Multiobjective Evolutionary Design of Deep Convolutional Neural Networks for Image Classification

Zhichao Lu, Student Member, IEEE, Ian Whalen, Yashesh Dhebar, Kalyanmoy Deb, Fellow, IEEE, Erik D. Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti, Member, IEEE

摘要

目前，卷积神经网络 CNN 已经作为深度学习的骨干网络架构经典范式来处理各种图像任务。起初，先进的 CNN 网络架构都是由人工智能专家精心设计的，需要大量的先验知识来设计出杰出的网络架构，使得神经网络架构设计十分繁琐与困难。现在，人们提出了神经网络架构搜索 NAS 任务，通过计算机自动设计神经网络架构，针对各类任务产生特有神经网络架构。目前存在的用于图像分类任务的神经网络架构搜索方法，存在无法适用于算力有限的环境中。其中原因有两个：1）当前方法产生的神经网络架构通常仅仅考虑单一地优化其分类性能，或者只产生在某一特定部署环境下的神经网络架构；2）在大部分方法下，神经网络架构搜索过程需要巨大的算力资源。为了解决这些问题，论文提出了使用多目标演化计算方法 NSGA-II 算法来完成搜索任务，同时将分类性能和模型复杂度作为优化目标。首先，通过 NSGA-II 算法可以通过 Pareto Front 来得出针对不同网络复杂度的性能优秀的架构，以适应不同的部署环境。其次，本文方法提出了在神经网络架构搜索过程合理缩小网络，以及在搜索后期使用贝叶斯网络优化方法对之前搜索出成功的架构进行结构共享得到新的架构，两种方法来提高计算效率。原文方法在以下数据集进行测试：CIFAR, ImageNet 和 human chest X-ray，并于已有的人工设计和其他神经网络架构搜索方法得到的神经网络架构效果进行比较。受资源限制，复现任务仅在 CIFAR-10 进行测试。相比于其他方法，本文方法的特点是能同时得到多个不同复杂度的神经网络架构供不同的部署环境选择。

关键词：卷积神经网络（CNN）；演化深度学习；遗传算法（GA）；神经网络架构搜索（NAS）

1 引言

请在此部分对选题背景，选题依据以及选题意义进行描述。

当前，在计算机视觉相关任务中，如分类，检测和语义分割任务，深度卷积神经网络已经获得了绝对的成功，提出了很多成功的架构，例如，图像特征提取中常用的 GoogLeNet[1], ResNet[2], DenseNet[3] 等，也有轻量化模型架构 ShuffleNet[4]，MobileNet[5] 和 LBCNN[6] 等。这些成果都是专家艰辛努力设计的模型架构。

神经网络架构搜索（NAS）提供了一种自动得到神经网络架构的方法，可以减少人们设计 CNN 架构的工作量。通常通过算法替换神经网络中的一些部件产生新的 CNN 架构，并在对应的任务中测试新架构的性能效果。NAS 已经在学术和工业界得到了广泛的关注，开始了大量的研究与应用，为各种任务定制特有的神经网络架构。

早期的 NAS 方法主要依靠强化学习（RL）方法来产生高性能的模型架构，但是强化学习方法最大的限制因素是需要极大的计算支援，常常需要上百个 GPU 跑上几周的时间才能实现。最近还有一

种基于松弛（relaxation）的方法，将不同层之间的连接关系通过实数变量和权重的方式进行近似表示，并使用梯度下降的方法进行优化，但是这类方法需要占用大量的显存，导致搜索空间受限。

在实际应用中，我们不仅关注预测模型的准确度，也要关注模型运行的效率，如部署在移动设备上低功耗和低延时。为此，在 NAS 任务中，人们也开始对模型的复杂度进行考虑。这就意味着 NAS 任务需要同时最大化预测性能和最小化网络复杂度，因此，需要进行多目标优化。强化学习和基于松弛的方法目前还没能够应用于多目标的任务。

演化算法受益于其基于种群的性质和编码的灵活性，在 NAS 任务中也受到了广泛的关注。演化算法通过迭代繁衍的方式，通过适者生存逐渐获得更优秀的个体组成的种群。虽然通过演化算法实现多目标优化已经是很成熟的工作了，但是大部分使用演化算法的 NAS 工作依然使用单目标优化方法。

本论文提出了基于多目标演化算法 NSGA-II[7] 实现 NAS 工作，称为 NSGANetV1，解决了前面提到的 NAS 遇到的问题。其贡献总结如下：

1) NSGANetV1 每一代族群迭代，都使得族群的分布更接近 Pareto front，再通过这些最优秀的个体融合和修改结构产生新的架构，从而得到下一代族群。另外，NSGANetV1 通过合理的缩小架构和通过贝叶斯网络共享过去得到的优秀架构来预估产生优秀模型的参数，两种方法来提高整个搜索过程的效率。

2) 通过 NSGANetV1 方法得到一系列不同复杂度的优秀性能架构，可以让设计者在完成 NAS 任务后选择符合部署环境的模型，而不是在执行 NAS 任务前需要根据部署环境去设置参数。另外，一系列的优秀架构可以揭示更多有价值的架构设计信息和原则，供设计者参考。

2 相关工作

当前，在多种应用和算力环境下，NAS 逐渐应用于特定任务的神经网络自动生成任务中。在早期的方法[8],[9]中，同时对模型的架构和权重参数进行优化，这些方法展现出优于人工设计的浅层全连接层的性能。接下来，我们来介绍下处理图像分类问题的 CNN 相关的 NAS 方法。

演化计算 NAS 方法：当前基于演化计算的 NAS 方法，一般都是仅对神经网络的拓扑结构进行演化计算，而将学习的参数留给梯度下降的方式解决。其中，Xie 和 Yuille 提出的 Generic CNN[10] 是最早提出的基于演化计算的方法之一。之后[11]提出一个大规模应用的基于简单演化算法的 NAS 方法。[12]对该方法进行了拓展，提出了 AmoebaNet 方法，这是首次在大规模应用中对比了简单演化算法（EA）和强化学习（RL）在 NAS 上的性能。他们的算法揭示了 EA 方法比 RL 或随机搜索收敛得要快。同时期，Liu 等人提出一种分层表示的算法[13]，其允许非模块化的层结构出现。尽管在不同的数据集上有着不错的提升，但是这些 EA 方法都非常计算低效，例如，在 AmoebaNet 方法中，一次运行就需要在 450 个 GPU 上跑一个星期才能完成。

同时，用于已预算好的 NAS 任务，另一种流水线型 EA 方法出现。Suganuma 等人[14]提出使用 Cartesian 遗传算法来从已有的架构模组中（如残差网络模块）组装网络架构。Sun 等人[15]提出使用随机森林实现通过离线代理模型预测架构的性能，通过梯度下降的方式消除部分低级优化。报告显示，他们于之前的方法相比剩下了三倍的时间。然而，这些节省算力的方法仅仅在小数据集上被展示出来，如 CIFAR-10 和 CIFAR-100，而在一般数据集上远比不上现在 state-of-the-art 的分类方法。

多目标 NAS 任务：Kim 等人[16]提出了 NEMO，其中一个最早使用对目标演化算法产生 CNN 架构的。他用了 NSGA-II[7]方法来最优化生成架构的分类性能和模型推理时间，其搜索空间包括每一层的输出通道数和每一层在给定的 7 种模块中选一种。DPP-Net[17]采用逐渐将模型从简单模型扩展起来，且仅仅留下在代理模型中最优的 k 个进行迭代。Elsken 等人[18]提出了 LEMONADE 方法，通过定制设计的近似网络态射降低了计算要求，这允许新生成的网络与其先行者共享参数，省去训练新模型的消耗。然而，LEMONADE 方法依然需要近 100 GPU-days 才能找到在 CIFAR 数据集上不错的架构。

搜索效率：当前 NAS 的瓶颈主要是评估模型时候需要的训练模型低级优化问题。每一次评估模型都需要消耗几小时的时间才能得到评估结果。为了提高实际在有限预算下的可行性，NAS 方法常使用一些不成熟的低级优化代替测量方法。一个常用的方法是减少架构的深度（层数）和宽度（通道数）产生一个小的代理模型来代替原有模型进行训练评估。然而，通常代理模型都是通过简单的启发来得到，导致其与实际结构性能相关性并不高。例如，NASNet[19]需要另外对 250 个架构进行 300 个轮次的训练（在单个 GPU 需要超过一年的时间）。与之相似的有 AmoebaNet[12]整个搜索过程需要对 27K 个模型进行评估。

在我们的工作中，我们同时聚焦到效率和代理模型的可靠性。通过一系列的系统性学习，我们权衡了代理模型与真模型的性能相关度与预计加速时间。我们提供了合理的设置在指定的搜索空间和数据集。

3 本文方法

在实际的 NAS 任务应用中，我们只需要考虑最大化模型的性能，这样单一的目标，而在多目标方法中，我们需要找一个与之相矛盾的目标来代表不同部署环境的情况。在这个工作中，我们在部署环境的复杂度多样性和设计高性能架构两个目标之间做双目标优化任务。其数学表达如下：

$$\begin{aligned} &\text{minimize} \boldsymbol{F}(\boldsymbol{x}) = \left(f_1\left(\boldsymbol{x}; \boldsymbol{w}^*(\boldsymbol{x})\right), f_2(\boldsymbol{x})\right)^T \\ &\text{subject to } \boldsymbol{w}^*(\boldsymbol{x}) \in \operatorname{argmin} \mathcal{L}(\boldsymbol{w}; \boldsymbol{x}) \\ &\boldsymbol{x} \in \Omega_x, \boldsymbol{w} \in \Omega_w \end{aligned} \tag{1}$$

其中，Ω_x 表示架构模块的搜索空间，Ω_w 表示架构中的权重参数。$\boldsymbol{F}$ 是由验证集中的分类错误率 f_1 和网络架构复杂度 f_2 所构成高级别优化目标向量。低级别优化目标 $\mathcal{L}(\boldsymbol{w}; \boldsymbol{x})$ 表示在训练集中的交叉熵损失函数。

3.1 本文方法概述

本文提出了 NSGANetV1 算法，通过迭代进化过程，使得架构种群越来越优秀。在每一个迭代中，用根据族群中已有的亲代架构，通过基因重组和基因突变的方式，生成新的一组后代架构。然后，所有的亲代和后代中进行性能评估，通过优胜劣汰的原则保留最优秀的个体架构作为下一代族群中的架构。初始化过程可以选择纯随机生成，也可以加入一些已有的优秀架构放入种群中。总体方法概括图，如图 1所示。

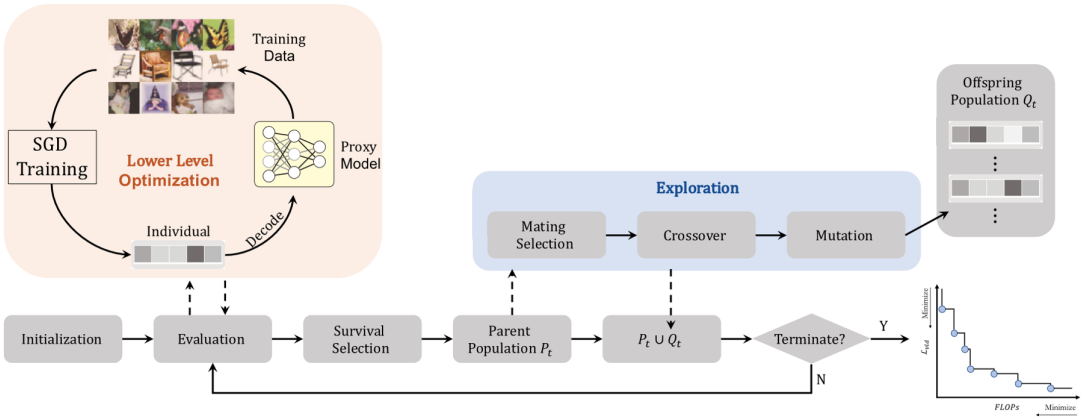

图 1: 方法示意图

3.2 搜索空间与架构表示编码

对优化神经网络架构的搜索可以基于很多种不同的搜索空间，而且对于所选空间具有的普遍性对可能的结果会有重大的影响。大部分的演化计算 NAS 方法[10],[15]，仅仅搜索单一的搜索空间，例如只搜索模块连接方式，或是超参数。与之不同的是，NSGANetV1 同时搜索运算模块和连接关系，使得搜索空间更加综合。

最新的 CNN 架构通常包括外部架构（网络级别）设计的设定，包括宽度（例如，通道数）、深度（例如，层数）和分辨率变化（例如，池化层操作），以及内部结构（模块级别）设计的设定，包括层间连接关系和运算设定，如 Inception Block[1]、ResNet Block[2]等。正如在 CNN 中所看到的那样，网络级别的决策大多是基于先验知识和手头任务的元启发式来手工调整的，在本工作中也是如此。而模块级别的决策通过演化计算方法实现。

一个模块（block）是指一个小的卷积模块，通过多次重复叠加形成一个完整的神经网络。为了实现分辨率的变化，本文将模块分为两种，一种是普通模块（Normal Block），另一种是下采样模块（Reduction Block）。其中，普通模块（Normal Block）的输入和输出的空间分辨率保持相同；下采样模块（Reduction Block）的输出的空间分辨率是输入的一半，通过一个步距为 2 的操作完成。如图 2所示。

我们使用同样的由 5 个节点（node）组成的无环有向图来表示两种模块（Reduction Block 使用步距为 2）。每个节点包括两个分支结构，映射到两个输入和一个输出。在模块 i 中的每个节点中，我们需要从 i-1 和 i-2 层的输出以及当前模块 i 内部在当前节点前面的节点输出中选择两个输出作为本节点的输入，如图 2(C) 所示。而运算块由以下 12 中常见的 CNN 模块进行选择：

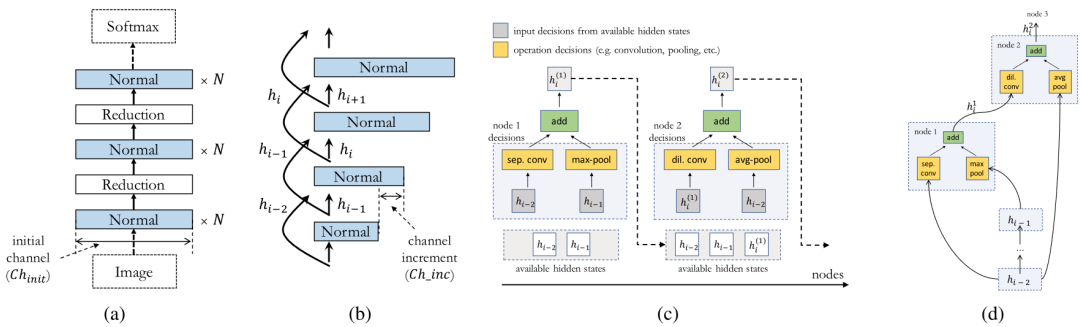

图 2: NSGANetV1 设计方式。(a) 模块堆叠形成架构。(b) 架构通道数随层数逐渐增加。(c) 每个模块（block）由 5 个节点（node）组成，每个节点由 2 个分支计算组成。(d) 模块结构可视化。

1) identity;
2) 3×3 max pooling;
3) 3×3 average pooling;
4) squeeze-and-excitation[20];
5) 3×3 local binary convolution[21];
6) 5×5 local binary convolution;
7) 3×3 dilated convolution;
8) 5×5 dilated convolution;
9) 3×3 depthwise-separable convolution;
10) 5×5 depthwise-separable convolution;
11 7×7 depthwise-separable convolution;
12) 1×7 then 7×1 convolution.

两个分支的结果通过相加产生新的隐藏层输出，提供给同一模块中后续的节点作为输入的选择。特别的有每一层的通道数不再是使用传统的分段变化，即每经过一个下采样层就倍乘通道数这种快速变化层数的方式，而是，通过每一层逐渐增加固定的通道数，慢慢增加通道数。如图 2(b)-(d) 的描述。

根据上面的描述，当我们在一个模块中有 20 个需要决定的组成块，也就是选择两对，分别含有 5 个节点的模块作为普通模块和下采样模块。我们有以下这么多种可能的结构：

$$\mathcal{B} = ((n+1)!)^2 \cdot (n\_ops)^{2n}$$

其中，n 表示节点的数量，n_ops 表示运算块的种类数量。因此，普通模块和下采样模块加起来的总共可以有大约 10^{33} 种搜索的结构。

3.3 性能评估策略

NSGANetV1 有两个性能评估指标，一个是分类准确度，另一个是模型架构的复杂度。

首先，对于分类准确度的目标，本方法通过新生成一个代理模型（proxy model），并使用 SGD 方法进行训练，将训练好的代理模型的分类性能作为评估指标。然而，即使使用缩小后的代理模型，使用梯度下降的方法进行训练来评估性能依然是一个十分耗时，消耗算力的方式。为了解决这个难题，本文通过消融实验，对架构缩小和优化过程做了合理的缩小，从而产生合适的代理模型。如图 3所示。本文使用秩序列相关性（Rank-order correlation）指标来判断，该指标表示缩小后的代理模型族群与原模型族群的排名情况相似性，当指标接近 1 时，认为代理模型的排名与原排名变化不大，可用于缩小模型。反之，则不适于以该数值缩小模型。如图 3所示，我们设定原模型初始化通道数为 36，层数为 20，训练轮次为 96。从图 3(b) 可知，缩小通道数并不能明显减少训练时间，故不考虑缩小通道数。从图 3(c) 可知，当层数缩小为 14 层时，排名变化较小，且训练时间减少。如图 3(d) 可知，当训练轮次减少到 36 次时，训练时间大幅减少，且排名变化也相对变化较小。故最终采用初始化层为 36，层数为 14，训练轮次为 36 的代理模型。

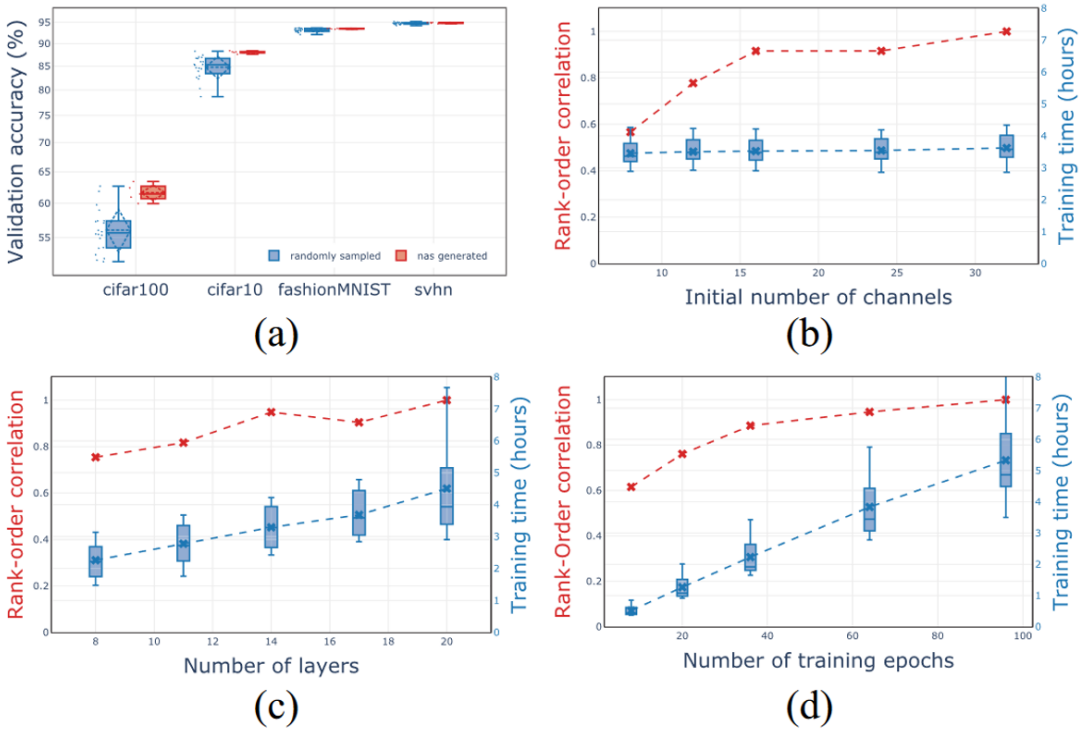

图 3: 代理模型的消融实验 (a) 随机生成的架构和同行 NAS 方法的架构在 4 个数据集上的平均分类精度分布。通过减少性能 (红线) 与梯度下降实际时间 (蓝色方框) 节省的 Spearman 相关性。(b) 初始化通道数。(c) 架构层数。(d) 训练轮次。注意，(b)-(d) 有两个 y 轴对应两个不同颜色的线

其次，是对复杂度的评估。复杂度的指标可以使用活跃节点数、活跃连接数、参数数量、推理时间、FLOPs 等指标，本文基于实验决定使用代理模型的 FLOPs 指标作为复杂度的目标。因为，实验表明推断时间会因为设备不同而得到不同的情况，而活跃节点数、活跃连接数和参数数量仅能代表单一方面的复杂度，而 FLOPs 指标更加全面可靠。

基于以上两个优化目标，本文使用 NSGA-II[7]进行优化。

3.4 新一代种群的生成

在生成新的后代种，主要包括两个操作，基因重组（crossover）和基因变异（mutation）。

首先，基因重组需要先作亲代选择，在当前族群中选取两个亲代进行基因重组操作。本文使用 binary tournament selection[22]方式，即在族群中随机抽取两个个体，并从中选择优秀个体作为亲代，执行两次操作即可得到两个亲代个体进行基因重组操作，这样既能满足择优，又能满足随机性。本文使用的基因重组方式以节点（node）为单位进行基因交换操作。如图 4所示。基因重组后会产生两个子代，我们只保留其中一个子代，以提高族群的多样性。每轮产生的子代数量与亲代族群数量相同。

其次，为增加族群的多样性，避免收敛至局部最优点，我们在基因重组产生后代以后，再进行基因变异操作。本文中的基因变异操作使用多项式变异操作（polynomial mutation operator，PM）[23]。基因变异又两种类型，分别是输入连接块的基因变异，如图 5左侧所示，以及运算块的基因变异，如图 5右侧所示。在使用 PM 方法进行基因变异的结果会尽量保持相似，变化幅度较小。例如，当前输入连接模块为 $h_i^{(2)}$，则其突变为 $h_i^{(1)}$ 的可能性会比突变为 h_{i-2} 的可能性大。而在运算块中，会根据运算快类型的计算复杂度进行预先排序，使其变异成计算复杂度相近的模块的概率更大。

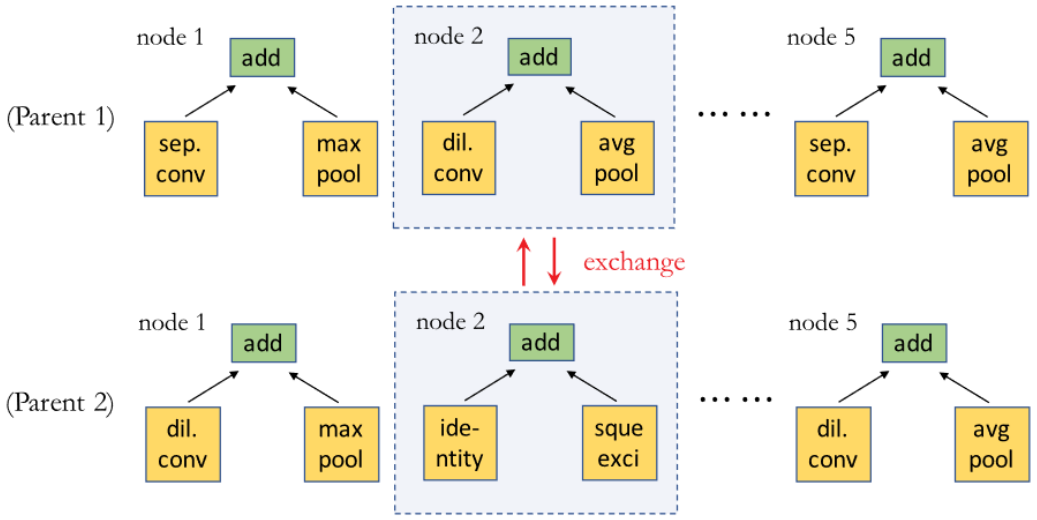

图 4: 基因重组方法

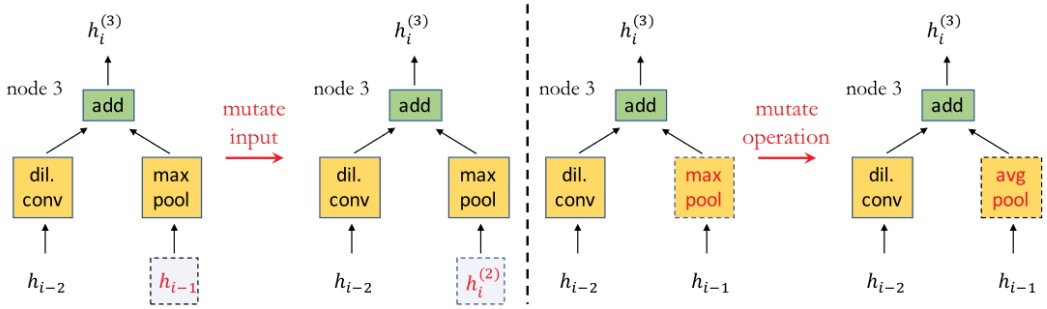

图 5: 基因重组方法

4 实验与结果

复现程序为基于 NSGANetV1 生成在 CIFAR-10 做图像分类任务的一组神经网络架构。

4.1 与已有开源代码对比

本论文没有开源代码，故没有对比。复现过程参考以下代码：https://github.com/ianwhale/nsga-net[24]，主要参考其中的 pymoo 多目标优化库在 NAS 任务中的使用方式，以及参考其将个体的基因信息转换为完整架构的方式。复现过程部分运算块种类，输入连接块基因选择，基因重组方式和基因变异方式。在设定运算块时候，除了将上述 12 种运算块操作设定好外，我们还需要注意到部分的运算块操作本身并不能满足下采样模块（Reduction block）的要求，如 Idendity 运算块不会使得分辨率减半，其他设置 stride 为 2 即可实现，所以，若这些无法减少分辨率的运算块出现在下采样模块中的时候需要进行处理。

4.2 实验环境搭建

实验环境设置如下：

1) Python 3.9.13
2) pymoo == 0.3.0 （最新版本更换了框架，故不能兼容）
3) Pytorch 1.7.1
4) torchvision 0.8.2

实验运行硬件环境为一张 Titan X Pascal。受设备资源的限制，并没有像原论文那样设置。搜索程

序运行设置为种群大小为 15，后代数为 10，迭代次数为 10，代理模型初始化层数 24，代理模型层数 11，代理模型训练轮次 30。训练完整模型程序是在 CIFAR-10 数据集中训练 600 个轮次。

4.3 使用说明

运行搜索架构程序指令，设置于上述一致：

python search.py –pop_size 15 –n_gens 10 –n_offspring 10 –init_channels 24 –layers 11 –epochs 30 –device cuda:0

每一代的数据会存入相对应的文件夹中，第 x 代演化结果会存入 gen_x 文件夹中，里面包括 fitness 和 individuals 两个文件。fitness 文件保存本代族群中每个个体架构的两个评估指标值，即 FLOPs 和分类准确度；individuals 则保存本代族群中所有个体架构的基因型。最终在完成最后一代族群生成后，只保留符合 Perato-Front 的个体作为最终的结果，该结果保存于 result 文件夹中的 fitness 和 individuals 文件中。

训练指定个体架构程序指令如下（假设搜索出结果保存于“./result/individuals”文件中，并获取其第一个架构做训练）：

python train.py –genofile ./result/individuals –n_geno 1 –epochs 600 –device cuda:0

其中，genofile 表示搜索结构保存路径，n_geno 表示读取里面第几个个体架构。从而训练指定个体架构的模型，并保存其模型参数。

4.4 创新点

为了让模型尽可能轻量化，本文只将轻量化的卷积核或运算块作为搜索空间，减少了模型的复杂度。

5 实验结果分析

下表是论文所给出的结果，以及与部分方法的对比。

Architecture	Search Method	GPU-days	Top-1 Acc.	#params
NSGANetV1-A0	**EA**	**27**	**95.33%**	**0.2M**
CGP-CNN[14]	EA	27	94.02%	1.7M
Large-scale Evo.[11]	EA	2,750	94.60%	5.4M
AE-CNN+E2EPP[15]	EA	7	94.70%	4.3M
ResNet[2]	manual	-	95.39%	1.7M
NSGANetV1-A1	**EA**	**27**	**96.51%**	**0.5M**
Hierarchical NAS[13]	EA	300	96.37%	61.3M
DenseNet[3]	manual	-	96.54%	25.6M
NSGANetV1-A2	**EA**	**27**	**97.35%**	**0.9M**
AmoebaNet-A[12]	EA	3,150	96.88%	3.1M
NSGA-Net[24]	EA	4	97.25%	3.3M
NSGANetV1-A3	**EA**	**27**	**97.78%**	**2.2M**
NASNet-A[19]	RL	1,575	97.35%	3.3M
LEMONADE[18]	EA	90	97.42%	13.1M
NSGANetV1-A4	**EA**	**27**	**97.98%**	**4.0M**
AmoebaNet-B[12]	EA	3,150	97.87%	34.9M

以下为复现得到的结果：

Architecture	Search Method	GPU-days	Top-1 Acc.	#params
1	EA	8	94.84%	1.4M
2	EA	8	95.01%	2.1M

6 总结与展望

本文提出了 NSGANetV1 方法实现同时优化架构性能和复杂度的多目标演化算法，通过设定基因重组和基因变异方式，不断搜索新的架构。特别提出了将原始模型缩小为代理模型的方法，大大提升了搜索的效率，提高了该方法的实用性。另外，原论文中还有一部分使用贝叶斯网络 BN 的方法，受时间和知识量的限制，这里暂未能实现，若感兴趣可自行实现。由本文可见，通过 EA 方式可以给不同的任务定制化对应的模型架构，其性能能比传统的人工设计网络架构优秀，且神经网络架构搜索时间和设备需求的减少，使得定制化模型架构变得可行。未来在基于多目标演化计算实现 NAS 的研究，可以对优化目标进行定制化设定，以满足不同的需求，也可以提供更自动化的方法，例如将总架构层数等因素纳入搜索空间范围内等等。

参考文献

[1] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.

[2] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[3] HUANG G, LIU Z, VAN DER MAATEN L, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4700-4708.

[4] ZHANG X, ZHOU X, LIN M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6848-6856.

[5] SANDLER M, HOWARD A, ZHU M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4510-4520.

[6] JUEFEI-XU F, NARESH BODDETI V, SAVVIDES M. Local binary convolutional neural networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 19-28.

[7] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II [J]. IEEE transactions on evolutionary computation, 2002, 6(2): 182-197.

[8] YAO X. Evolving artificial neural networks[J]. Proceedings of the IEEE, 1999, 87(9): 1423-1447. DOI: 10.1109/5.784219.

[9] STANLEY K O, MIIKKULAINEN R. Evolving Neural Networks through Augmenting Topologies[J]. Evolutionary Computation, 2002, 10(2): 99-127. DOI: 10.1162/106365602320169811.

[10] XIE L, YUILLE A. Genetic CNN[C]//2017 IEEE International Conference on Computer Vision (ICCV). 2017: 1388-1397. DOI: 10.1109/ICCV.2017.154.

[11] REAL E, MOORE S, SELLE A, et al. Large-scale evolution of image classifiers[C]//International Conference on Machine Learning. 2017: 2902-2911.

[12] REAL E, AGGARWAL A, HUANG Y, et al. Regularized evolution for image classifier architecture search[C]//Proceedings of the aaai conference on artificial intelligence: vol. 33: 01. 2019: 4780-4789.

[13] LIU H, SIMONYAN K, VINYALS O, et al. Hierarchical representations for efficient architecture search [J]. arXiv preprint arXiv:1711.00436, 2017.

[14] SUGANUMA M, SHIRAKAWA S, NAGAO T. A genetic programming approach to designing convolutional neural network architectures[C]//Proceedings of the genetic and evolutionary computation conference. 2017: 497-504.

[15] SUN Y, WANG H, XUE B, et al. Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor[J]. IEEE Transactions on Evolutionary Computation, 2019, 24(2): 350-364.

[16] KIM Y H, REDDY B, YUN S, et al. Nemo: Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy[C]//ICML 2017 AutoML workshop. 2017: 1-8.

[17] DONG J D, CHENG A C, JUAN D C, et al. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 517-531.

[18] ELSKEN T, METZEN J H, HUTTER F. Efficient multi-objective neural architecture search via lamarckian evolution[J]. arXiv preprint arXiv:1804.09081, 2018.

[19] ZOPH B, VASUDEVAN V, SHLENS J, et al. Learning transferable architectures for scalable image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 8697-8710.

[20] HU J, SHEN L, SUN G. Squeeze-and-Excitation Networks[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018: 7132-7141. DOI: 10.1109/CVPR.2018.00745.

[21] JUEFEI-XU F, BODDETI V N, SAVVIDES M. Local Binary Convolutional Neural Networks[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017: 4284-4293. DOI: 10.1109/CVPR.2017.456.

[22] MILLER B L, GOLDBERG D E, et al. Genetic algorithms, tournament selection, and the effects of noise[J]. Complex systems, 1995, 9(3): 193-212.

[23] DEB K, AGRAWAL R B, et al. Simulated binary crossover for continuous search space[J]. Complex systems, 1995, 9(2): 115-148.

[24] LU Z, WHALEN I, BODDETI V, et al. Nsga-net: neural architecture search using multi-objective genetic algorithm[C]//Proceedings of the genetic and evolutionary computation conference. 2019: 419-427.