汕頭大學
SHANTOU UNIVERSITY

硕 士 学 位 论 文

题　　目 基于禁忌搜索和进化算法的神经网络架构搜索

英文题目 **Neural Architecture Search Based on Tabu Search and Evolutionary Algorithms**

姓　　名 龙周彬　　学　　号 **151909114**

所在学院 工学院　　导师姓名 范衡

专　　业 电子与通信工程

入学日期 **2019 年 9 月**　答辩日期 **2022 年 5 月**

学位论文原创性声明

本论文是我个人在导师指导下进行的工作研究及取得的研究成果。论文中除了特别加以标注和致谢的地方外，不包含其他人或其他机构已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在论文中以明确方式标明。本人完全意识到本声明的法律责任由本人承担。

作者签名：龙周彬　日期：2022 年 5 月 30 日

学位论文使用授权声明

本人授权汕头大学保存本学位论文的电子和纸质文档，允许论文被查阅和借阅；学校可将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存和汇编论文；学校可以向国家有关部门或机构送交论文并授权其保存、借阅或上网公布本学位论文的全部或部分内容。对于保密的论文，按照保密的有关规定和程序处理。

作者签名：龙周彬　　导师签名：

日期：2022 年 5 月 30 日　　日期：2022 年 5 月 30 日

硕 士 学 位 论 文

论文中文题目：基于禁忌搜索和进化算法的神经网络架构搜索

论文英文题名：Neural Architecture Search Based on Tabu Search and Evolutionary Algorithms

指 导 教 师 ：范　衡
申　请　人　：龙周彬

论文答辩委员会成员

主席：肖　刚　教授　(韩山师范学院)

委员：陈耀文　教授　(汕头大学工学院电子系)

邹安民　教授　(汕头大学工学院电子系)

赖李洋　副教授　(汕头大学工学院电子系)

李　兵　副教授　(汕头大学工学院电子系)

摘要

随着人工智能技术的飞速发展，神经网络已经在日常生活中的许多领域得到了广泛的应用，如人脸识别、机器翻译、语音识别和医疗图像诊断等。通常来说，一个神经网络的性能主要由网络模型的对应的结构和权重参数决定。截止目前，网络模型的权重参数往往可以使用随机梯度法进行训练，网络模型的结构则主要由相关专业人士进行设计。然而在现实中，并不是所有的人都具备针对特定任务设计神经网络的能力，这严重阻碍了神经网络算法的落地与应用。鉴于此，针对特定任务自动化地生成神经网络模型结构成为急需解决的问题，同时神经网络架构搜索作为一种可以自动设计神经网络结构的技术也越来越受到广泛的关注。

现有大多数神经网络架构搜索的优化方法，包括进化算法、强化学习和基于梯度的方法，都没有明确地使用记忆策略来记录搜索的过程，这在一定程度上会导致重复的搜索，降低搜索效率。为了解决这一问题，本文提出了一种新的神经网络架构搜索算法，主要内容如下：

（1）提出了双种群协同机制。在搜索过程中维持最优子种群和禁忌子种群，通过一定的概率来选择父代种群。从而保证父代种群混合了两个子种群的信息，避免重复的搜索。

（2）设计了禁忌规则。禁忌机制建立了一个禁忌列表来记录近期搜索过程中所选择的位置和操作，并从中选择那些近期未被选择过的个体用以建立禁忌子种群。

（3）引入了替代模型。在搜索过程中使用替代模型对个体进行粗略评估，从中选择出较有潜力的个体，从而加速搜索过程中的性能评估。

为了验证本文所提方法的有效性，本文在 CIFAR-10 和 Fashion-MNIST 公开数据集上进行了多次对比实验。综合实验结果表明本文所提方法与当前先进方法的具有可比性。

关键词：神经网络架构搜索；进化算法；禁忌搜索；替代模型

Abstract

With the rapid development of artificial intelligence, neural networks have been widely used in our daily life, such as face recognition, machine translation, speech recognition and medical image diagnosis. In generally, the performance of neural network mainly depends on the network topology and its connection weights. Up to now, the connection weights of the neural network are trained by using stochastic gradient algorithm, and the topology of the neural network is mainly designed by relevant professionals. However, most of us don't have the ability to design neural networks for specific tasks, which seriously hinders the application of neural network algorithms. Therefore, generating neural network topology for specific tasks automatically has become an urgent issue to be resolved, while neural architecture search (NAS) as a technology for automatically designing neural network topology is also attracting more and more interests.

Most of the existing optimization methods for NAS, including evolutionary algorithms, reinforcement learning and gradient-based methods, have not employed memory strategies explicitly to record the search process, which may lack of efficiency when searching neural architectures. To solve this issue, we propose a new NAS method, the main ideas are list as follows

(1) We proposed a dual-population mechanism. In the search process, the archive subpopulation and tabu subpopulation are maintained, and the parent population is selected through a certain probability in the two subpopulations. This mechanism ensures that the parent population combimes the information of the two subpopulations and avoids repeated searches process.

(2) We designed tabu rules. The tabu rules build a tabu list to record the selected positions and operations in the recent search process. And the tabu subpopulation are selected from these individuals which not in the tabu list.

(3) We employed surrogate model to accelerate the search process. In the search process, the surrogate model is employed to get the performence of individual rather than use stochastic gradient method directly.

To confirm the superior performance of our approach, our method was evaluated on CIFAR-10 and Fashion-MNIST datasets. The comprehensive experimental results demonstrate the superiority of the suggested method.

Keywords： Neural architecture search, Evolutionary algorithms, Tabu search, Surrogate model

目 录

第 1 章 绪论

1.1 研究背景及意义

近 10 多年来，深度神经网络因具有强大的自动特征表示能力已经在许多领域,如图像分类、目标检测、自然语言处理和语音识别等上取得了巨大的突破。相对于传统方法的手工提取特征然后再处理的方法，深度神经网络可以从原始数据自动学习并提取有用的特征，大大节约人力成本并提高效率。通常认为，一个神经网络模型在上述任务上取得优异性能与两个因素有关，分别是网络模型结构和网络模型结构对应的参数，只有这两者相互达到最优才能让深度神经网络发挥最大的性能。对于一个给定的神经网络模型，可以通过一个损失函数来衡量模型输出值和实际值之间的误差，再利用梯度算法来不断调整网络模型参数以便实现在给定任务上的最优性能。但是如何设计一个好的神经网络模型，这一直以来都是学术界和工业界在不断探索的问题。

截止目前，几乎所有成功的深度神经网络模型，如分类任务上的 ResNet[1]、GoogLeNet[2] 和 DenseNet[3]；目标检测任务上的 YOLO[4]和 Faster R-CNN[5]；分割任务上的 FCN[6]、SegNet[7]和 U-Net[8]等模型往往是由专业的研究人员或者专家来设计的，他们往往拥有丰富的神经网络设计经验和图像处理领域的相关知识。但是在现实中绝大多数的人都不同时具备这种丰富的设计经验和相关领域的知识，同时在设计神经网络模型的时候往往由于个人的经验和思维的局限性很难设计出性能更优的网络模型。而且当前流行的神经网络模型通常都是针对于某个特定数据集来进行设计的。然而，在实际应用中需要面对各种复杂多变的场景，如果场景发生了改变或者更换了使用场景，则可能需要重新调整神经网络模型。这就对网络模型的设计者提出了更高的要求。基于上述原因，神经网络架构搜索 (Neural Architecture Search: NAS)[9,10]，一种可以让计算机在某种特定任务自主地设计出性能可以比拟或者超越传统人工设计的复杂神经网络的方法，成为了一个非常有意义的研究课题。

1.2 研究的难点和挑战

$$\begin{cases} \arg\min_{A} = \mathcal{L}\left(A, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{fitness}}\right) \\ \text{s.t. } A \in \mathcal{A} \end{cases} \tag{1-1}$$

通常来说，神经网络架构搜索可以归结为如公式(1-1)所示的优化问题，其中 $\mathcal{A}$ 表示潜在神经网络结构的搜索空间，$\mathcal{L}(\cdot)$ 表示在训练数据集 $\mathcal{D}_{\text{train}}$ 上训练后的网络模型在适应度评估数据集 $\mathcal{D}_{\text{fitness}}$ 位上的性能。$\mathcal{L}(\cdot)$ 通常是非凸的和不可微的。实际上，神

经网络架构搜索是一个复杂的优化问题，其面临着一些难点和挑战，例如：

（1）复杂约束：神经网络是一个强约束的网络，每一个模块对输入和输出的特征尺度有着严格的要求。想要实现神经网络的自动化搜索需要灵活设计每一个模块的输入和输出，让其可以正确地工作在网络的各个部分。

（2）离散表示：神经网络架构搜索的过程可以简单理解为一个组合优化的过程，从大量的模块库中选择一个模块并组装，再观察其在测试集上的性能。所以，在组合的过程中通常需要将神经网络进行编码并离散表示。如何实现高效地将神经网络编码，且快速地从编码翻译为神经网络，这也是一个研究的难点。

（3）双层结构：一般来说，一个神经网络的性能往往由网络的模型结构和其所对应的参数决定。所以在搜索过程中不仅需要优化网络结构，还需要对所产生网络模型的参数进行优化。所以神经网络架构搜索是一个典型的双层优化问题，外层是网络结构的优化，内层是网络权重参数的优化。

（4）计算昂贵：在神经网络搜索的过程中会产生很多候选的个体，这些候选个体都需要经过梯度下降算法算法来进行评价，根据现有的计算资源，训练一个网络需要几个小时甚至几十个小时，在搜索过程中会产生几千甚至几万个候选个体。所以神经网络架构搜索的计算会非常昂贵，同时这也是一个制约神经网络架构搜索发展的一个重要因素。

（5）多个冲突准则：一般来说，在神经网络中模型越大，模型的特征表示能力就越强，性能就越好。但是模型越大，对硬件资源的要求就越高。所以在神经网络架构搜索中，如何在性能和模型大小等多个指标中取得一个权衡，这也成了一个研究的难点问题。

1.3 研究现状

神经网络架构搜索算法的初始工作通常被认为是由谷歌团队于 2016 年首次在 arxiv 网站上发布的论文[11]，该论文同时也在 2017 年被国际学习表示会议(The International Conference on Learning Representations, ICLR)正式接受出版。从那时起，大量的研究人员前赴后继投入了巨大的精力，不断开发新的算法用以解决神经网络架构搜索中存在的各种问题。

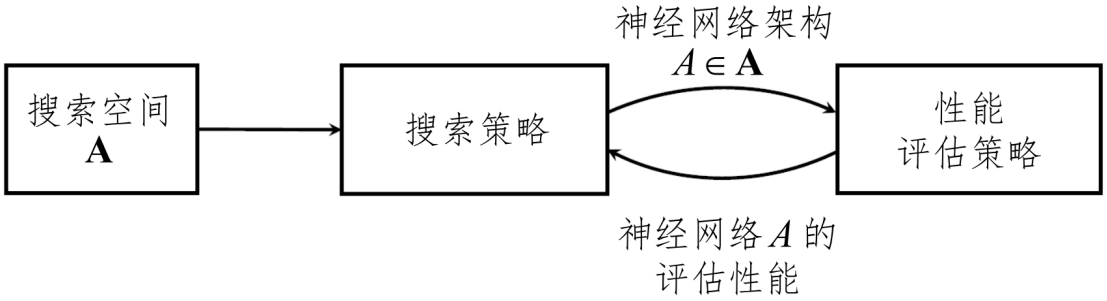


图 1-1 神经网络架构搜索算法框架图

神经网络架构搜索的算法流程框架如图 1-1 所示，从图中可以看出神经网络架构搜索主要分为搜索空间、搜索策略和性能评估策略三大块。

（1）搜索空间：搜索空间定义为一个神经网络结构理论上所有的表示可能。搜索空间设置的好坏直接影响到了最后的搜索效果的好坏，也决定了搜索的效率。在实际，针对不同的应用，设计者可以在搜索空间中融入该应用的知识体系结构，通过这种方式可以在某种程度上提高搜索效率，简化搜索过程。然而，加入了人的先验知识的同时也融入了个人的设计偏见，在提高搜索效率的同时也有可能阻碍了新网络结构的发现。

（2）搜索策略：搜索策略定义为如何去探索搜索空间，从而达到更快更好地产生所需要神经网络模型的目的。搜索策略的设计是一个经典的探索与开发问题，一方面要尽可能多地探索未知的领域，另一方面需要避免过早陷入局部最优中，从而获得次优解。

（3）性能评估策略：神经网络架构搜索的目的在于搜索一个性能更好的网络模型，而网络模型的性能评估策略指的就是评估网络性能优劣的过程。通常可以将一个网络模型在指定的训练集从头开始训练到收敛，然后在对应的测试集上进行评估该网络的性能。然而很不幸的是，这个评估方法计算代价非常昂贵，也正是这个原因阻碍了在搜索过程中探索更大搜索空间的可能。

1.3.1 搜索空间研究现状

搜索空间定义了一些基本的操作，通过这些不同的操作可以组合成不同的网络结构。搜索空间的定义直接影响了最终的网络形态，构建一个合适的搜索空间可以让搜索算法更加高效地找到合适的网络结构。现有的研究中，搜索空间主要有三种，分别是链式结构、多分支结构和基于 cell 结构的搜索空间。

1.3.1.1 链式结构

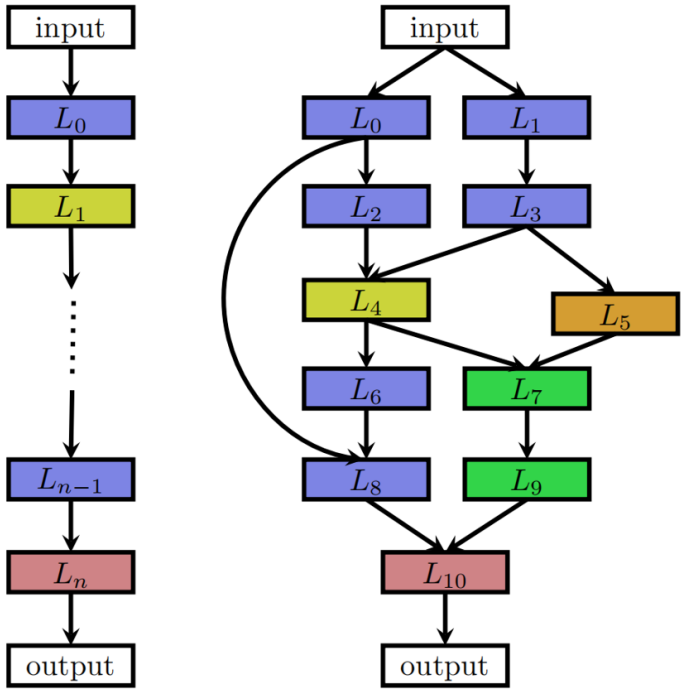
基于链式结构的搜索空间是最为简单的一种搜索空间结构，早期的神经网络模型，如 VGG-Net[12]，AlexNet[13]等都是这种架构。简单来说，链式结构的每一网络层，如卷积层，池化层等功能层的输入和输出都仅仅只和该层的前一层或者后一层相连接，

不会出现跨层连接的情况。图 1-2 左边就是一个典型链式结构的神经网络架构。针对链式结构的搜索空间，设计者需要考虑的问题有如下几个：

（1）网络的层数：由于链式结构的搜索空间相对简单，所以增强模型性能的方法主要来自于模型的长度（深度）。一般认为，模型越深，模型可表示的特征能力就越强，模型性能就越好。但是模型层数过深会造成模型参数量够大，就需要消耗更多的计算资源。同时模型的层数过深会出现梯度消失和梯度爆炸等问题，让模型性能变差。现有解决这个问题的方法有，如合理设置模型的深度，加入一些正则化处理或者加入 Bath Normalization (BN)[14]，等策略，但是这些策略都无法从根本上解决这些问题。

（2）网络各层的操作和相关参数：虽然链式结构的各个层的功能大体固定，但是每一个功能模块的选择较多。例如，卷积层除了常规的卷积外还可以选择空洞卷积[15]、深度可分离卷积[16]等不同类型的卷积等。卷积的具体超参数也是需要考虑的，如卷积核为3×3，5×5，7×7 的卷积等[17, 18, 19]，同时还需要考虑是否补零，步长和通道数等具体参数。

（3）网络各个功能层的排列顺序：在人工设计的神经网络中，池化层一般位于卷积层的后面。



a. 链式结构　　b. 多分支结构

图 1-2 基于链式结构和多分支结构的搜索空间[9]

1.3.1.2 多分支结构

链式结构的缺点是显而易见的。为了在一定的网络深度内增加网络模型的特征提

取能力，参考 GoogLeNet[2]的 Inception 模块可知，网络模型可以“变胖”，即前一层网络的输出可以经过多个分支然后再进行汇总操作。网络模型除了“变胖”，还可以使用跳跃连接，如经典的 ResNet 网络的残差模块（Residual block）就证明了使用跳连可以将浅层提取的特性信息直接传给高层网络，通过这种方式可以大大提高网络模型的性能[11,20,21]。基于分支结构搜索空间的网络模型如图 1-2 右边所示，从图中可以看出数据的流动不是单一的流动方向，而是多路并行且会在一定的支点汇总，然后再多路并行前进。

针对多分支结构的搜索空间，设计者除了需要考虑网络的层数和功能层的相关操作外，还要特别注意跳连的位置和数目。

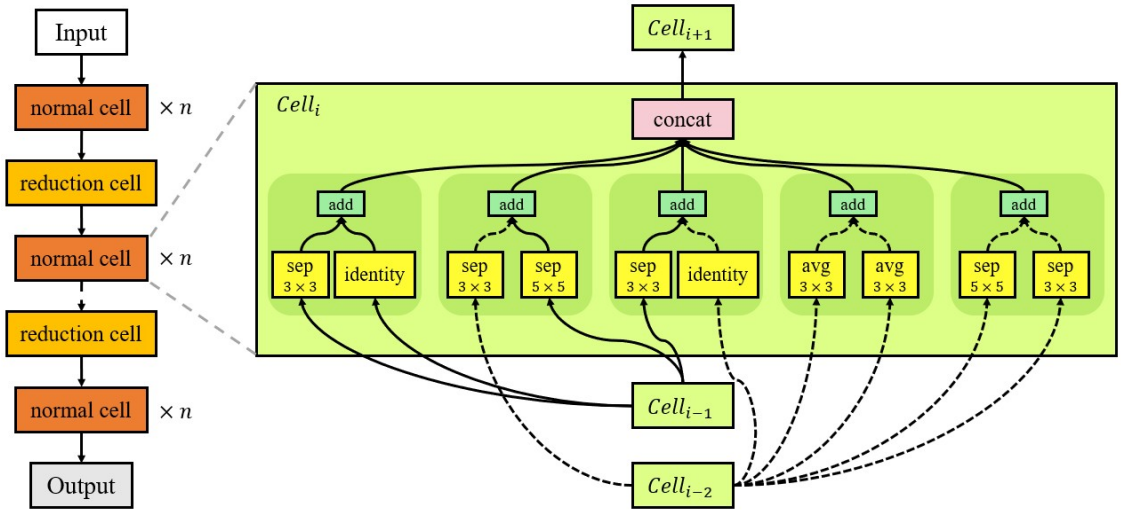


图 1-3 基于 cell 结构的搜索空间

1.3.1.3 基于 cell 的结构

链式结构和分支结构各有特点，将两者结合起来，优势互补，就形成了基于 cell 结构搜索空间的神经网络，如图 1-3 所示。从图中看出，该结构最外层如图 1-3 左边显示，和链式结构类似，其中的每一个 cell 可以看作是链式结构里面的一个功能层。而在每一个 cell 内部，如图 1-3 左边所示，内部是一个带有多路并行分支的结构，且第 i 层 cell 的输入有两个，分别是第 $i-1$ 和 $i-2$ 层 cell 的输出。相对于链式结构和分支结构的网络，基于 cell 的结构在外层具有链式结构的简单优势，在 cell 内部又同时具有多路分支结构和跳连结构的优势。因此基于 cell 结构的搜索空间是现在主流的搜索空间。

基于 cell 结构搜索空间的网络结构由两种不同功能结构的 cell 模块按照一定的排列顺序堆叠而成，这两种模块分别是 normal cell 和 reduction cell。从结构上看，这两种 cell 的内部结构都是一样的，如图 1-3 右边所示，一个 cell 有两个输入，一个输出，cell 内部具有 k 个节点（一般将 k 设置为 5）。对于 normal cell 来说，其输出的

特征图（feature map）大小与输入的一致，而 reduction cell 的输出特征图的维度是其输入的一半。在搜索过程中，主要精力在于 normal cell 和 reduction cell 的搜索。在搜索完成后，再按照一个的规则进行堆叠，最后将堆叠后的网络训练验证其性能结果。

对比现有的三种不同的搜索空间，链式结构的搜索空间相对比较直接明了，设计者通常通过增加网络的层数来增强网络的性能，这种设计具有一定的局限性。多分支结构可以通过跳连和多路并行来增加网络宽度来提高模型性能，但是在搜索过程中会比较复杂，搜索空间过大。基于 cell 结构的网络架构同时兼备了上述两种结构的特点，在外层通过堆叠 cell 来组成网络模型，同时在 cell 内部使用多分支结构和跳连方式。在搜索过程中，只关注于 normal cell 和 reduction cell 的搜索，较为简单。

1.3.2 搜索策略研究现状

搜索策略的目的在于能够快速地在搜索空间中找到一个满足要求的网络模型。因为在当前的研究中，搜索空间往往是巨大的，按照目前的计算资源，根本无法或者很难进行穷举或者遍历所有搜索空间得到一个最优的网络模型。基于此，一个稳定且高效的搜索策略显得尤为重要。在当前的研究中，常用的搜索策略有包括强化学习、进化算法和基于梯度的可微分方法等。

1.3.2.1 强化学习

基于强化学习[22]的搜索策略是神经网络架构搜索中常用的搜索策略。在搜索过程中，神经网络架构的生成可以看作是智能体（Agent）的行为（Action），行为的空间等同于定义的搜索空间，生成网络模型架构所得到的性能看作是智能体的奖励（Reward）。不同的强化学习方法的主要不同在于他们如何表示代理策略以及如何优化策略。

Zoph 和 Le[11]在 2017 年的时候使用一个循环神经网络（Recurrent neural network：RNN）[23]作为控制器，然后对搜索空间进行采样，紧接着将采样的数据组合为神经网络模型。在搜索过程中使用强化学习来优化 RNN 模型的参数，引导控制器更好地采样。最后该方法采样出来的网络在公开数据集上取得了与人工设计网络性能相近的准确率。随后在 2018 年 Zoph[24]等人提出了 NASNet，在 NASNet 中采用基于 cell 的搜索空间，并将搜索出来的 normal cell 和 reduction cell 进行堆叠形成网络模型。最后经过验证，搜索出来的网络模型在公开数据集 ImageNet[25]和 COCO[26]上都取得不错的成绩。同期还有 Baker[17]等人使用 Q-learning[27]来选择每一层网络的参数并提出 MetaQNN 网络。然后 Zhong[28]等人提出 BlockQNN，在该方法中使用了早停策略和分布式异构框架来提高搜索效率。同样地，ENAS[29] 网络为了提高搜索过程的效率，使用了权重共享的策略，将网络中对应的权重参数进行继承。通过这种方式，ENAS 避免了从零开始训练网络中的权重参数，从而大大提高了效率。Tan[30]等人提出的

MnasNet 使用强化学习的方法将在终端的资源约束也考虑到智能体的奖励中，以此提高了搜索的性能。

1.3.2.2 进化算法

在自动机器学习[31]（Auto machine learning，Auto ML）中，可以采用进化算法来优化神经网络的模型结构、权重参数以及其他一些超参数。早在 20 多年前 Miller[32]等人使用了进化算法来对神经网络结构进行优化，对应的网络权重参数则使用随机梯度法进行训练。随后 Angeline[33]，Stanley[34]等人使用了进化算法来同时优化神经网络的架构和网络权重参数。特别是 Kenneth[35]等人提出 NEAT，其主要思想是通过从最小的网络结构进化到一个完整的神经网络模型，同时使用保护个体的创新机制，在进化过程中使用不同的标记来区分不同的种群。实验结果表明 NEAT 同时进化的网络拓扑结构和权重可以成为人工进化的神经网络主要的优势，且其提出的网络在一些任务上的性能优于当时的固定网络结构而只优化参数的算法。

随着神经网络模型的增大，使用进化算法去优化网络模型的权重变得不太现实。同时，基于梯度的反向传播机制来优化网络模型权重也越来越得到学术界的认可。所以在近几年的工作中，如 Real[36]，Suganuma[18]，Liu[37]，xie[38] 等人的工作中都是使用进化算法来优化网络模型的结构，而网络模型参数部分则交给了随机梯度下降法[39]（stochastic gradient descent，SGD）进行优化。在 2007 年，Salimans[40]等人的工作中证明了，进化策略[41]作为一种黑盒优化算法可以作为强化学习的一种替代方式，可以替代如 Q-Learining 和策略梯度等相关操作。同期，Real[42]等人在大规模图像分类上使用进化算法来进行网络模型的自动化搜索，并在 CIFAR-10[43]和 CIFAR-100 数据集上获得与人工设计网络相近的性能结果。随后，Real[42]等人在 2019 年提出的 AmoebaNet，改进了对后代个体的选择算法，在进化过程中尽可能地保留多年轻的个体，实验结果表明，进化算法的收敛速度比强化学习和随机搜索更快，且首次在 ImageNet[25]数据集上打败了人工设计的网络模型。同期，Miikkulainen 等人提出的 CoDeepNEAT[44]网络，在 NEAT 的基础上对网络的拓扑结构，模块和超参数进行了优化，其在视觉、语音识别和自然语言处理等任务上均取得了与人工设计网络的性能相近的结果。在近期，CARS[45]使用进化算法从搜索空间中的超网络（supernet）采样子网，并使用权重继承的方式来对采样的子网进行权重参数的初始化，从而达到加速搜索的过程。

与此同时，还有一些其他的基于进化算法的网络自动化搜索算法，如 CGP-CNN[46]就使用笛卡尔遗传编程[47]从一些现有的网络模块中，如卷积模块，池化模块，残差模块等模块中相互组合形成新的网络模型。另外，NSGA-NET[48]使用了 NSGA-II[49]算法来考虑模型性能和模型参数量两个指标来指导神经网络的进化过程。

1.3.2.3 基于梯度的可微分方法

神经网络架构搜索往往被认为是一个不连续且不可导的问题，因为其搜索空间就是一个不连续的空间。但是，不连续的问题同样可以使用基于梯度的反向传播算法来求解。具体来说，使用连续松弛操作以实现直接的基于梯度的优化。和前面的方法不同，在一个网络层中，基于强化学习或者进化的方法往往是固定一个操作然后再评价该模型的性能，而在基于梯度的方法中，研究者[50]将所有的后端操作$\{o_1,\ldots,o_m\}$都赋予一个权重$\{\alpha_1,\ldots,\alpha_m\}$，对于输入的$x$，则输出：

$$y=\sum_{i=1}^{m}\alpha_i o_i(x),\alpha_i\geq 0,\sum_{i=1}^{m}\alpha_i=1 \tag{1-2}$$

在 2018 年，Liu 提出了 DARTS[50]算法，该算法首次将神经网络架构搜索问题转换为可微问题，其预先定义了3×3，5×5，7×7卷积和池化等 8 种操作，最后使用 softmax 函数将每个操作进行加权并保留权值最大的两条边。随后 Xuanyi Dong 等人基于 DARTS 提出了 GDAS[51]，在该工作中，GDAS 定制了一个 reduction cell，在搜索过程中只搜索 normal cell。同时 GDAS 不在整个超网络更新每一个 node 的权重，而是在搜索过程中仅采样节点间的一种操作，因此 GDAS 的搜索效率大大提高。在同期 P-DARTS[52]采用渐进式增加网络层数，同时使用了一下正则化规则来减少代理数据集和真实数据集之间的误差。Xu 提出的 PC-DARTS[53]将每种操作的通道连接起来，从而大大减少了搜索时候的显存占用，同时也提高了搜索的效率。

1.3.3 性能评估策略研究现状

由于网络模型的性能评估是一个非常耗时的一个过程，从零开始训练一个网络直到收敛需要几个小时甚至几十个小时。在神经网络架构搜索的过程中，需要评价几百上千，甚至几万个网络。如果对全部网络都从零开始训练至收敛，那么这将是一次巨大的资源消耗，同时这也是基本不可能完成的任务。基于此，一个高效稳定的性能评估策略显得尤其重要。

在现有的研究中，为了减少网络模型的评估时间，提高搜索效率，学者们做了大量的工作，也提出了许多解决方案。归结起来主要有低保真度评估、早停机制、代理模型和权重共享。近几年来，替代模型和权重共享机制越来越受到学者的欢迎。

1.3.3.1 低保真度评估

为了得到一个网络模型的真实性能，常用的方法往往是将训练集的所有数据都拿去训练网络模型直至模型收敛，但是这往往是比较消耗时间和计算资源的。所以在神经网络架构搜索中，有学者就提出了低保真度的网络模型性能评估的方法[24,54,55]。具体来说，在搜索过程中，可以使用训练集的部分样本而不是全部训练集，或者可以降

低训练集的图像分辨率来减少计算成本，还可以通过减少网络的层数以减少网络的参数来加速训练过程。

使用低保真度评估的方法虽然在一定程度上可以加速网络的评估，但是由于其使用的是缩小版的网络模型或者是部分的数据集，所以导致其评估结果和真实的网络结果存在一定的误差。

1.3.3.2 早停机制

早停机制是指在训练网络模型的时候不训练到收敛。具体来说在训练的时候，使用完整的网络模型，同时使用全部的训练集数据，但是可以不把网络训练到收敛，只训练较小的迭代次数，例如所有待评估网络模型都只训练 10 代，并以此作为网络模型的精度[53]。Zheng 等人在 2019 的 MdeNAS[56]的工作中指出，在相同的训练代数下，如果一个网络的性能比另一个网络的性能要好，那么将这两个网络都训练到收敛，结果还是原来性能好的网络最后的性能要好。

基于早停机制，神经网络的搜索效率可以大大提升，且可以在搜索过程中得到网络性能排序。

1.3.3.3 替代模型（surrogate model）

替代模型是工程问题中常用的一种优化方法。因为在很多实际问题中，往往无法直接或者很难求解是，可以使用计算量相对较小、求解迅速的简化模型来替代原来的复杂模型。从而达到加速工程求解的目的。

神经网络架构搜索过程的网络模型评估就是一个典型的计算量大的问题。现有的研究中也有许多学者使用替代模型来加速神经网络架构搜索的研究。如 Sun[57]等人使用随机森林训练一个离线的替代模型来预测网络模型的性能。同样地，Lu 等人在 NAT[58]中使用了包括径向基函数、高斯函数、多层感知机等多种模型训练在线的替代模型来预测网络模型的性能，在这个工作中，与以前的离线替代模型需要准备替代模型训练样本不一样，该工作在搜索过程中使用替代模型来挑选部分有潜力网络模型，并使用权重继承的形式来训练挑选出来的网络，最后将训练过的模型作为样本用来更新替代模型的参数。该方法在每一轮迭代中都会计算各个替代模型的误差，并在下一轮的迭代中选择误差最小的一个替代模型。

1.3.3.4 权重共享

除了上面那几种性能评估策略外，权重共享的评估方法也是常用的方法之一。对于权重共享，最常用的就是 One-Shot[59,60,61]方法。简单来说需要设计一个超网络，该网络包含了所有搜索空间的所有可能性，一个新网络结构可以看作是超网络的采样。通过超网络，只需要在搜索开始之前对超网络进行训练，然后在搜索阶段对新采样的网络结构的初始化权重设置为超网络中对应部分的权重。通过这种方式新搜索出来的

网络结构就可以获得比较好的初始参数，基于这些参数初始化的网络结构只需要较小的训练迭代次数就可以获得比较好的性能。因此通过权重共享的方式，也可以大大提高搜索效率。值得注意的是，在构建和训练超网络的时候由于该网络包含了所有的子网络，所以超网络会比较庞大且难于训练，这对计算资源又提出了新的要求。

1.4 主要研究内容

本文以研究基于分类任务的神经网络架构搜索算法作为核心研究内容，以进化算法和禁忌搜索算法作为主要研究手段，利用进化算法和禁忌搜索建立双种群进化机制，在搜索过程中同时保留进化算法的最优种群和禁忌算法的禁忌种群，实现快速高效地找到性能更好的网络。具体研究内容如下：

（1）基于进化的神经网络架构搜索，实现神经网络的自动化设计。利用双种群协同进化的优势，研究结合进化算法和禁忌搜索算法的神经网络架构搜索方法。

（2）研究禁忌规则的设计，实现对已经搜索过的区域进行合理避让。研究搜索过程中的禁忌列表设计，实现对搜索历史的保存。

（3）研究基于种群进化的神经网络架构搜索，设计并使用替代模型来粗略评估新产生个体的性能差异，从而大大提高搜索效率。

1.5 论文章节安排

本文共包含五个章节，各个章节的具体内容如下：

第一章：绪论。在本章节主要介绍了神经网络架构搜索的研究背景和意义，同时调研了神经网络架构搜索的研究现状，并指出其研究的重难点所在。为以下各个章节打下基础。

第二章：基础原理。本章会介绍本文研究的一些基础知识，如神经网络的基础知识、进化算法的基础理论和禁忌搜索的一些基本知识。

第三章：基于禁忌搜索和进化算法的神经网络架构搜索。在基于进化神经网络的基础上，融入禁忌搜索的记忆特性，形成最优种群和禁忌种群的双种群协同进化，提高搜索模型的性能。同时引入替代模型，使用替代模型来对网络模型的性能进行评估排序，从而达到加速神经网络架构搜索的过程。

第四章：实验结果与分析。在神经网络架构搜索的流行数据库 CIFAR-10 进行搜索和测试，分析实验结果，同时和其他当前先进方法进行比较。

第五章：总结与展望。在该章节中，将会对全文的研究内容、研究方法以及实验结果进行总结，然后凝练出本文的主要工作和贡献点。最后再并对本文研究领域的未来工作和研究前景进行展望。

第 2 章 相关基本原理

2.1 进化算法

进化算法(Evolutionary Algorithm: EA)是一种基于达尔文进化论所提出的启发式算法，其流程图如图 2-1 所示。进化算法的步骤大体可以分为初始化种群、计算适应度值、终止条件设置、后代选择和交叉变异等基本操作。

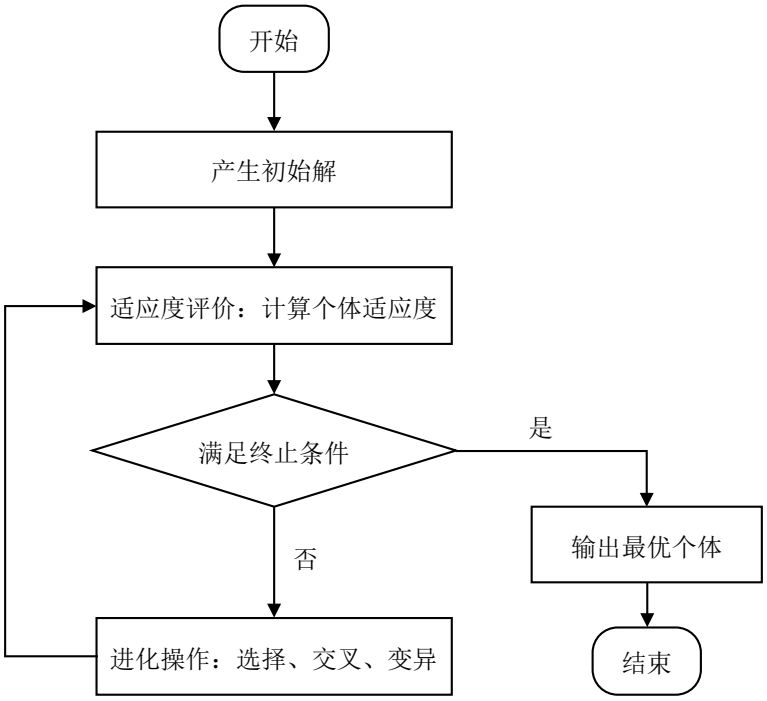


图 2-1 进化算法流程图

关于进化算法的一些基本概念如下：

（1）个体和种群：在进化算法中优化问题的一个解就可以当作是种群的一个个体，多个个体组合在一起就被称为一个种群。在现实使用中，由于在进化过程中使用直接编码比较麻烦，所以一般会通过一定的规则将个体编码为字符串。

（2）适应度值：为了综合评估个体的性能差异，需要设计合理的适应度函数来对个体的性能进行评估。同时适应度值也是用来指示父代种群选择的一个重要指标。

（3）后代选择：进化算法中在每一轮的循环迭代中通过优胜劣汰的机制选择性能较好的个体保留下来作为父代，并开始下一轮的循环。常见的选择算法有轮盘赌、锦标赛、截断选择、蒙特卡洛选择和概率选择等。

（4）交叉变异：交叉和变异是进化算法中产生后代的两种算子。交叉操作是指从父代种群中选择两个个体，然后两个个体交换部分编码，把优秀父代中的编码片段

遗传给子串。通过交换基因片段就有可能组合出性能更佳的后代个体。相对于交叉操作需要从父代种群中选择两个个体，变异操作只需要选择一个个体即可。具体来说，变异操作是对所选择的个体中的单个或者多个基因编码片段进行变异。

进化算法具有的较强鲁棒特性和灵活性，可以轻易找到大部分复杂问题的全局解。实验证明，结合较好的初始化解和胚胎模型设置，使用进化算法可以在优化的初期就有很大的概率找到较好的解。进化算法具有如下优点：

（1）进化算法不需要任何类型的梯度信息即可进行优化。目前大多数优化算法，如梯度下降法都严重依赖于损失函数的梯度信息来进行优化，但是在现实中很多优化问题对于梯度信息很难甚至完全不可能得到。

（2）进化算法非常的易于并行化运算。因为是基于种群的优化算法，所以很容易就可以实现并行化的计算，从而大大加速算法运行的效率。

（3）进化算法可以跳出局部最优解陷阱。基于种群优化和不依赖损失函数的梯度信息，使用适者生存的策略让进化算法不会轻易地陷入局部最优解的陷阱。

（4）进化算法可以很好地处理单目标和多目标的优化问题[62]。与解决多目标优化问题的传统方法相比，进化算法具有很大的优势，因为它们可以与整数、不连续或离散的设计变量同时应用；同时进化算法对帕累托前沿形状不敏感，并且能够找到位于非凸或不连续区域的解决方案。因此，进化算法可以适用于具有随机特征、不确定性或噪声适应性的问题。

2.1.1 进化算法常用的选择策略

在进化算法中常常使用不同的选择机制从种群中选择个体到交配池（mating pool）中，紧接着将交配池中的个体作为父代，并由父代产生子代。为了产生更好的子代，所以常常认为交配池里面的父代性能一般都是比较“好”的个体，而选择机制就是从种群中选择“好”个体的一个过程。选择压力是指性能好的个体在选择过程中被选到的几率的一个度量，一般来说选择压力越大，那么越好的个体就越容易被选择。这种选择压力促使遗传算法提高了后代种群的适应度。进化算法的收敛速度在很大程度上由选择压力决定的，选择压力越大，收敛速度越大。进化算法可以在很大范围的选择压力下找到最优或者近似最优的解。然而，如果选择压力很低，收敛速度将会很慢，而遗传算法将花费更长的时间来找到最优的解。如果选择离子压力过高，进化算法则会过快的收敛而增加了错失了找到最优解的可能性。针对选择策略，本文着重介绍常用的两种，包括锦标赛选择和轮盘赌选择。

（1）锦标赛选择（Tournament selection）

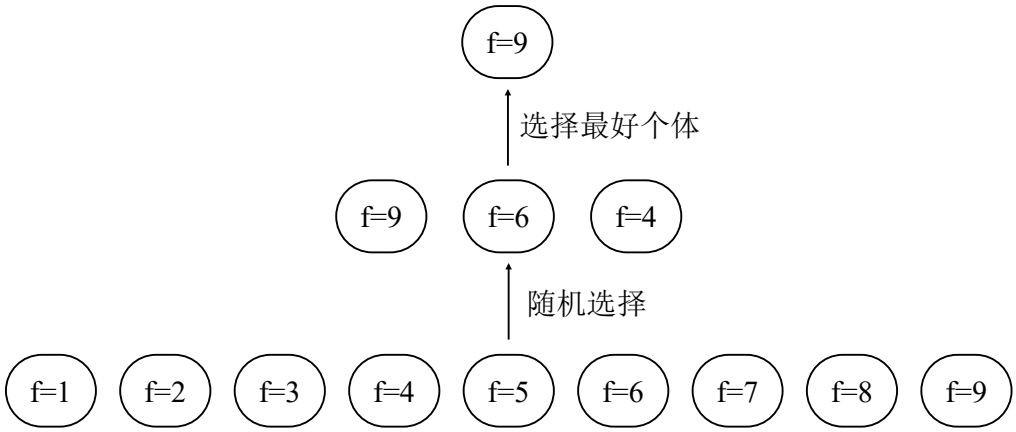


图 2-2 三锦标的选择过程，图中的 f 表示适应度值

在锦标赛选择[63]中，一个重要的参数就是竞标数量 s。锦标赛选择就是在锦标数量 s 中提供了选择压力。在 s 锦标中被选择出来的优胜者往往都是这 s 个个体中性能最好的一个，随后优胜者被放入到交配池中。通过迭代，使用锦标赛选择机制选择出来的种群平均适应度值往往比原始种群的平均适应度值要好。在锦标赛选择机制中，可以通过增大锦标数量 s 来增大选择的压力。具体来说，随着锦标数量 s 的增大，选择出来的优胜个体的适应度值会更好。

上图 2-2 就是一个简单的三锦标的选择过程。首先，从种群中随机选择出来 3 个个体，然后再对其进行排序并从中选择适应度值最好的一个作为优胜者。通过锦标赛选择的规则可以看出，选择出来的个体不一定是最好的个体，但是在锦标的过程中可以选择了最好的个体作为优胜者，所以从整体来看这个优胜者起码不是整个种群中最差的个体。从中可以认为，随着锦标数量 s 的增大，选择出种群中适应度值最高个体的可能性就越大，当锦标数量等于种群数量时，选择出来的即为种群中适应度值最优的个体。在使用中，一般将锦标数量设置为 2 或者 3，也就是二锦标和三锦标。

（2）轮盘赌选择（Roulette Wheel Selection）

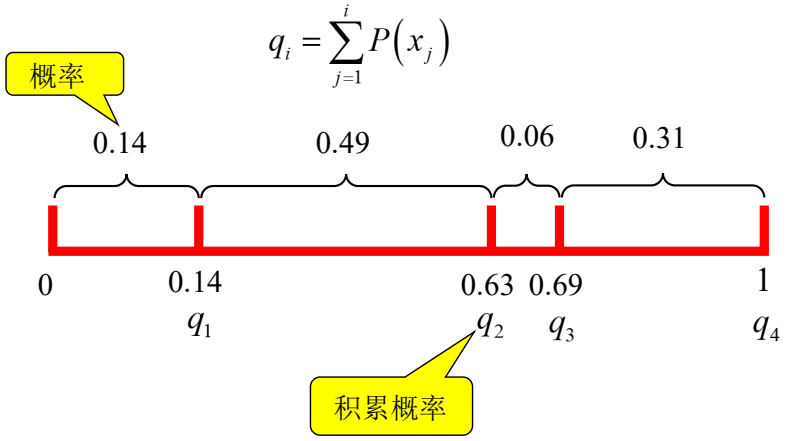


图 2-3 轮盘赌选择算法的例子

轮盘赌选择[64]又可以叫做比例选择，因为其主要思想就是根据个体的不同适应

度值将对应个体设置为不同的选择比例。在轮盘赌选择机制下种群中个体的适应度值越大则该个体被选择的机率就越大，如果已知种群 A 中个体 a_1 的适应度值为 $f(a_1)$，个体 a_2 的适应度值为 $f(a_2)$，$a_3 \cdots a_n$ 个体的适应度值分部为 $f(a_3) \cdots f(a_n)$，其生成的概率分别为 $p_s(a_1), p_s(a_2) \cdots p_s(a_n)$ 的和为 1。所以个体 j 被选择的概率为：

$$p_s(a_j) = \frac{f(a_j)}{\sum_{i=1}^{n} f(a_i)}, j = 1,2,...,n \tag{2-1}$$

被选择的积累概率为：

$$q_s(a_j) = \sum_{i=1}^{j} p_s(a_j) \tag{2-2}$$

在轮盘赌算法中所有个体被选择的概率和为 1，所以实际操作如图 2-3 所示。首先计算出每个个体的积累概率 q，然后产生一个 0 到 1 的随机数 p，通过对比 p 和积累概率，若 $p \le q_s(a_1)$，则选择个体 a_1，否则选择个体 a_k，使得：$q_s(a_{k-1}) < p \le q_s(a_k)$ 成立。

2.1.2 变异和交叉

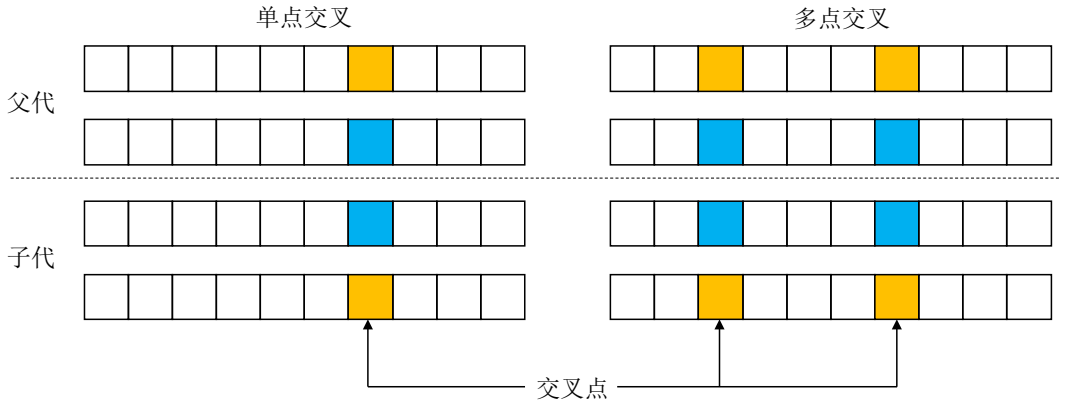


图 2-4 单点交叉两点交叉

在进化算法中往往通过变异和交叉两种方式产生后代。一般情况下，交叉是两个个体相互交换其基因片段，这个过程是参考了自然界中的雌性和雄性的交配行为，通过这种交换基因片段组合出新的基因组合个体。相对于交叉方式中需要两个个体参与，变异方式只需要选择出一个个体即可。具体来说，变异操作是将其自身的某些基因片段发生改变，这种方式有点类似于生物个体受到外界因素而产生的基因突变，这种突变的方向是不确定的，有可能会变得越来越差，因此在使用中一般会将突变的概率和突变的长度设置尽量小。

如上图所示就是常用的两种交叉例子，单点交叉（Single-point Crossover）和两点交叉（Double-point Crossover）。从上图 2-4 中可以看出，在交叉是由两个父代产生

两个子代。交叉操作的第一步就是随机产生交叉点，如果是单点交叉则只需要一个交叉点，如果是两点交叉则需要两个交叉点。在确定好交叉点后再将两个父代中对应的基因片段进行交完，最后产生了两个新的子代。当然，除了以上两种交叉方式以外，也有其它多种交叉方法，如均匀交叉（Uniform Crossover）、半均匀交叉（Half Uniform Crossover）和三亲杂交（Three Parents Crossover）。

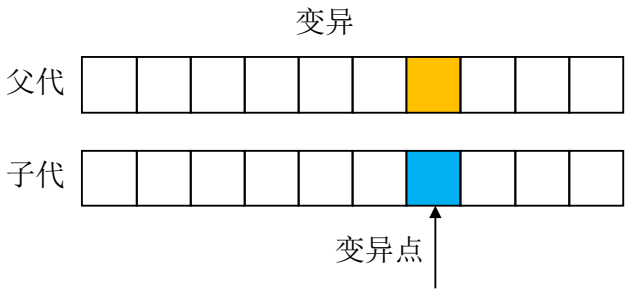


图 2-5 变异操作

如上图 2-5 为变异的一个例子，在变异操作中只需要选择一个父代，然后随机选择一个变异点将变异点的编码随机变异为其他编码。在实际应用中，由于变异操作可以产生更多区分度比较大的个体，所以为了避免进化过程中种群陷入局部解，往往可以使用变异操作来保持种群的多样性。除了上面所描述的变异操作外还有其它一些变异的方法，如功率变异（Power Mutation）、均匀变异（Uniform Mutation）、非均匀变异（Non- uniform Mutation）、收缩变异（Shrink Mutation）和高斯变异（Gaussian Mutation）等。

2.2 禁忌搜索算法

禁忌搜索算法(Tabu Search Algorithm：TS) [65]是由 Glover 于 20 世纪 90 年代提出的智能启发式算法。相对于进化算法的基于适应度值的适者生存的原则，禁忌搜索是一种基于局部领域搜索的全局逐步寻优算法。和人类的搜索活动相似，禁忌建立一个存储列表（禁忌列表）来记录搜索的过程，在接下来的几个迭代中不会再访问在禁忌列表访问的数据。禁忌搜索算法通过这种方式来避免了重复性的搜索，将其搜索方向引导出局部最优解，让算法更容易求得全局最优解。

具体来说，禁忌搜索是建立在邻域结构设计上的，将给定的初始化设为当前解，然后在当前解的基础上产生出邻域解并同样计算其适应度值。若邻域中存在比当前解更优的解集且该解集或者该解满足藐视准则，则将更优解设为当前解。如果当前解发生了变化，则需要将当前解加入禁忌列表中，同时将其任期设置为最大，禁忌列表中的其他解的任期相应减 1 直到任期为 0。

禁忌搜索的一些基本概念：

（1）初始解和候选解：现阶段绝大多数算法的初始解集都是随机产生的，也可以使用部分人工的解集。候选解集一般是邻域空间中的适应度值较高的解组成。

（2）适应度函数：和进化算法类似，禁忌搜索算法同样需要定义一个适应度函数来对解进行评估。适应度函数的定义是禁忌搜索算法中最重要的步骤之一，适应度值的好坏直接影响算法的性能。一般认为个体适应度值越高，该解的性能越好。

（3）邻域结构：通常将产生解的准则或者方法叫做邻域结构。在禁忌搜索算法中，一般通过这种邻域结构产生新的解集。当前的算法研究中，常用的邻域结构有互换、插值、逆序等。邻域结构的设计也是禁忌搜索算法中一个重要的部分。

（4）禁忌对象：禁忌的目的是为了引入记忆特性，从而达到减少无效搜索，提高搜索效率，避免陷入局部最优解的目的。所以禁忌的对象一般为邻域结构的编码状态变量，用来表示编码状态的变化过程。例如在交换的邻域结构中，禁忌对象为每一次迭代的交换对。

（5）藐视准则：在搜索过程中会遇到一些特殊情况，例如所有候选解集都被禁忌了，或者出现了某个非常优秀的解，它的解性能比历史出现的所有解都要优秀，但是他被禁忌了。在以上这两种情况下，一般会使用藐视准则来忽视其禁忌属性并将该候选解集作为当前解集，从而避免了优秀解集的浪费。

2.3 神经网络基础

近年来，神经网络在许多重要领域取得突破性进展，如图像处理任务、机器视觉和自然语言处理等场景。在这些任务上，传统的方法大多比较复杂。如在传统的图像处理中，往往需要人工设计很多特征。但是在复杂任务上，如目标检测，行人重识别等领域上，由于需要的特征比较复杂，人工设计的特征往往满足不了现有要求。而神经网络可以自动地提取有用特征，并可以高效表达特征，减少人工的参与，可以大大提高了任务效率和性能。神经网络因其具有强大的特征学习能力，在未来可以大大地提高生产力，因此具有重要的研究价值。

2.3.1 卷积神经网络

为了解决前馈神经网络[66,67]中的层与层之间的全连接而造成参数量多大的问题，学者们提出了卷积神经网络。在设计上，卷积神经网络可以使用卷积核的局部连接来取代全连接，同时这种局部连接的方式很符合生物学中的稀疏响应特性，可以让网络模型的性能更好。

从生物学的角度看，20 世纪 60 年代 Hubel 和 Wiesel[68]等人在对猫的视觉皮层细胞的研究中提出了感受野的概念。随后日本学者[69]在此基础上提出了神经认知机，这可以看作是卷积网络的首次实现。相对于全连接的前馈网络，卷积网络对图像等二维

数据更加敏感，同时使用了权重共享的机制是的卷积神经参数相对更少。卷积神经网络是早期在商业应用中表现最好的深度网络模型之一，而且其研究也是当前人工智能的关键研究热点，特别是卷积神经网络的架构搜索。

2.3.2.1 卷积操作

相比于前馈全连接网络，卷积神经网络的卷积层采用了卷积的这一数学运算的规则来处理连接的参数。在卷积层中需要学习的参数一般包括了卷积核（Convolution Kernel）和偏置参数（Bias），类比于传统的图像处理方法的滤波器，一个卷积核就是一个提取特征的算子，所以在实际应用中一个卷积层里往往需要多个卷积核来同时提取抽象层级的不同特征。

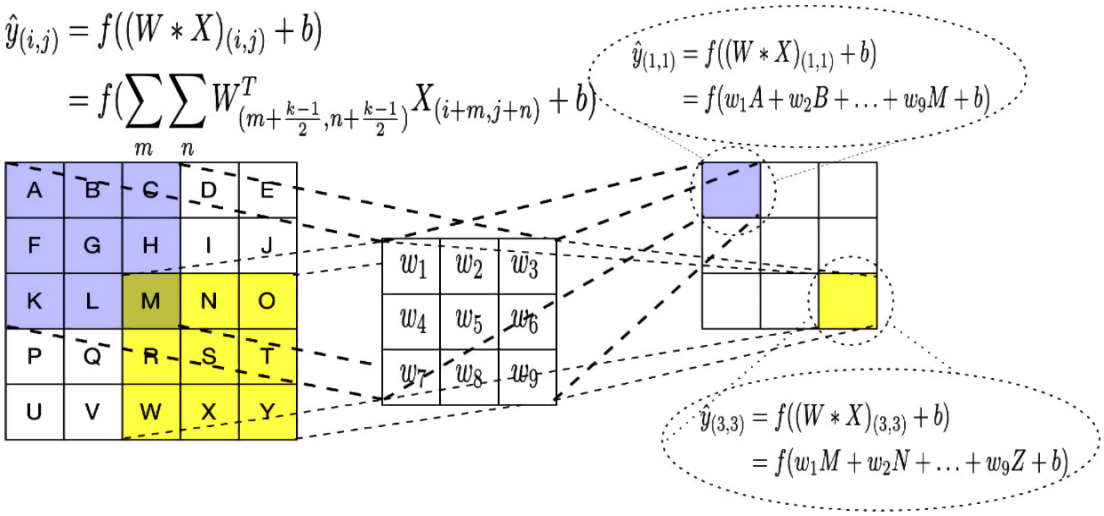


图 2-6 单通道卷积操作计算的样例（图中仅标注了第一步和最后一步的运算过程）

假设在激活函数为 f，卷积核步长为 l，无边缘补植的卷积层中，输入的特征数据为 X，需要学习的卷积核参数为 W，偏置记为 b，那么输出的特征为：

$$\hat{y}_{(i,j)} = f\left((W*X)_{(i,j)} + b\right) = f\left(\sum_m \sum_n W^T_{\left(m+\frac{k-1}{2}, n+\frac{k-1}{2}\right)} X_{(i+m,j+n)} + b\right) \tag{2-3}$$

具体的计算样例如图 2-6 所示。在实际应用中，需要学习的卷积核有多个 $W = \{W_1, W_2 \ldots W_K\}$，对应的偏置 $\beta = \{b_1, b_2 \ldots b_k\}$，那么在这个卷积层中输出的特征图就是由这 k 个卷积核和偏置组成的 k 通道特征图。

相对于全连接网络，一方面卷积操作输出特征与输入特征之间的连接关系只有卷积核大小，输出的神经元接受的视野也取决于卷积核大小、步长等参数，所以其连接满足了稀疏连接的特性。另一方面，在卷积操作中通过卷积核的上下左右移动，可以共享卷积核的权重参数，同时通过计算输出每一个位置的特征图信息，大大减少了网络的参数，避免了训练参数过多而造成的各种问题。

针对于图像特征的空间位置问题，卷积层则使用了一组卷积核对每一个局部感受

野进行了互相关的运算操作，以此得到了输出特征图中不同空间位置的特征响应。

2.3.2.2 池化操作

池化层（Pooling Layer）是卷积网络中除了卷积层外的另一种重要网络层，其主要的作用就是减少特征的空间尺寸大小，同时也可以进一步减少下一卷积层的计算量。所以，在网络设计中往往会周期性地插入池化层。

现有的研究中，常见的池化操作包括有最大池化层（Max Pooling Layer）和平均池化层（Average Pooling Layer）。具体来说，最大池化层输出了其感受野中最大的特征值，而平均池化则输出感受野内的特征平均值。另外池化层还具有特征旋转和平移的不变形，因为在使用了池化后，即使感受野内的特征平移或者旋转了，其最大池化输出的特征仍然是一致的。

2.3.2.3 全连接层

在经过了一系列的卷积操作和池化运算后，图像特征的空间尺寸就会变的越来越小，在最后需要一个全连接层将所有的特征连接起来，然后将特征输出至分类器（如，softmax 分类器）。在功能上来看，全连接层可以通过全连接将卷积或者池化后的特征与所在任务的分类相结合。

随着研究的发展，为了进一步减少全连接层的参数，也出现了使用1×1卷积层来替代全连接层。同时，使用1×1卷积层还可以输入任意尺寸，避免了像全连接层需要固定网络的结构，使得网络模型灵活性更强。

2.3.2 经典的神经网络结构

2.3.2.1 LeNet-5

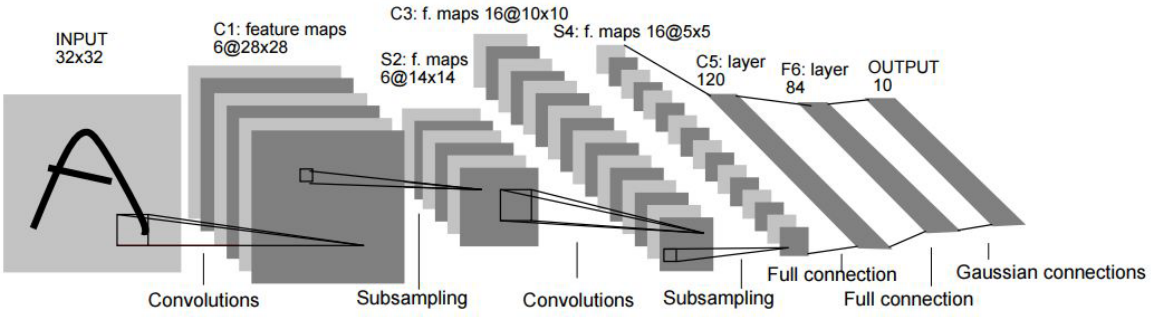


图 2-7 LeNet-5 的网络结构[70]

LeNet-5[70]是由 LeCun 等人于 1998 年提出的，用于解决手写字体字符识别的卷积神经网络，同时也是卷积神经网络在实际应用中较早的网络模型之一。如图 2-7 为一个 LetNet-5 的网络模型图，从图中可以看出网络模型主要有 7 层，输入为单通道的 32×32 的图像，输出为 10 个类别的分类信息。这 7 个网络层分别是：

（1）卷积层 C1：C1 层为网络的第一个卷积层，在该层使用了 5×5 的卷积核，步

长设为 1，无填充。通过 C1 层得到 6 个大小为 28×28 的特征图。

（2）下采样层 S2：S2 层为池化层，在这里使用了 2×2 的平均池化，步长设置为 2，经过池化后还使用了激活函数 sigmoid 非线性输出。经过 S2 后得到 6 个14×14 的特征图。

（3）卷积层 C3：C3 为网络的第二个卷积层，在该层中同样使用了 5×5 的卷积核，最后得到了 16 个10×10 的特征图。

（4）下采样层 S4：同 S2 层类似，最后得到了 16 个 5×5 的特征图。

（5）卷积层 C5：由于经过 S4 层后，输出为5×5 的特征图，在 C5 同样使用了 5×5 的卷积核。经过运算，全连接生成了 120 个特征。

（6）全连接层 F6：与 C5 层的输出全连接，生成 84 个特征。

（7）输出层：输出 10 个类别的分类信息。

综上所述，LeNet-5 网络设计与现在通用的网络设计有所不同，例如，在激活函数的选择上 LeNet-5 选择的是 sigmoid 激活函数，但是现在一般选择 tanh、ReLU 或者 leakly 较多。在池化层的选择上，现在普遍采用最大池化，而不是平局池化。但不可否认的是 LeNet-5 的设计成为了卷积神经网络的基础，打开了卷积神经网络设计的大门。

2.3.2.2 AlexNet

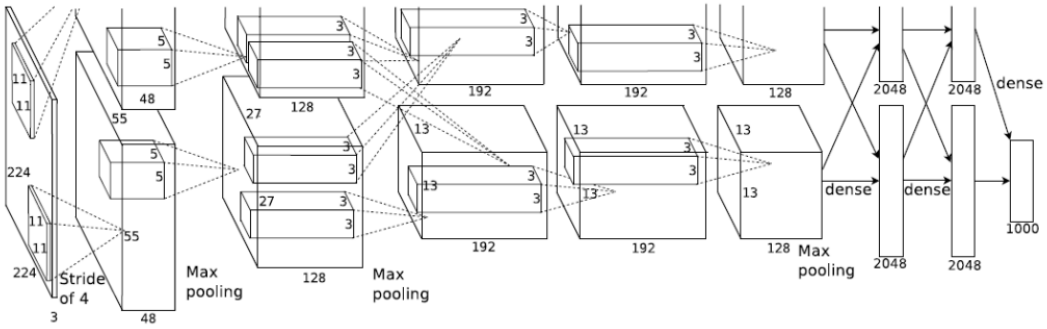


图 2-8 AlexNet 的网络结构[13]

Alex 等人于 2012 年提出经典的网络模型结构 AlexNet[13]，该网络在同年夺得了 ILSVRC-2012 图像分类大赛的冠军。AlexNet 的网络结构如图 2-8 所示，从图中可以看出该网络中卷积层共有 5 层，而全连接层则有 3 层。相较于 LeNet-5，AlexNet 采用了大量的新技术，如激活函数采用 ReLU 函数替代 sigmoid 函数，局部响应归一化（Local Response Normalization，LRN），采用 Dropout[71]思想以及使用 GPU 来加速网络参数的训练。具体来说：

（1）激活函数：AlexNet 开创性地在卷积神经网络上应用了 ReLU 作为激活函数，相对于 sigmoid 函数，ReLU 可以防止网络过深造成的梯度弥散问题，同时 ReLU 具有的稀疏激活性可以选择性地激活神经元。

（2）局部响应归一化：AlexNet 在激活函数后使用了局部响应归一化来提高网络模型的泛化能力。

（3）Dropout：采用 Dropout 以后，网络中的神经元将会以一定的概率舍弃，这在一定程度上可以避免过拟合问题，提高网络性能。

（4）使用 GPU 训练：AlexNet 使用了 GPU 来训练网络的参数，在当时由于 GPU 的计算能力受限，AlexNet 设计了分组卷积，训练时让卷积运算分布在两个不同的 GPU 上进行运算，最后再将这两个 GPU 生成的特征图进行组合成为最后的输出。

2.3.2.3 GoogLeNet

GoogLeNet[2]是 ILSVRC-2014 图像分类大赛中的冠军网络模型。在 GoogLeNet 之前，提升网络性能的方法一般可以通过增加网络的层数，但是随着网络层数的增加会出现参数过多、梯度爆炸等问题。针对这些问题，GoogLeNet 受到 Network in Network（NiN）[72]的启发，通过多路计算拓宽网络的宽度，并提出 Inception 模块，通过模块的堆叠组建成网络模型，方便增添和修改。随着网络的不断改善，Inception 模块可以分为 4 个不同的版本，分别是为 Inception v1，Inception v2，Inception v3 和 Inception v4。

（1）Inception v1：

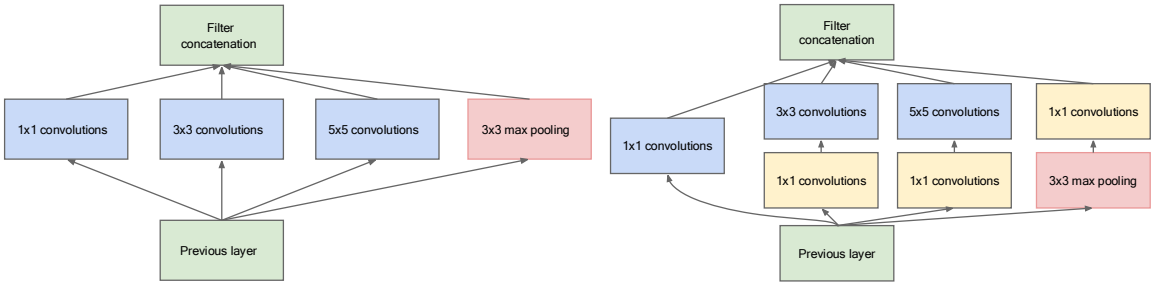


图 2-9 Inception v1 的网络结构[2]

相对于以往在卷积层只使用一种卷积操作不同，Inception v1 在同一网络层中使用了好几种不同的卷积操作，包括1×1,3×3 和5×5 的卷积操作来拓宽了网络的宽度，Inception v1 的原始版本如图 2-9 左边所示。通过这几种不同的卷积核可以同时捕捉图像不同规模的特征信息，通常来说，使用大的卷积核可以更容易捕捉到图像中的比例比较大的物体特征，反之图像中的较小目标可以选小尺寸的卷积核。但是在同一卷积层里使用了多种不同尺寸的卷积操作，不可避免地造成网络参数过多的问题，为了解决这个问题，Inception v1 还引入了1×1 卷积来对输入的特征进行映射，同时还减少网络参数。

（2）Inception v2：

Inception v2 利用了 Batch Normalization（BN）[14]来让每一层的输出都被可以归

一化到正态分布的形式，以此提高网络模型的鲁棒性，还可以以更大的学习率来对网络进行训练，加快网络的搜索速度。除此之外，Inception v2 为了进一步减少参数量，使用了两个3×3卷积核来对一个5×5卷积核进行替换，同时还增加了网络的深度。

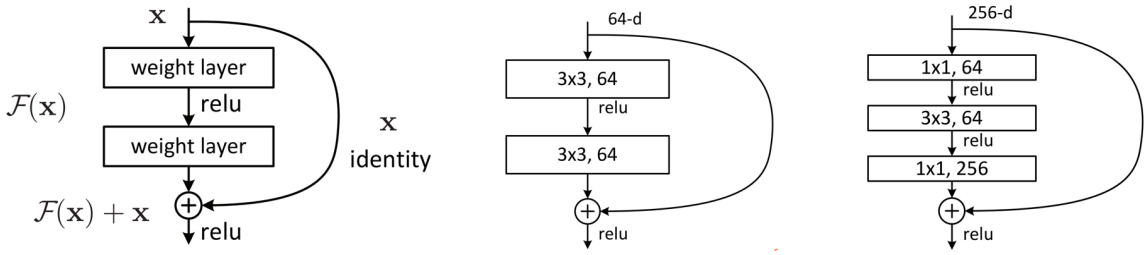
（3）Inception v3：

Inception v3[73]在 Inception v2 的基础上进一步将5×5和3×3的卷积核进行分解，其中将5×5卷积核拆分为1×5和5×1的卷积核，其中将3×3卷积核拆分为1×3和3×1的卷积核，以此来达到减少参数和提高网络深度的目的。同时 Inception v3 的优化器使用了 RMSProp，网络的输入尺寸由244×244变为了299×299，增加了输入图像的尺寸，保留了更多的原始信息。

（4）Inception v4：

Inception v4[74]主要借鉴了 ResNet 了残差模块（Residual Block），设计了 Inception-ResNet-v1 和 Inception-ResNet-v2 结构，实验证明这种设计可以在加快训练速度的同时提高网络模型精度。

2.3.2.4 ResNet



（a）残差学习单元　（b）常规残差模块　（c）瓶颈残差模块

图 2-10 残差模块[1]

为了解决网络层数加深而出现的梯度爆炸和梯度消失等问题，He 等人在 2016 年提出了 ResNet[1]网络模型。ResNet 网络结构的基本模块为残差模块（Residual Block）。残差模块的主要思想为将输入直接跳跃连接到模块的输出，如图 2-10 中 a 所示，假设输入的特征为x，模块输出为$H(x)$，则残差学习的特征为$F(x)=H(x)-x$。

ResNet 主要提出了两种残差模块，一个主要在模块中使用两个3×3的卷积核进行串联，如图 2-10 中 b 所示。另一种是把1×1，3×3和1×1卷积操作串联起来组成瓶颈残差模块（Bottleneck Module），如图 2-10 中 c 所示，在两端使用1×1卷积可以让参数更少，同时让网络层数加深，也可以提升网络模型的非线性。

2.4 进化神经网络架构搜索

通过上节可知，经典的神经网络架构都是由专业的人士进行设计，这不仅仅要求设计者拥有丰富的相关专业知识，同时也是消耗大量的人力资源。这就使得神经网络架构的自动化设计成为了一种迫切的需求。

为了应对以上难点和挑战，许多方法被提出来用以解决上述的问题。在众多方法中，基于进化算法的神经网络架构搜索（Evolutionary based Neural Architecture Search algorithms：ENAS）[75]是当前研究的主流方法之一。

基于进化算法的神经网络架构搜索流程如图 2-11 所示，从图中可以看出主要流程如下：

（1）种群初始化：种群初始化可以使用随机初始化（Random Initialization）和富有初始化（Rich Initialization）。随机初始化就是指在定义的搜索空间范围内使用随机的方法产生种群的个体。而富有初始化是指在初始化的过程中使用已有的经验或者专业人士设计的经典网络模型作为初始化个体。

（2）种群选择：从种群中根据适应度值挑选出优秀的个体。

（3）繁殖操作：可以使用变异操作和交叉操作产生后代，在这个过程中需要。

（4）网络模型训练操作：在这个过程中需要将上一步繁殖操作得到的个体先进行译码操作，将网络编码翻译为神经网络模型。然后再在训练集上进行训练。通常来说这个过程是整个流程中最为耗时的。

（5）计算适应度操作：在这个过程需要先将神经网络模型在训练集上进行验证，然后在验证集上进行验证测试，并把验证准确率作为适应度值。

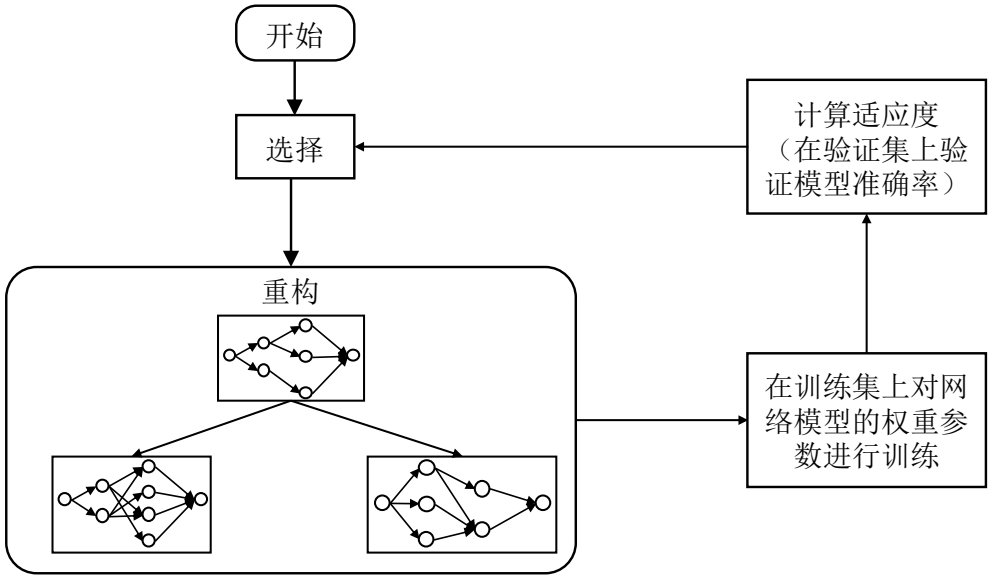


图 2-11 基于进化算法的神经网络架构搜索通用流程图

2.3.1 基于 block-level 的进化

Genetic CNN[38]是一种基于 block-level 的进化的，可以自动学习如何卷积神经网络的算法之一。该方法主要是以进化算法中的遗传算法（Genetic Algorithm）[76]作为主导，将遗传算法应用到卷积神经网络的架构搜索中，从而学习和构建最优的卷积神经网络架构。

首先，为了不让所构建的神经网络无限地发展，该方法约束了网络的最大层数，这些层（Stage）是由池化层为边界进行划分的，并且每一层都包含一系列预定义的模块（Block）。通常来说，这些预定义模块都是常用的卷积操作和池化操作。接着，该方法还提出了一种新编码方式将整个网络模型编码为固定长的二进制编码。最后再利用遗传算法对在搜索空间内进行优化。

在 Genetic CNN 中对网络模型和编码做出了一下的限制：

（1）该方法将网络结构分为不同的层（模块），每个模块由相同的卷积操作组成，在这里将卷积操作称为“节点”（Node）。在操作过程中输出的特征图的维度大小不变。

（2）每个模块的前后都有一个默认的节点相连接，这样设置的目的是为了让模块里的每个节点之间的连接都具有意义。

（3）每两个相邻的层之间固定使用池化层进行相连接，这也是卷积神经网络中常用的连接方式。

（4）允许层中的节点孤立，这表示模块内的节点不与其他节点相连接。同时也意味着这个层是空的。在这种情况下，需要将模块内默认的前后两个节点融合为一个。

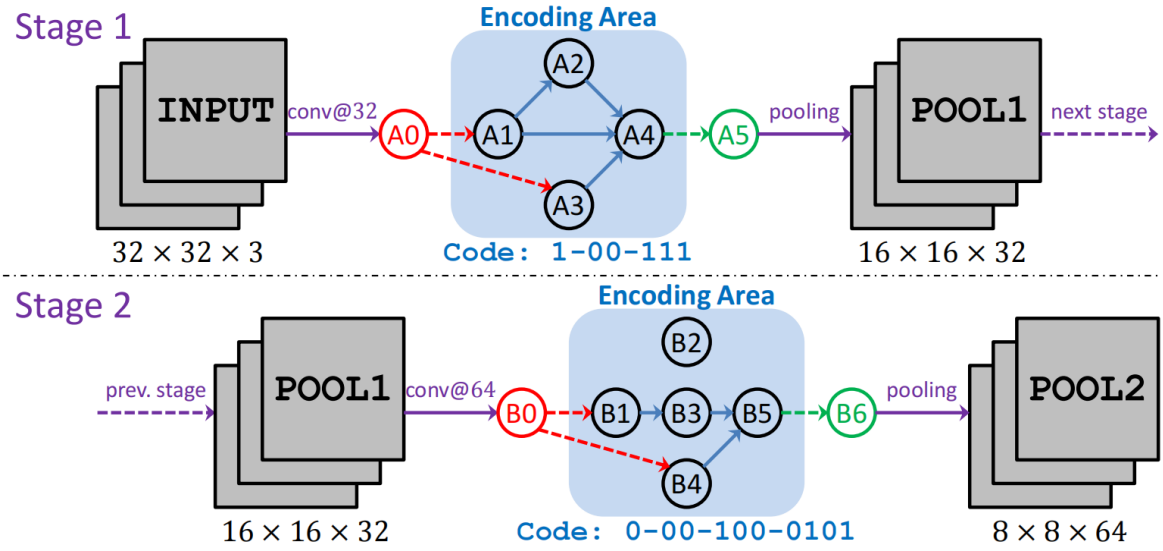


图 2-12 Genetic CNN 的编码方式，其中红色虚线表示默认的输入连接，绿色虚线表示默认的输出连接[38]

Genetic CNN 的编码方式如图 2-12 所示。该图展示了 2 层网络的编码，在每一层中都默认有两个节点，分别为该层的输入节点和输出节点，如第一层的 A0 和 A5 节点；第二层的 B0 和 B6 节点。对于默认输入节点来说，其主要接收来自上一层的输出并执行卷积操作。对于默认输出节点来说，其主要接收前面经过卷积操作后的数据并进行求和，最后再进行卷积操作。

在一层中，除了默认的两个节点外，其他的节点被定义为普通节点，也就是需要被编码的节点。在一层中所有的节点都具有唯一的有序编号，每一个节点代表一个对

应的卷积操作。为了满足卷积神经网络的运算约束，规定了在同一层中的所有卷积操作都设置相同的卷积核大小和通道数量。

对于图 2-12 所示的网络中的编码具体。超参数设置为$S=2$，$K_1=4$，$K_2=5$，其中S表示整个网络的层数，K_i表示第i层的普通节点的数量。对于节点连接之间的逻辑编码，该论文定义了 0 表示无连接，1 表示有连接。所以在第一层中$K_1=4$则需要 6 个节点来表示编码，具体来说编码顺序为：

（1）第一层：1 和 2；1 和 3、2 和 3；1 和 4、2 和 4、3 和 4。

（2）第二层：1 和 2；1 和 3、2 和 3；1 和 4、2 和 4、3 和 4；1 和 5、2 和 5、3 和 5、4 和 5。

按照上述的规则，对应到图 2-12 中的编码为：

（1）第一层：1-00-111。

（2）第二层：0-10-000-0011。

Genetic CNN 所采用的遗传算法步骤如下：

（1）初始化：该工作先固定长度为L的编码长度，然后使用随机的方法产生种群的个体，每个编码字节都采用伯努利分布进行独立的采样。

（2）选择：采用随机选择的方法选择个体，通过计算每一个个体的适应度值并将其作为选择概率，适应度值越高的个体被选择概率越大。

（3）变异：每个个体根据一定的概率改变自身结构。通过变异操作，可以减少进化过程中陷入局部最优解的概率。

（4）交叉：可以让相邻的个体互换特定的层结构。

（5）评估：将网络结构在验证集上的准确率作为个体的适应度值。

总的来说，在该工作中使用的约束机制虽然可以减少了搜索空间的范围，但是也带来了一定的局限性。例如，固定每个节点的卷积操作，设置相同的卷积核大小和通道数。这在某种层度上限制了网络模型的初始阶段信息。

2.3.2 基于 node-level 的进化

在上一节的 Genetic CNN 中提到，由于固定了网络的结构，给网络的搜索带了许多限制。针对这个问题，在基于 node-level 中的大规模演化的图像分类器（Large-Scale Evolution of Image Classifiers）[77]就放松了对网络的限制，减少在网络模型的设计上的人为参与。即采用最简单的无卷积的网络做出初始化，同时在网络模型上不对网络深度和跳连进行限制，让网络随意进化。该工作使用进化算法从一个最简单的网络架构开始进化，通过预先设定的不同变异方式对原始个体进行变异优化。同时为了加快进化的过程，该方法采用了网络权重继承的方法来加快训练的速度。

对于种群初始化操作，在该工作中将种群大小设置为 1000，同时种群中所有个

体统一采用如图 2-13 所示的网络模型作为初始化。

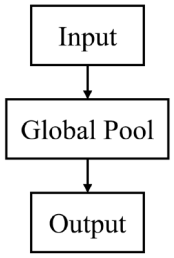


图 2-13 初始化个体[77]。

对于选择操作，该工作采用了经典的二锦标选择的方式，从种群中随机抽取两个个体，将适应度好的个体进行复制以此作为产生子代的父代个体，同时将性能差的个体从种群中剔除。

在变异算子的设置上，该工作设置了 11 种变异算子，具体为：

（1）改变学习率。

（2）网络架构保持不变，继续训练。

（3）重新设置网络权重。

（4）插入卷积操作（卷积核大小设置为 3x3；将步长随机设置为 1 或者 2；卷积的通道数和输入通道数一致；最后再随机选择一个位置上，然后在随机决定是否加上 batch normalization 和 ReLU 激活函数）。

（5）移除卷积层。

（6）改变卷积核的步长（只能设置为 2 的次幂）。

（7）改变卷积的通道数（随机选择网络中的某个卷积）。

（8）改变卷积核的大小（随机选择网络中的某个卷积，然后随机改变水平或者垂直方向上的大小，其中限制必须为奇数）。

（9）添加跳连操作。

（10）移除跳连操作。

（11）插入一对一层(卷积核大小1×1，步长随机 1 或 2，通道数和输入通道数一样，最后在随机决定是否加上 Batch Normalization 和 ReLU 激活函数)。

在适应度值的计算上，该工作引入了网络权重继承机制，新产生的子代将从父代中继承部分网络权重参数。具体来说，如果某一层的尺度完全相同，那么父代的这一层权重参数将被保留。所以对于上述定义的一些变异算子，一些可以完全保留父代权重参数（例如，改变学习率和网络架构不变，继续训练的变异算子）；另一些则完全不可能保留父代权重（例如重新设置权重的变异算子）；还有一些则可以保留部分权重（例如，变异卷积核大小的操作）。

进化过程图如 2-14 所示。图中灰色部分表示在搜索过程中被淘汰的个体，而蓝

色部分则表示在搜索过程中保留下来的个体。从图中的采样点可知，初始化的个体比较简单，进化过程中网络架构越来越复杂。

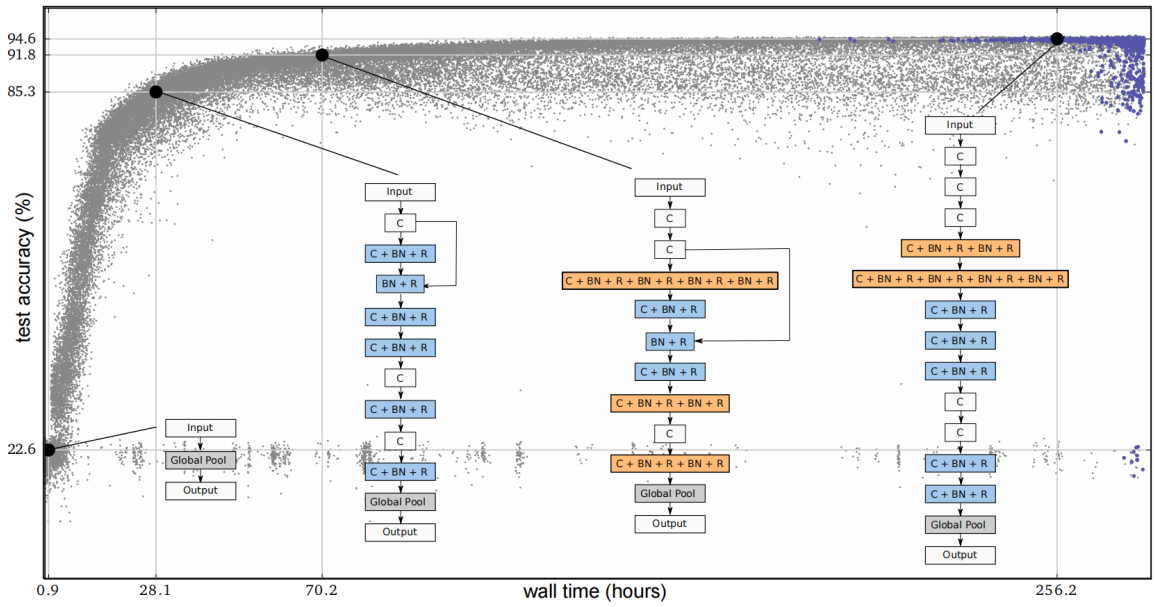


图 2-14 进化过程图[77]

在大规模演化分类器的这个工作中，作者减少人工的干预，通过大规模进化将一个简单网络进化为一个复杂的网络，这充分表明了进化算法的有效性。同时其在公开数据集 CIFAR-10 上也达到了 7.10%的错误率，但是由于该工作需要大量的显卡进行计算，一般个人或者机构无法很好对该工作进行复现。

2.3.3 基于 cell-level 的进化

上节提到的大规模演化分类器减少了人工知识的引入。具体来说，在种群初始化中采用了相对简单的网络结构，通过这种设置可以引导算法在进化过程中生产性能更佳的网络结构。那么引入更多先验知识的方法会不会让进化出来的效果更加理想呢？由此，谷歌提出了 AmoebaNet[42]，该方法基于 cell 的搜索空间，在 cell 内部采用了类似残差模块的跳连结构和 Inception 模块的多路分支结构来有效提取图像的特征信息。最后在网络的构建上采用了和 GoogLeNet 和 DenseNet 类似的结构，在网络中大量重复搜索得到的 cell 结构。由此可见，AmoebaNet 在搜索空间和网络架构上都引入了比较强的先验知识。

AmoebaNet 的主要改进有：

（1）传统的进化算法通常保留性能最好的个体，而在该工作中对每个个体（网络结构）则引入了年龄标签，在进化过程中更加偏向于选择较为年轻的个体，从而保持种群的活力。

（2）为了和基于强化学习的 NASNet[24]作比较，该方法使用了和 NASNet 相同

的变异操作。每一个节点表示一个隐藏层的特征，边代表相关运算操作（卷积操作，池化操作等）。每次变异均可选是连接变异或者是操作变异。

图 2-15 是 AmoebaNet 的搜索空间，从 2-15 的左图可知网络骨干是一种基于 cell 的线性连续堆叠空间；相邻 cell 之间采用了类似残差的跳连结构，如图 2-15 中间所示。在 cell 的内部则为一个多路分支结构，如图 2-15 右图为搜索过程中的一个例子。

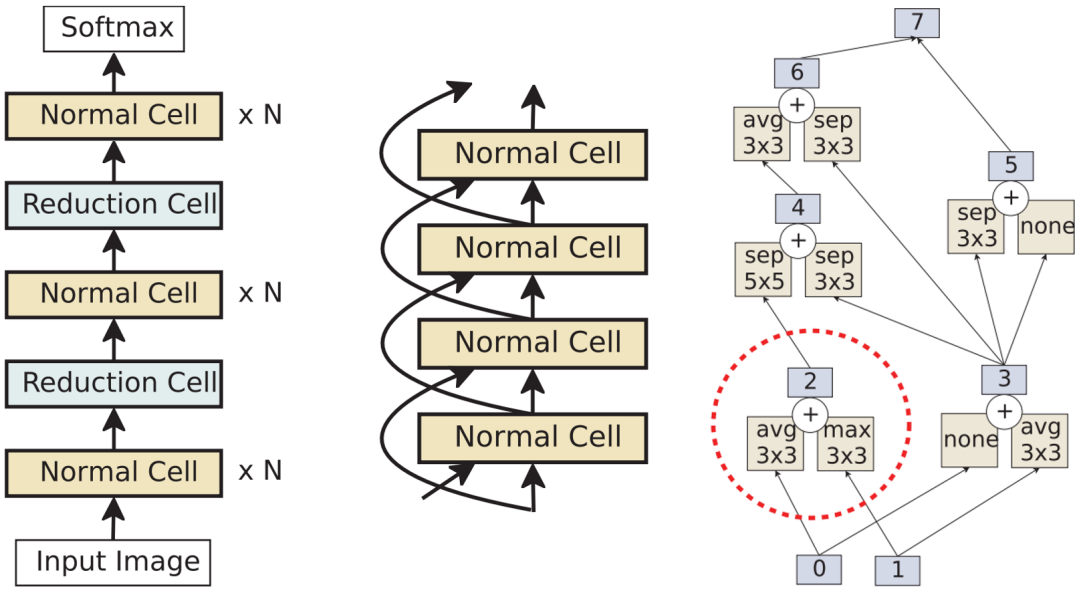


图 2-15 AmoebaNet 的网络结构和搜索空间[42]

在 AmoebaNet 的工作中，由于网络的骨干是预先设定好的，如果想要得到更大的网络结构，只能通过增加网络的深度，堆叠更多的 cell。进化过程中只有两种不同的 cell，这在一定程度上缩小了搜索空间。最后 AmoebaNet 搜索出来的网络结构在公开数据集的性能首次超越人类设计的网络。这也表明引入恰当的先验知识可以大大提高搜索速度和性能。

第 3 章 基于禁忌搜索和进化算法的神经网络架构搜索算法

大多数现有的神经结构搜索策略都没有在搜索的过程中使用记忆策略来避开已经搜索过的空间区域，这在一定程度上会造成搜索过程的冗余，降低搜索效率。为了解决这一问题，本文在基于 cell 结构的搜索空间基础上提出了一种结合了禁忌机制的进化神经网络架构搜索算法。在这个算法中首先需要根据搜索空间，结合禁忌规则，设计了一个高效的编码规则。接着，建立一个禁忌列表来保存历史中搜索过的空间区域，并对已经搜索过的区域进行合理避让，通过双种群设计让算法在尽可能地搜索更多的区域的同时保持种群的优越性。为了进一步提升搜索效率，本文还使用了替代模型来对网络的性能进行粗略的预估以便快速得到网络模型的性能。

3.1 整体网络框架设计

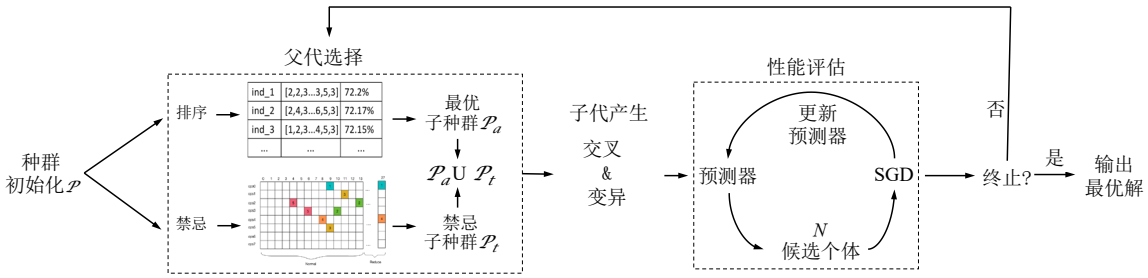


图 3-1 本文所提方法架构图

本文提出的算法框架和伪代码如图 3-1 和算法 1 所示。从图中可以看出，本文所提出算法的步骤为：首先初始化种群；然后进入了父代种群的选择环节，这个环节中将保持两个子种群，分别是最优子种群和禁忌子种群；紧接着，选择完父代种群后，本文进入子代生成环节，在这个环节中本文将采用变异和交叉的方式来产生后代种群；随后进入适应度值评估环节，同时这个环节中是整个算法流程最耗时的环节，在本文中将网络模型在验证集上的准确率作为个体的适应度值。为了提高效率，在本文中采用了替代模型来加速适应度值计算的过程。最后，将判断是否达到了终止条件，然后开启下一轮循环。

算法 3-1 本文所提算法框架

输入:

nArchive：最优子种群的大小；

nTabu：禁忌子种群的大小；

$T_{\max}$：最大代数；

$nTenure$：禁忌任期；

K：父代种群大小；

$nCan$：进化过程中的候选个体个数；

输出：最优的可行解。

1 **初始化：**

2 初始化种群 POP；初始化禁忌列表 $tabuList$；

3 根据初始化种群初始替代模型 $predictor$；

4 **while** $gen \le T_{max}$ **do**

5 从 POP 获得 $nArchive$ 个最优子种群 $archiveSubpop$；

6 从 POP 通过 $tabuList$ 获得 $nTabu$ 个最优子种群 $tabuSubpop$；

7 初始化父代代种群 P；

8 $P = getParent(archiveSubpop, tabuSubpop, K)$ //算法 3-2

9 初始化子代代代种群 Q；

10 **for** $j \leftarrow 1$ **to** K **do**

11 **foreach** $p \in P$ **do**

12 个体 p 通过变异的方式产生子代个体 q；

13 $Q := Q \cup q$

14 **end**

15 **for** $j \leftarrow 1$ **to** K **do**

16 **foreach** $q \in Q$ **do**

17 通过 $predictor$ 获得个体 q 的适应度预测值；

18 **end**

19 通过对子种群 Q 的预测适应度值排序获得前 $nCan$ 个候选个体 $C=\{c_1, c_2, ... c_{nCan}\}$；

20 **for** $j \leftarrow 1$ **to** $nCan$ **do**

21 **foreach** $c \in C$ **do**

22 使用随机梯度算法获得候选个体 c 的真实性能；

23 **end**

24 扩充种群 $POP := POP \cup C$；

25 使用候选个体 C 的信息更新 $tabuList$；使用候选个体 C 的信息更新 $predictor$；

26 $gen = gen + 1$；

27 **end**

28 **输出：**最优可行解。

3.1.1 种群初始化

在搜索开始阶段，需要对算法进行初始化，具体如算法 3-1 的 1-4 行。为了减少初始化种群的生成时间，在种群初始化的设置上，本文将使用为替代模型准备的随机样本作为初始化的种群。具体来说，本文使用二锦标的选择机制从样本中随机抽取 100 个不重复的个体，并以此作为初始化种群。通过这种方式，可以在初始化的阶段就可以让种群拥有了较好的性能。同时，由于样本是使用随机的方法产生的，在这个过程中没有引入太多的先验知识，可以让初始化种群更加多样性。

3.1.2 父代选择

从图 3-1 框架流程图中可以看出，父代种群包括了最优子种群和禁忌子种群两部分。更具体地说，如算法 3-2 所示，父代种群通过双种群协同机制来选择。种群主要由两部分组成，其中一部分类似于传统的进化算法，即通过个体的适应度值排序来选择最优子种群；另一部分则是使用了本文设计的禁忌规则来选择禁忌子种群。

通过保留两个子种群的这种方式，被选择出来的父代种群个体不仅考虑了个体适应度的性能指标，还考虑了在之前的搜索过程中是否选择了当前个体的邻域空间。通过这两个种群的相互配合，一方面可以保留了搜索过程中出现的优良个体基因，另一方面可以避让已经搜索过的邻域空间。该子种群在搜索过程中可以避免已经探索过的邻域空间，从而避免无用的搜索过程，大大提高搜索效果。

3.1.3 子代个体的产生

在本文中采用了交叉的方式产生子代，由于本文设计的编码位之间存在着约束关系，所以常规的交叉和变异方式不能直接应用。基于此，结合搜索空间和编码规则，本文采用了一种自定义的变异和交叉方式，具体如 3.5 小节所述。

3.1.4 适应度值计算

在计算适应度值的环节。为了减少搜索过程中计算适应度值的消耗时间，本文引入了替代模型来预测每一个新产生个体的性能，从而达到加速搜索过程的目的。

前人工作[38,42]中大多将产生的子代全部都使用随机梯度法来获取子代个体的适应度值，虽然这种方法可以获得所有个体准确的适应度值，但是比较消耗计算资源。在本文中，将引入替代模型来对子代种群的适应度值进行加速。具体来说，在每一次迭代的中，先使用替代模型来对所有产生的子代进行性能预测，然后再取预测性能较好的前几个网络模型作为候选子代并使用随机梯度法进行训练，得到其真实的性能，最后将这几个候选个体加入到种群中并以此作为新样本用以更新替代模型的参数。

3.2 搜索空间和网络编码设计

3.2.1 搜索空间

搜索空间的设计是神经网络架构搜索中的一个重要过程，一个好的搜索空间可以大大提高搜索效率和搜索结果。在当前的研究中，常用的搜索空间有链式的搜索空间、分支结构的搜索空间和基于 cell 结构的搜索空间。经过综合对比，在本文中采用了基于 cell 结构的搜索空间，因为在该搜索空间下只需要搜索出 normal cell 和 reduction cell，然后通过一定的规则进行堆叠就可以形成一个性能比较好的神经网络。相对其他结构来说，这种结构更加容易编码且对资源要求相对较少。

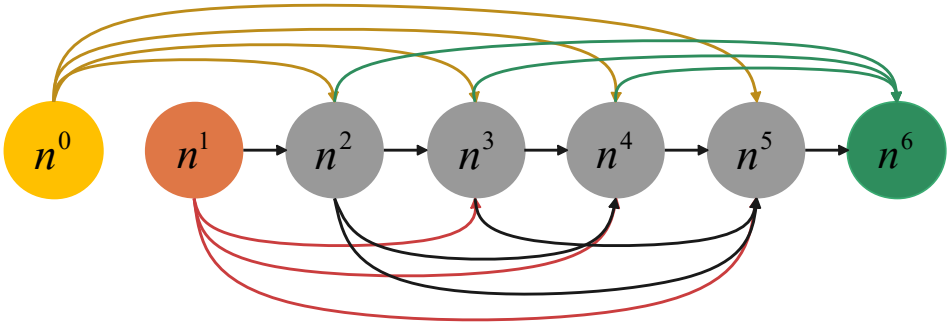


图 3-2 cell 内部的连接方式

在搜索过程中，normal cell 和 reduction cell 的内部结构是一样的。对于 normal cell 来说，其输出的特征图维度和输入的是一致的，在每一个 normal cell 内部只做了特征的运算和提取，没有特征的降维。而在 reduction cell 中，该 cell 主要的任务是做了特征的降维，主要技巧是将步长（stride）设为 2，让其输出的特征图的长宽变为输入的一半，以此来达到特征降维和提取高级特征的目的。

如图 3-2 是一个 cell 中节点的连接情况，在每一个 cell 中都是由 n 个有序节点组成的有向无环图。对于每一个节点 n^i 来说，其代表的是每一次计算后的特征图（feature map）。对于每一条边 (i,j) 表示从节点 n^i 到节点 n^j 的运算操作 $o^{(i,j)}$，在实际中每一条边代表的是具体的操作运算，如卷积操作、池化操作和跳连操作等。本文规定了在每一个 cell 内部中，前两个节点是该 cell 的数据输入点，最后一个节点是该 cell 的数据输出点，其余的节点为 cell 的内部节点。一般来说，在 cell 中的数据输入点都是前两个 cell 的数据输出点。对于 cell 里面的内部点来说，每一节点的输入都可以由其驱节点计算而来，具体计算公式为。

$$n^{(j)} = \sum_{i<j} o^{(i,j)}(n^{(i)}) \tag{3-1}$$

参考了其他先进方法的搜索空间设计，本文同时规定了每一个节点只能连接两个前驱节点，运算为对所选前驱运算结果的 add 操作。对于输出节点来说，其运算为所

有内部节点的 concat 操作。

对于 cell 中每一条边的候选运算操作，本文参考了其他先进工作，设置了 8 种常用的候选操作，分别为：3×3 深度可分离卷积、5×5 深度可分离卷积、3×3 空洞卷积、5×5 空洞卷积、3×3 平均池化、3×3 最大池化、跳跃连接和无连接，共 8 种候选运算操作。

3.2.2 编码设计

在搜索过程中为了更好表示神经网络模型，本文采用了间接编码的形式来表示神经网络模型。在一个 cell 中，共有 7 个节点，按顺序从 n^0-n^6 对其进行编号。根据前文所述，节点 n^0 和 n^1 是输入节点，n^2-n^5 为内部节点，n^6 为输出节点。由于输入和输出的节点的连接方式已经固定，所以在本文中，只需要对内部节点 n^2-n^5 进行编码。由于在 cell 内部采用了有向无环的图结构，所以对于内部节点 n^2-n^5 依次有 2 个到和 5 个候选的前驱节点。

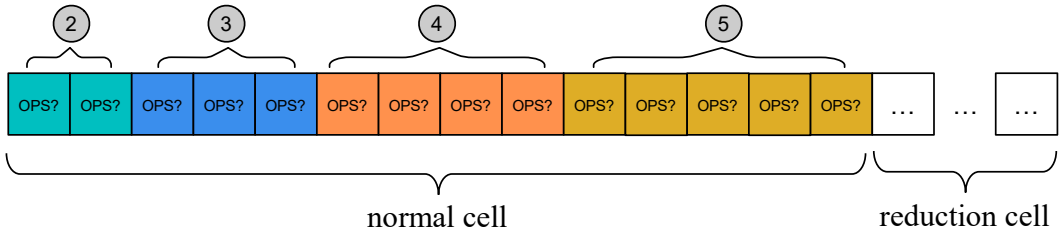


图 3-3 normal cell 和 reduction cell 的编码，图中的 OPS? 表示候选的操作运算

在编码中，本文采用位编码的形式。具体来说，对于每一个节点的前驱节点，本文都使用 1 个位来表示与其的连接，位上的数字表示连接的操作运算。如对于节点 n^2，使用 2 个位来表示其和节点 n^0 和 n^1 的连接。对于节点 n^5，则需要使用 5 个位来表示其和节点 n^0-n^4 的连接。所以，在对一个 cell 的网络模型进行编码的时候，共需要 2+3+4+5=14 个位编码。在本文中采用了一个 28 个整数的字符串来编码，每个位的范围是从 0 到 k，其中 k 为候选操作的数量，在本文中定义了 8 中候选操作，所以 k=8。如图 3-3 所示，28 个整数字符串编码分为两部分，包括 normal cell 网络结构的编码和 reduction cell 网络结构的编码。前 14 位表示 normal cell 的网络结构，最后 14 位表示 reduction cell 的网络结构，。例如，在 normal cell 编码部分中，它包含四个内部节点 n^2-n^5，每个节点的输入都来自于其前驱。节点 n^2 是图 3-3 中的一个绿色的部分，它有来自节点 n^0 和 n^1 的两个输入，节点 n^5 的候选输入为 n^0 到 n^4。

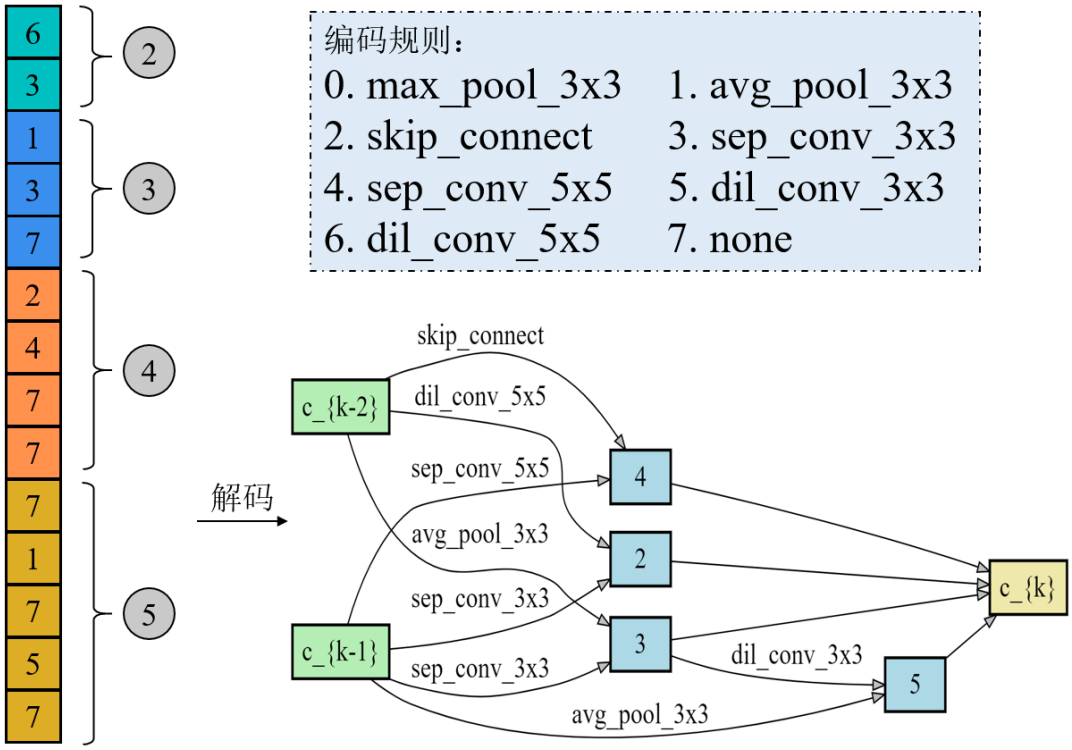


图 3-4 cell 的编码样例

上图 3-4 展示了一个 cell 编解码的实例，左边为一个 cell 的编码，右上角为具体的编码规则，右下角为对应解码后的网络图。

从图中编码可知，节点 n^2 的前驱分别连接了 n^0 和 n^1，对应的操作编码为 6 和 3，通过查找编码规则可知 $o^{(0,2)}$ 为5×5 空洞卷积，$o^{(1,2)}$ 为3×3 深度可分离卷积。同理，对于节点 n^5，通过查找编码规则，编号 7 为无操作，所以节点 n^5 同时连接 n^1 和 n^3，通过查找编码规则可知 $o^{(1,5)}$ 为3×3 平均池化，$o^{(3,5)}$ 为3×3 空洞卷积。

3.3 基于双种群协同设计的父代选择

现有的基于进化算法的神经网络架构搜索算法在搜索过程中往往只使用单种群的进化，该种群一般只考虑网络模型在验证集上的准确率这一个因素。在这种机制下随着搜索的迭代，进化的种群就会容易陷入局部解难以跳出来。在本文中采用双种群进化的机制，通过在父代选择过程中同时维持最优子种群和禁忌子种群。通过双种群的互相配合，避免陷入局部解，加快搜索效率，提高搜索性能。

在本文中，分别获得最优子种群和禁忌子种群后需要从这两个子种群中选择出产生后代的个体。如算法 3-2 所示，为了可以在搜索过程中可以随机地从两个子种群选择父代，本文设置了一个概率 p，在选择的过程中，如果当前随机概率 $c < p$，那么从最优子种群中使用锦标赛机制来选择一个个体。反之，从禁忌子种群中随机选择一个个体。通过这种概率的选择，可以保证选择出来参与变异的父代就混合了两种子种

群的信息。

算法 3-2 双种群协同

```
1 Function  pop = getParent(archiveSubpop, tabuSubpop, K)
2       父代代种群 P ← φ ;
3       for  k ← 1  to  K  do
4             if  random() < ρ  then
5                   使用锦标赛从 archiveSubpop 中选择一个个体 p ;
6             else then
7                   使用锦标赛从 tabuSubpop 中选择一个个体 p ;
8             end
9             P := P ∪ p ;
10      end
11      return  P ;
12 end
```

3.4 禁忌规则的设计

3.4.1 禁忌对象

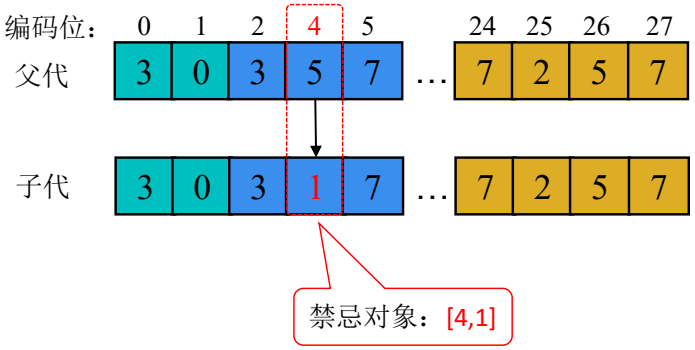


图 3-5 单个禁忌对象样例

禁忌对象指的是在禁忌表上被禁忌的那些变化元素，也可以理解为是从父代个体变成子代个体需要改变的对象。结合编码设计，本文将禁忌对象设置为编码串上的位与其对应的操作。具体来说，编码串上共有 28 位，每一位上共有 8 中选择，那么在本文中共有 28×8=224 个禁忌对象。禁忌对象的样例，如图 3-5 所示，图中展示了单个点的禁忌对象例子。从图中红色部分可以看出，子代生成的过程是将父代的 4 号编码位变成了 1，所以这一次变化的禁忌对象就是[4, 1]。如果从父代到子代的变化中出

现超过两个及两个以上的变化位置，则禁忌对象为所有变化位的并集。

3.4.2 禁忌子种群的选择

在本文的设计中，禁忌子种群的选择优先考虑的是候选个体的禁忌对象是否在禁忌列表里面。如果在，则不选择该个体，除非该个体的适应度值可以达到当前最优；否则继续挑选下一个体。

禁忌子种群的选择详细过程如算法 3-3 所示。具体步骤如下：

第一步，根据每个产生的个体适应度值对当前种群的个体进行从高到低的性能排序。

第二步，如果发现当前个体的适应度值要高于目前为止发现的最好性能，这就证明在该个体中出现了某些新的基因片段导致性能的提升，为了保留出现的良好基因片段，本文将忽略所有的禁忌规则，直接将该个体选择到禁忌种群中；如果该个体的性能表现一般，那么接下来将查询禁忌列表，如果该个体的禁忌对象不在当前的禁忌列表中，那么证明产生该个体的编码位和对应的操作已经有一段时间未被访问了，为了探索该禁忌对象的潜在可能，则会将该个体选入到禁忌子种群中；如果该个体的禁忌对象还在列表中，则认为在不久前的搜索中就访问过该对象，为了探索更广的搜索空间，将舍弃该个体，转而挑选下一个对象直到找到符合要求的子种群。

算法 3-3 禁忌子种群的选择

```
1 Function tabuSubpop=TabuSelection(POP,nTabu,tabuList)
2     初始化禁忌子种群 tabuSubpop ← ∅；
3     初始化种群索引 j ← 0；
4     对种群 POP 进行性能排序；
5     for i ← 1 to nTabu do
6         while true do
7             获取 POP 第 j 个体 p；
8             if 个体 p 的适应值为当前最优 or p 的禁忌对象不在 tabuList 中 then
9                 break;
10            end
11            else then
12                种群索引自增 j := j + 1;
13            end
14        tabuSubpop := tabuSubpop ∪ p；
```

15 **end**

16 return *tabuSubpop*；

17 **end**

3.4.3 禁忌列表和禁忌任期

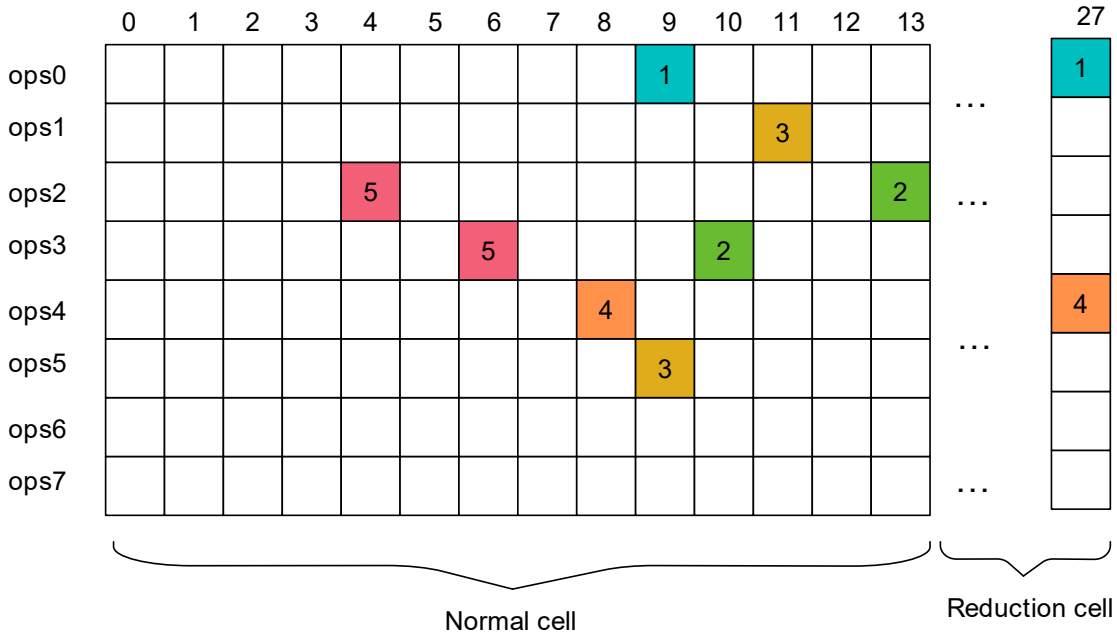


图 3-6 禁忌列表的一个样例

禁忌搜索本质是一种短期的记忆搜索机制，而禁忌列表则是这短期机制的实现，禁忌列表是禁忌搜索算法中的一个重要部分。禁忌列表是一个二维存储结构，在这个存储结构中包含了所有可能的禁忌对象。在搜索过程中，禁忌列表保存了在搜索过程中最近几次搜索过的区域。

禁忌列表中二维网格的数字代表的是禁忌的任期时长，也叫禁忌长度，指的是禁忌对象不能被选取的周期。如果禁忌期限设置过短，则容易出现循环，跳不出局部最优；如果设置过长，则会造成计算时间过长。禁忌任期时长一般为整数，假设某个对象的任期被设置为 n 则表示该禁忌对象在 n 轮的迭代次数内不会再次被搜索到。每一个禁忌对象的禁忌任期会随着迭代的进行而逐渐减少，直到减少至 0。

基于此，如上图 3-6 所示为一个搜索过程中的一个禁忌列表样例。上图禁忌表的横坐标表示 28 位的编码，纵坐标表示的是 8 个运算操作编码，中间的网格点数字表示的第 i 个编码位下的第 j 个操作的任期为 n，也就是说在往后的 n 轮迭代中不能出现第 i 个位下第 j 中操作这种禁忌结构的新个体（除非新产生个体可以打破禁忌规则）。以图中最左边粉红色方框中的数字 5 为例，该位置当前迭代中编码位 4 的操作 2 的禁忌任期为 5。

3.5 子代生成设计

3.5.1 变异操作设计

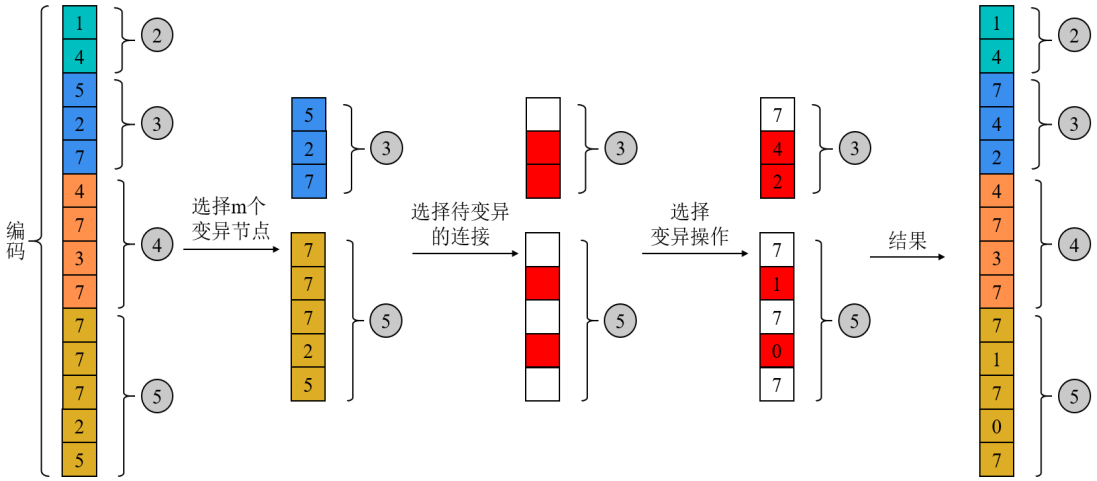


图 3-7 变异过程样例

在本文中将应用变异操作从父代中生成新的网络结构。本文将变异过程分为三步。第一步，从 28 位的编码的 8 个节点中（normal cell 和 reduction cell 各 4 个节点）随机抽取 m 个节点（$m \le 8$），并将选择出来的点叫做变异节点。第二步，在所选的变异节点上随机选择其两个前驱的连接边（由于规定了每个节点只能有两个前驱，对于部分节点，如 normal cell 中的节点 n^3， n^4， n^5 可能有超过 2 个候选前驱）。第三步，在所选的前驱上随机选择一种操作运算。

如图 3-7 为变异操作的一个样例，从图中可知第一步选择的变异节点为 n^3 和 n^5；随后在第二步中分别对节点 n^3 和 n^5 选择节点 n^1、n^2 和节点 n^1、n^3 连接的边；第三步再选择对应的运算操作编号，上图中分别将 $o^{(1,3)}=4$、$o^{(2,3)}=2$，$o^{(1,5)}=1$、$o^{(3,5)}=0$。值得注意的是，在第三步中由于部分节点的候选节点大于 2 的时候，需要将没被选中的边的操作赋值为“无操作”。如上图中的节点 n^3，将 $o^{(0,3)}=7$；如节点 n^5，将 $o^{(0,5)}=7$，$o^{(2,5)}=7$， $o^{(4,5)}=7$，其中编号“7”表示“无操作”。

在变异的过程中，本文发现如果对所有候选个体都采用同一种变异的变异长度（每一个个体变异的节点数都一致），那么变异产生的新个体有很大几率会变得更差。在优化问题中，探索与开发（Exploration and Exploitation）[78]是一个经典的问题，一方面希望在有限的时间内探索更多的空间，找到更多的可能。但是另一方面，也希望可以在已经探索过的区域进行开发，以便在这个区域内找到更好的解。基于此，本文采用了可控的变异长度机制，不再采用统一的变异长度，而是根据所选父代的在验证集上的精确率来调节变异的长度，一般将性能好的个体变异更少的节点以便进行在该个体附近进行开发，性能差的个体变异较多的节点以便探索更多的可能区域。

3.5.2 交叉操作设计

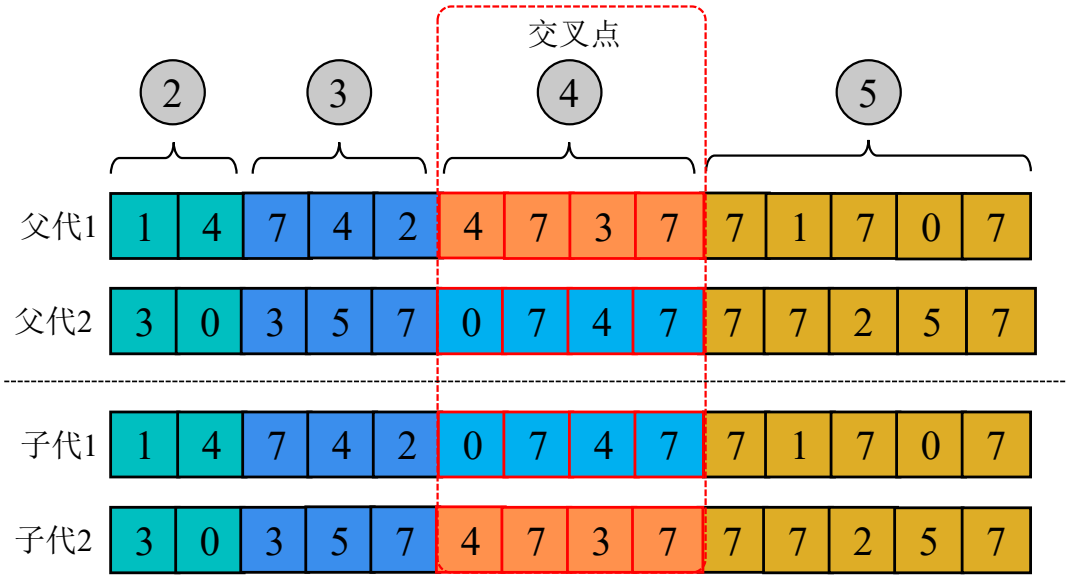


图 3-8 交叉过程样例

在本文中同样使用交叉的方法从父代中产生子代。传统的交叉方式，如 2.1.2 节所述是基于编码位来进行交叉的，可以在编码中任意选择交叉点，然后互换交叉点的内容。然而这种交叉方式只适用于每一个编码位都相互独立的任务，在本文的编码设计中根据搜索空间，每个编码位之间可能相互关联和约束，例如对于编码节点 n^3 来说，其编码为上图 3-8 蓝色部分，具体编码位为 2-4（编码位从 0 开始）。在同一组编码节点中的编码位要满足相关的约束，例如在一组节点内只允许与至多两个前驱相连接，其他不相连的边对应的编码位需要设置为 7（无操作）。所以，在本文中如果直接使用常规的以位为基础的单点交叉，那么交叉后的子代就有可能无法满足上述所说的约束条件了。

针对本文所使用的搜索空间和编码规则，本文采用基于节点的交叉规则。具体来说在搜索过程中，在交叉前先使用锦标赛选择出两个不同的个体作为父代；紧接着使用随机的方法从 8 个节点（4 个 normal cell 节点和 4 个 reduction cell 节点）的编码中选择出交叉点；最后，交换父代个体中交叉点部分的编码。具体过程如图 3-8 所示，其展示了一个 cell 中单点交叉的过程，在图中交叉点为 n^4 节点，通过交换父代个体 1 和 2 的节点 n^4 相关的全部编码，交换后的结果如上图 3-8 下方所示。

3.6 替代模型的设计

在评估神经结构的性能时，大多数现有的方法都使用反向传播算法从零开始训练网络至收敛。然而，这种做法通常是很耗时的，在早期神经网络架构搜索的工作中往往需要消耗几千个 GPU DAY 才能完成一次搜索的过程。而使用替代模型的方法往往

可以在很短的时间内获得一个网络模型的近似准确率，虽然这准确率一般情况下是不太准确的，但是在这个阶段通常只需要获得一个近似的网络性能并有所区分度即可。常用的替代模型有径向基函数、多层感知机模型和高斯函数等。

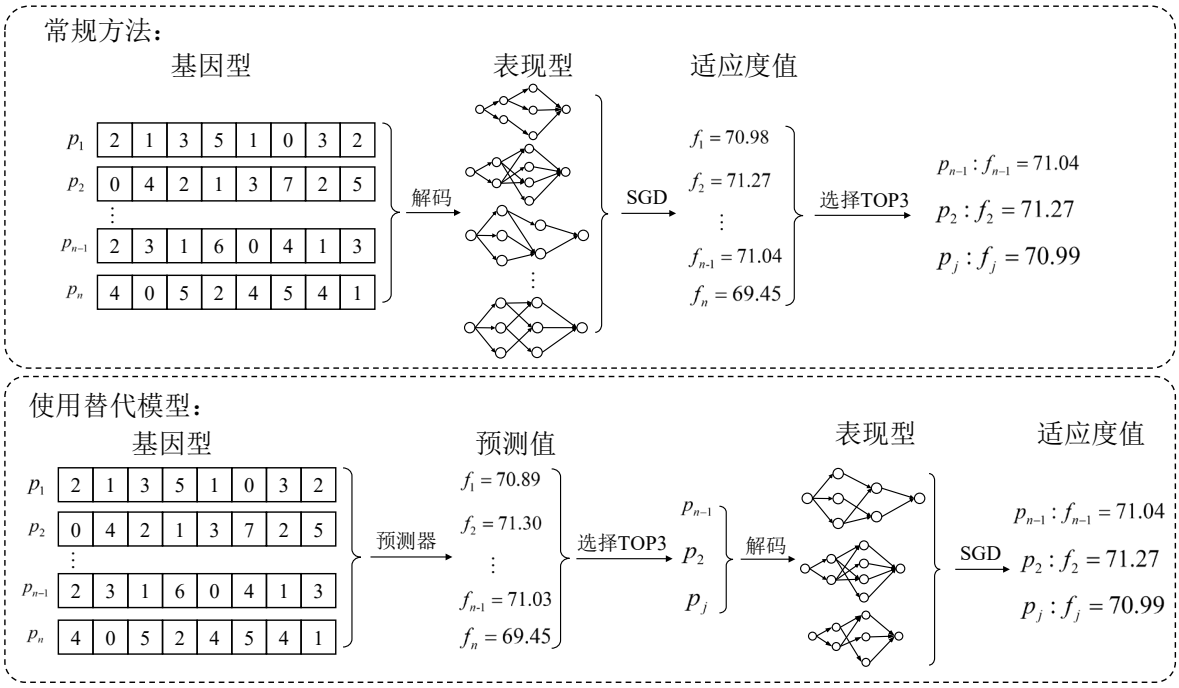


图 3-9 使用随机梯度法和替代模型得到网络性能的示例对比

经过对比，在本文的工作中使用了径向基函数（Radial Basis Function, RBF）作为替代模型来对网络模型的准确率性能进行快速地预测。径向基函数也是神经网络的一种，但是其网络结构比较简单，是一种单隐藏层的网络。在性能上径向基函数具有很强的逼近能力，其分类能力和学习能力的速度都要强于传统的反向传播神经网络。在结构上，径向基函数结构比较简单，只有一层隐藏层。具体来说，第一层是由输入信号节点组成的输入层；第二层为隐藏层，该层的节点数往往和所求解的问题相关；第三层为输出层，该层一般只起到信号传输的作用。由于网络结构简单，所以径向基函数的训练也相对简单，收敛速度也很快。通过在隐藏层的核函数，隐藏层对输入矢量进行变换，将低维的模式输入数据变换到高维空间内，使得在低维空间内的线性不可分问题在高维空间内线性可分，同时更容易逼近任意的非线性函数，克服陷入局部极小值的问题。

径向基函数作为神经网络的一种，同样需要样本数据对其进行训练。在本文中，将网络模型的 28 位编码作为特征，同时将使用随机梯度法得到的网络模型准确率作为结果标签输入到网络中，让径向基函数学习输入的网络模型编码与输入出准确率的映射关系。为了训练径向基函数模型，本文在搜索开始前随机采样了一组网络模型，并用反向梯度算法对网络模型进行训练，以便得到对应模型的真实性能。然后将径向

基函数模型在这些数据上进行训练，以便更好地拟合数据。值得注意的是，由于训练径向基函数的样本数据都是大体一致，所以在这里只需要随机采样一次，并将样本保留即可，而不需要每一次都进行采样和训练，从而大大节约了搜索阶段的时间。

如图 3-9 展示了直接使用随机梯度法和引入替代模型来获得子代适应度值的对比图。如图上部分中可知，在进化过程中如果直接常规评估方法，需要将所有产生的个体全部解码为神经网络，然后使用随机梯度法进行评估，最后在将性能较好的个体才会被选择加入到种群中，而其他较差的个体则会被丢弃掉。在这个过程中也对被舍弃的个体进行了训练，这在某种程度上是一种资源的浪费。我们假设评估一个个体的时间为T_{train}，产生的子代个体为n那么该方法就需要消耗的时间为$T_{total}=T_{train}\times n$。

相比较于常规方法需要对所有个体都进行随机梯度训练，使用替代模型可以在训练前对网络模型进行预测，如图 3-9 下图所示。通过这种方式选择 TOP3 个体作为候选个体，然后再将候选个体解码为神经网络，紧接着使用随机梯度法对候选个体进行训练并获取其适应度值，最后将候选个体加入到种群中。

在搜索过程中，为了实时更新替代模型的性能，也会将这些个体加入到替代模型的训练样本中，如图 3-1 所示。通过这种方式，可以形成了一个良性的循环，一方面既使用替代模型加速了网络模型评价的速度，另一方面，只评估少量个体获取其真实性能，同时更新替代模型训练样本。

具体来说。首先，在搜索过程中使用径向基函数来对预测新产生的网络模型个体的性能，并对性能进行排序。其次，将预测性能前t的网络作为候选个体，并使用随机梯度算法对后选个体进行训练以便得到其真实的性能。紧接着，再将候选个体作为样本去更新径向基函数的模型参数，同时统计预测误差，以便进行下一轮的迭代。

第 4 章 实验结果与分析

本节对上文提出的基于禁忌搜索和进化算法的神经网络架构搜索算法进行试验验证。为了和现有方法进行比较，本文同时在公开数据集 CIFAR-10 和 Fashion-MNIST 上进行实验。通过结果对比分析，本文所提方法搜索出来的网络结构在网络性能上和搜索所需要的计算资源上与当前较好算法具有一定的竞争力。

本文的实验主要分为两个阶段，分别是模型搜索阶段和模型评估阶段。在模型搜索阶段，主要负责搜索出性能最优的 normal cell 和 reduction cell 模快。在模型评估阶段，主要负责将搜索阶段搜索出来的两个 cell 模块通过一定的方式进行堆叠组建为神经网络，然后再使用随机梯度算法将网络模型训练至收敛，以便获得网络模型的真实性能。

4.1 数据库

4.1.1 CIFAR-10

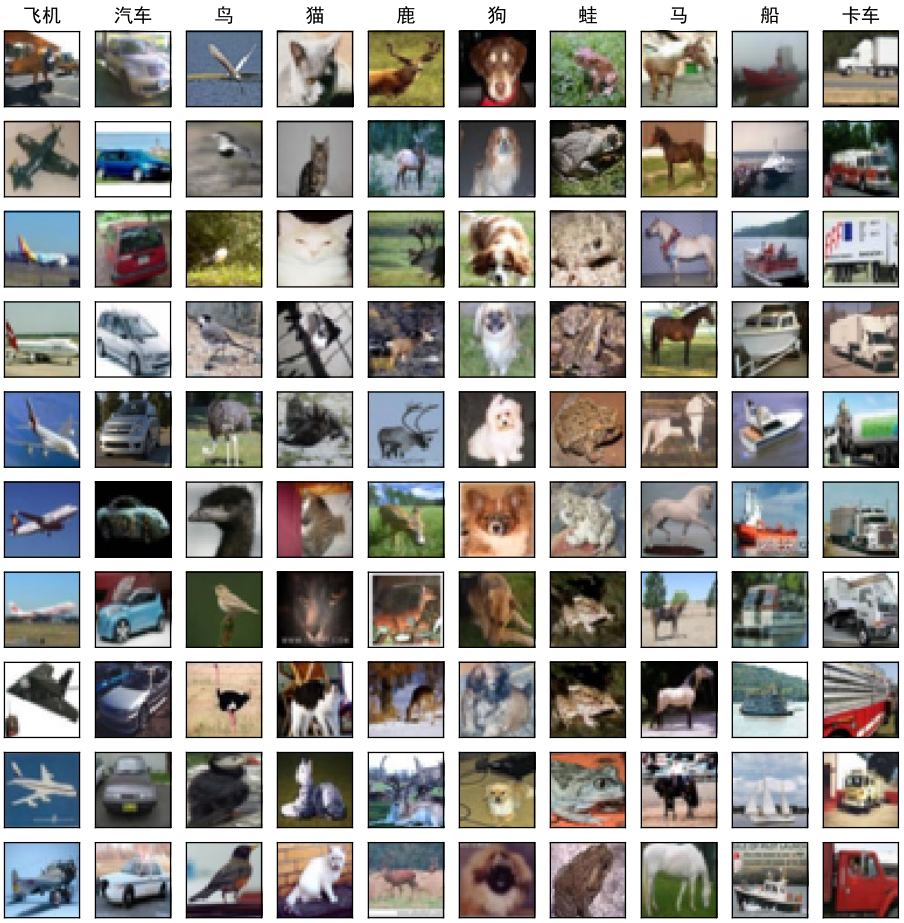


图 4-1 CIFAR-10 数据集样例

CIFAR-10 数据集[43]如图 4-1 所示，是当前神经网络架构搜索工作中最流行的公开数据集之一。CIFAR-10 是由 Alex 于 2009 年提出小型图像分类数据集，在 CIFAR-10 中的数据集中有 10 个分类的数据，分别是飞机、汽车、鸟、猫、狗、蛙、鹿、马、船和卡车，每一个类别分别有 6000 个图像，每个图像为32×32 大小的 RGB 图像。CIFAR-10 整个数据集共有 60000 整图像，其中测试图像有 10000 张，训练图像有 50000 张。

4.1.2 Fashion-MNIST

Fashion-MNIST[79]数据集是肖等人于 2017 年公开的，旨在代替 MNIST[80]手写数据集的图像数据集。Fashion-MNIST 是由德国的一家时尚科技公司 Zalando 提供的服装的分类数据集，其类别为 T 恤、裤子、套头衫、连衣裙、外套、凉鞋、衬衫、运动鞋、包包和靴子共 10 个类别。和 MNIST 类似，Fashion-MNIST 的图像也是单通道的灰度图像，图像大小为28×28，其中包括 6 万张训练图片和 1 万张测试图像，共 7 万张图片。

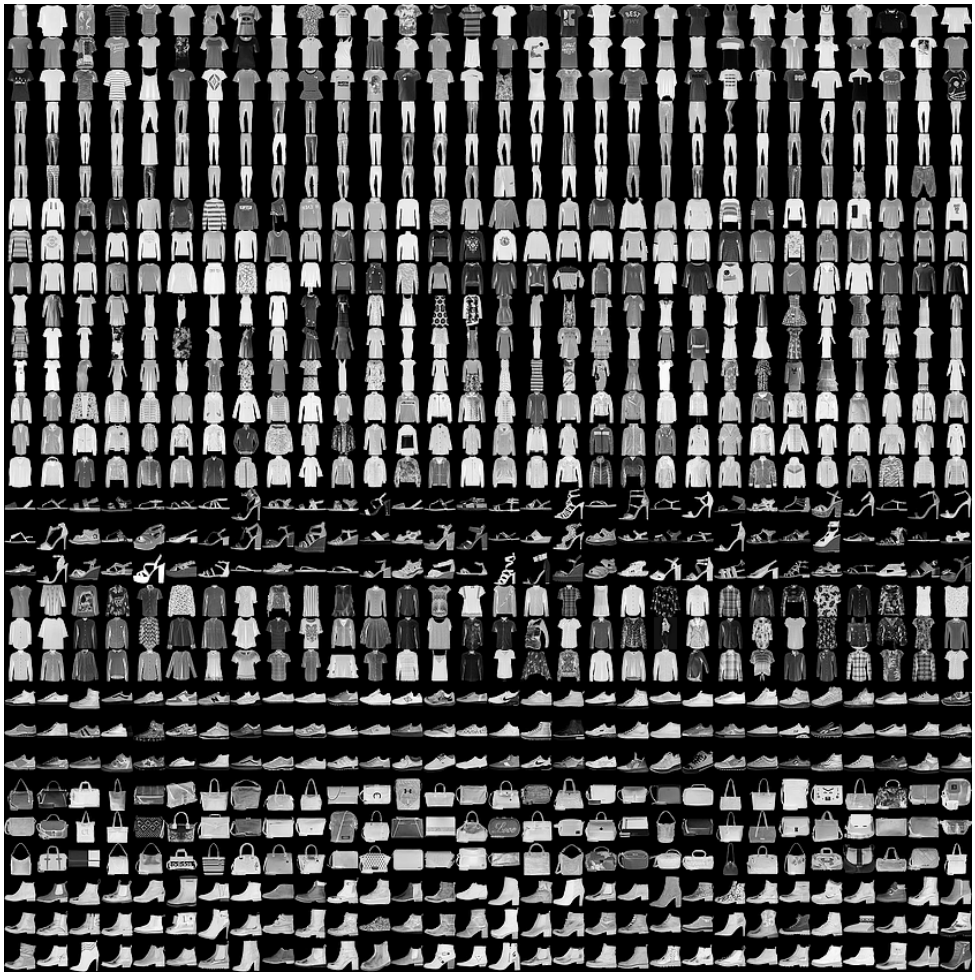


图 4-2 Fashion-MNIST 数据集样例

4.2 评价指标

本文实验的评价包括搜索得到的网络模型的准确率指标和搜索阶段消耗计算资源的性能指标。在网络模型的准确率指标上，由于本文的任务是经典的图像分类问题，因此使用的是分类问题上的准确率(*Accuracy*)指标，具体计算公式如下：

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{4-1}$$

其中 *TP, TN, FN, FP* 的相关概念如下表 4-1 所示：

表 4-1: 二元混淆矩阵

	人工标定为正类	人工标定为负类
算法判别为正类	真阳性（True Positive, TP）	假阳性（False Positive, FP）
算法判别为负类	假阴性（False Negative, FN）	真阴性（True Negative, TN）

针对搜索阶段的计算资源的评价指标，本文采用的是 GPU DAYS 的评判标准。GPU DAYS 是指在搜索阶段使用的 GPU 个数乘以搜索所需要的天数。

4.3 实验设计

4.3.1 实验环境

本文的实验以基于 Linux 操作系统的 GPU 集群服务器作为实验的硬件平台，其操作系统版本为 Ubuntu18.04；其 GPU 型号为 NVIDIA-GTX-Titan-XP，计算能力为 3.6，显存容量为 12GB；其 CPU 型号为 56 核心的 Intel(R) Xeon(R) E5-2680，主频为 2.40GHz；本文涉及到的神经网络搭建平台均为 Pytorch。除了在验证搜索出来的最优模型性能上使用多 GPU 数据并行（Data Parrellel）的方式加速训练外，如无特别说明，本文的实验均在单 GPU 模式下训练、测试及部署。

4.3.2 实验参数设置

4.3.2.1 搜索阶段参数设置

在搜索阶段的网络模型参数设置上，为了提高搜索效率，本文采用了 6 个 cell 进行堆叠，其中包括 4 个 normal cell 和 2 个 reduction cell。在模型训练参数上，本文将输入的图片的尺寸设置为16×16；将每一次训练的 batch size 设置为 128；将原始的通道数设置为 8；将最后的损失函数设置为交叉熵函数；在优化器的设置上，选择了 SGD 优化器，同时将动量（momentum）设置为 0.9，将学习率设置为 0.03。

在进化的参数设置中，本文将最优子种群和禁忌子种群的大小均设为 10；将父代种群的大小同样设置为 10，这意味着将从最优子种群和禁忌子种群里选出 10 个个

体；将进化迭代的次数设置为 200；在父代的选择中，本文采用的经典的二锦标选择算法。在禁忌操作的设置中，本文将禁忌任期设置为 5；根据编码长度和 cell 内部每一条边可选的运算操作数，本文将禁忌列表设置为一个 8×28 大小的二维表格。

4.3.2.2 评估阶段参数设置

在性能评估阶段，本文将搜索阶段搜索到的最优个体的网络参数进行随机初始化使用随机梯度法进行训练（在搜索阶段训练的网络参数将被丢弃）。在网络模型的设置上，本文将堆叠 20 个 cell 组成网络模型，其中 normal cell 有 18 个 normal cell 和 2 个 reduction cell，两个 reduction cell 的位置位于整个网络 1/3 和 2/3 的位置。

为了将模型训练至收敛，本文参考 DARTS[50]的网络参数设置，将输入的图像尺寸设置为 32×32；将每一次训练的 batch size 设置为 96；同时为了数据增强，本文使用了 cutout 机制，将 cutout 的长度设置为 16；在模型通道设置上，将模型的初始通道数设置为 36；在模型参数优化器的选择上，本文选择了 SGD 优化器，同时也将动量设置为 0.9，将初始学习率设置为 0.025，将权值衰减设置为 0.0003；为了可以将模型从头开始训练至收敛，本文将最大的训练迭代次数设置为 600 个 epoch；同样地，为了增强模型的性能，本文也是用了 drop path 正则化。为了避免训练过程出现梯度爆炸和梯度消失的问题，本文同样在网络结构中使用了梯度裁剪（Gradient Clipping）技术。

4.4 实验结果与分析

4.4.1 在 CIFAR-10 上的搜索过程的结果和分析

表 4-2：在搜索过程中设置不同种群的对比实验结果

种群设置	搜索阶段的输出结果（准确率%）
最优子种群 + 禁忌子种群	71.98
只保留最优子种群	71.70
只保留禁忌子种群	71.51

表 4-2 展示了在 CIFAR-10 数据集上的搜索过程中分别只保留最优子种群，只保留禁忌子种群和同时保留最优子种群加禁忌子种群的搜索结果。从结果不难看出，同时保留最优子种群和禁忌子种群可以提高在搜索阶段的搜索能力。在搜索过程中同时保留了最优子种群和禁忌子种群的时候，可以搜索得到准确率性能为 71.98%的最优个体。而只有最优子种群或者只保留禁忌子种群的时候，搜索得到的最优个体性能分别为 71.70%和 71.51%的准确率。

根据表 4-2 的结果分析可知，在只保留禁忌子种群的时候，搜索结果较差。这表明，由于没有充分保留种群中的最优个体，虽然可以通过禁忌规则探索更多区域，然

而无法利用最优子种群进行更多的开发，导致最优结果欠佳。只保留最优子种群的时候，最终结果和同时保留两个子种群的结果差距较小，这表明在搜索过程中最优子种群发挥的作用比较大，同时在结合禁忌子种群后，搜索能力有所提高。

图 4-3 展示了设置不同的子种群在 CIFAR-10 数据集上的搜索过程图，图中的横坐标表示种群进化迭代的次数，纵坐标表示在当前种群中最优的个体在测试集上的准确率。图中不同颜色线条代表不同的实验设置，其中蓝色线表示在搜索过程同时保留最优子种群和禁忌子种群，橙色线表示只保留最优子种群，而绿色线只保留了禁忌子种群。

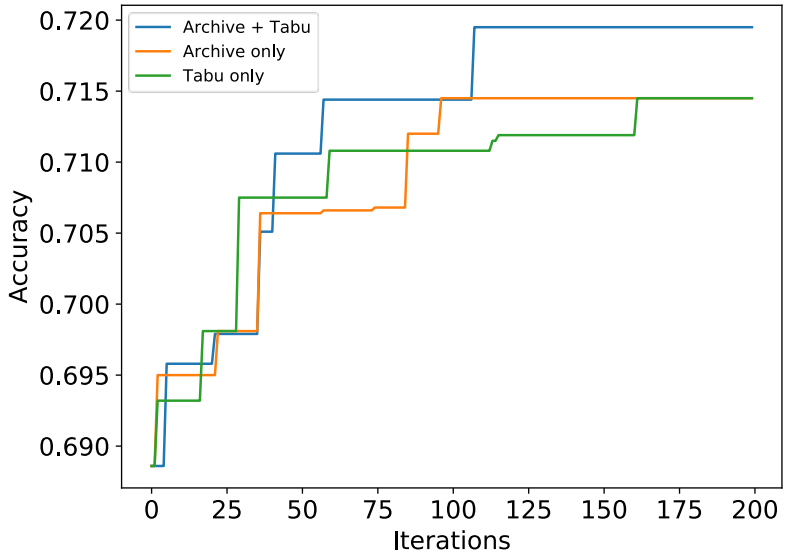


图 4-3 三种不同方法在 CIFAR-10 搜索过程中每一代最优性能对比

从图 4-3 的结果分析可知，从整体看，三种方法的大体的趋势上都是一致的，具体来说，整个搜索过程在前期的时候变化较大，特别是在前 75 代搜索的过程，这表明在进化前期种群的个体潜力都比较大，所以最优个体更新的频率和幅度都相对较大。同时从图中可以看到三种方法在 25 到 50 代之间都有一次比较大的跳变。在进化的中期，也就是在 70 代到 120 代的期间，种群变化较小，期间最优种群只发生了少量的更新。在接近进化后期的时候，种群的个体趋于收敛，同时种群最优个体性能趋近稳定。

对图 4-3 中三种不同方法进行单独分析可知，在同时保留最优子种群加禁忌子种群的时候，其最后结果的性能最好，且在中后期均有提升；对于只保留最优子种群或者只保留禁忌子种群来看，这两种方式在前期都有不俗的性能表现，特别是在只保留最优子种群的时候，在前期有一段时间的性能甚至要比其他两种都要好，但是在后期却无法继续提升；对于在搜索过程只保留禁忌子种群来说，其在后期有增长，可是由于前面的提升太慢，虽然后期有较少的提升，但是不足以弥补前面的差距。

图 4-4 展示了本文所提方法在 CIFAR-10 搜索过程中每一代新增个体的平均性能。通过图中分析可知，在整个搜索过程中每一代产生的个体的平均性能在整体上是呈上升趋势。其中在前 75 代的进化中，子代种群性能提升较为明显，特别是在 50 代和 75 代中出现了一次比较大的突变，这和图 4-3 可以相互佐证。在进化的中期，新产生的子种群从性能上稳中向好，整体变化不大。然而在进化的最后几代中，特别是 175 代后，依稀出现了性能下降的情况，如在 185 代附近出现了一次较大的下跌，这表明在进化的后期种群多样性变差，同时种群进化开始往不好的方向进化，结合图 4-3 可知在进化的后期没能产生性能更好的个体。

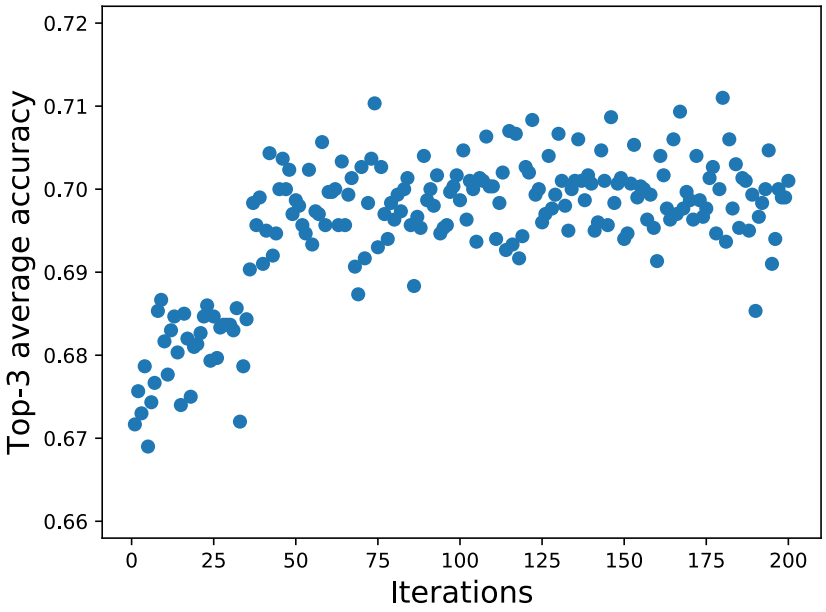


图 4-4 本文所提方法在 CIFAR-10 搜索过程中每一代新增个体的平均性能

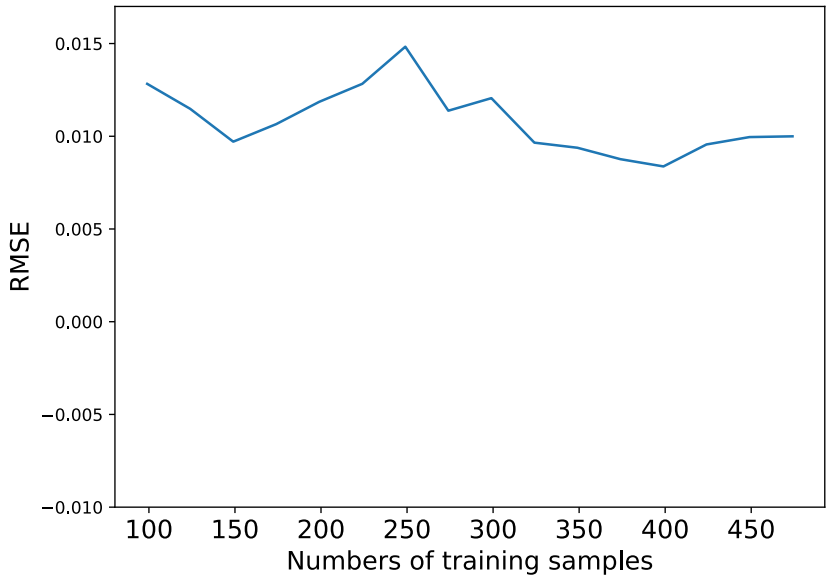


图 4-5 不同训练样本个数对替代模型的性能影响

图 4-5 展示了不同采样个数对替代模型（径向基函数）输出适应度值的均方根误差（Root Mean Square Error，RMSE）。图中分别使用了从 100 到 500 个采样点的数据，从纵坐标结果看出，径向基函数的 RMSE 的范围为 0.01 至 0.015 之间，误差较小。这也说明，本文采用的替代模型具有一定的准确性。

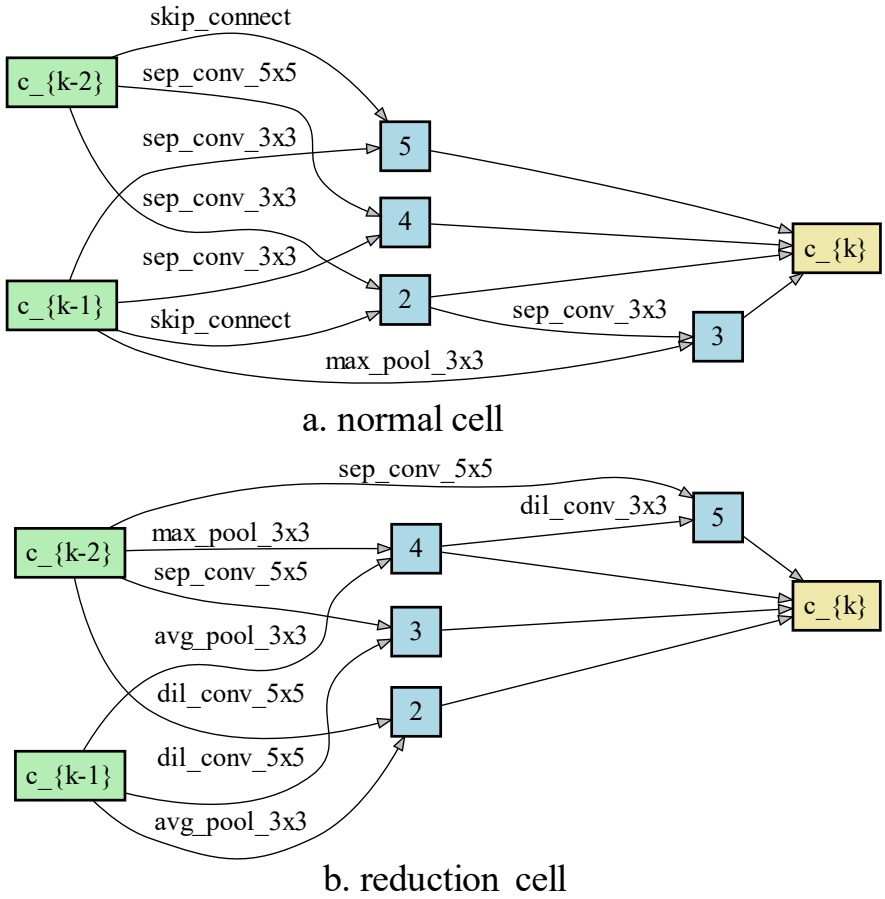


图 4-6 在 CIFAR-10 中搜索得到的最好结果

本文所提方法在 CIFAR-10 数据集上搜索得到的最优 normal cell 和 reduction cell 如图 4-6 所示。从图中可以看出，对于 normal cell 来说， 其对深度可分离卷积更为敏感，在该 cell 中共有 5 个深度可分离卷积，其中有 4 个卷积核大小为3×3，只有 1 个为5×5，且该 cell 还有出现了一个跳跃连接，将c_{k-2}（前前一个 cell）的输出直接输入到n^5节点。对于 reduction cell 来说，其内部构造比较复杂，该 cell 使用了 5 种不同的运算操作，分别是，5×5深度可分离卷积、5×5空洞卷积、 3×3空洞卷积、3×3平均池化和3×3最大池化，没有使用任何的跳连。

4.4.2 在 CIFAR-10 和 Fashion-MNIST 上的模型评估结果和分析

表 4-3 展示了本文所提方法和当前先进方法在 CIFAR-10 数据集上的实验对比结果。直观上看，本文所提方法在 CIFAR-10 数据集上只使用了 1.25 个 GPU DAYS 就搜索出错误率为 2.52%，模型的参数量大小为 3.73M 的网络结构。

同人工设置的神经网络相比较，本文得到的网络结构在错误率模型大小要优于大多数当前人工设计的经典卷积神经网络模型，如 VGGNet、ResNet 和 DenseNet 等。虽然 ResNet (depth=110)在模型大小上相对较小，然而其错误率相对较高。

对于强化学习的方法相比，NASNet 在搜索时间上消耗了 1800 个 GPU DAYS，耗时比较大；Block-QNN 使用了早停机制，在搜索时间上要比 NASNet 要节约不少，但是，由于早停机制无法准确评估一个个体的性能，所有造成搜索出来网络的错误率相对较高；同样地，ENAS 首次使用了权重继承机制，所以在搜索时间有较大的提升，只使用了仅仅 4 个 GPU DAYS 的时间，同时在错误率上也仅仅比 NASNet 少了 0.2%。

对于基于梯度可微分的方法，原始工作是 DARTS，其主要的特点是搜索时间相对较少，在当时仅仅使用了 4 个 GPU DAYS，但是在搜索过程中 DARTS 所需要的显存相对较大。其后续的工作都是针对如果更加加快搜索速度而设计的，如 BayesNAS 和 GDAS 分别仅仅需要 0.2 和 0.17 个 GPU DAYS。其中 Fine-Tuning DARTS 在 DARTS 的基础上，在网络的构建上加入了注意力模块，让其实现了较好的性能，其达到了 2.48% 的错误率，本文所提方法没有可以在网络中加入注意力模块，所以比改方法在性能上高了了 0.04%的错误率，但是本文得到的模型在参数量上小了 0.4M。

对于基于进化的方法，早期的 AmoebaNet 消耗资源较大，需要了 3150 个 GPU DAYS，但是其错误率和模型大小都比较好。对于 CARS 的方法，由于其使用了超网络来进行网络的权重参数继承，所以搜索时间很快，同时其搜索出来的模型大小也相对较小。

综合对比可知，本文得到的最优网络模型参数量与和 DARTS 和 CARS 搜索的结果相近。在性能上和先进方法 AmoebaNet 和 Fine-Tuning DARTS 都相差不大。同时在搜索时间上，本文通过替代模型的加速可以把模型搜索的过程限制在 1.25 GPU DAY 中，相对于 NASNet 和 AmoebaNet 的几千个 GPU DAY，本文所提方法具有明显的速度优势。综上所述，本文所提方法可以在搜索时间、网络性能和模型参数大小得到一个比较好的平衡，与当前先进方法具有可能性。

表 4-3：本文方法和其他先进方法在 CIFAR-10 数据集上的对比结果

Architectures	Test Error (%)	Params (M)	Search Cost (GPU days)	Search Method
VGGNet	7.66	20.04	-	Manual
DenseNet-BC	3.46	25.6	–	Manual
ResNet (depth=110)	6.43	**1.7**	-	Manual
ResNet(depth=1001)	4.6	10.2	-	Manual
NASNet-A+cutout	2.65	3.3	1800	RL
ENAS + cutout	2.89	4.6	4	RL
Block-QNN-S	4.38	6.1	90	RL
DARTS	2.76	3.3	4.0	Gradient-based
RC-DARTS	2.81	3.3	1.0	Gradient-based
GDAS	3.75	**2.5**	**0.17**	Gradient-based
Amended-DARTS	2.81	3.5	1.0	Gradient-based
BayesNAS + cutout	2.81	3.4	0.2	Gradient-based
Fine-Tuning DARTS	**2.48**	3.9	1	Gradient-based
Genetic CNN	7.1	-	17	EA
AmoebaNet-A+cutout	3.34	3.2	3150	EA
AmoebaNet-B+cutout	2.55	34.9	3150	EA
CARS	2.62	3.6	0.4	EA
NSGANet	3.85	3.3	8	EA
OURS	2.52	3.73	1.25	EA

为了验证本文所提方法的通用性，本文同时在 Fashion-MNIST 数据集上进行实验，实验结果如表 4-4 所示，通过与当前方法对比可得，本文所提方法同样具有可比性。

表 4-4：本文方法和其他先进方法在 Fashion-MNIST 数据集上的对比结果

Architecture	Test Error (%)	Params(M)	Search Method
ResNet-18	6.65	11.2	Manual
DeepCaps	5.54	7.2	Manual
Neupde	7.60	0.4	Manual
WaveMix	6.22	9.62	Manual
DARTS	4.26	2.6	Gradient-based
OURS	3.92	3.5	EA

第 5 章 总结与展望

5.1 研究总结

本文主要研究的是在分类任务上的神经网络架构搜索算法。神经网络架构搜索作为当前的一个研究热点，可以通过使用搜索优化算法，利用现有的一些网络基本模块，在一些通用的数据集或者某些针对性的问题上完成网络模型的自动化搭建。这在某种程度上可以大大减少神经网络设计的门槛，同时加快神经网络算法的落地和使用。但是，现有的神经网络架构搜索算法中在搜索阶段由于没有合理避让已经搜索过的区域容易造成重复搜索和效率低下的问题，同时神经网络架构搜索的模型评价也是一个典型的计算难问题，为了精确得到网络模型的准确率，动辄需要花费上千个 GPU DAYS。而且神经网络结构内部之间的存在各种复杂的约束，这些问题都给神经网络架构搜索带来了极大的挑战。

在本文中提出了一种新颖的基于进化算法和禁忌搜索的神经网络架构搜索算法，实现在分类任务上的网络模型高效自动化搜索。本文的主要贡献主要包括：

（1）提出了基于进化算法和禁忌搜索的双种群进化机制，相对于以往的基于进化算法的神经网络架构搜索算法可能会重复搜索的相近邻域空间的问题，本文所提方法可以在搜索过程使用了禁忌列表来对近期的邻域空间进行记录，从而避免了重复搜索相同邻域空间的问题，提高搜索性能的同时可以提高搜索效率。

（2）在基于 cell 结构的搜索空间上对 normal cell 和 reduction cell 进行统一编码，并在变异过程中采用可控变异长度机制，对不同适应度值的个体采用不同的变异长度，提高了变异的效率。

（3）设计禁忌规则，建立禁忌列表。将搜索过程中被选择个体的禁忌对象保存到禁忌列表中。同时在选择禁忌子种群的实收，对近期搜索过的邻域空间进行避让，提高搜索性能。

（4）引入了替代模型用以加快搜索过程中的网络模型性能评估速度。在搜索开始前，通过预先采样的样本数据训练径向基函数用以作为预测网络模型的替代模型，在搜索过程中选择预测性能较好的几个候选个体使用随机梯度下降算法获得模型的真实性能，然后将候选个体加入训练径向基函数的样本中，从而达到减少个体评估同时提高替代模型预测性能的目的。

5.2 研究展望

在神经网络架构搜索这个领域中，当前的搜索过程还不是完全自动化的，在搜索

空间的设计上仍然需要许多相关领域的经验知识。例如，在图像处理任务上，卷积模式是非常的有效的，所以在搜索空间上就包括了许多卷积模块、池化模块以及在网络的最后需要设计全连接层模块。针对时间序列的数据处理，如自然语言处理，循环神经网络的效果非常的好，所以在网络的编码时需要考虑如长短期记忆（Long short-term memory, LSTM）模块、门控循环单元（Gated Recurrent Unit，GRU）和最小门单元(minimal gated unit, MGU)等。在未来是否可以在编码空间上使用统一的编码，让算法可以实现所有主流任务的自动化搜索。使得神经网络架构搜索可以更少地依赖相关领域的知识，并同时实现多任务和多目标的问题。

目前神经网络架构搜索可以针对特定的问题或者数据集设计搜索空间，选择合适的搜索算法，就可以搜索出一个性能较好的网络模型。但是神经网络架构搜索无法指出网络结构具体好在哪里，具有什么不一样的特征。因此，在未来的相关工作中探索某个特征对性能的影响是非常有必要的。

参考文献

[1] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, Jun. 2016: 770–778.

[2] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Santiago, Chile, Dec. 2015: 1-9.

[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, Hawaii, USA, Jul. 2017: 4700–4708.

[4] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, USA, Jun. 2016: 779-788.

[5] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards real-time object detection with region proposal networks[J]. Advances in Neural Information Processing Systems, 2015, 28.

[6] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, USA, Jun. 2015: 3431-3440.

[7] Badrinarayanan V, Kendall A, Cipolla R. SegNet: A deep convolutional encoder-decoder architecture for image segmentation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(12): 2481-2495.

[8] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional networks for biomedical image segmentation[C]. Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, Cham, Munich, Germany, Oct. 2015: 234-241.

[9] Elsken T, Metzen JH, Hutter F. Neural architecture search: A survey[J]. The Journal of Machine Learning Research, 2019, 20(1): 1997-2017.

[10] 耿飞，王春楠，王宏志．神经网络架构搜索综述[J]．智能计算机与应用，2020, 10(6): 25-30. DOI:10.3969/j.issn.2095-2163.2020.06.006.

[11] Zoph B, Le QV. Neural architecture search with reinforcement learning[J]. arXiv preprint arXiv:1611.01578, 2016.

[12] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[J]. ArXiv preprint arXiv, 1409.1556, 2014.

[13] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Advances in Neural Information Processing Systems, 2012, 25.

[14] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]. Proceedings of the International Conference on Machine Learning, Lille, France, Jul. 2015: 448-456.

[15] Yu F, Koltun V. Multi-scale context aggregation by dilated convolutions[C]. InProceedings of the International Conference on Learning Representations, San Juan, Puerto Rico, May. 2016.

[16] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

[17] Baker B, Gupta O, Naik N, Raskar R. Designing neural network architectures using reinforcement learning[C]. Proceedings of The International Conference on Learning Representations, Toulon, France, Apr. 2017.

[18] 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, Berlin, Germany, July. 2017: 497-504.

[19] Cai H, Chen T, Zhang W, et al. Efficient architecture search by network transformation[C]. Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, Feb. Vol. 32. No. 1. 2018.

[20] Brock A, Lim T, Ritchie JM, et al. SMASH: One-shot model architecture search through hypernetworks[C]. Proceedings of the 6th International Conference on Learning Representations, Vancouver, Canada, Apr. 2018.

[21] Elsken T, Metzen JH, Hutter F. Simple and efficient architecture search for convolutional neural networks[C]. Proceedings of the NIPS Workshop on Meta-Learning, Long Beach, USA, Jan. 2017.

[22] Kaelbling L P, Littman M L, Moore A W. Reinforcement learning: A survey[J]. Journal of Artificial Intelligence Research, 1996, 4: 237-285.

[23] Medsker L R, Jain L C. Recurrent neural networks[J]. Design and Applications, 2001, 5: 64-67.

[24] Zoph B, Vasudevan V, Shlens J, Le QV. Learning transferable architectures for scalable image recognition[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Pecognition, Salt Lake City, USA, Jun. 2018: 8697-8710.

[25] Deng J, Dong W, Socher R, Li LJ, Li K, Fei L. ImageNet: A large-scale hierarchical image database[C]. Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Florida, USA, Jun. 2009: 248-255.

[26] Lin T Y, Maire M, Belongie S, et al. Microsoft COCO: Common objects in context[C]. Proceedings of the European Conference on Computer Vision. Springer, Cham, 2014: 740-755.

[27] Watkins C J C H, Dayan P. Q-learning[J]. Machine Learning, 1992, 8(3): 279-292.

[28] Zhong Z, Yan J, Wu W, Shao J, Liu CL. Practical block-wise neural network architecture generation[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, Jun. 2018: 2423-2432.

[29] Pham H, Guan M, Zoph B, et al. Efficient neural architecture search via parameters sharing[C]. Proceedings of the International conference on machine learning, Stockholm, Sweden, Jul. 2018: 4095-4104.

[30] Tan M, Chen B, Pang R, et al. MnasNet: Platform-aware neural architecture search for mobile[C]. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, Jun. 2019: 2820-2828.

[31] He X, Zhao K, Chu X. AutoML: A survey of the state-of-the-art[J]. Knowledge-Based Systems, 2021, 212:106622.

[32] Miller GF, Todd PM, Hegde SU. Designing neural networks using genetic algorithms[C]. Proceedings of the Third International Conference on Genetic Algorithms, 1989: 379-384.

[33] Peter J. Angeline, Gregory M. Saunders, and Jordan B. Pollack. An evolutionary algorithm that constructs recurrent neural networks[H]. IEEE Transactions on Neural Networks, 1994, 51: 54–65.

[34] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies[J]. Evolutionary Computation, 2002, 10: 99–127.

[35] Stanley KO, Miikkulainen R. Evolving neural networks through augmenting topologies[J]. Evolutionary Computation, 2002, 10(2): 99-127.

[36] Real E, Aggarwal A, Huang Y, Le QV. Aging evolution for image classifier architecture search[C]. Proceedings of the Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence, Hawaii, USA, Jan. 2019.

[37] Liu C, Zoph B, Neumann M, Shlens J, Hua W, Li LJ, Fei L, Yuille A, Huang J, Murphy K. Progressive neural architecture search[C]. Proceedings of the European Conference on Computer Vision (ECCV), 2018: 19-34.

[38] Xie L, Yuille A. Genetic CNN[C]. Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, Oct. 2017: 1379-1388.

[39] Robbins H, Monro S. A stochastic approximation method[J]. The Annals of Mathematical Statistics, 1951: 400-407.

[40] Salimans T, Ho J, Chen X, Sidor S, Sutskever I. Evolution strategies as a scalable alternative to reinforcement learning[J]. arXiv preprint arXiv:1703.03864, 2017.

[41] Beyer H G, Schwefel H P. Evolution strategies–a comprehensive introduction[J]. Natural Computing, 2002, 1(1): 3-52.

[42] Real E, Aggarwal A, Huang Y, Le QV. Regularized evolution for image classifier architecture search[C]. Proceedings of the AAAI Conference on Artificial Intelligence, Hawaii, USA, Jan. 2019: 4780-4789.

[43] Krizhevsky A, Hinton G. Hinton. Learning multiple layers of features from tiny images (technical report)[R]. University of Toronto, 2009.

[44] Miikkulainen R, Liang J, Meyerson E, et al. Evolving deep neural networks[J]. Artificial Intelligence in the Age of Neural Networks and Brain Computing, Academic Press, 2019: 293-312.

[45] Yang Z, Wang Y, Chen X, et al. Cars: Continuous evolution for efficient neural architecture search[C]. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, China, Jue. 2020: 1829-1838.

[46] Suganuma M, Shirakawa S, Nagao T. A genetic programming approach to designing convolutional neural network architectures[C]. Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, Jul. 2018: 5369-5373.

[47] Miller JF, Harding SL. Cartesian genetic programming[C]. Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference, Montreal, Canada, July. 2009: 3489-3512.

[48] 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, Prague, Czech Republic, Jul. 2019: 419-427.

[49] 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):182-97.

[50] Liu H, Simonyan K, Yang Y. Darts: Differentiable architecture search[C]. Proceedings of the International Conference on Learning Representations, Vancouver, Canada, Apr. 2018.

[51] Dong X, Yang Y. Searching for a robust neural architecture in four gpu hours[C]. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, Jun. 2019: 1761-1770.

[52] Chen X, Xie L, Wu J, Tian Q. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation[C]. Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019: 1294-1303.

[53] Xu Y, Xie L, Zhang X, et al. H. Pc-darts: Partial channel connections for memory-efficient architecture search[C]. Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, May.2019.

[54] Li L, Jamieson K G, DeSalvo G, et al. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization[C]. ICLR (Poster). 2017.

[55] Zela A, Klein A, Falkner S, et al. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search[J]. arXiv preprint arXiv:1807.06906, 2018.

[56] Zheng X, Ji R, Tang L, et al. Multinomial distribution learning for effective neural architecture search[C]. Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, Oct. 2019: 1304-1313.

[57] 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: 350-64.

[58] Lu Z, Sreekumar G, Goodman E, el al. Neural architecture transfer[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021; 2971-89.

[59] Vinyals O, Blundell C, Lillicrap T, Wierstra D. Matching networks for one shot learning[J]. Advances in Neural Information Processing Systems, 2016; 29.

[60] Xie S, Zheng H, Liu C, et al. SNAS: stochastic neural architecture search[C]. Proceedings of the International Conference on Learning Representations. 2018.

[61] Cai H, Zhu L, Han S. ProxylessNAS: Direct neural architecture search on target task and hardware[C]. International Conference on Learning Representations. 2018.

[62] 郑金华. 多目标进化算法及其应用[M]. 科学出版社, 2007.

[63] Blickle T. Tournament selection[J]. Evolutionary Computation, 2000, 1: 181-186.

[64] Lipowski A, Lipowska D. Roulette-wheel selection via stochastic acceptance[J]. Physica A: Statistical Mechanics and its Applications, 2012, 391(6): 2193-2196.

[65] Glover F. Tabu search—part I[J]. ORSA Journal on Computing, 1989; 190-206.

[66] Bebis G, Georgiopoulos M. Feed-forward neural networks[J]. IEEE Potentials, 1994, 13(4): 27-31.

[67] Svozil D, Kvasnicka V, Pospichal J. Introduction to multi-layer feed-forward neural networks[J]. Chemometrics and Intelligent Laboratory Systems, 1997, 39(1): 43-62.

[68] Hubel, David H., and Torsten N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cats visual cortex[J]. The Journal of Physiology, 1962, 160.1: 106-154.

[69] Fukushima K, Miyake S. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position[J]. Pattern Recognition, 1982, 15(6):455-69.

[70] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition[C]. Proceedings of the IEEE, 1998, 86(11): 2278-324.

[71] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The Journal of Machine Learning Research, 2014, 15(1):1929-58.

[72] M. Lin, Q. Chen, S. Yan. Network In Network[C]. International Conference on Learning Representations, Banff, Canada, Apr. 2014.

[73] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, Jun. 2016: 818-2826.

[74] Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-resnet and the impact of residual connections on learning[C]. Proceedings of the Thirty-first AAAI Conference on Artificial Intelligence, California, USA, Feb. 2017.

[75] Liu Y, Sun Y, Xue B, et al. A survey on evolutionary neural architecture search[J]. IEEE Transactions on Neural Networks and Learning Systems, 2021.

[76] Mirjalili S. Genetic algorithm[M]. Evolutionary Algorithms and Neural Networks, Springer, Cham, 2019: 43-55.

[77] Real E, Moore S, Selle A, et al. Large-scale evolution of image classifiers[C]. International Conference on Machine Learning. Sydney, NSW, Australia, Aug. 2017: 2902-2911.

[78] Gupta A K, Smith K G, Shalley C E. The interplay between exploration and exploitation[J]. Academy of Management Journal, 2006, 49(4): 693-706.

[79] Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms[J]. arXiv preprint arXiv:1708.07747, 2017.

[80] LeCun Y. The MNIST database of handwritten digits[J]. http://yann. lecun. com/exdb/mnist/, 1998.

攻读学位期间主要研究成果

一、 发表的学术论文

[1] Fan Z, **Long Z**, Li W, et al. Neural Architecture Search Based on Tabu Search and Evolutionary Algorithm[C]. 2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER). IEEE, 2021: 805-811.(EI 索引，与第 3、4 章有关)

[2] Lin P, Li X, **Long Z**, et al. Pipeline Leak Detection, Location and Repair[C]. 2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER). IEEE, 2021: 102-108.

[3] Fan Z, Yang Z, Tang Y, Li W, Xu B, Wang Z, Sun F，**Long Z**, and Zhu G. An Improved Epsilon Method with M2M for Solving Imbalanced CMOPs with Simultaneous Convergence-Hard and Diversity-Hard Constraints[C]. International Conference on Evolutionary Multi-Criterion Optimization. Springer, Cham, 2021: 248-256.

二、申请专利

（一）已授权实用新型专利

[1] 范衡,**龙周彬**,林培涵,李晓明,马培立,陈文钊,叶志豪,黄铭威,李建立. 一种集成视觉与压力反馈的带压堵漏末端执行器[P]. 广东省：CN214745029U,2021-11-16.

（二）已申请发明专利

[1] 范衡,林培涵,陈梓泓,王诏君,李晓明,**龙周彬**,董朝晖,马培立. 一种无中心机器人集群包围任务控制方法及系统[P]. 广东省：CN112527012A,2021-03-19.

[2] 范衡,李晓明,**龙周彬**,林培涵,马培立,韦家弘,姜志成,何伟源,钟铸威. 一种带压堵漏机器人自主漏点定位方法[P]. 广东省：CN113001559A,2021-06-22.

[3] 范衡,李晓明,王诏君,王柳,黄华兴,林培涵,马培立,**龙周彬**. 基于基因调控网络的群体机器人动态围捕控制方法及系统[P]. 广东省：CN112462779A,2021-03-09.

三、软件著作权

[1] 群体机器人聚合形态自动化设计平台。

四、奖项

[1] 第六届中国国际“互联网+”大学生创新创业大赛广东省铜奖。

致谢

毕业伴随着凤凰花的悠悠香气与夏季的一丝热情如约而至。此时此刻，在毕业论文完成之际，特别感谢在汕头大学认识的老师、同学与及朋友们，你们的指导、陪伴和鼓励让我在汕大度过了难忘的三年，从你们身上我收获了很多的学习和成长。有志、有识、有恒、有为的校训牢记心中，感谢汕头大学美丽的校园环境、丰富的校园活动、完善的教学体系以及浓厚的科研氛围，让我在这三年的学习生涯中可以专注于科研的探索。

特别感谢我的导师范衡教授，感谢您三年来的指导与帮助。您对科研的热情和严谨让我获益匪浅。您对我的悉心指导，教会了我如何进行科学研究。是您让我在三年的研究生生涯中不迷失方向。

同时也要感谢实验室团队的老师和师兄师弟们。特别是李文姬老师和王诏君博士，感谢你们在科研和生活上对我的照顾。另外还要感谢杨知师姐、孙福赞师兄、林培涵同学、李晓明同学、王柳同学、董朝辉同学、蔡堉伟同学和石泽同学等人在研究生期间对我的帮助。

感谢舍友肖尔达，在四年的生活中教会了我许多，在我生活和科研上遇到困难的时候总能给我意见和帮助。

特别感谢黄炜莹同学，感谢你们在生活上对我的包容。是你们让我了解了汕头这座城市，带我领略了潮汕文化的魅力。

最后还要感谢我的家人们，是你们无声的支持，让我感受到了家人们的温暖，是你们的支持和鼓励让我可以更加专心于学习而不用为家里的琐事而烦恼。

作者：龙周彬

2022 年 4 月 12 日