肯德尔相关系数提高了7.45%,训练时间减少了1.55倍.

关键词 预测器;NAS;Transformer;GCN;Embedding

中图法分类号 TP18 **DOI号** 10.11897/SP.J.1016.2024.01469

An Architecture-Enhanced Performance Predictor for Transformer-Based NAS

WANG Ji-He[1)] WU Ying[1),2)] CHI Heng-Zhe[1),2)] WANG Dang-Hui[1),3)] MEI Kui-Zhi[4)]

[1)](*School of Computer Science, Northwestern Polytechnical University, Xi'an*, 710129)

[2)](*Engineering Center for Embedded System Integration, Ministry of Education, Xi'an*, 710129)

[3)](*National Engineering Laboratory for Integrated Aero-Space-Ground-Ocean Big Data Application Technology (ASGO), Xi'an*, 710129)

[4)](*Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an*, 710049)

Abstract Neural Architecture Search (NAS) represents a paradigm shift in the design of neural

收稿日期:2023-08-17;在线发布日期:2024-04-17.本课题得到国家自然科学基金"面向边缘智能的随机近存计算专用体系结构研究"(No.62272393)、"无人系统自主定位及环境建模的计算优化与软硬件协同设计"(No.62076193)资助.**王继禾**(通信作者),博士,副教授,中国计算机学会(CCF)会员,主要研究领域为高效能人工智能系统、边缘计算等.E-mail:wangjihe@nwpu.edu.cn.**吴 颖**,硕士研究生,主要研究领域为图计算、分布式计算.**迟恒喆**,硕士研究生,主要研究领域为图计算.**王党辉**,博士研究生,副教授,中国计算机学会(CCF)会员,主要研究领域为计算机体系结构,边缘智能系统等.**梅魁志**,博士,教授,主要研究领域为图像分析,嵌入式机器视觉等.

network architectures, transitioning from manual, intuition-based processes to automated, algorithm-driven methods. This evolution has significantly reduced design costs and improved model performance across various tasks in computer vision, natural language processing, and beyond. Despite these advances, the computational expense associated with evaluating the performance of candidate architectures remains a formidable challenge, consuming the lion's share of resources in NAS endeavors. The emergence of NAS predictors based on the Transformer architecture has offered a promising pathway to mitigating these computational demands. The Transformer's superior capability for encoding topological information of neural networks has made it an attractive foundation for developing predictors that can efficiently approximate the performance of vast numbers of architectures without the need for exhaustive training. However, the deployment of Transformer-based NAS predictors has encountered significant obstacles. The initial challenge lies in the pretreatment phase, where traditional encoding schemes like One-hot encoding fall short in capturing the full spectrum of node features within a network architecture. Such schemes can identify different types of operations but lack the granularity to describe the operational parameters with sufficient detail, such as the sizes of convolutional kernels, which are critical for accurately modeling network behavior. During the encoding stage, the reliance on the Transformer's self-attention mechanism introduces a second challenge by potentially overlooking essential structural information. The self-attention mechanism, while powerful for capturing global dependencies, may not fully preserve the hierarchical and spatial relationships inherent in neural network architectures, which are crucial for understanding their performance. The evaluation phase presents a third hurdle, where existing predictors mainly utilize Multilayer Perceptron (MLP) to estimate the performance of architectures based on forward propagation alone. This approach neglects the critical influence of back propagation gradients, an oversight that can lead to substantial inaccuracies. The dynamic interplay between forward and backward propagation in training neural networks is complex, and any predictor that fails to account for this interplay is likely to suffer from considerable predictive error, with discrepancies ranging widely between the estimated and actual performance. Addressing these challenges, our proposed solution, "An Architecture-Enhanced Performance Predictor for Transformer-based NAS," introduces innovative strategies at each stage of the prediction process. In the pretreatment phase, we advocate for a novel encoding approach, Hyper-dimensional Embedding, which significantly expands the input feature space to provide a richer, more nuanced representation of node operations. This method allows for a more detailed characterization of architectural components, thereby enhancing the predictive model's ability to discern subtle differences between architectures. To resolve the issues identified in the encoding stage, our approach integrates the Transformer's output with additional structural information through a Graph Convolutional Network (GCN). This strategy ensures that the spatial and hierarchical relationships between architectural components are maintained and factored into the prediction process, addressing the deficiencies of the self-attention mechanism in preserving structural integrity. Finally, in the evaluation phase, we construct a more comprehensive model of network behavior by incorporating both forward and backward propagation dynamics into our predictive framework. This involves feeding a combination of dataset characteristics, structural encodings, and gradient information into the GCN predictor, thereby achieving a more accurate and holistic assessment of architectural performance. Our experimental results demonstrate the effectiveness of this approach, showing a marked improvement in the Kendall correlation coefficient by 7.45%

and a reduction in training time by 1.55 times compared to state-of-the-art methods. This significant advancement underscores the potential of our proposed solution to enhance the efficiency and accuracy of NAS processes, paving the way for more rapid and cost-effective development of high-performing neural networks.

Keywords　predictor; NAS; transformer; GCN; embedding

1　引　言

近年来，神经网络架构搜索（NAS）作为一种通过搜索算法设计神经网络架构的方法，在计算机视觉和自然语言处理等领域得到了广泛应用，相较于人工设计网络，NAS 方法可以减少设计成本并提高模型性能. 如图 1 所示，NAS 过程首先从搜索空间中寻找候选架构，其次由 NAS 性能预测器对候选架构的推理精度进行预测，最后根据预测器的性能评估结果调整搜索策略并寻找下一个候选架构，直至找到搜索空间中的最优架构. 因此性能评估阶段预测器的性能直接影响到 NAS 的搜索结果.

NAS 的任务目标是优化下述目标函数：

$$a^* = \mathrm{argmax} f(\boldsymbol{a}_{\mathrm{Inf}}, \boldsymbol{D}) \tag{1}$$

其中 $\boldsymbol{a}_{\mathrm{Inf}}$ 表示搜索空间 A 内的所有候选推理图架构，$\boldsymbol{D}$ 表示对候选架构性能评估时应用的数据集信息，函数的目的是在 $\boldsymbol{a}_{\mathrm{Inf}}$ 中挑选出最佳架构 a^*. 在这些预测器中，基于 Transformer 的预测器因具有较高的推理精度而被广泛应用，具体来说，预测器通过结构 Embedding 和数据集 Embedding 学习特征信息，使得预测结果不仅依赖于架构本身，还考虑了数据集的特性，如数据量和复杂度. 然而，这些方法仍然面临着预测精度不足的问题. 如图 1 所示，这主要是因为现存基于 Transformer 的 NAS 性能预测器在预处理、编码和评估三个阶段中分别存在以下问题.

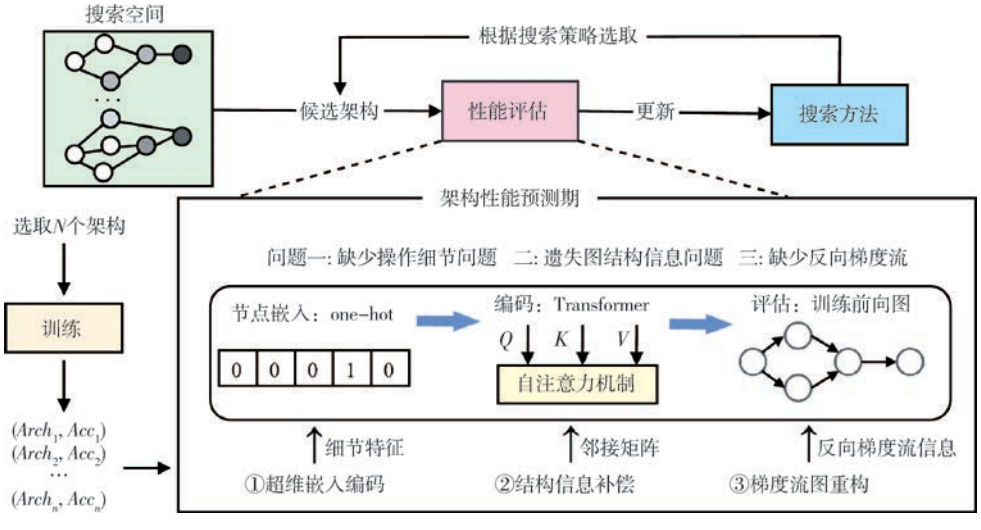

图 1　基于 Transformer 的 NAS 性能预测器及问题剖析

在预处理阶段，传统的 One-hot 编码方式只能描述节点操作的类型，例如输入、输出、最大池化、卷积、跳过连接和置零，而难以定量描述操作的细节特征，如卷积核大小等，这在 Transformer 的自注意力机制期间无法提供充足的架构信息.

在编码阶段，预测器会将输入数据中所有节点的信息编码为 Embedding 向量，但在权重分配过程中，Transformer 的自注意力机制会导致部分节点因权重过低而被忽略[1,2]，从而导致编码后的 Embedding 无法完整描述图结构，这被称为“结构缺失”问题. 因此，将编码后的结构向量输入 MLP[3] 进行评估时，评估结果与原架构的相关性较低（肯德尔系数＜0.73）[4,5]. 因此，应在此阶段对结构信息进行补偿.

在评估阶段，图神经网络（GNN）将候选架构视为图数据，传统的 GNN 模型仅评估了每个候选架构的前向传播过程，以拟合前向推理的特征流[6,7]. 但是，仅仅评估前向图是有缺陷的，因为在真实的神

经架构搜索过程中,神经网络的训练既包括前向传播也包括反向传播[8].反向传播梯度流的丢失意味着对候选架构的性能描述不完整,这将误导预测器的评估和判断.因此,如图 1 所示,将反向传播梯度流信息作为参数纳入到整体评估中,是一种较为符合实际训练过程的预测方法.

基于上述问题及分析,本文提出了基于 Transformer 结构增强的 NAS 性能预测器,首先,在预处理阶段,为了丰富图结构信息,本文设计了专用的节点编码方法来描述节点信息,称为"超维嵌入",与传统的 One-hot[9] 方法相比,超维嵌入方法增加了输入数据的维度并输入 Transformer,该方法可以描述节点操作的更多细节特征,如内核尺寸,填充尺寸等.通过超维嵌入方法,本文对图节点特征进行了更加全面的描述;其次,在编码阶段,为了补偿由 Transformer 编码器引起的"结构缺失",本文将编码后的图特征向量与结构信息共同作为参数输入 GCN 进行二次编码,得到完整图结构的特征向量,进而对缺失的图结构进行补偿;最后,为了引入反向梯度流信息,本文根据链式法则将原推理图拓展为包含反向传播过程的全训练图并用以训练,全训练图中明确定义了每个反向传播过程,以馈入反向梯度流信息.通过增加训练图的反向传播过程,构建同时包含前向传播和反向传播的全训练图,便可以在评估阶段将架构信息,数据集信息和梯度信息共同作为输入数据馈入图卷积网络(GCN)预测器,并将训练后的 GCN 作为预测器输出网络精度预测结果.

实验验证中,本文在通用的 NAS 基准测试上进行的实验表明,与目前最先进的方法相比,预测器的肯德尔相关性系数提高了 7.45%,训练时间降低了 1.55 倍.

本文的贡献主要包括四个方面:

(1)本文为图神经网络提出了一种"超维嵌入"编码方法,可以丰富对节点特征的描述.

(2)本文提出了一种增强图结构信息的图编码方法,以补偿由 Transformer 编码器自注意力机制引起的结构缺失问题.

(3)本文引入了包含反向梯度流的全训练图,以替代前向推理图,以在评估阶段包含梯度信息,能模拟真实的 NAS 数据流动过程,提高预测精度.

(4)实验结果证明了本文提出的方法在多任务 NAS 中的有效性,与目前最先进的方法相比,有着更高的预测精度和更快的搜索速度.

2 相关工作

2.1 One-hot 节点编码

在预处理阶段均对图节点操作采用 One-hot 编码方式可以有效区分不同节点的操作类型,然而,One-hot 编码的劣势也非常明显.首先,它无法表达更多的操作细节.例如,在神经网络架构中,仅仅知道一个节点执行的是卷积操作并不足够,因为卷积操作有许多参数,如卷积核的大小、步长、填充方式等,这些都是影响网络性能的重要因素.One-hot 编码无法描述这些细节特征,从而限制了它在表示复杂网络结构时的效力.其次,One-hot 编码无法表达节点间的关系,网络架构中节点的连接方式对于网络性能同样至关重要,但 One-hot 编码无法捕捉这种结构信息,这进一步限制了其在复杂网络架构表示中的应用.尽管 One-hot 编码在一定程度上简化了神经网络架构的表示,使模型能够区分不同的操作类型,但其在描述操作细节、处理高维数据和表达结构信息方面的局限性也促使研究者们寻求更高效、更全面的编码方法,以更准确地反映神经网络架构的性能和特性.

2.2 基于 Transformer 的 NAS 性能预测器

由于 Transformer 具有强大的编码能力与特征提取能力[10],可以提高 NAS 预测器的精度,因此 Lu 等人在 2021 年提出的基于 Transformer 的 NAS 性能预测器[3]被广泛应用.该方法处理流程主要分为三个阶段:在预处理阶段,使用 One-hot 编码将候选架构的节点特征信息编码为节点特征 Embedding;在编码阶段,特征 Embedding 与图的拉普拉斯矩阵被共同馈入 Transformer 编码器以编码位置信息,最终输出图特征 Embedding,以表示图结构信息;在预测阶段,图特征 Embedding 被馈入到一个两层 MLP,以预测架构的性能精度.

但是 Nguyen[1] 和 Ying[2] 等人在 2022 年随后指出,这种将 Transformer 应用于图编码将导致结构缺失问题,这是因为 Transformer 的自注意力机制可能会导致部分图结点的权重过低而被忽略,进而导致对图结构的描述不完整,所以此方法的预测精度仍然较低.Lu 等人[11] 于 2023 年进一步提出了使用基于置换不变性的 Transformer 编码器来增强架构图的编码信息,但是将 Transformer 应用于图编码的结构缺失问题仍然存在.

2.3 神经网络架构图编码

图神经网络[12,13] 将神经网络架构视为图数据

进行处理,图的编码方式[14]是影响图神经网络性能的重要因素.目前的 NAS 预测器[15]普遍使用同步 GNN 聚合方法进行图编码.但是同步聚合也存在信息流动相对受限的缺点.Lukasik 等人[16]进一步通过 T 轮前向和反向传播来更好地学习图 Embedding 向量.Ning 等人[1]则将神经网络架构中的节点操作视为特定类型,并使用信息传播机制来模拟神经网络中数据的前向传播.Jing 等人[17]使用基于图掩码自动编码器的增强预测器 GMAE 和自监督的预训练方法进行架构图的学习.然而,上述方法只能从输入的图编码信息中学习非常有限的架构知识,导致很多关键信息被遗漏,例如控制架构性能的超参数信息,这将影响图神经网络的学习能力以及最终的推理精度.

在评估阶段,上述方法在推理的最后一层都仅使用 MLP 对前向传播图进行精度预测,忽略了反向传播梯度流对预测精度的影响,这意味着无法捕捉网络在训练过程中的动态学习行为.因此难以真正拟合 NAS 评估中的正反向交替信息流过程.为了提高 NAS 评估的准确性和效率,开发能够综合考虑前向传播和反向传播过程的评估方法极为重要.

基于上述分析,本文将三个关键参数,即数据集信息、图结构编码和梯度流编码共同馈入到一个专用 GCN 预测器中,通过实验验证,本文的 NAS 预测器在常用基准测试数据集 NAS-bench101、NAS-bench-201、NAS-bench-301 上经过定性与定量分析,都具有更好的精度,同时降低了 NAS 的搜索速度.

3　基于 Transformer 结构增强的 NAS 性能预测器

3.1　结构增强的 NAS 性能预测器整体架构

在图结构的性能预测精度方面,Transformer 的优势在于:结合图结构中不同局部的子空间信息以获得整体结构信息[3].然而,它的预测结果仍然与真实性能差距较大(肯德尔系数<0.73),同时,其注意力机制所引起的结构缺失问题还会误导 NAS 最优架构的搜索.这主要是因为将公式(1)中的 $\boldsymbol{a}_{\mathrm{Inf}}$ 视为图数据时,图中只包括候选网络的前向推理图,而并不涉及反向传播的梯度信息,但真实的训练过程是通过反向传播梯度流对参数进行调整,两者在行为上差异较大.因此以公式(1)为搜索目标的 NAS 过程难以寻找到较高精度的网络.

首先,Transformer 的自注意力机制可能会弱化图编码中的结构信息.在自注意力机制编码之前,需要先将节点信息编码为向量,传统的方法只使用 One-hot 编码和低维编码粗略表示节点的操作类型(例如输入、输出、最大池化、卷积、跳过连接和置零等),这在自注意力机制期间无法提供充足的架构信息.而更糟糕的是,虽然自注意力机制可以突出源节点与距离较远的节点之间的关系,但实际在特征传递过程中,相邻节点对源节点的影响更大,因此自注意力机制可能会导致相邻结点的权重被远距离节点干扰,从而误导了 NAS 对整个图结构信息的判断.因此,丰富节点信息以及增加额外的结构信息注入对于 Transformer 这种基于自注意力机制的预测器来说很有必要.

其次,基于 $\boldsymbol{a}_{\mathrm{Inf}}$ 的预测函数忽略了反向传播的梯度流信息,与神经网络不同,在图神经网络的数据流动过程中,反向传播是提高架构性能的重要过程,忽略反向传播而仅考虑前向传播过程将导致梯度信息不足,进而影响 NAS 对图结构信息的学习,目前仅基于 $\boldsymbol{a}_{\mathrm{Inf}}$ 的预测器找到的最优架构也只能达到较低的预测精度(低于最先进技术的精度).因此,如果用一个包含特征流(前向传播)和梯度流(反向传播)的全训练图 $\boldsymbol{a}_{\mathrm{Tra}}$ 来替换 $\boldsymbol{a}_{\mathrm{Inf}}$,预测器的精度就能更接近 NAS 的真实水平.

为了解决上述问题,本文在公式(2)中提出了基于 Transformer 结构增强的 NAS 性能预测器,如图 2 所示,为了提高预测精度,本文将三个参数,即架构信息($\boldsymbol{a}_{\mathrm{Tra}}$)、数据集信息($\boldsymbol{D}$)和梯度流信息($\nabla$),共同输入预测器,以丰富预测器的学习内容.

$$a^{*}=\underset{\boldsymbol{a}_{\mathrm{Tra}}\in\boldsymbol{A}}{\operatorname{argmax}} f^{*}(\boldsymbol{a}_{\mathrm{Tra}},\boldsymbol{D},\nabla) \tag{2}$$

为实现公式(2),本文提出了一个新的编码器 f^{*},并通过以下方法来突出结构信息,1)使用超维嵌入编码方法丰富节点操作的细节特征,2)将图的结构信息与编码信息共同输入 GCN 网络以补偿 Transformer 引起的"结构缺失",3)为了引入反向传播的梯度流,本文将反向梯度图追加到了原始的前向推理图中并构建了一个拓展的训练图($\boldsymbol{a}_{\mathrm{Tra}}$).梯度图中的边及方向由链式法则定义,边的方向即是反向传播期间的梯度流(∇)方向.本文在 3.2 节和 3.3 节中将分别介绍新的图节点编码方式和图编码方式,在 3.4 节中本文将介绍全训练图的生成过程,并在 3.5 节中详细说明将该预测器置入 NAS 的方法.

3.2　预处理阶段:超维嵌入节点信息

图神经网络的神经元架构关系可以用一个有向

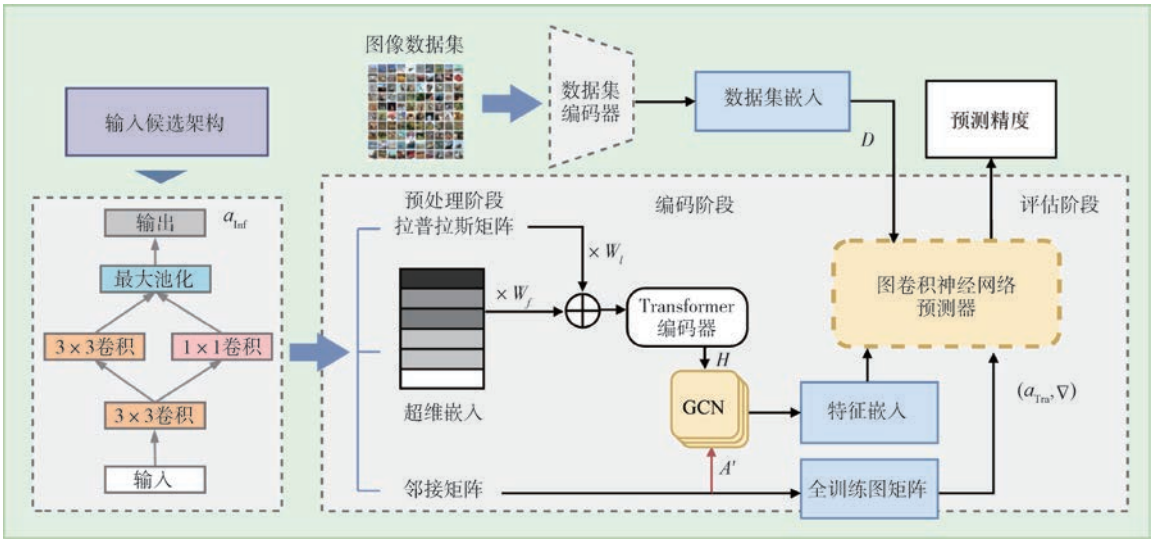

图 2 基于 Transformer 结构增强的 NAS 性能预测器设计图

无环图(DAG)，$G=(V,E)$ 来描述，假设图的节点集合是 $V=\{v_1,v_2,\cdots,v_n\}$，且每个节点具有 m 个操作(如卷积、池化或跳过连接等)，则其特征矩阵为 $\boldsymbol{X}\in R^{n\times m}$. 边 $e_{i\to j}\in E$ 表示节点 v_i 到 v_j 的一条有向连接. 图中边的关系可以用邻接矩阵 $\boldsymbol{A}\in R^{n\times n}$ 和度矩阵 $\boldsymbol{D}\in R^{n\times n}$ 表示，图的拉普拉斯矩阵 $\boldsymbol{L}\in R^{n\times n}$ 可由 $\boldsymbol{L}=\boldsymbol{D}-\boldsymbol{A}$ 计算得出.

如图 3(a)所示，在对图节点编码时，传统的 One-hot 编码方法使用 n 个比特位来表示 n 种操作类型，但这种方法仅仅可以区分神经元节点的不同操作模式，例如卷积、最大池化、全连接等. 然而图结构的性能不仅取决于每个神经元节点的操作类型，还取决于操作的细节特征.

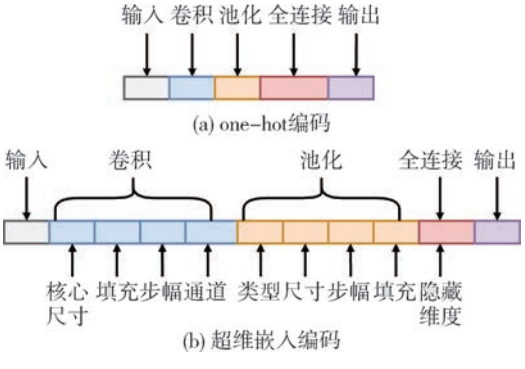

图 3 传统的 One-hot 编码与本文提出的超维嵌入编码

例如，同样是卷积模式，3×3 卷积能够保留特征的高频信号，而 7×7 卷积只能传递低频信号，操作类型相同，但不同的卷积规模意味着卷积在图结构中发挥着不同的作用. 如图 3(b)所示，为了在编码节点信息时，既要包括节点操作类型信息也要包括操作的细节特征，本文在 One-hot 编码基础上，增加了更多比特位来描述节点的细节特征，称为超维嵌入，以表示不同操作类型下的超维嵌入细节特征，见表 1.

表 1 图神经网络中不同 CNN 操作的超维嵌入细节特征

操作模式	超维嵌入细节特征
卷积	核心尺寸，填充，步幅，通道
池化	类型，尺寸，填充，步幅
全连接	隐藏维度

在超维嵌入编码后，本文将编码向量输入到一个变换矩阵 $\boldsymbol{W}_f\in R^{m\times k}$，并将其转换为 k 维矩阵，以匹配 Transformer 的操作空间(k 维)并训练. 通过在特征矩阵上应用 Transformer 编码器的 Multi-Head 自注意力机制[18]，可以增强图中不同子空间的结构信息.

$$\boldsymbol{H}_N=Attention[(\boldsymbol{X}\boldsymbol{W}_f+\boldsymbol{L}\boldsymbol{W}_p)] \tag{3}$$

如公式(3)所示，本文使用拉普拉斯矩阵 $\boldsymbol{L}$ 进行位置编码[3]，使输入数据可以携带位置信息，然后通过一个变换矩阵 $\boldsymbol{W}_p\in R^{n\times k}$ 将 $\boldsymbol{L}$ 转换为 k 维.

3.3 编码阶段：补偿结构缺失

3.3.1 图编码器

节点特征矩阵 $\boldsymbol{H}\in R^{n\times k}$ 的定义是 $\boldsymbol{W}_f\boldsymbol{X}$ 和 $\boldsymbol{W}_f\boldsymbol{L}$ 的加和，在自注意力机制中，输入数据为查询矩阵 $\boldsymbol{Q}$，键矩阵 $\boldsymbol{K}$ 和值矩阵 $\boldsymbol{V}$，公式如图(4)所示：

$$Attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d_k}})\boldsymbol{V}\in R^{n\times k} \tag{4}$$

如公式(5)所示，在本文的预测器中，Multi-

Head 自注意力机制的意义是组合图结构中不同位置的子空间信息,

$$\begin{cases} head_i = Attention(\boldsymbol{QW}_i^{\boldsymbol{Q}}, \boldsymbol{KW}_i^{\boldsymbol{K}}, \boldsymbol{VW}_i^{\boldsymbol{V}},) \\ MultiHead(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = (head_1,\cdots,head_h)\boldsymbol{W}^{\boldsymbol{O}} \\ \boldsymbol{W}_i^{\boldsymbol{Q}} \in R^{d_{model}\times d_k}, \boldsymbol{W}_i^{\boldsymbol{K}} \in R^{d_{model}\times d_k}, \\ \boldsymbol{W}_i^{\boldsymbol{V}} \in R^{d_{model}\times d_v}, \boldsymbol{W}^{\boldsymbol{O}} \in R^{d_v\times d_{model}}. \end{cases} \tag{5}$$

为了补偿结构信息,本文的预测器使用 GCN 而不是 MLP 作为编码器的最后一层,这可以解决传统方法中的"结构缺失".用于二次编码的 GCN 输入数据为 1)Transformer 的输出数据 $\boldsymbol{H}_N$ 和 2)邻接矩阵 $\boldsymbol{C}$.公式(6)在聚合层中,每个节点 h_j 都会结合相邻节点与自己的信息更新参数.公式(7)定义了层间的传播法则,其中输入 GCN 中的结构信息由对称邻接矩阵 $\boldsymbol{A}'=\boldsymbol{A}+\boldsymbol{A}^T$[19]决定.

$$\begin{cases} \boldsymbol{a}_i^{(l)} = AGGREGATE^{(l)}(\{\boldsymbol{h}_j^{(l-1)}:\boldsymbol{j} \in N_{(v_i)}\}) \\ \boldsymbol{h}_i^{(l)} = COMBINE^{(l)}(\boldsymbol{h}_j^{(l-1)}, \boldsymbol{a}_i^{(l)}) \end{cases} \tag{6}$$

$$\boldsymbol{H}_N^{(l+1)} = \sigma(\boldsymbol{A}'\boldsymbol{H}_N^{(l)}\boldsymbol{W}_{graph}^{(l)}) \tag{7}$$

其中,$\boldsymbol{W}_{graph} \in R^{k\times d_{out}}$ 是 GCN 层中的参数矩阵,σ 是 ReLU 激活函数. $\boldsymbol{H}_N^{(l)}$ 的每一行代表了一个节点的 Embedding,因为需要的是每个节点的 Embedding,而不是整个图的描述,所以本文在编码器中没有用到读出层.GCN 层在此过程中只用于编码作用,在训练中与最终的 GCN 预测器形成堆叠关系,GCN 预测器的标签为 NAS-Bench 数据集中的候选架构精度,通过联合训练更新 GCN 编码器的权重参数,编码器最终输出的是整个图的特征 Embedding 向量.

3.3.2　数据集编码器

在 NAS 任务中,数据集编码器的作用是描述数据集 D 的特征.本文数据集编码器与论文 MetaD2A[20] 类似,即在 ImageNet-1K 上预训练 ResNet18 模型以获得图像 Embedding.如公式(8)所示,本文使用了集合注意力块(SetAttentionBlock,SAB)和多头注意力池化操作(Pooling by Multi-Head Attention,PMA)[21]来学习数据集 D 每个类别中采样图像的特征,如公式(8)所示,SAB 模块使用自注意机制学习集合中每个元素的特征,PMA 将特征数据压缩为固定长度的向量.

$$\begin{cases} SAB(\boldsymbol{D}) = LN(\boldsymbol{D} + MLP(\boldsymbol{H}_{SAB})) \\ PMA(\boldsymbol{D}) = LN(\boldsymbol{D} + MLP(\boldsymbol{H}_{PMA})) \end{cases} \tag{8}$$

如公式(9)所示,$\boldsymbol{H}_{SAB}$ 和 $\boldsymbol{H}_{PMA}$ 由 Multi-Head 注意力层计算得到,LN 为层归一化操作.

$$\begin{cases} \boldsymbol{H}_{SAB} = MultiHead(\boldsymbol{D},\boldsymbol{D},\boldsymbol{D}) \\ \boldsymbol{H}_{PMA} = MultiHead(\boldsymbol{D},MLP(\boldsymbol{D}),MLP(\boldsymbol{D})) \end{cases} \tag{9}$$

由于集合的编码应该与输入元素的顺序信息无关,因此 SAB 和 PMA 模块中的函数都是逐行计算的,满足排列等价性(Permutation Equivalence,PE).

3.4　评估阶段:引入反向梯度流信息

本文提出的预测器使用 GCN 作为预测函数,其设计思路是将图结构信息,数据集信息和梯度流信息共同作为评估参数输入预测器,前两者已在前文中分别进行了介绍,在本节中,本文将介绍如何构建一个包含前向特征流(推理子图)和反向梯度流(梯度子图)的训练图,成为全训练图.

3.4.1　构建前向图和反向图

在训练的前向传播阶段,模型的参数是固定的,输入数据通过给定图结构的邻接矩阵 $\boldsymbol{C}_{n\times n}$,形成向前传播的特征流,直至得到此次传播的损失值.

$$\boldsymbol{M}_{\mathrm{Inf}} = \begin{pmatrix} x_1 & x_2 & \cdots & x_{n+1} \\ y_1 & c_{0,0} & \cdots & c_{0,n} \\ y_2 & c_{1,0} & \cdots & c_{1,n} \\ \cdots & \cdots & & \cdots \\ y_n & c_{n,0} & \cdots & c_{n,n} \end{pmatrix}, \boldsymbol{M}_{\mathrm{Grad}} = \boldsymbol{C}^{\mathrm{T}} \tag{10}$$

如图 4 和公式(10)所示,在传播过程中,本文添加了 1 个头节点 S_0,并在邻接矩阵的基础上,增加了 S_0 对应的行数据($\boldsymbol{x}=[x_1,x_2,\cdots,x_{n+1}]$)和列数据($\boldsymbol{y}=[y_1,y_2,\cdots,y_{n+1}]^{\mathrm{T}}$)以得到拓展的矩阵 $\boldsymbol{M}_{\mathrm{Inf}}$,由于图 4 中头节点($S_0$)只链入了神经网络的第一个神经单元($S_1$),所以除了 x_2 的值为 1 外,x 和 y 中其余元素的值均为 0.

在训练的反向传播中,网络中的每个神经元根据误差信号计算本层参数的梯度,并通过梯度下降法不断调整本层节点的参数(op_i).因此,其反向图的邻接矩阵,即 $\boldsymbol{M}_{\mathrm{Inf}}$ 的反向传播图便是梯度流 $\boldsymbol{M}_{\mathrm{Grad}}$.本文将 $\boldsymbol{M}_{\mathrm{Grad}}$ 追加到 $\boldsymbol{M}_{\mathrm{Inf}}$ 的末尾,构建了一个既包含前向特征流也包括反向梯度流的全训练图 $\boldsymbol{a}_{\mathrm{Tra}}$,以描述训练中的全部数据流.如图 4 所示,在 $\boldsymbol{a}_{\mathrm{Tra}}$ 中,$\boldsymbol{M}_{\mathrm{Inf}}$ 和 $\boldsymbol{M}_{\mathrm{Grad}}$ 处于对角位置.需要注意的是,特征流和梯度流之间存在许多相互依赖关系,这需要清楚的在 $\boldsymbol{a}_{\mathrm{Tra}}$ 中表示,以描述完整训练过程.接下来本文将展示全训练图的构建过程.

3.4.2　构建全训练图

由于特征图和梯度图之间存在相互依赖性,所

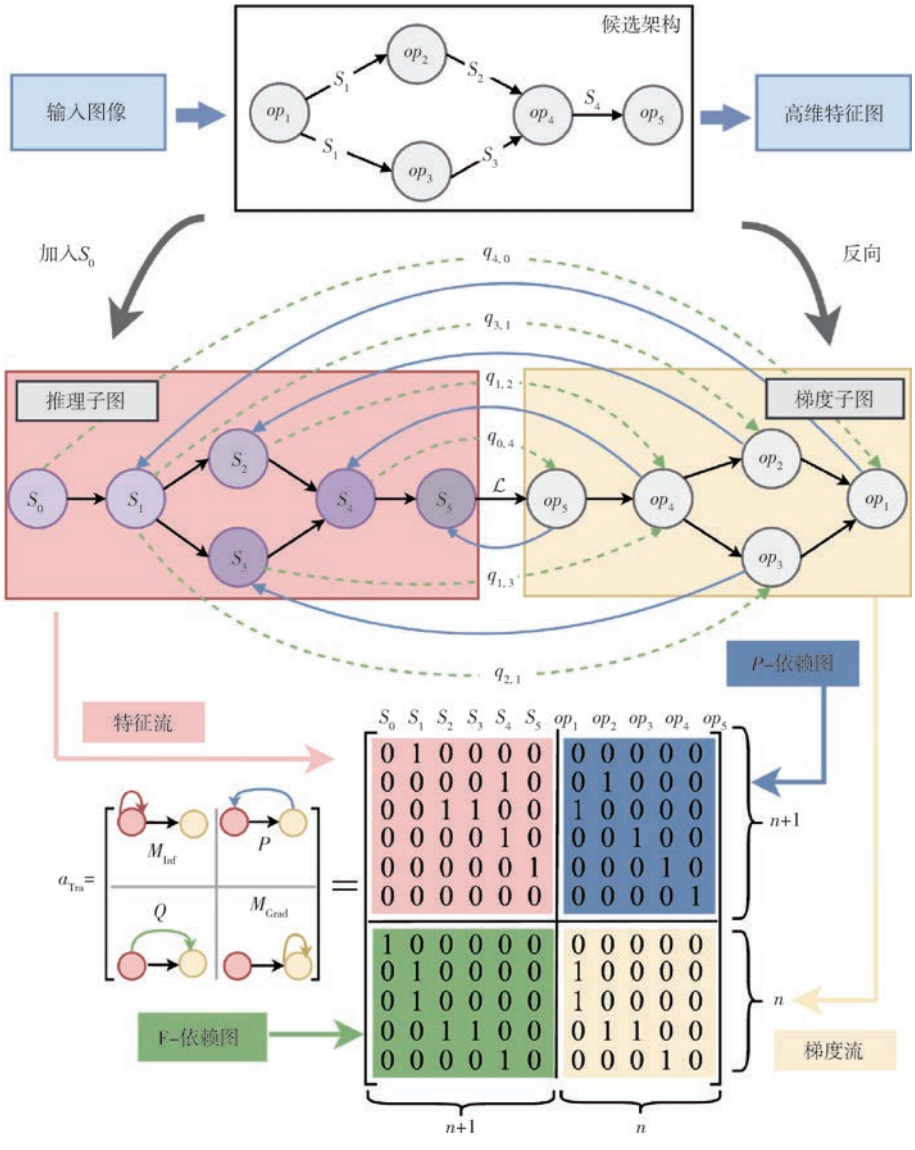

图 4 构建训练图

以除了这两张图之外，训练图中还应分别包含推理和梯度子图之间的两种依赖关系，即 p 依赖矩阵（参数依赖矩阵）和 f 依赖关系（特征依赖关系）. 在图 4 中，推理子图和梯度子图中的实线表示推理图 $\boldsymbol{M}_{\text{Inf}}$ 和梯度图 $\boldsymbol{M}_{\text{Grad}}$ 中的数据流动方向. 图中两个子图间的实线表示 p 依赖矩阵的操作对数据流的影响，而虚线表示 f 依赖关系. 设神经网络架构中共有 n 个操作节点，从 op_1 输入节点起，经过 n-2 个神经网络中的不同操作后最终输出节点为 op_n. 而数据流动的过程中，S_1 为输入图像信息 S_0 经过 op_1 节点后的输出结果，记作 $S_1 = op_1(S_0)$. 以此类推，最后一个输出特征信息 S_n 为数据流动过程中从 S_0 至 S_n 的不同路径的叠加和，记总路径为 p 条，如公式(11)所示：

$$S_n = \sum_{path=1}^{p} op_n\{op_k \cdots op_{k-1}[op_1(S_0)]\}$$
$$k \in path(op_1 \sim op_n) \tag{11}$$

在 p 依赖矩阵中，两个特征节点间的静态参数定义如公式(12)所示.

$$\frac{\partial S_n}{\partial S_{n-1}} = op_n \tag{12}$$

梯度流▽的参数更新操作如下所示：

$$\frac{\partial L}{op_1} = \sum_{path=1}^{p} \frac{\partial L}{S_n}\frac{\partial S_n}{\partial S_k}\frac{\partial S_k}{\partial S_{k-1}}\cdots\frac{\partial S_1}{\partial op_1}$$
$$= \sum_{path=1}^{p} \delta_l S_0 op_k op_{k-1}\cdots op_f \tag{13}$$

其中 op_f 是在路径 p 上最后一个连接到 op_1 的操作. 根据公式(13)可以得出结论，操作 op_i（参数更新）取决于两部分，即操作 $op_j(j > i)$ 和静态特征

$S_k(k<i)$. 在反向梯度子图中,op_i 只与其直接前驱相连,而其间接前驱则通过链式法则向它传递信息. 例如图 4,在梯度子图中,op_1 节点连接到了 op_2 和 op_3,并在 f 依赖矩阵中与静态特征 S_0 相关联.

对于 p 依赖矩阵 $\boldsymbol{P}\in R^{(n+1)\times n}$ 中的一个元素 $p_{i,j}$,其行索引 $i\in[0,n]$,列索引 $j\in[1,n]$. 静态参数 op_j 只有在 S_i 更新为新的特征 (S_j) 时才起作用,这可以用公式(14)表示:

$$\partial S_j=\begin{cases}op_j\times\partial S_i, & \text{when } i,j\in[1,n]\\ 0, & \text{when } i=0\end{cases}\tag{14}$$

因为 S_0 节点是人为添加的,实际并不参与任何特征生成过程,所以 $\boldsymbol{P}$ 矩阵的首行($i=0$ 时)数值恒定为零. 基于公式(14)中的推理关系,只有当 $i=j$ 时,$p_{i,j}$ 才被置为 1,见图 4.

对于 f 依赖矩阵 $\boldsymbol{Q}\in R^{n\times(n+1)}$ 中的一个元素 $q_{j,k}$,行索引 $j\in[1,n]$,列索引 $k\in[0,n]$. 基于链式法则关系,更新后的操作参数 (op'_k) 由静态特征 S_j 及原操作参数 (op_k) 共同决定,如公式(15)所示:

$$\begin{cases}op'_k=op_k+\nabla_k S_j\\ \nabla_k=\dfrac{\partial L}{\partial op_k}\end{cases}\tag{15}$$

如图 4 所示,基于公式(15)中的推理关系,只有当 $M_{\text{Inf}}(j,k)=1$ 时,$q_{j,k}$ 才被设置为 1. 由于矩阵 $\boldsymbol{P}$ 记录了 $S\rightarrow op$ 的依赖关系,所以它被放置在 $\boldsymbol{a}_{\text{Tra}}$ 的右上角. 矩阵 $\boldsymbol{Q}$ 则因为其相反的依赖关系方向被放置在左下角. 本文将训练 $\boldsymbol{a}_{\text{Tra}}$ 的整个过程视为神经网络训练中的消息传递过程,它是一个包含了"数据集信息"和"结构信息"的完整训练模型.

综上所述,假设神经网络图架构中有三个节点 V_i,V_j 和 V_k 形成链式依赖关系 $(V_i\rightarrow V_j\rightarrow V_k)$,那么(1)对于推理子图中的一个节点 (S_j),它的输出特征取决于 V_i 节点动态输入 (S_i) 和 V_j 节点的静态参数 (op_j),这可以在一张图中表示为由 op_j 链入到 S_j 的一条边,称为 p 依赖图;(2)对于梯度子图中的一个节点 (op_j),在训练过程中,链式法则定义了每层节点的参数更新规则,它取决于动态的梯度 (op_k) 参数和静态的特征 (S_j),这可以在另一张图中表示为从 S_j 到 op_k 的一条边,称为 f 依赖图,这两种图的生成过程如算法 1 所示.

算法 1.　建立全训练图 $\boldsymbol{a}_{\text{Tra}}$

输入:初始神经元 C;推理子图 M_{Inf};梯度子图 $\boldsymbol{M}_{\text{Grad}}$;

输出:p 依赖矩阵 $\boldsymbol{P}$ 和 q 依赖矩阵 $\boldsymbol{Q}$;

过程:

S1. FOR $i=0$ to n DO

S2.　FOR $j=1$ to n DO

S3.　　IF s_j 是 C 中 op_j 的外边 THEN

S4.　　$p_{i,j}\leftarrow 1$ //矩阵 $\boldsymbol{P}$ 中的元素

S5. FOR $j=1$ to n DO

S6.　FOR $k=0$ to n DO

S7.　　IF $\boldsymbol{M}_{\text{Inf}}(j,k)=1$THEN

S8.　　$\boldsymbol{q}_{j,k}\leftarrow 1$ //矩阵 $\boldsymbol{Q}$ 中的元素

3.4.3　基于 GCN 进行精度评估

在基于 GCN 进行精度预测阶段,架构数据 $\boldsymbol{H}_N$ 和数据集 Embedding 向量 $\boldsymbol{H}_D$ 被共同输入 GCN 网络. 然后在训练图 $\boldsymbol{a}_{\text{Tra}}$ 上传递特征流和梯度流信息. GCN 的传播公式如下:

$$\boldsymbol{H}_G^{(l+1)}=\sigma(\boldsymbol{a}_{\text{Tra}}(\boldsymbol{H}_N,a_{\text{Tra}})^{(l)}\boldsymbol{W}_{graph}^{(l)})\tag{16}$$

本文使用均方误差(MSE)损失函数来训练 GCN[4]. 如公式(17)所示,输出层将最终得到的节点 Embedding$\boldsymbol{H}_i^{(l)}$ 和图架构 $\boldsymbol{G}$ 二次编码为一个图编码 $\boldsymbol{H}_G$ 以补偿"结构缺失",本文使用了全局汇聚池化方法[19]来编码图 Embedding.

$$\boldsymbol{H}_G=ReLU\Big(\sum_{i=1}^{n}\boldsymbol{H}_i^{(l)}\Big)\tag{17}$$

其中 $\boldsymbol{H}_G$ 既包含了候选架构的结构信息,也包含了 NAS 任务的数据集信息. 本文将这个 Embedding 向量输入进一个两层的全连接神经网络,网络的输出是一个标量值 $\hat{y}$,其意义是在给定数据集 D 下架构的预测精度.

3.5　基于预测器的 NAS

本文所提出的预测器与其他常见的 NAS 预测器相比,在搜索空间和搜索策略方面采用相同的设计思路[3]. 因此本文可以在算法 2 中用 GCN 预测器替换传统 NAS 预测. NAS 首先从搜索空间中采样搜索架构,然后使用 GCN 预测器来预测性能精度. 最后,预测器将采用"赢家通吃"的方法[22]来决定性能最高的架构. 在 NAS 的监督学习期间,预测器的泛化能力主要取决于训练样本的数量 N. 而架构向量能否充分描述整个架构信息则取决于网络节点信息是否充足以及结构信息的残余量. 就超参数设置而言,节点的特征维度 k 和训练的 Batch 大小也会影响图编码器. 将在 4.4 节中对上述参数进行详细分析.

算法 2.　将预测器应用到 NAS

输入:A:架构搜索空间;P:NAS 性能预测器;D:图像数据集;N:样本架构数量

输出:具有最高性能 y^* 的架构 $a_i\in A$

S1. 从 A 中随机采样架构集 $N=\{a_0,\cdots a_N\}$;

S2. 使用 P 来预测 N 中架构的准确性;

FOR $i=1$ to n DO

建立 z_i 的全训练图 $\boldsymbol{a}_{\mathrm{Tra}}$；

通过 GCN 预测器获得 $\bar{y}_i$.

S3. 用相应的数据集 D 评估 Top-K 架构$\{z_0,\cdots z_k\}$，得到 z 的真实性能数据集 $Y=\{y_0,\cdots y_k\}$；

S4. 赢家通吃[22].

4 实验验证

4.1 实验设置

4.1.1 NAS 数据集

NAS-bench-101 和 NAS-bench-102 包含了在多种图像数据集（如 CIFAR-10、CIFAR-100、ImageNet 等）上预先训练好的神经架构搜索模型的测试精度数据. 因此研究人员可以基于 NAS-bench-201 中已有的测试精度结果来评估他们提出的新模型的性能，而不需要自己训练大量模型. 这可以更快更方便地对新提出的 NAS 方法进行评测. 因此本文采用了 NAS-bench-101 和 NAS-bench-102 两个数据集作为基准，如图 5 所示.

NAS-bench-101 是一个 NAS 的基准数据集，它包含了超过 423624 个神经网络架构及其在 CIFAR-10 图像分类任务上的性能数据. 该数据集旨在为 NAS 研究提供一个标准化的基准测试，以便更系统地比较不同的 NAS 方法. 如图 5(a)所示，NAS-bench-101[23]搜索空间中的神经元节点操作模式包括 5 种，即输入、最大池化、3×3 卷积、1×1 卷积和输出. 数据集中的一个架构最多包含 7 个节点和 9 条边. 该数据集采用的是 OON(Operation On Nodes 在节点上操作)模式，即“以节点为操作”.

NAS-bench-201 是另一个常用的 NAS 基准数据集，用于评估不同的 NAS 方法在图像分类任务上的效果. 如图 5(b)所示，NAS-bench-201[24]的节点操作有 7 种，即输入、3×3 最大池化、3×3 卷积、1×1 卷积、跳过连接、置零和输出. 每个架构固定由 4 个神经元节点和 6 条边组成. 由于 NAS-bench-201 数据集中采取的是 OOE(Operation On Edges，在边上操作)格式，即“以边为操作”，而 GCN 处理数据的格式为 OON，所以在将数据输入 GCN 之前，应先将 NAS-bench-201 中每个图的格式转换为 OON 格式[25]，以便 GCN 处理，如图 5(c)所示，转换后的图中共有 8 个节点和 10 条边. 在格式转换后，它共有 15625 个候选神经网络架构. 本文在 CIFAR-10、CIFAR-100 和 ImageNet16-120 三个数据集上进行了预训练.

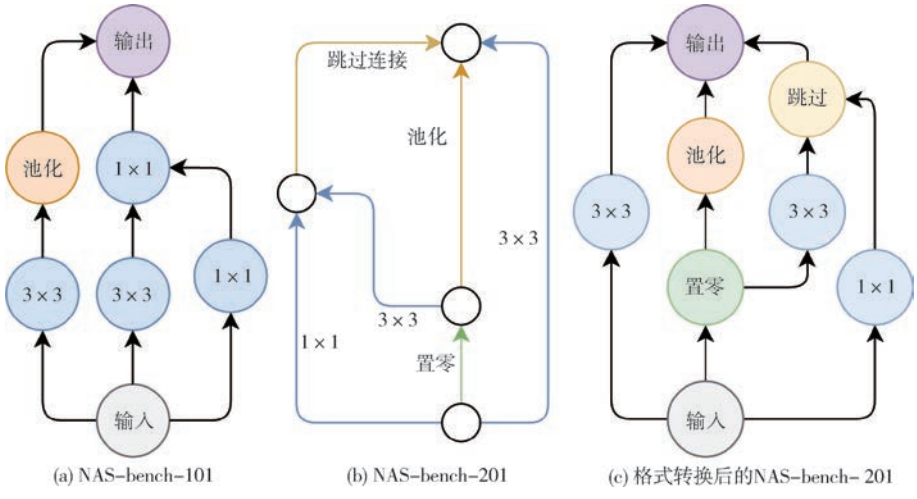

图 5 NAS-bench-101 和 NAS-bench-201 基础结构图

NAS-bench-301 数据集[26]包含了 10^{21} 个架构在 CIFAR-10 数据集上进行训练的性能. 其搜索空间与 DARTS 模型相同，是一个具有 7 个顶点的 DAG，并采取了 OOE 格式，因此在本实验中同样被转为 OON 格式. 该数据集的主要操作类型有 9 种，除了输入和输出外，还包括平均池化、最大池化、跳过连接、3×3 分离卷积、5×5 分离卷积、3×3 空洞卷积与 5×5 空洞卷积.

4.1.2 图像数据集

CIFAR-10 数据集共包含 10 个类别，每个类别有 6000 张图像，总共有 60000 张 32×32 的 RGB 图像. 这 60000 张图像被分为 50000 张训练样本和 10000 张测试样本. NAS-bench-101 的结果是在 CIFAR-10 数据集上训练得到的.

CIFAR-100 数据集包含 100 个类别,每个类别有 600 张 32×32 的彩色图像,共计 60000 张图像.在每个类别的 600 张图像中,有 500 张是训练集,100 张是测试集.每张图像都有一个标签:清晰标签或粗糙标签.

SVHN 数据集是从谷歌街景图像中提取的数字图片数据集,用于数字识别和定位任务,数据集规模适中,场景复杂,是数字识别领域常用的公开数据集.每张图片中都包含一组"0-9"的数字,数字 1-9 对应的标签为 1-9,"0"对应的标签为 10.训练集包含 73257 张图像,测试集包含 26032 张图像.

Aircraft 数据集共有三十种分类,其中共 10000 张图片,其中三分之二用作训练集.

4.1.3　对照组设置

为了确保实验的有效性.本文选取了多个常用的 NAS 方法作为对照实验组.在 NAS 预测器的相关研究中,Peehole[4] 使用 MLP 层来对架构进行编码.Neural Predictor[27] 仅使用 GCN 层作为图编码器预测 NAS 性能.NAO[5] 使用单路径超网络和统一采样来优化一次性超网络.TNASP[3] 在图编码器中使用了 Transformer 的 Multi-Head 自注意力机制,然后接 MLP 层作为预测模块,输出预测准确率.PINAT[11] 在 TNASP 模型的基础上增加了转换不变性的模块.GMAE[17] 使用图掩码自动编码器学习神经网络图架构.

ENAS[8] 使用 RNN 控制器来确定每个节点的操作类型以及在候选架构中选择哪些边,其子网络的初始化权重由 HyperNet 生成的.EPE-NAS[28] 可以对未训练的网络评分,并且评分结果与其训练后的性能相关联.STEN[29] 在架构上应用了一个可微分采样器,这是一种基于梯度反向传递的搜索方法.PCDARTS[30] 是一种基于梯度的 NAS,通过部分通道连接改进了传统的 DARTS[31].

NAS-bench-201 包含了在多种图像数据集(如 CIFAR-10、CIFAR-100、ImageNet 等)上预先训练好的 NAS 模型的测试精度数据,所以被视为图编码器的基准数据集.本文以 NAS-bench-201 为基准进行实验.

用作图编码器的图卷积网络的层数设置为 3 层,隐藏层尺寸为 144.最后一层的输出层是 2 层全连接层,隐藏维度为[144,256].使用 Adam 优化器[32] 可以最小化 MSE 误差,学习率为 1×10^{-4}.与 TNASP[3] 相同,本模型训练时 Batch 大小为 10,均训练 300 个 epoch,使用 NVIDIA-TITAN-XPGPU 和 Pytorch 1.11.0 框架进行训练.

4.1.4　超参数设置

超参数的值如表 2 所示,本文用于 NAS-Bench-201 训练集和测试集的超参数设置与论文[24]相同.

表 2　实验的超参数设置

超参数	数值
节点特征 k	64
Transformer 输出维度	64
Transformer 的 Head 数量	8
图编码器 GCN 的输出维度	144
Dropout	0.1
数据集编码器输出维度	64
每个类的输入图片数量	20

4.2　图编码器的性能

在实验中,本文使用肯德尔相关性系数来衡量预测值 $\hat{y}_i$ 和真实值 y_i 之间的相关性,此系数被广泛应用于 NAS 预测器的性能预测实验中[33,4].肯德尔相关性系数 τ 的值在 -1 和 1 之间.$\tau=1$ 表示两个随机变量之间的正相关性最强,$\tau=-1$ 表示两随机变量的负相关性最强.如果 $\tau=0$,则两个变量是相互独立的[34-35].

不同编码器在 NAS-bench-101、NAS-bench-201 和 NAS-bench-301 数据集上的测试结果如表 3 所示.由于部分实验的对照组模型未公开源码,且并未在所有 NAS-bench 数据集上进行实验验证,因此本文选取了在各数据集上的 SOTA 模型进行了对照试验.与其他方法相比,本文提出的预测器在给定样本数量较少时,τ 值增加的较多,例如,在 NAS-bench-101 测试中,本文的方法仅用了 100 个样本就取得了与 Neural Predictor 使用 424 个样本时[27] 相似的性能,即本文采取的方法仅需四分之一的数据量即可与 Neural Predictor 达到相同的预测精度.而在 NAS-bench-201 测试中,这一比例进一步降低到十分之一.从 NAS-Bench-301 数据集中随机抽取部分数据进行训练与测试后,本方法依旧可以取得较高的预测准确性.实验结果表明,我们的方法通过丰富节点 Embedding 信息和补偿自注意力机制成功突出了图架构信息,本文方法的学习率相比于 Neural Predictor[27] 提高了 1.91 倍,相比 TNASP[3] 提高了 1.55 倍.本文方法的预测相关性相比当前最优结果提高了 7.45%.

同时,从图 6 中针对 NAS-bench-201 数据集的统计结果可以看出,在模型复杂度方面,本文中图编码器模型的参数量相较于目前最新的 Transformer

模型降低 0.07M，图卷积网络模块相较于置换不变性模块的计算量[36]可减少 38.0%. 本模型可突破其他方法所形成的参数量-精度帕累托边界. 总的来说，本文所述方法能够进一步提升模型的计算效率.

表 3 在 NAS-bench-101、NAS-bench-201 和 NAS-bench-301 数据集上不同编码器的肯德尔系数

NAS-Bench-101	100(0.02%)	172(0.04%)	424(0.1%)	4236(1%)
MLP[4]	0.284	0.356	0.373	0.415
Neu Predictor[27]	0.480	0.615	0.624	0.732
NAO[5]	0.501	0.493	0.704	0.775
D-VAE[5]	0.530	0.549	0.626	0.698
TNASP*[3]	0.549	0.597	0.683	0.757
Our*(MLP)	0.536	0.603	0.678	0.733
Our*(GCN)	**0.635** ↑	**0.686** ↑	**0.728** ↑	**0.787** ↑
NAS-Bench-201	78(0.05%)	156(0.1%)	469(3%)	1563(10%)
MLP[4]	0.446	0.450	0.509	0.520
NAO[5]	0.467	0.493	0.470	0.526
NeuPredictor[27]	0.343	0.413	0.584	0.646
TNASP*[3]	0.532	0.589	0.706	0.787
PINAT*[11]	0.571	0.642	0.707	0.792
Our*(MLP)	0.557	0.602	0.725	0.788
Our*(GCN)	**0.623** ↑	**0.647** ↑	**0.730** ↑	**0.799** ↑
NAS-Bench-301	100(0.1%)	500(0.5%)	1000(1%)	10000(10%)
MLP[4]	0.346	0.538	0.561	0.586
NeuPredictor[27]	0.520	0.621	0.656	0.675
GMAE[17]	0.599	0.653	0.657	0.681
TNASP*[3]	0.521	0.736	0.786	0.877
Our*(GCN)	**0.655** ↑	**0.801** ↑	**0.849** ↑	**0.887** ↑

*：使用 Transformer 编码.

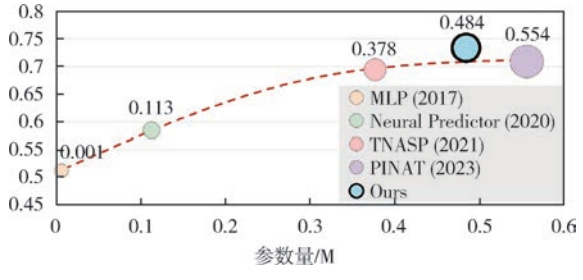

图 6 基于 NAS-bench-201 数据集测试的模型参数量与肯德尔相关系数的关系
圆点面积代表模型参数量.

4.3 图像数据集的可迁移性

为了测试模型的可迁移性，本文先使用较小的数据集(cifar-10)对预测器进行预训练，然后在更大的数据集(cifar-100、svhn、aircraft)上测试预训练得到的模型. 同时，在模型中加入了图像数据集编码部分[37]. 表 4 的结果表明，在 cifar-100 数据集上，本文预测器的预测精度(73.51%)明显优于其他四种方法，表明本文提出的预测器对架构有着高度泛化的记忆能力. 此外，在这些数据集上，本文方法的搜索时间也有大幅降低，例如，我们的方法在 cifar-100 上只需要 16 秒就可以找到最佳架构，与在 cifar-10 上的时间相同. 此外，在该评估部分进行了引入反向梯度流的训练图的消融实验. 增加该部分后，预测模型的肯德尔系数由 0.764 增加至 0.801，由此可以说明该训练图在学习架构信息与相关图像数据集信息融合时的有效性.

表 4 在 NAS-bench-201 上不同 NAS 方法在多个数据集上的测试精度

目标数据集	NAS 方法	参数(M)	搜索时间(s)	测试准确度(%)
cifar-10	ENAS[8]	4.6	13315	54.3±0.00
	EPE-NAS[28]	—	206.2	91.31±1.69
	STEN[29]	—	30200	87.64±0.00
	PC-DARTS[30]	1.17	10395	93.66±0.17
	Ours	4.3	16	**94.36±0.01** ↑
cifar-100	ENAS[8]	4.6	13315	15.61±0.00
	EPE-NAS[28]	—	206.2	69.58±0.83
	STEN[29]	—	58808	59.09±0.24
	PC-DARTS[30]	0.26	19951	66.64±2.34
	Ours	4.3	16	**73.51±0.00** ↑
svhn	STEN[29]	0.48	85189	96.02±0.12
	PC-DARTS[30]	0.47	31124	95.40±0.67
	Ours	5.1	110	**96.12±0.50** ↑
aricraft	STEN[29]	0.44	18564	44.84±3.96
	PC-DARTS[30]	0.32	3524	26.33±3.40
	Ours	5.1	120	**50.51±2.44** ↑

4.4 消融实验

4.4.1 基于 GCN 和 MLP 的预测器比较

从表 3 的数据可以看出，基于 MLP 的预测器(w/o GCN)会受到结构缺失影响，其预测精度普遍低于基于 GCN 的预测器，在 NAS-bench-101 和 NAS-bench-201 上分别低了 11.22%和 3.76%. 本文在图 7 中分别说明了 GCN 和 MLP 的预测值与真实精度之间的相关性. 图 7(a)中的 GCN 版本预测器，大多数预测点都遵循了与真实值相同的分布. 然而，对于图 7(b)中的 MLP 预测器，性能较差架构(预测精度<10%)的预测数据与真实值的分布发生显著偏离，这意味着 MLP 预测器对架构信息存在误判.

4.4.2 相关性系数

为了证明上述 MLP 预测器对架构信息存在误判，实验需测量编码后的结构信息残余量. 在本文

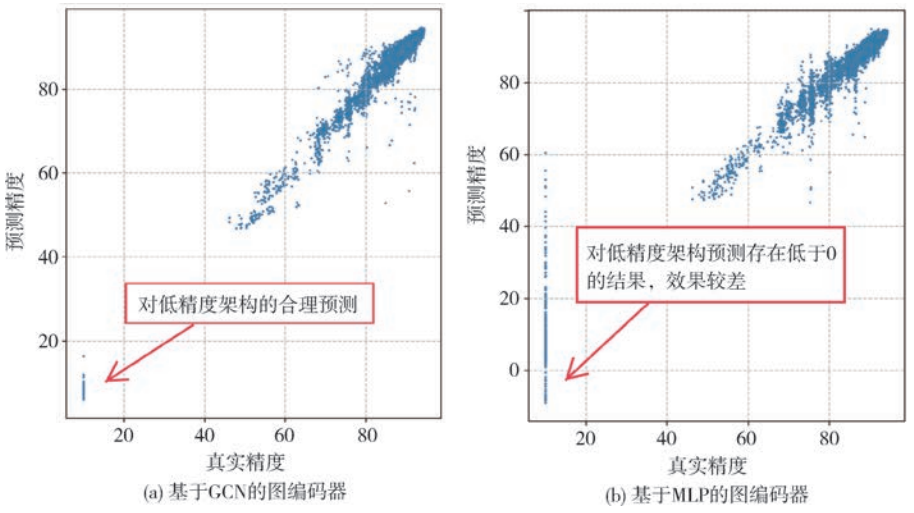

图 7　基于 GCN 的图编码器与基于 MLP 的图编码器的预测结果与真实结果的对比

中,我们采用两种测量系数,即余弦相似度和 RV 系数,以量化架构信息和邻接矩阵之间的相关性.余弦相似度一般被用于测量架构局部区域的一致性,它通过测量两个向量之间的夹角度数来确定它们的相似度,如公式(18)所示.当 x 和 y 指向方向较为一致时,余弦相似度值接近 1;反之则余弦相似度接近 −1.本文测量的是邻接矩阵 $\boldsymbol{C}$ 的行向量与图 Embedding$\boldsymbol{H}_{\mathrm{N}}$ 的余弦相似度:

$$\cos(\theta)=\frac{x\cdot y}{\|x\|\times\|y\|}=\frac{\sum_{i=1}^{k}(x_i\cdot y_i)}{\sqrt{\sum_{i=1}^{k}x_i^2}\times\sqrt{\sum_{i=1}^{k}y_i^2}} \tag{18}$$

而 RV 系数更适合被用于测量全局架构.RV 相关系数是皮尔森相关系数的多元广义化,它的取值范围是从 0 到 1.1 表示两组数据间相关性最高,而 0 表示完全独立.表 5 中的数据是在每个样本的条件下测量五组,取其平均值的结果.两个指标皆表明:GCN 版本预测器中存在更多的结构信息[38,39],具体来说,局部和全局区域的架构信息残余分别增加了 14.32%和 2.38%.

表 5　添加 GCN 层后的余弦相似性和 RV 系数比较

测试	模型	样本			
		78	156	781	1563
余弦	MLP	0.297	0.318	0.333	0.295
	GCN	**0.356** ↑	**0.350** ↑	**0.354** ↑	**0.361** ↑
RV	MLP	0.421	0.444	0.435	0.424
	GCN	**0.442** ↑	**0.445** ↑	**0.441** ↑	**0.437** ↑

为了测试本文提出超维嵌入的表达能力,本文在 NAS-bench-101 和 NAS-bench-201 的数据集上进行了验证.图 8 的结果表明,超维嵌入可以提升预测器的性能,但是在 NAS-bench-201 上提升的效果较小.此现象表明,如果图中神经元节点的细节特征较少,则不适合使用超维嵌入方法编码,如 NAS-bench-201 中的置零和跳过操作本身缺少细节特征,因此无法对其进行细节信息扩展,针对此类操作,本文采用全 0 编码.然而,对于细节特征较丰富的操作,如 NAS-bench-101 中的卷积和池化操作,超维嵌入表示可以更好记录节点的细部特征.

4.4.3　特征维度和 Batch 大小

本文特征维度的选择采用的是启发式解决方案,在图 9(a)中,特征维度分别选取 16、32、64、128 和 256.数据的 Batch 大小为 1、10、32、64 和 128.分别测试了三种训练样本数占比情况(0.05%,1%,5%).当特征维度为 64 时预测器整体性能达到最佳值,之后即使继续扩大参数规模,预测器的性能也不会有明显提升.因此本文设置特征维度为 64.

随着本文在实验中将数据的 Batch 大小从 1 增大到 128,预测器的性能随之下降,如图 9(b)所示.这个结果为本文的设计提供了一个设计原则:由于图结构的自由度远大于普通图像,不利于模型收敛,因此较大的 Batch 不适合学习架构表示.因此本文统一将 Batch 大小设置为 10.

4.4.4　典型网络结构自动生成

图 10 是本文从 NAS-bench-201 中找到的典型架构,其中 3×3,1×1 均指卷积操作的规模.值得一提的是,如图 10(a)所示,本文提出的方法在没有先验知识的情况下,从 NAS-bench-201 数据集中成功找到了 Resnet 残差结构[40],充分说明本文提出的结构增强方法具备全局图评估的能力,并能够通过

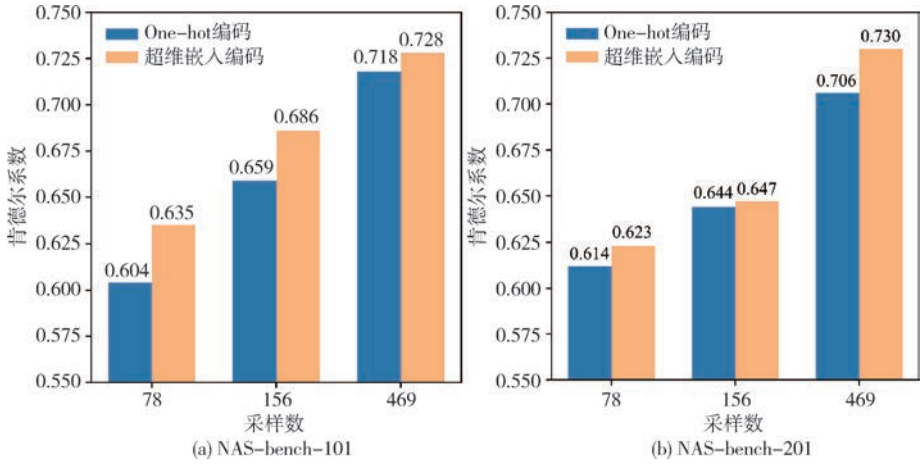

图 8 One-hot 编码与超维嵌入的效果对比

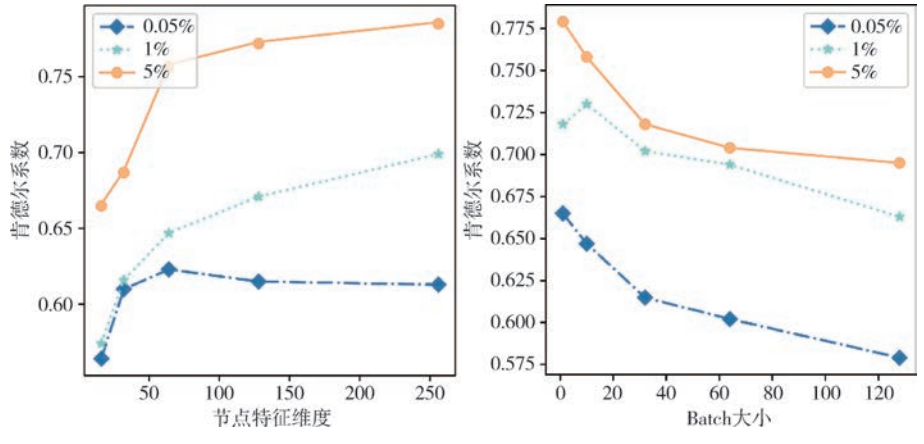

图 9 特征维度和 Batch 大小对 NAS-bench-201 结果的影响

引入反向梯度流图，发现并指示出正确的架构搜索方向，印证了本文提出方法的有效性.

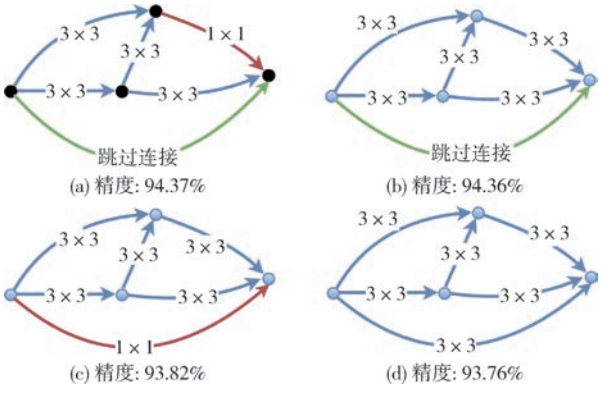

图 10 自动生成的典型网络架构

5 结 论

本文提出了一个基于 Transformer 结构增强的 NAS 性能预测器. 为了丰富节点操作细节特征，本文在预处理阶段提出了超维嵌入方法，为了弥补结构缺失问题，增强架构信息，在编码阶段将 Transformer 编码后的信息与结构信息共同输入一个图卷积网络. 此外，为了引入反向梯度流信息，本文构建了全训练图并将其馈入到 GCN 中进行精度预测. 结果表明，本文的预测器能在训练中获得更丰富的架构信息，从而使预测值更接近真实值 NAS 性能.

参 考 文 献

[1] Dai Quoc Nguyen, TuDinh Nguyen, Dinh Phung. Universal graph transformer self-attention networks// Companion Proceedings of the Web Conference 2022. New York, USA, 2022, 193-196

[2] Ying C, Cai T, Luo S, et al. Do Transformers Really Perform Bad for Graph Representation? //Proceedings of the Advances in Neural Information Processing Systems, 34 (NeurIPS). LA, USA, 2021: 28877-28888

[3] Lu S, Li J, Tan J, et al. TNASP: A transformer-based NAS predictor with a self-evolution framework// Proceedings of the