一种高效可解释的全自动化通道剪枝算法

黄源翔

北京科技大学

密　　级：公开

论文题目：一种高效可解释的全自动化通道剪枝算法

学　　号：G20208807

作　　者：黄源翔

专 业 名 称：计算机技术

2023 年 05 月 30 日

一种高效可解释的全自动化通道剪枝算法

An Efficient and Interpretable Fully Automatic Channel Pruning Algorithm

研究生姓名：黄源翔

指导教师姓名：齐悦

北京科技大学计算机与通信工程学院

北京 100083，中国

Master Degree Candidate： Yuanxiang Huang

Supervisor： Yue Qi

School of Computer & Communication Engeering

University of Science and Technology Beijing

30 Xueyuan Road，Haidian District

Beijing 100083，P.R.CHINA

分类号：TP391　　密　　级：公开

ＵＤＣ：　　单位代码：１０００８

北京科技大学硕士学位论文

论文题目：一种高效可解释的全自动化通道剪枝算法

作者：黄源翔

指 导 教 师：齐悦 副教授　**单位：**北京科技大学

指导小组成员：何杰 教授　**单位：**北京科技大学

单位：

论文提交日期：2023 年 05 月 30 日

学位授予单位：北 京 科 技 大 学

致　谢

逝者如斯夫，不舍昼夜。三年的研究生生涯已至尾声，回首在北京科技大学的无数个日子，求学、科研、历练、生活，受益良多。

感谢我的导师齐悦老师！师者匠心，止于至善！在学期间，您传授的知识、展示的治学态度和行事风格等等，都对学生产生了深远的影响。从入学初的培养计划到当下的毕业设计，学生在您的指导下踉步前行，而有今日。

感谢何杰老师！师者如光，微以致远！您对项目整体的把握还是对细节的关注，不仅教会学生如何在项目中做好，更教会学生如何做好项目。

感谢各位授课老师！片言之赐，亦吾师也！没有各位老师的传道授业解惑，学生永远不知世界之大、万物之妙。

感谢各位师兄弟姐妹，感谢各位同门！学贵得师，亦贵得友！与你们在一起的学习生活，亦是我永远的财富。

感谢我的父母和亲人！无父何怙，无母何恃！感谢你们的照顾与支持，孩儿一路走来，既是为了自己，也是为了你们脸上的笑容。

最后感谢参与答辩的评审老师!感谢各位老师于百忙中审阅学生的文章，也感谢各位老师提出的宝贵意见！

海阔凭鱼跃，天高任鸟飞！老师、朋友，理念、知识，我会带着在北京科技大学收获的一切，驶向前路。

摘 要

结合神经网络结构搜索的自动化通道剪枝相比于传统通道剪枝方法，可以大幅降低人工参与。然而现有研究无法摆脱对超参数的依赖，针对模型剪枝率组合的高维搜索空间使它们难以兼顾高效与全自动化。

本文提出一种高效可解释的全自动化通道剪枝算法：首先，将卷积层通道权重置换引发的输出特征图变化量作为通道重要性的度量指标，使通道重要性的计算在保持高效的同时更具可解释性；其次，将通道重要性转换为信息贡献率，并基于累计信息贡献率计算卷积层的通道保留数，该算法可以在同一个超参数的指导下为不同卷积层定制通道保留数；最后，将自动化通道剪枝中针对剪枝率组合的多元搜索转换为更高效的针对累计信息贡献率的一元搜索，并采用二分搜索策略，大幅降低了搜索成本。本文算法只需使用者设置可接受精度损失和最小搜索区间即可高效地自动搜索最佳剪枝结果。

实验数据表明：（1）相比于其它经典算法，基于权重置换的通道剪枝算法得到的剪枝结果在精度上平均提高 0.04%，压缩率平均提高 10.9%；（2）对于通道数相同的卷积层，基于累计信息贡献率的通道保留数算法能够在一个超参数的指导下为不同层计算合适的通道保留数；（3）本文算法在不同模型和数据集上具备通用性，相比于其它经典算法，本文算法可以获得相近甚至更好的剪枝结果，并且所需搜索成本平均降低 36%。

关键词： 深度学习，卷积神经网络，模型压缩，通道剪枝，神经网络结构搜索

An Efficient and Interpretable Fully Automatic Channel Pruning Algorithm

Abstract

Automatic channel pruning with neural architecture search can significantly reduce human involvement compared to traditional channel pruning. Existing studies still heavily rely on hyperparameters, and the high-dimensional search space for the pruning rate combinations makes it difficult to balance efficiency and full automation.

In this essay, we propose an efficient and interpretable fully automatic channel pruning algorithm. Firstly, the output feature map changes caused by channel weight permutation are used to measure channel importance, making it both efficient and interpretable. Secondly, the channel importance is converted to information contribution rate and the cumulative information contribution rate is used to calculate the to-remain channel number, which is customized for different layers under one hyperparameter. Finally, the multi-dimensional search for pruning rate combinations is transformed into a more efficient one-dimensional search for the cumulative information contribution rate with a dichotomous search in automatic channel pruning, which reduces the search cost. The proposed algorithm only requires the user to set the acceptable accuracy loss and the minimum search interval to automatically search for the best pruning results efficiently.

Our experiments show that (1) the channel pruning based on weight permutation achieves an average improvement of 0.04% in accuracy and 10.9% in compression rate for pruning results compared to other classic algorithms; (2) for convolutional layers with the same number of channels, the to-remain channel number calculated based on the cumulative information contribution rate can adapt for them under one hyperparameter; (3) the proposed algorithm is universal for different models and datasets, and can achieve comparable or even better pruning results with 36% less seach cost on average than other classic algorithms.

Key Words：Deep Learning, Convolutional Neural Network, Model Compression, Channel Pruning, Neural Architecture Search

目　录

插图清单

附表清单

1 绪论

1.1 研究背景与意义

以卷积神经网络为主要内容的深度学习已然广泛地应用于各个领域，如医学界采用图像识别等技术进行辅助诊断，工业界采用目标检测等技术进行分拣和筛选等工作，智能家居更是集计算机视觉、语音识别和自然语言处理等技术于一身。这些应用极大地解放了人力，提高了生产效率，也为人们提供了更优质的生活体验。

现代卷积神经网络通常包含特征提取器和分类器两部分，前者主要由卷积层、池化层和激活层等组成，其任务在于将样本映射到高抽象级的特征表示空间，后者主要由全连接层组成，其任务则在于将样本的特征表示映射到样本标签空间，从而完成对样本的预测[1]。AlexNet[2] 的成功证明了深度模型拥有更强的特征表达能力，由此卷积神经网络的设计朝更宽更深的方向发展以便胜任复杂任务，如 ResNet[3] 和 DenseNet[4] 等。深度模型在展示其强大特征表达能力的同时，也对部署设备提出了更高的计算资源需求（包括算力和存储），表 1-1 展示了部分经典深度模型在输入为单张 $3 \times 128 \times 128$ 图片时的浮点数计算量与参数量。在资源丰富的服务器中这样的需求不值一提，然而在计算资源十分受限的边缘设备中，深度模型的运算速率将大打折扣，这对具有实时性要求的任务而言是不可接受的，如辅助驾驶的道路检测和军事领域的目标识别等。鉴于深度学习在边缘计算中愈加广泛的应用，减少模型的计算量和参数量以适配资源有限的边缘设备是十分必要的。

表 1-1 部分经典深度模型的浮点计算量和可学习参数量

模型	计算量（M FLOPs）	（可学习）参数量（M）
AlexNet	254.85	61.10
VGG16 [5]	5134.81	138.36
GoogLeNet [6]	494.19	6.62
ResNet18	595.95	11.69
ResNet50	1351.17	25.56
ResNet101	2570.01	44.55
DenseNet121	946.65	7.98

模型压缩，包括权重量化、网络剪枝、知识蒸馏、神经网络结构搜索（Neural Architecture Search，NAS）和轻量化模型设计[7]，旨在达到模型精度和计算资源之间的权衡。通道剪枝是一种有效的模型压缩方法，它通过移除原模型中不必要的通道，从而达到压缩模型的目的。本文出于以下四个动机

就自动化通道剪枝展开研究：

（1） 通道剪枝在某种准则的指导下进行模型压缩，与知识蒸馏和神经网络结构搜索等方法相比，它可在更短的周期内得到剪枝结果；

（2） 通道剪枝在原模型上直接进行压缩，与知识蒸馏和轻量化模型设计等方法相比，它不需要重新设计模型，降低了对使用者的技术要求；

（3） 通道剪枝是通道级的模型压缩方法，与层剪枝（网络剪枝的另一方向）相比，剪枝后模型的计算序列与原模型相同，不会影响样本特征提取的过程；

（4） 自动化通道剪枝可在较少的人工参与下完成模型的压缩任务，这类方法因其具备的高易用性而受使用者青睐，并且是未来通道剪枝研究的重要方向之一。

1.2 国内外研究现状

深度学习的概念由 Hinton 等人于 2006 年提出[8]，随后在计算机视觉、自然语言处理和语音识别等领域中取得丰硕成果。**计算机视觉**是以图像处理为主题的技术。自 1998 年的 LeNet[9] 之后，传统机器学习曾一度超越神经网络，如支持向量机（SVM）。直到 2012 年，AlexNet[2] 证明了神经网络自行学习到的特征可以超过人工设计的特征，自此研究者们提出各种不同的深度网络，如经典的 VGG[5] 和 ResNet[3] 等。除了基本的图像分类问题，这些经典网络也常作为骨干网应用于目标检测和分割等其它图像处理问题，如 SSD[10] 和 YOLO 系列[11,12]。近年来的相关研究开始将注意力放在三维空间上，如三维重建[13,14]和姿态估计[15]等。**自然语言处理**是致力于计算机和人类自然语言之间相互沟通的研究领域。2013 年的 word2vec[16] 在机器翻译等问题上取得重大突破，随后 GRU[17] 等模型更上一层楼，如今循环神经网络[18]（RNN）和注意力机制[19,20]等已成该领域最常用的技术。目前自然语言处理主要集中于情感分析、机器翻译和文本分析等问题上，随着 ChatGPT 的问世，智能问答、智能对话甚至对话翻译等都将是未来研究的热点问题。

在深度学习应用愈加广泛、能力愈加强大的同时，模型体量和硬件的配置要求也随之飞升。当今人们对边缘计算的智能化要求越来越高，如智能家居和智能驾驶等，边缘设备有限的资源与深度模型对硬件资源的高要求矛盾使模型压缩势在必行。

现有模型压缩有许多不同的方向。**权重量化**[21,22]通过将原模型权重的存储格式由 FP32 转换为更低 bit 的方式降低模型所占存储空间和计算时延。

权重量化较为成熟，8-bit 及以上的量化在工业界已十分常见，而更低 bit 量化的相关研究也有很多，如何在降低 bit 位数的同时维持模型精度是相关研究的重点问题。**知识蒸馏**[23–26]通过将大模型所含任务信息转移到小模型中实现模型压缩，目前其应用已扩展到了知识迁移等领域中。知识蒸馏相关研究主要关注知识体系和蒸馏策略，在师生模型结构上关注度不足，鉴于其在知识蒸馏应用中的重要地位，师生模型结构的研究将会是未来的一个重点。**神经网络结构搜索** [27–29]通过在模型结构空间中的搜索和评估寻找当前任务下的最优解，这类方法主要面临的挑战是搜索成本过高，因此如何在保证搜索效果的同时降低搜索成本是相关研究的重点。**网络剪枝**[30–34]通过移除网络中不必要的部分达到模型压缩的效果，根据剪枝粒度可分为层剪枝与通道剪枝。鉴于使用者对自动化的向往，近年来网络剪枝相关研究逐步与 NAS 相结合。**轻量化模型设计**[35–39]通过降低特征图通道数和应用分组卷积等方法设计参数量和计算量更少的模型。

通道剪枝是一种有效的、重要的模型压缩方法，它不仅可以在保持模型结构不变的同时降低模型体量、提升模型前向传播速度，还可以有效地抑制模型在训练中可能出现的过拟合现象。通道剪枝的主要问题在于待剪枝通道的选择和剪枝比例的设定，前者的相关研究主要有基于度量[40]、基于误差重建[41,42]和基于稀疏化训练[43,44]等方向，后者即具化为剪枝率设置问题，这类研究主要可分为预定义和自动两种[45]。然而，有研究指出随机性的通道剪枝在微调后同样能得到理想的结果[45,46]，这几乎推翻了一些传统的通道剪枝理论，并促使通道剪枝与 NAS 相结合。自 He 等人提出的 AMC[33] 开始，越来越多结合 NAS 的相关研究出现，实验证明这类方法可以取得极具竞争力的剪枝结果，并且其自动化的剪枝流程对使用者而言十分友好。反过来，通道剪枝为 NAS 提供了搜索空间定义和搜索策略的指导，使 NAS 在理论性和实用性上得到明显提升。可以预见，与 NAS 的结合将是未来通道剪枝乃至网络剪枝的重要研究方向。

1.3 本文研究思路与研究内容

本文的目标是设计一种高效可解释的全自动化通道剪枝算法，总体研究思路如图 1-1 所示。在对现有相关研究进行分析后，本文将算法设计分为三个问题：（1）现有相关研究中的通道重要性度量主要分为基于度量的方法和基于误差重建的方法，这些方法难以兼顾通道重要性的计算效率和可解释性；（2）针对剪枝率组合的高维搜索空间将为算法带来高额的搜索成本，这在实际应用中是难以接受的；（3）遍历式的搜索策略虽然能够找到最优解，然而

大量的探查同样为算法带来了巨大的计算和时间成本。

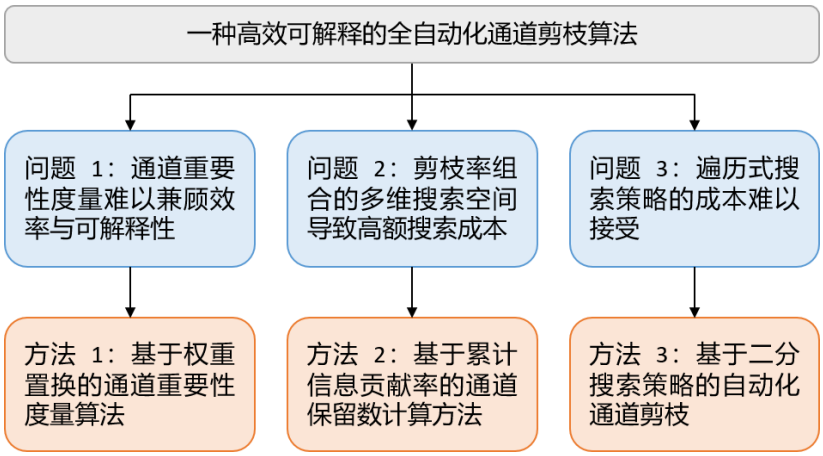


图 1-1 研究思路

根据上述分析，本文进行如下研究：

（1） 本文探究了通道剪枝中衡量通道重要性的标准，提出了基于权重置换的通道重要性度量算法，并基于此提出了基于权重置换的通道剪枝方法。具体而言，本文将卷积层通道权重置换前后引发的输出特征图变化作为该通道的重要性衡量标准，在完成所有通道的重要性计算后，结合算法设定的剪枝率计算出需要移除的通道数，最后将重要性最低的若干个通道移除。本文提出的基于权重置换的通道重要性衡量标准结合了模型权重本身与训练集样本，对通道重要性的评估更加全面、更具解释性。

（2） 本文分析了基于剪枝率指导通道保留数计算中存在的问题，在基于权重置换的通道重要性度量算法之基础上提出了基于累计信息贡献率的通道保留数计算方法，调整了通道剪枝算法的实现方式。具体而言，本文将通道的重要性评估转换为其对所属卷积层信息量的贡献率评估，摒弃了基于剪枝率的通道保留数计算，设计了累计信息贡献率指导通道保留数的计算。剪枝过程将保留信息贡献率最高的若干个通道，并保证这些通道的信息贡献率之和不小于算法设定的累计信息贡献率。本文提出的基于累计信息贡献率的通道保留数计算方法可以在同一超参数的指导下为不同卷积层定制合适的通道保留数，调和了基于剪枝率的通道保留数计算中全局共享剪枝率的简单性与逐层定制剪枝率的合理性。

（3） 本文在基于基于累计信息贡献率的通道保留数计算方法之上提出了高效的全自动化通道剪枝方法。具体而言，本文观察到模型压缩率与其

精度的近似负相关关系，采用了二分搜索的策略，将累计信息贡献率的定义域作为搜索空间，对最优剪枝模型进行求解，将传统自动化剪枝中的高维搜索空间降至一维，降低了自动化通道剪枝的搜索成本。

（4） 本文提出的全自动化通道剪枝方法将累计信息贡献率转换为算法内部变量，使自动化通道剪枝摆脱人工调整超参数的繁杂过程，使用者只需要设定可接受精度损失和最小搜索区间，即可在无人工干预的情况下快速得到剪枝模型。

1.4 本文结构

本文共 5 章，各章安排如下：

第一章：绪论。本章首先阐述本课题的研究背景与研究意义，然后介绍模型压缩的研究现状，最后说明本课题的研究思路与研究内容。

第二章：相关基础理论与相关研究。本章首先介绍深度学习相关的基础理论与相关研究，包括深度学习方法以及卷积神经网络，然后具体介绍不同的模型压缩思路，并总结现有的部分经典研究。

第三章：基于权重置换的通道剪枝算法。本章首先介绍不同通道剪枝方法以及其中存在的问题，然后说明并定义通道重要性，提出基于权重置换的通道重要性计算方法，并分析算法实现中的相关细节，最后通过实验对本章提出的基于权重置换的通道剪枝方法进行验证与分析。需要说明的是，本章工作是为第四章中提出的全自动化通道剪枝算法做基础。

第四章：基于累计信息贡献率的全自动化通道剪枝算法。本章首先介绍自动化通道剪枝方法以及其中存在的问题，然后说明以剪枝率指导通道保留数计算的不合理性，提出基于累计信息贡献率的通道保留数计算方法和基于累计信息贡献率的自动化剪枝算法，并描述算法的实现，最后通过实验对本章提出的基于累计信息贡献率的自动化通道剪枝方法进行验证与分析。

第五章：结论。本章首先梳理本课题工作，然后分析本文的贡献与不足，并提出可能的改进方向，最后对通道剪枝以及模型压缩的未来方向进行展望。

2 相关基础理论与相关研究

2.1 深度学习

深度学习由 Hinton 等人在人工神经网络[47]（ANN）和卷积神经网络[9]（CNN）的基础上提出，它模拟人脑对事物的认知方式建立起神经网络，通过逐层特征变换将样本的特征表示变换到不同抽象级别的特征空间，进而解决复杂的模式识别问题[8]。作为机器学习领域的一个重点研究方向，深度学习在计算机视觉[3,5,48,49]、自然语言处理[19,20]、搜索推荐[50–52]和语音合成[53–56]等诸多领域取得了丰硕成果，是人类向人工智能迈进的重要一步。

2.1.1 深度学习方法

监督学习旨在构建一个能够表示样本与标签间映射关系的模型，它被广泛应用于回归任务和分类任务中，如数值回归[57,58]、图像分类[3–5]和目标检测[10–12,59,60]等。监督学习根据拟合目标的方式可分为生成式和判别式：前者根据数据集学习联合概率分布 $\mathrm{P}(X,Y)$，然后根据标签先验概率 $\mathrm{P}(\mathrm{Y})$ 求解条件概率分布 $\mathrm{P}(Y|X)$ 作为预测结果，如朴素贝叶斯算法[61,62]和隐式马尔科夫模型[63–65]等；后者根据数据集直接学习条件概率分布 $\mathrm{P}(Y|X)$ 用作预测，如支持向量机[11,66,67]（SVM）、随机森林[68,69]（RF）和线性回归[70–72]（LR）等。

无监督学习根据无标签样本集解决各种模式识别任务。在现实中，样本包含的信息往往不仅局限于人工给定的标签，在无标签情况下模型可以学习到更多的样本特征，从而实现对样本的理解。聚类[73,74]和数据降维[75,76]等都是无监督学习的典型应用。

强化学习采用一种奖惩机制，依据激励信号和反馈信号指导智能体在与环境的交互中不断地分析环境反馈和做出决策，从而使得智能体在一个时间段内获得的收益最大或完成既定任务。强化学习与监督学习和无监督学习的一大不同点在于它不通过学习既有数据集获得信息，而是以行为主义心理学的“探索-利用”模式直接从环境中学习知识并且权衡存在冲突的收益。鉴于这一特点，强化学习常被应用于博弈[77,78]和多智能体等领域[79]。

2.1.2 卷积神经网络

卷积神经网络是深度学习的主要内容，最早可追溯至 LeCun 等人于 1998 年提出的用于数字识别的 LeNet[9]。LeNet 由特征提取器和分类器构成，

前者通过卷积层的局部感受野和基于滑动窗口的权重共享，解决了传统 BP 神经网络在图像处理中纵向特征丢失和模型体量过大等问题，此外 LeNet 的网络结构对后来的网络设计产生了深远的影响。随着大规模数据集的出现和 GPU 的普及，Krizhevsky 等人沿用 LeNet 的结构提出了 AlexNet[2]，并在 2012 年 ImageNet 图像识别挑战赛以大优势取胜，卷积神经网络再次成为热点。与 LeNet 相比，AlexNet 设计了更多卷积层和全连接层，大幅提升了模型的特征表达能力，同时它也成为了浅层网络和深度网络的分界。虽然 AlexNet 展示了深度模型的强大特征表达能力，却没有提出设计深度模型的指导性准则。NIN[80] 提供了一种模块化的深度模型构建准则，它将多个小型网络串联起来构成深度网络。Simonyan 等人的 VGG[5] 在特征提取器中重复使用基本块构造深度模型，提供了另一种深度模型设计准则，他们还采用两个串联 3×3 卷积核取代一个 5×5 卷积核，在维持模型表现的同时将参数量减少了 28%。与 VGG 同一时期的 GoogLeNet[6] 设计了多分支并行的 Inception 结构，不同分支上设置不同尺寸的卷积核以便提取不同尺度的特征信息，并借鉴 NIN 的设计串联多个 Inception 以构建深度网络。虽然 AlexNet 的成功揭示了模型越深其特征表达能力越强的规律，但实际应用中深度模型却出现网络退化现象[3,81]，这严重阻碍了深度模型的应用与发展。He 等人提出的 ResNet[3,81] 通过残差结构解决了网络退化问题以及梯度消失/爆炸问题[82]，使深度模型在实际应用变得可行。

卷积神经网络发展至今涌现了许多设计巧妙的结构和诸如 VGG 和 ResNet 等经典模型[83]，现代卷积神经网络通常包含以下基本组件：

（1） 卷积层

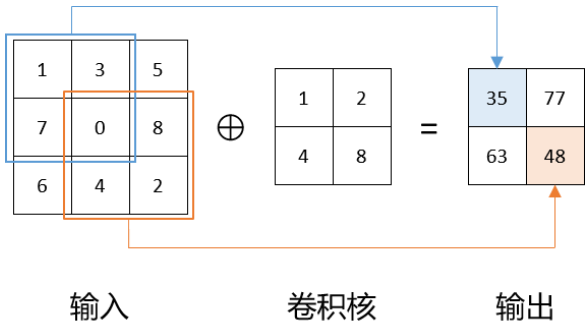


图 2-1 二维卷积核的运算过程

卷积层是卷积神经网络的核心组件，它由若干个卷积核构成[84]，其输入输出都被称为特征图（feature map）。这些特征图包含若干个通道，每个通道都是对目标某一特征的表达。卷积层的任务是提取输入特征图中各个通道表示的特征信息，并将这些信息映射到更高抽象级的特征空间中。卷积层虽然

得名于卷积（convolution）运算，但其实际的操作是互相关（cross-correlation）运算，其运算过程如图 2-1 所示。对于一个输入特征图，卷积核与其所有尺寸相等的子图分别进行点乘运算和求和运算得到输出特征图。在实际应用场景中，单个卷积核对目标特征的提取是远远不足的，因此现代卷积核通常是三维的，卷积层参数尺寸可表示为 $(c_{out}, c_{in}, k_h, k_w)$，其中 c_{out} 表示输出通道数（卷积核数），c_{in} 表示输入通道数，k_h 和 k_w 分别表示卷积核的高度和宽度。在实际应用中，为了从样本中提取出足以用于预测的特征信息，卷积神经网络通常需要多个卷积层。

（2） BN 层

BN 层（batch normalization）通过对输入数据的归一化解决内部协变量偏移（internal covariate shift）现象引起的深度网络训练困难问题[85]。由于训练时数据分批次输入，不同批次数据在一定程度上拥有独特的分布，这使得网络中间输出的分布不断变换，且这种变化随网络深度的增加而累积，这种现象被称为内部协变量偏移，它将增加深度网络的训练时间和难度。BN 层对其输入进行归一化处理，将任意分布的数据转换成均值为 0 标准差为 1 的分布，从而消除内部协变量偏移，降低网络的训练难度[86]。在卷积神经网络中，BN 层通常设置在卷积层和激活层之间，对卷积层的每个输出通道分别做批归一化。由于输入数据分布被强制变换，网络将无法学习到数据原始的信息，因此 BN 层在训练过程中为各通道训练一个缩放因子 γ 和偏置 β 用于变换重构，BN 层的训练和预测分别如算法 2-1 和算法 2-2 所示。

算法 2-1： 训练时 BN

输入： 当前批次样本集 $\mathrm{B} = \{x_1, x_2, \cdots, x_m\}$，可学习参数 γ、β

输出： 批归一化样本集 $\{y_i = BN_{\gamma,\beta}(x_i)\}$

1. $\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$; //当前批均值
2. $\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$; //当前批方差
3. $\widehat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$; //归一化
4. $y_i \leftarrow \gamma\widehat{x_i} + \beta \equiv BN_{\gamma,\beta}(x_i)$; //变换重构
5. **return** $\{y_i\}$;

算法 2-2： 预测时 BN

输入： 待预测样本 x，可学习参数 γ、β，训练时样本均值 μ_B，训练时样本方差 σ_B^2

输出： 批归一化样本后的待预测样本 y

1. $\mathrm{E}[x] \leftarrow E[\mu_B]$;
2. $\mathrm{Var}[x] \leftarrow \frac{m}{m-1}E[\sigma_B^2]$;
3. $\mathrm{y} \leftarrow \frac{\gamma}{\sqrt{\mathrm{Var}[x] + \epsilon}}x + \left(\beta - \frac{\gamma \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}}\right)$;
4. **return** y;

（3） 激活层

与卷积层和 BN 层不同，激活层不具有可学习参数，它实质上是一个增加网络中间输出非线性因素的函数。从卷积层的计算方式可知神经网络对样本特征的提取与加工是线性的仿射变换（affine transformation），因此在线性的任务中，如图 2-2 左所示，网络能够很好地找到分类依据。然而现实中的任务存在大量的非线性特征，如图 2-2 右所示，线性的学习器无法对这些样本进行正确预测，需要为学习器增加非线性的拟合能力才能完成预测任务。常见的激活函数有 sigmoid、relu 和 tanh 等。

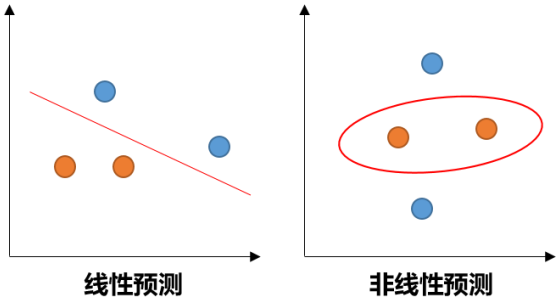


图 2-2 线性预测和非线性预测

（4） 池化层

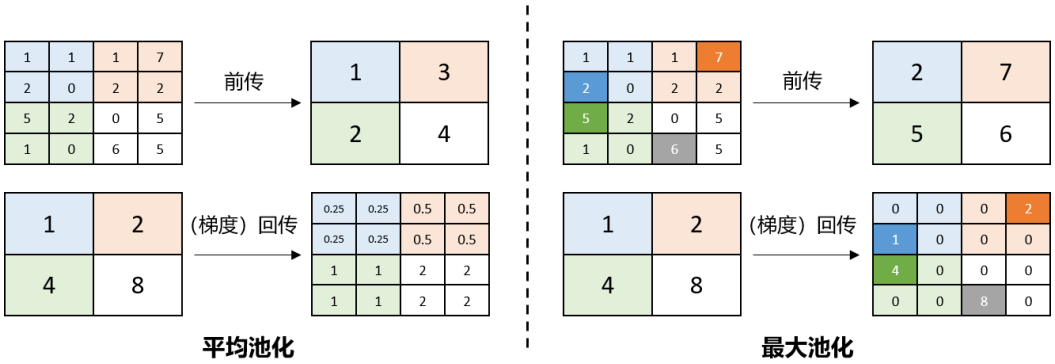


图 2-3 池化层的前向传播与反向传播

池化层也不具有可学习参数，它的任务是压缩输入特征图。一方面它将输入特征图中若干相邻像素点收缩成一个像素点，起到减小计算量的作用，例如一个 2×2 的池化层可以将特征图尺寸缩小至 1/4，进而将下一层计算量降低至 1/4；另一方面通过压缩特征图提取主要特征，降低模型在提取目标特征时对空间位置的依赖性[87]，比如目标检测中，样本往往将目标置于图片中央，池化层能够防止网络在学习目标轮廓、纹理和颜色等特征的同时错误地将空间位置也视作目标特征之一，避免目标处在图片边缘时的漏检误检问题。另外，池化层也起到增加非线性和扩大感受野的作用[88]。池化操作一般

有平均池化和最大池化两种。平均池化将所选区域内像素点的平均值作为池化后的像素值，在模型训练时的回传（backward propagation）机制中，一个像素点的梯度将平均分给所选区域内每个像素点，如图 2-3 左所示；最大池化将所选区域内最大的像素值作为池化后的像素值，并且标记下这个像素点的位置，以便池化层在回传中将梯度传给这个像素点，如图 2-3 右所示。

（5） 全连接层

卷积层和池化层等将目标特征映射到高抽象级的表示空间，全连接层的任务则是将这些特征表示映射到目标标签空间[89]，从而完成对样本的分类或回归任务。对于目标在某一抽象级上的特征表示（输入特征图）X，全连接层的计算过程如式（2-1）所示。

$$\mathrm{Y} = \mathrm{WX} + \mathrm{b} \tag{2-1}$$

式中 Y 表示全连接层输出，W 和 b 分别表示全连接层的权重矩阵和偏置向量。当全连接层作为网络的最后一层时，它的输出向量就代表这网络对当前样本的预测结果：在分类任务中，输出向量表示样本在各个标签上的得分；在回归任务中，输出向量表示样本的回归值。由于全连接层参数量与计算量较大，一些高性能网络在实现中采用其它层替代全连接层，如 ResNet 和 GoogLeNet 等采用全局平均池化（global average pooling）实现样本特征表示到标签空间的映射取得了更高的模型性能[6,81]。但全连接层并非完全可取代，Zhang 等人在实验中发现包含全连接层的网络在迁移学习中，特别是源域和目标域差异较大的情况下，展现出更强的迁移能力[90]。

2.2 模型压缩

模型压缩旨在达到模型精度于计算开销的均衡，使计算规模庞大的神经网络适配资源受限的边缘设备。

2.2.1 权重量化

权重量化是一种以低 bit 数据格式存储神经网络权重的模型压缩方法，如 16-bit、8-bit 和 4-bit 等，它能有效地降低模型所占存储空间并提高模型的前向传播速度，其核心过程是原始权重到量化权重的映射，如式（2-2）。

$$\begin{aligned} q &= \frac{r}{S} + Z \\ S &= \frac{r_{max} - r_{min}}{q_{max} - q_{min}} \\ Z &= q_{max} - \frac{r_{max}}{S} \end{aligned} \tag{2-2}$$

式中 r 和 q 表示原始权重和量化后权重，脚标的 max 和 min 分别表示最大值和最小值，S 表示缩放因子，Z 表示零点。Rastegari 等人提出的 XNOR-Net 第一次在 ImageNet 上进行对二值网络的评估[21]，Hubara 等人探究了神经网络量化与二值网络的联系[91]，Lin 等人试图建立多个 1-bit 模块并以组合的形式逼近全精度（FP32）模型的权重分布[22]，Zhou 等人提出了自适应的量化方法从而灵活地为模型的不同层进行权重量化[92]，Liu 等人在 ResNet 中加入全精度的下采样连接提升了 1-bit 网络的精度[93]。现代权重量化技术已经比较成熟，根据量化间隔可分为线性量化和非线性量化，根据量化进行的时机可分为后量化（PTQ）和量化感知训练（QAT）[94]。

（1） 线性量化与非线性量化

线性量化采用相同的量化间隔对网络权重进行量化，根据量化前后的零点变化情况又可分为非对称量化和对称量化[95]。非对称量化前后的零点是不对应的，即原始权重分布中的零点在量化后不再是零点，在这种量化方式中权重的映射可表示为式（2-3）。

$$\begin{aligned} \mathrm{x} &= \mathrm{round}\left(\frac{r}{S}\right) + Z \\ q &= clamp\left(0, 2^{bit}-1, x\right) \end{aligned} \tag{2-3}$$

式中 round(·) 表示四舍五入，$2^{bit}-1$ 表示量化后位数可表示的最大值（无符号），clamp(·) 表示式（2-4）所示计算过程。

$$clamp(a,b,x) = \begin{cases} a, x \le a \\ x, a < x < b \\ b, b \le x \end{cases} \tag{2-4}$$

在对称量化中原始权重的零点量化后仍是零点，其权重映射表示为式（2-5）。

$$\begin{aligned} x &= round\left(\frac{r}{S}\right) \\ q &= clamp\left(-2^{bit-1}, 2^{bit-1}-1, x\right) \end{aligned} \tag{2-5}$$

对称量化受零点不变的约束，当原始权重分布空间在正负区间上不均匀时将导致非饱和量化。非对称量化不受此约束，具有更好的动态映射范围，然而在量化后模型的前向传播过程中具有比对称量化更高的计算成本。一般而言，对于卷积层这类计算量大且数据分布固定的模型组件采用对称量化，而对于激活层这类计算简单且与数据分布是动态的模型组件采用非对称量化。

非线性量化中量化间隔与网络权重相关，根据量化后权重分布空间是否覆盖全部的取值空间又可分为非饱和量化与饱和量化。非饱和量化是采用一种直接的权重映射方式，即将原始权重分布空间的绝对值的最大值映射为量化后权重分布空间的边界值。当原始权重在正负区间分布不均匀（最大值与最小值的绝对值不相等）时，非饱和量化后的权重分布空间同样是不均匀的，

即权重分布空间不能覆盖全部的取值空间，如图 2-4 左所示。由于不饱和量化中有一部分动态范围被浪费，量化后模型精度将出现一定程度的损失。饱和量化的权重映射方式则是在原始权重分布空间中寻找一个截断点 T，并将它映射为量化后权重分布空间的边界值，原始权重中绝对值大于 T 的值也将被映射为量化后权重分布的边界值，如图 2-4 右所示，由于量化后权重分布空间能够覆盖全部的取值空间，饱和量化可以保证量化后的动态范围不被浪费。另外，饱和量化一般需要使用者对不同截断点进行评估才能够确定最优解，如 tensorRT 通过最小化量化前后权重分布的相对熵选取截断点。

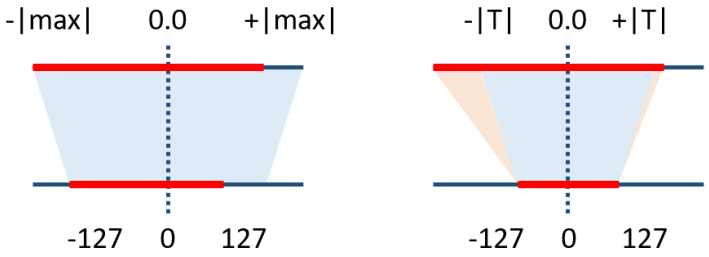


图 2-4 非饱和量化与饱和量化

（2） 后量化与量化感知训练

根据量化进行的时机可分为后量化是在模型训练完成后进行的量化，它只需要很少的数据作为校准集，可以在几乎不调整超参数的情况下完成权重量化。量化感知训练则是在模型训练过程中进行的量化，它通过在原模型训练过程中插入伪量化节点，用以寻找原模型权重分布（主要是最大值和最小值），并向原模型损失函数中引入模拟量化后的精度损失，一起指导模型训练。量化感知训练将整个训练集当作校准集，并从中寻找最合适的量化因子，因此其量化效果在理论上优于后量化。

还有一些专注于低 bit 量化的研究，如二值量化和三值量化等。二值量化是一种 1-bit 量化，其权重取值为 $\{-1,1\}$，它起源于二值化网络[96]。这种量化的权重映射方式有决策式和随机式两种。决策式二值量化采用符号函数对原始权重进行量化，如式（2-6），

$$\mathrm{q} = \begin{cases} -1, r < 0 \\ +1, 0 \le r \end{cases} \qquad (2-6)$$

式中 q 和 r 分别表示量化后权重和原始权重。随机式二值量化则对原始权重进行 sigmoid 运算，运算结果的表示量化后权重取值为 1 的概率，如式（2-7）。

$$\mathrm{q} = \begin{cases} +1, with\ \ probability\ \ \sigma(r) \\ -1, with\ \ probability\ \ 1-\sigma(r) \end{cases} \qquad (2-7)$$

式中 σ 表示 sigmoid 运算。三值量化的权重取值为 $\{-1,0,1\}$，它将原始权

重分布空间中寻找量化阈值 Δ，并按照式（2-8）对原始权重进行量化。

$$q = \begin{cases} -1, r < -\Delta \\ 0, -\Delta \leq r \leq \Delta \\ 1, \Delta < r \end{cases} \quad (2-8)$$

式中 q 和 r 分别表示量化后权重和原始权重。而量化阈值 Δ 的选择标准是最小化原始权重和量化后权重的欧氏距离。相比之下，三值量化比二值量化的存储多 1-bit，但量化后的模型精度更高[97]。

虽然 8-bit 及以上精度的量化可以达到量化后模型精度几乎不下降，但低 bit 量化在大规模数据集上仍然会出现明显的精度损失，如二值量化在 ImageNet 上相比于全精度模型精度损失超过 10%。如何降低低 bit 量化带来的模型精度损失将会是未来权重量化的一个重要研究方向。

2.2.2 知识蒸馏

知识蒸馏是一种基于“教师-学生”模式的模型压缩方法，其基本思路是将大体量的预训练模型（教师模型）的知识迁移至小体量的待训练模型（学生模型）[26]。在一般的模型训练中，模型接收输入数据后通过调整内部可学习参数使输出分布与标签相拟合。在知识蒸馏中，学生模型接收与教师模型相同的输入数据，其输出分布拟合的目标是教师模型的输出分布，如图 2-5 所示。

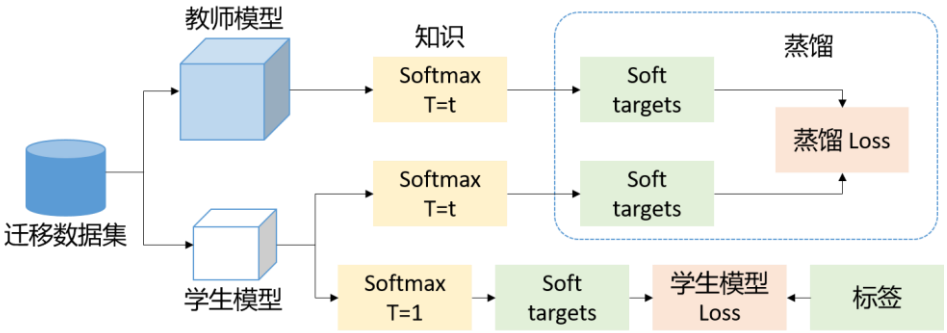


图 2-5 知识蒸馏框架[26]

常规分类模型的输出层采用 softmax 强化各输出节点的数值差异，再通过归一化得到一个近似 one-hot 的输出向量。这样的输出向量关注一种标签的预测结果，忽略其它标签的相关信息，这并不利于学生模型的软性拟合。知识蒸馏为 softmax 引入参数 T，如式（2-9）。

$$p_i = \frac{e^{o_i/T}}{\sum_j e^{o_j/T}} \quad (2-9)$$

式中 p_i 和 o_i 分别表示第 i 个输出节点的预测评分和 softmax 输入。在知

识蒸馏过程中选用较大的 T，使教师模型的 softmax 输出包含所有标签的预测信息，从而利于学生模型对教师模型的软性拟合。当学生模型用于预测样本时 T 将取 1，此时式（2-9）变为传统 softmax。由于学生模型训练中 T 的变化与化学中利用沸点不同分离混合物的蒸馏方法相似，知识蒸馏因此得名。

知识蒸馏中主要关注知识（knowledge）、蒸馏策略（distillation schemes）以及师生结构（teacher-student architecture）[98]。

知识形式主要分为基于响应的知识（response-based knowledge）、基于特征的知识（feature-based knowledge）和基于关系的知识（relation-based knowledge）三类[99]。基于响应的知识指的是教师模型的最后一层逻辑输出，即让学生模型学习教师模型的输出，这是最简单有效的知识形式，也是 Hinton 等人提出的最初的知识系统[26]；基于特征的知识将教师模型的中间层输出也纳入知识系统，可以看作基于响应的知识的扩展，如 Passalis 等人通过特征空间概率分布的匹配实现知识迁移[100]等；基于关系的知识则深入地探索不同模型组件和不同样本之间的关系，如 Lee 等人使用奇异值分解提取特征图的相关信息[101]，Passalis 等人让学生模型拟合教师模型内的信息流动[102]，Liu 等人通过个体和关系图进行知识蒸馏使迁移的知识包含个体特征[103]。

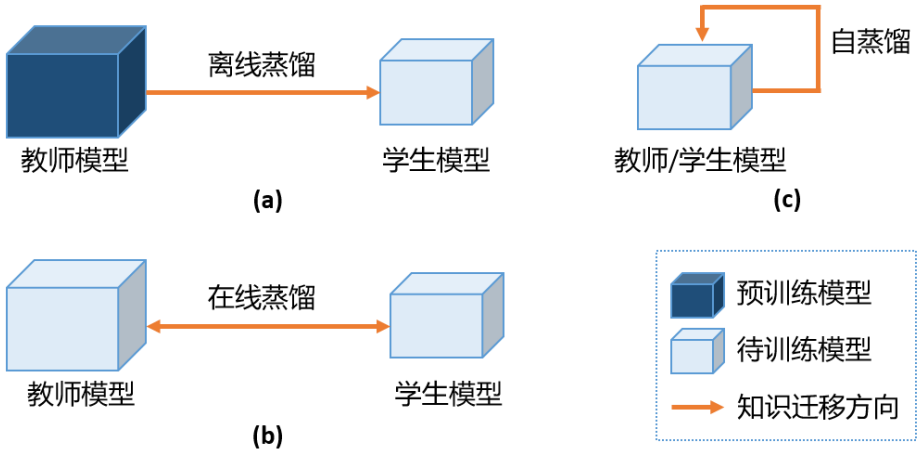


图 2-6　三种蒸馏策略

蒸馏策略主要分为离线蒸馏（offline distillation）、在线蒸馏（online distillation）和自蒸馏（self-distillation），如图 2-6 所示。离线蒸馏包括训练教师模型和知识迁移两个阶段，教师模型单方面地指导学生模型，虽然知识迁移十分高效，但也导致学生模型对教师模型的过度依赖，这种方法是 Hinton 等人在提出知识蒸馏时所使用的[26]；在线蒸馏中，知识迁移不再是教师模型到学生模型的单向行为，而是两个模型相互学习同时更新，如 Zhang 等人设置多个未经训练的模型相互学习[23]，这种方法在高性能教师模型不易

获得时尤为适合；自蒸馏则是同一个模型即当教师又当学生，可以视作在线蒸馏的一种特例，如 Zhang 等人在模型内部不同抽象级的特征图之间进行知识蒸馏[24]，Ji 等人通过不同抽象级特征图的整合与细化在同级特征图中进行知识蒸馏[25]。

师生结构是知识迁移的载体，若教师模型与学生模型的特征表达能力差距太大，蒸馏容易发生知识溢出，而过于复杂的学生模型虽然能有效避免知识的溢出，却难以提供理想的模型压缩效果，因此教师模型和学生模型的结构设计是知识蒸馏中的重要问题。然而知识蒸馏相关研究并没有予以足够的重视，现有的学生模型通常有以下选择：（1）教师模型的简化版，（2）教师模型的量化版，（3）一个结构完整的小型网络或（4）与教师模型相同。为了使学生模型与教师模型匹配，近年来有一些研究提出了结合神经网络结构搜索的知识蒸馏。

2.2.3 神经网络结构搜索

模型效果的好坏很大程度上取决于超参数的设置，在传统机器学习中有许多自动化搜索超参数的算法，如随机搜索（random search）、贝叶斯优化（bayesian optimization）和进化算法（evolution algorithm）等，这些方法统称超参数优化（hyperparameter optimization）。对于深度学习而言，超参数不仅限于学习率（learning rate）、批大小（batch size）和权重衰退因子（weight decay）等训练参数，还包括模型深度、各层算子和卷积层宽度等定义模型结构的超参数，这些超参数具有离散和相互关联等特点。训练参数的自动化搜索仍然可以采用超参数优化算法，而定义模型结构的超参数则需要另一种搜索策略，即神经网络结构搜索。

NAS 流程通常包括定义搜索空间、基于搜索策略寻找候选网络结构以及模型评估与反馈等步骤，其中寻找候选网络结构与模型评估是一个寻找最优解的循环过程。最原始的搜索思想是在搜索空间下遍历所有网络结构并从中选择最优解，然而现代神经网络结构复杂多样，遍历式的搜索难以实现，这驱使着研究者们寻找更优的搜索策略：**（1）基于强化学习的搜索策略。**Baker 等人提出了 MetaQNN，通过对一系列神经网络算子和相关参数的选取生成候选模型，并将模型训练后的测试集精度作为强化学习的环境反馈，用以指导 MetaQNN 下一轮的候选模型生成[104]。Zoph 等人将神经网络结构描述成一种特征表示（包括模型深度、卷积层数和卷积核尺寸等定义模型结构的超参数），并采用循环神经网络（RNN）作为控制器搜索“好模型”的特征，从而实现神经网络结构的启发式搜索[27]。**（2）基于进化算法的搜索策略。**Real 等

人对神经网络进行编码，在进化过程中维持一个网络结构集合，在这个集合中每个网络结构的测试集精度作为它们的适应度（fitness）。如图 2-7 所示，每次进化将随机选择两个网络，淘汰其中适应度较低的网络，另一个网络得到保留并作为父节点发生随机的结构变异从而生成子节点，然后将子节点加入网络结构集合[28]。**（3）基于梯度的搜索策略。**基于梯度的搜索策略关键在于将搜索空间转换为连续的。Liu 等人将神经网络结构搜索建模为含有 n 个有序节点的有向无环图，其中节点和遍分别表示隐式特征和算子，并且采用 softmax 对候选操作进行混合，实现了搜索空间从离散转为连续，继而采用梯度优化的方法搜索最优网络结构[105]。Luo 等人则将网络结构嵌入一个连续空间，在这个空间中定义模型精度的预测函数并以之建立目标函数，从而实现了目标函数的连续和可导[29]。

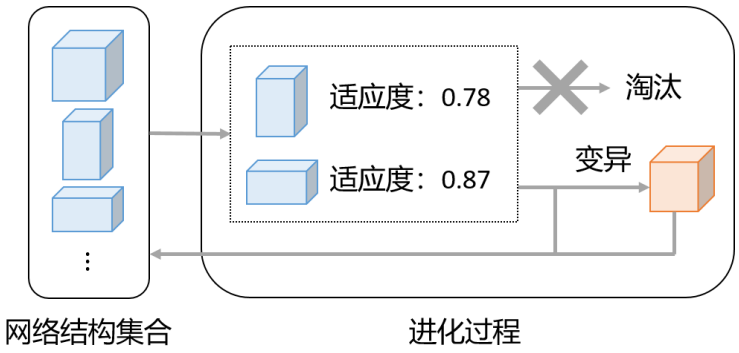


图 2-7　基于进化算法的搜索策略

另外，还有一些其它类型的搜索策略，如基于集成学习[106]和基于贝叶斯优化[107]的搜索策略。

这些搜索策略虽然使神经网络结构搜索得到实现，却无法避免巨大搜索空间带来的高昂搜索成本，如 Baker 等人在他们的工作中为 SVHN 和 CIFAR10 等数据集花费了超过 200 个 GPU 小时[104]，Zoph 等人在他们的工作中为 CIFAR10 动用了 800 颗 GPU[27]，这使得 NAS 成为一种奢侈技术。为此，一些研究者将关注点放在 NAS 的加速上：**（1）层次化表示（hierarchical representation）。**Zoph 等人提出的 NASNet 将神经网络视作由基本单元（cell）叠加而成的，将 NAS 转换成对两种基本单元（normal cell 和 reduction cell）的结构搜索，大幅压缩了搜索空间[108]。Liu 等人将神经网络的搜索空间分割成不同层次的子空间，最底层为卷积层和激活层等基本网络组件，中间层为网络组件构成的图，最高层为多个图构成的神经网络[109]。**（2）权重共享（weight sharing）。**考虑到 NAS 最为耗时的部分在于累次的候选模型训练，一些研究采用了网络权重共享，在下一个候选模型中复用上

一个候选模型的权重。Elsken 等人采用网络态射（network morphisms）让网络在发生形变时保持功能不变[110]。Pham 等人将 NAS 视作大图中寻找子图的过程，在一个大体量、高性能的网络子结构中进行最优模型的搜索[111]。Zhang 等人提出了一种只需搜索一次的 NAS，它在一个经过充分训练的大型网络中寻找并移除不重要的连接，从而得到最优网络结构[112]。**（3）表现预测（performance prediction）。**在 NAS 中对候选模型的训练是为了获得它们的测试集精度从而评估候选模型，一些研究企图通过其它方式评估候选模型避免或降低训练成本。Liu 等人使用 LSTM 作为代理模型，根据候选模型的结构预测其测试集精度，从而指导 NAS[113]。Klein 等人利用贝叶斯神经网络对候选模型的学习曲线进行建模和预测，从而降低 NAS 过程中候选模型的训练开销[114]。

NAS 可以在只有数据等少量输入的情况下自动地完成网络结构设计和模型训练，极大地降低了人工参与。除了搜索空间的定义外，搜索策略和加速方法等皆可沿用到模型压缩以外的其它领域，如优化器[115]等。

2.2.4 网络剪枝

网络剪枝通过移除模型中不必要的组件达到减小模型体量和提高模型前向传播速度的目的。早期的网络剪枝方法主要是非结构化剪枝[116]，这类剪枝方法针对的目标是单个神经元，由于剪枝后的网络权重是一个稀疏矩阵，若没有专门的硬件和计算库支持，无法带来实质性的性能提升。20 世纪 80 年代末，Hanson 等人提出了基于幅值的剪枝方法，通过对网络中每个隐藏单元施加与其绝对值相关的权重衰减来最小化隐藏单元数量，在最大限度保持网络精度的基础上进行裁剪[117]。由于当时的神经网络的影响力远不如今天，他们的研究在很长一段时间内并没有继续推进。20 世纪 90 年代初，LeCun 等人提出的 OBD[30] 方法和 Hassibi 等人提出的 OBS[31] 方法在反向传播中采用损失函数相对于权重的二阶偏导（Hessian 矩阵）衡量网络权重的重要程度，然后根据选定的阈值进行剪枝。由于 Hessian 矩阵计算复杂，这种剪枝方法难以应用在深度网络中。2015 至 2016 年间，Han 等人针对深度神经网络的压缩提出了一系列方法[118–120]，对 AlexNet 和 VGG 进行剪枝，将模型体量压缩至数十分之一。

不同于非结构化剪枝，结构化剪枝不会对网络原有的结构造成破坏，现有的设备能很好地支持剪枝后的网络推理，因此近年来的研究主要集中在结构化剪枝。根据剪枝目标粒度，结构化剪枝可以细分为层剪枝和通道剪枝。

层剪枝以网络中卷积层为剪枝目标。在神经网络中，由于存在逻辑先后关系，不同层的计算必须是串行的，而卷积层内各通道的计算没有这种限制，在多核设备的支持下它们是可以并行的，因此层剪枝的研究认为裁剪串行计算序列中的节点可以有效地降低网络推理耗时，如 Chen 等人将每一层特征表示对模型最终表现的贡献程度作为选择待剪枝层的标准[121]。还有一些根据硬件部署的情况来决定待剪枝的层，如 Yang 等人对比每个层的功耗或计算时延，将功耗大（时延高）的层作为优先剪枝的对象[122,123]。

通道剪枝以卷积层中的通道作为剪枝目标。最常见的通道剪枝是基于贪心思想的方法，它通过衡量卷积层中各通道的重要性，然后根据设定的阈值裁剪那些不重要的通道，理论上这样可使网络精度损失最小。现有通道剪枝方法主要可分为基于度量的通道剪枝与基于误差重建的通道剪枝。**（1）基于度量的通道剪枝。**这类方法将模型自身的某个指标作为待剪枝通道的选择标准，这类通道剪枝方法的基本思路是将幅值作为衡量通道重要性的指标。Li 等人将权重的绝对值大小用作衡量通道重要性的依据[40]。Liu 等人在 BN 层被广泛应用的背景下，为 BN 层引入通道级缩放因子并采用 L1 范数诱导其稀疏化，最后以这些缩放因子作为通道重要性的评估指标[32]，其剪枝算法的框架如图 2-8 所示，这类方法的算法框架基本上与之相似。基于幅值的方法是建立在 “smaller norm less informative” 假设之上的，然而 Ye 等人认为这个假设未必成立[124]，因此一些研究者提出其它度量方法。Molchanov 等人提出基于泰勒展开式的标准，通过循环地进行网络微调和剪枝，直到满足停止剪枝的条件[125]。Lee 等人将归一化的目标函数相对于参数导数的绝对值作为衡量通道重要性的指标[126]。**（2）基于误差重建的通道剪枝。**这类方法不再只关注于模型参数本身，而是将数据和输出也纳入考虑范围，其基本思想在于：如果对某一卷积层进行剪枝后，它的特征输出对后续的网络推理不会带来明显的影响，说明裁剪掉的是不重要的部分，如 Luo 等人和 He 等人通过最小化特征重建误差（feature reconstruction error）确定可以裁剪的通道[41,42]。

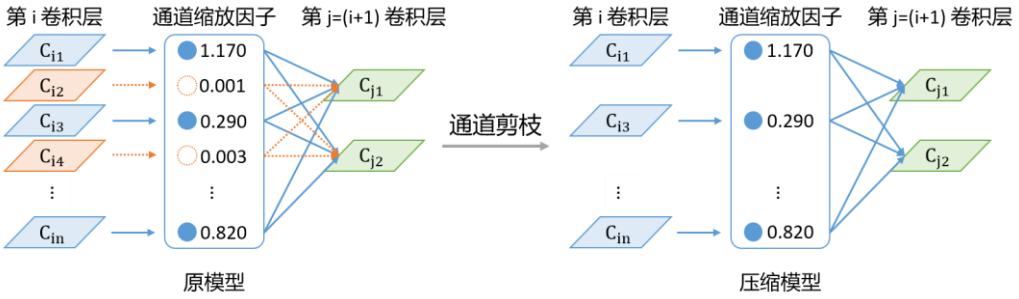


图 2-8　通道剪枝算法框架[32]

还有一些采用其他标准衡量通道重要性的方法，如 He 提出的 FPGM 将剪枝的目标从不重要的通道转移到冗余的通道[127]。

有一些研究认为通道剪枝不应忽略模型参数间的联系，否则剪枝难以得到最优解，如 Ashok 等人利用强化学习分别进行网络的层移除和层缩小[128]，Peng 等人将剪枝通道的选取建模为约束下的二次规划问题[129]，Ding 等人采用聚类的方法计算通道间或特征图间的相似性，通过裁剪冗余通道完成网络剪枝[34]。

近年来一些研究将关注点放在自动化上，这些研究将 NAS 的思想引入网络剪枝中。He 等人提出的 AMC 将强化学习引入通道剪枝[33]，开创了自动化剪枝的先河。Yu 等人借鉴 Cai 等人的 SuperNet 设计，在原模型上采用贪心法逐步地进行剪枝[130,131]。鉴于自动化剪枝的易用性，与 NAS 的结合将是未来网络剪枝的重要研究方向。

2.2.5 轻量化模型设计

与权重量化、知识蒸馏和网络剪枝等方法相比，轻量化模型并不在大体量模型上做压缩，而是设计计算更高效的网络结构，在提高模型前向传播的同时维持较高的特征表达能力，是一种非常规的模型压缩方法。

轻量化模型设计的一个重要方式是将标准卷积转换为深度可分离卷积（depth-wise convolution）。标准的卷积层计算如图 2-9 左所示，每个输出特征都由全部的输入特征计算而得，这种卷积层的参数量 $params$ 和浮点数计算量 $FLOPs$ （含 bias）可近似表示为式（2-10）和式（2-11）。

$$params = c_{out} \times c_{in} \times k_h \times k_w \tag{2-10}$$

$$FLOPs = c_{out} \times h \times w \times (c_{in} \times k_h \times k_w + 1) \tag{2-11}$$

式中 c_{out} 和 c_{in} 分别表示卷积层输出特征数和输入特征数，k_h 和 k_w 分别表示卷积核的高和宽，h 和 w 分别表示特征图的高和宽。分组卷积如图 2-9 右所示，输入特征和输出特征被分为若干组，每个组内的特征进行标准卷积层运算，这种卷积层的参数量 $params_g$ 和浮点数计算量 $FLOPs_g$ （含 bias）可近似表示为式（2-12）和式（2-13）。

$$\begin{aligned} params_g &= \frac{c_{out}}{g} \times \frac{c_{in}}{g} \times k_h \times k_w \times g \\ &= \frac{1}{g} \times c_{out} \times c_{in} \times k_h \times k_w \end{aligned} \tag{2-12}$$

$$\begin{aligned} FLOPs_g &= \frac{c_{out}}{g} \times h \times w \times \left(\frac{c_{in}}{g} \times k_h \times k_w + 1\right) \times g \\ &= \frac{1}{g} \times c_{out} \times h \times w \times (c_{in} \times k_h \times k_w + g) \end{aligned} \tag{2-13}$$

式中 g 表示卷积层的分组数。可以看到一个 g 组的分组卷积层使参数量和计算量下降至 $1/g$。深度可分离卷积是分组卷积的极端情况，它的分组数量与输出特征数量相等，在深度可分离卷积中有 $g = c_{out} = c_{in}$，此时卷积层参数量 $params_d$ 和浮点数计算量 $FLOPs_d$ （含 bias）可近似表示为式（2-14）和式（2-15）。

$$\begin{aligned} params_d &= \frac{c_{out}}{c_{out}} \times \frac{c_{out}}{c_{out}} \times k_h \times k_w \times c_{out} \\ &= c_{out} \times k_h \times k_w \end{aligned} \tag{2-14}$$

$$\begin{aligned} FLOPs_d &= \frac{c_{out}}{c_{out}} \times h \times w \times \left(\frac{c_{out}}{c_{out}} \times k_h \times k_w + 1 \right) \times c_{out} \\ &= h \times w \times (c_{out} \times k_h \times k_w + c_{out}) \end{aligned} \tag{2-15}$$

分组卷积可以带来参数量和计算量的减少，但因为每个输出特征仅由一部分输入特征计算得到，无法像标准卷积那样综合全部的输入信息，这个现象在深度可分离卷积中更加明显。这种通道间的信息割裂是每个基于深度可分离卷积设计的轻量化模型必须要面对的问题。MobileNet 中在深度可分离卷积后添加一个输入输出维度相等的 1×1 卷积，在不改变通道数量的情况下综合各通道信息[36]。ShuffleNet 设计两个串联的分组卷积，将第一层的输出特征重组后传给第二层，如图 2-12 所示[132]。

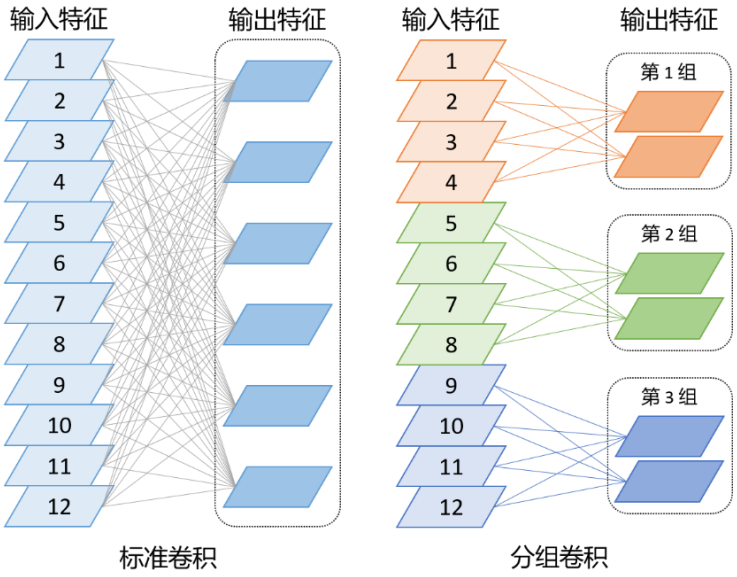


图 2-9 不同卷积中输入特征与输出特征的映射关系

现有研究已提出很多轻量化模型，如早期的 SqueezeNet，采用深度可分离卷积的 MobileNet 系列和 ShuffleNet 系列，以及 GhostNet 等。

（1） SqueezeNet

Iandola 等人提出的 SqueezeNet 是最早期的轻量化模型之一，它设计了 Fire Module 模块[35]。如图 2-10 所示，Fire Module 由 squeeze 层和 expand

层组成，前者采用 1×1 卷积降低特征图通道数，后者则并联 1×1 卷积和 3×3 卷积将特征图通道数回升，这种设计可以在保证网络对样本的充分学习下降低参数量和计算量。另外，Iandola 等人还总结了三种轻量化模型设计策略：1）用 1×1 卷积取代模型中的部分 3×3 卷积；2）减少 3×3 卷积的输入通道数；3）具有下采样功能的层部署在模型后部。值得一提的是，SqueezeNet 中采用 1×1 卷积调整特征图通道数的设计对后来的相关研究影响深远。

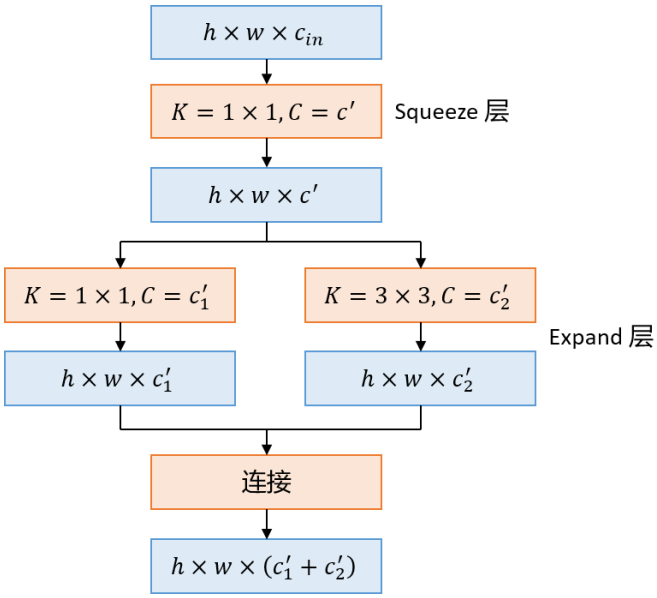


图 2-10 SqueezeNet 的 Fire Module 结构[35]

（2） MobileNet

MobileNet 系列有三代。MobileNet V1 提出了重要的深度可分离卷积，并用以取代标准的 3×3 卷积，明显地降低了神经网络的参数量与计算量[36]。MobileNet V2 在 V1 的基础上设计了倒置残差模块，如图 2-11 所示，第一个 1×1 卷积提升特征图通道数量，第二个 1×1 卷积再将之恢复，通道数量更多的 3×3 卷积能够更充分地提取样本特征[37]。MobileNet V3 则借鉴了 NAS 相关技术和针对硬件的优化[133]。

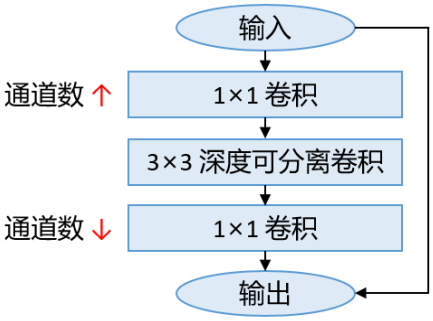


图 2-11 MobileNet V2 中的倒置残差结构[37]

（3） ShuffleNet

ShuffleNet 系列有两代。ShuffleNet V1 同样采用了 3×3 深度可分离卷积和残差块，与 MobileNet V2 不同的是，它采用 1×1 分组卷积与通道重组取代了 1×1 标准卷积[132]，如图 2-12 所示。ShuffleNet V2 不仅考虑前向传播中计算的耗时，还考虑了内存读写的时间成本，并且设计了新一代的 ShuffleNet block，如图 2-13 所示[38]。另外，Ma 等人还总结了轻量化模型设计的四个指导原则：1）当输入输出通道数相等时卷积层运算所需 MAC（memory access cost）最少；2）大规模分组卷积会增加 MAC 开销；3）设计网络的分支结构会降低并行能力；4）诸如 ReLU 和 bias 等单元级操作的计算访存比很小。

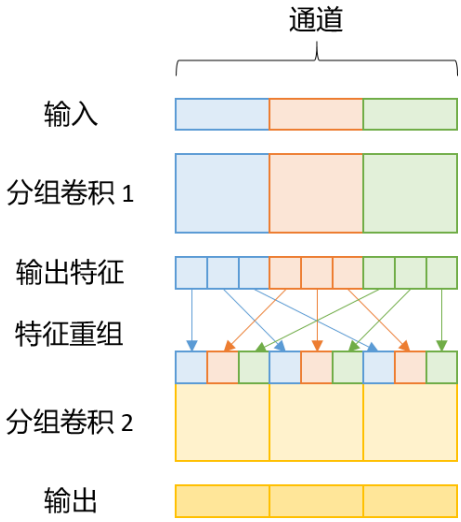


图 2-12 ShuffleNet 中的特征重组过程[132]

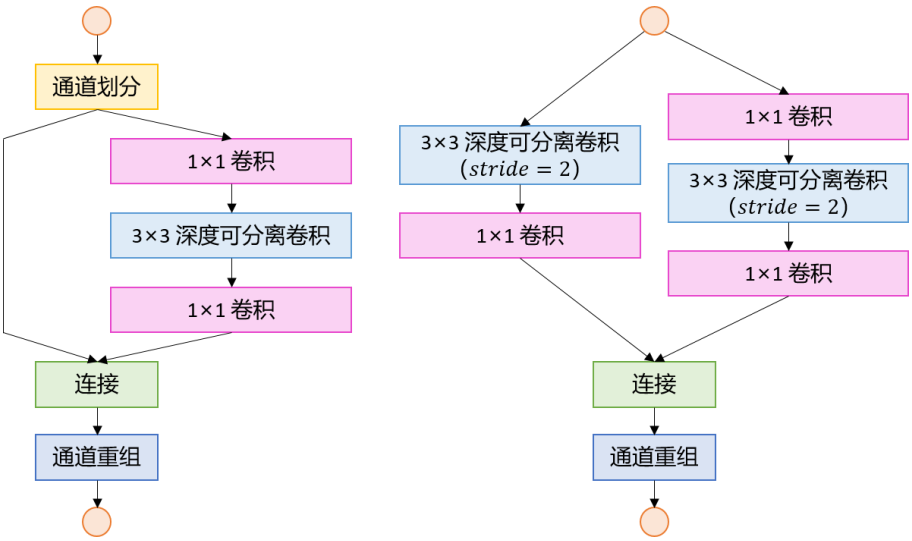


图 2-13 ShuffleNet V2 中的 ShuffleNet block 结构[38]

（4） GhostNet

Han 等人在对神经网络中间输出的可视化实验中发现，这些中间输出存在相似的特征图，他们通过计算成本更低的线性运算替代卷积运算生成这些特征图，如图 2-14 所示，在不改变特征图通道数量的情况下降低神经网络的参数量和计算量[39]。

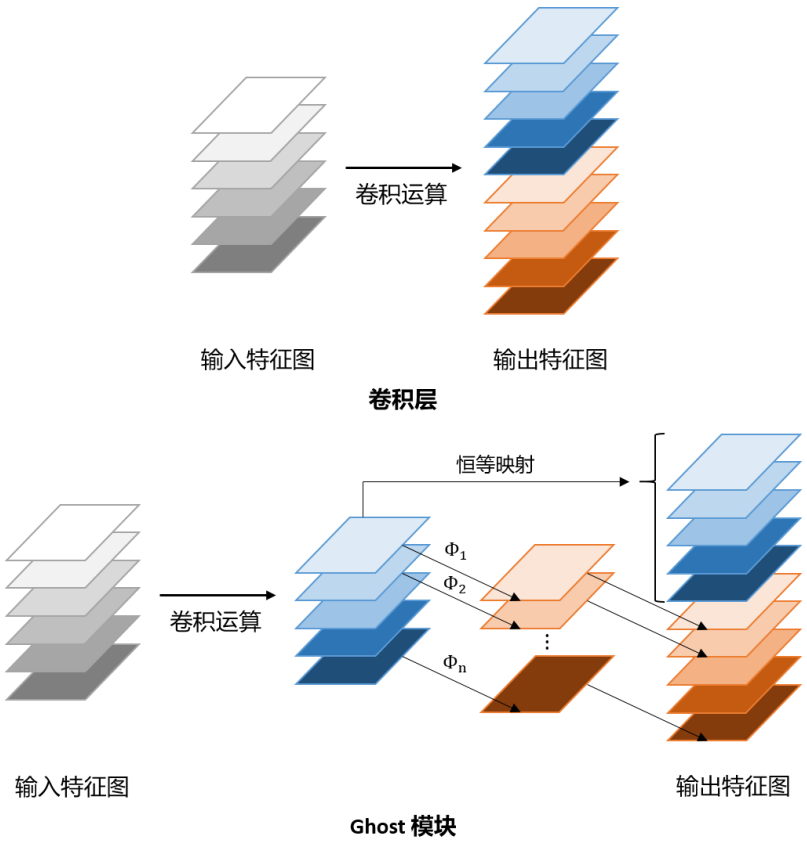


图 2-14 GhostNet 卷积层运算过程[39]

2.3 小结

本章介绍了深度学习方法、常见卷积神经网络及组件以及包括网络剪枝和神经网络结构搜索等模型压缩技术的相关理论和研究。本文认为通道剪枝的核心问题在于提出待剪枝通道的选择标准和确定适合给定任务的剪枝比例，结合 NAS 的自动化通道剪枝将这两个问题融合起来，为使用者呈现出一种端到端的剪枝方案。本文认为目前自动化通道剪枝方法仍然存在一些不足：（1）待剪枝通道的选择标准难以兼顾效率和可解释性；（2）针对原模型子结构空间的搜索使算法难以兼顾效率和自动化；（3）遍历式的搜索策略会为自动化通道剪枝算法带来高昂的成本。

3 基于权重置换的通道剪枝算法

3.1 引言

通道剪枝的一个关键问题在于提出一种合理的标准指导待剪枝通道（或待保留通道）的选取。现有相关研究大致分为三类：（1）基于度量的通道剪枝，这类方法旨在寻找一个通道重要性的衡量标准，结合使用者设置的剪枝率将重要性最低的若干通道移除。这类方法又可细分为基于权重、基于激活值和基于梯度三种。Li 等人基于 L1 正则化判断通道的重要性[40]，He 等人通过计算不同通道权重的相似性找出并移除冗余通道[127]，Luo 等人利用信息熵衡量通道重要性[134]，Molchanov 等人将激活值和权重零点处的泰勒展开式一阶项作为待剪枝通道的选择依据[125,135]。（2）基于误差重建的通道剪枝，这类方法通过最小化输出重建的误差指导剪枝过程，依靠通道对模型性能的影响程度选择待剪枝通道。Luo 等人提出的 ThiNet 在卷积层内寻找能够替代整个卷积层输出结果的通道子集[41]，He 等人采用 Lasso 回归最小化剪枝前后的误差[42]，Yu 等人通过最小化网络倒数第二层的误差指导待剪枝通道的选择[136]。（3）基于稀疏训练的通道剪枝，这类方法着眼于模型训练，通过各种正则化手段诱导网络权重变得稀疏，并选择值为零或接近零的权重进行剪枝。这类方法起源于非结构化剪枝，如 Hanson 等人采用的基于幅值的剪枝方法[117]，现在也有一些研究将这种方法转换为结构化剪枝。Huang 等人为网络引入可学习的 mask 并采用 APG 算法将之稀疏化进而实现结构化剪枝[137]，Zhang 等人采用 ADMM 算法诱导网络稀疏化[44]，Liu 等人通过 L1 正则化诱导 BN 层缩放系数稀疏化并将之当作选择待剪枝通道的依据[32]。

基于度量的通道剪枝无需经过计算即可完成待剪枝通道的选择，但选择标准主要着眼于网络权重本身，对数据（输出）在其中发挥的作用关注不足，并且这类方法基于 “smaller norm less informative” 假设，即根据幅值大小决定通道的裁剪与否，然而这种假设却未必是成立的[124]，因此这类剪枝的合理性难以证明。而在基于误差重建的通道剪枝中，研究者将 Loss 等结合数据的指标纳入考虑范围，虽然解决了潜在的剪枝合理性问题，却也引入了高额计算成本，如 Luo 等人提出的 ThiNet，对一个 n 通道卷积层的计算复杂度为 $O(2^n)$ [41]，He 等人在单个卷积层的输出特征图上做 Lasso 回归以寻找最具代表性的通道[42]，其计算复杂度更是难以量化。综合上述分析可以看出，这两类算法难以兼顾高效性和可解释性。

本章提出一种基于权重置换的通道剪枝算法，在待剪枝通道的选择标准

中兼顾了网络权重本身和数据（输出），结合了基于度量的方法的高效性和基于误差重建的方法的可解释性。受**排列重要性**[138,139]（permutation importance）启发，本章首先对卷积层通道进行权重置换，然后计算权重置换前后与该通道对应的输出特征图的变化，并且根据输出特征图的变化程度衡量通道的重要性，最后在给定剪枝率的指导下按照通道重要性对模型进行通道剪枝。模型在经过充分训练后，其任意一个卷积层的输出特征图都是目标在相应抽象级上的特征表示，而这些特征表示是由卷积层中各个通道的权重提取出来的。当通道权重发生置换后，与之对应通道的输出特征图将不再是目标的特征表示，并且特征图的变化程度反映了该通道对卷积层输出的影响程度。采用通道权重置换前后输出特征图变化程度衡量其重要性，在关注模型权重本身的同时将数据（输出）纳入考虑，使通道重要性的衡量结果更具有说服力，并且逐一计算通道重要性的复杂度为 $O(n)$。实验结果表明，本章提出的算法在不同模型和不同数据集上都能取得理想的压缩效果，并且在更高压缩率下能够获得更高的精度。因此，基于权重的通道剪枝算法不仅是第四章中提出的基于累计信息贡献率的全自动化通道剪枝算法的基础，同时也是一种独立可用的通道剪枝算法。

3.2 算法原理与实现

3.2.1 通道重要性

通道重要性是衡量网络中通道对模型性能影响程度的抽象指标，在实际研究中它通常由模型的某项数值表示，如权重幅值[40,127]等。一个卷积神经网络由许多层组成，包括卷积层、BN 层、激活层和全连接层等，每个层又由若干个通道组成。以多层感知机为例，它由若干隐藏层构成，每个隐藏层又由若干神经元组成，如图 3-1 左所示，对于输入数据 $\mathbf{x} = (x_1, x_2, x_3)^T$（其中 x_i 表示第 i 项特征），隐藏层的每个神经元对其进行计算并将中间结果传播至下一层直到最终输出层得到 $\mathbf{o} = (o_1, o_2)^T$（其中 o_i 表示第 i 项预测值），传播过程如式（3-1）所示：

$$\begin{aligned} \boldsymbol{h} &= \sigma(W_1\boldsymbol{x} + \boldsymbol{b_1}) \\ \boldsymbol{o} &= W_2\boldsymbol{h} + \boldsymbol{b_2} \end{aligned} \tag{3-1}$$

式中 W_1 和 W_2 分别表示 4×3 的权重矩阵和 2×4 的权重矩阵，σ 表示隐藏层的激活函数，$\boldsymbol{h}$ 表示隐藏层的中间输出，$\boldsymbol{b_1}$ 和 $\boldsymbol{b_2}$ 分别表示 4 维偏置向量和 2 维偏置向量。本文观察输出层 $\boldsymbol{o}$：

$$(o_1, o_2)^T = \left(w_{2,1}, w_{2,2}, w_{2,3}, w_{2,4}\right)\boldsymbol{h} \tag{3-2}$$

式中 $w_{2,i}$ 表示权重矩阵 W_2 第 i 个列向量（第 i 个通道权重），每个 $w_{2,i}$ 都对 $\boldsymbol{o}$ 有直接的影响，这种影响的程度即反映了 $w_{2,i}$ 的重要性（权重矩阵 W_1 的通道重要性与之同理）。

基于上述分析，可以认为神经元对其所属隐藏层输出的影响程度反映了其重要性。在卷积层中，通道（卷积核）是对输入进行处理的基本单元，它的地位等价于多层感知机中的神经元，因此本文将卷积层的通道重要性定义为某通道对其所属卷积层输出影响程度。

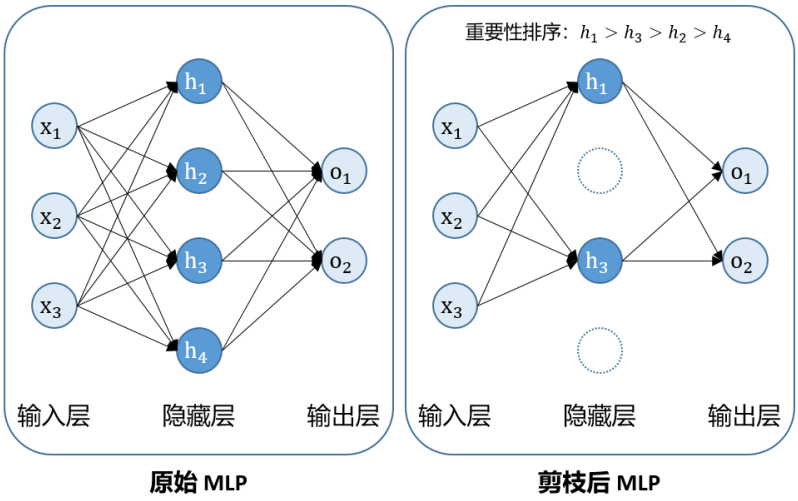


图 3-1 基于神经元重要性的 MLP 剪枝

基于通道重要性的通道剪枝算法考察网络中各通道对网络输出的影响力（即通道重要性），将重要性最低的若干通道作为剪枝目标从而平衡网络精度与计算开销，如图 3-1 右所示。由式（3-2）可知，当第 i 个通道的重要性较低时，$w_{2,i}$ 的元素取值变化对输出 $\mathbf{o}$ 的影响并不大，基于 $\mathbf{o}$ 的目标预测结果也不会产生明显的偏差，在此情况下移除第 i 个通道，即可在减少通道数量的同时尽可能维持网络对目标的预测精度。

3.2.2 基于权重置换的通道重要性度量算法

本文提出了一种结合待剪枝网络权重本身以及数据（输出）的通道重要性度量算法。在描述算法原理前，本章在此做符号定义：

（1） 每个卷积层对应的卷积核集合表示为 $\mathrm{K}=\{k_1,k_2,\cdots,k_{c_{out}}\}$，其中 k_i 表示卷积层中第 i 个通道对应的卷积核，c_{out} 表示当前卷积层的输出通道数，由于卷积层输出通道与卷积核一一对应，本文将它们视作同一概念。鉴于通道剪枝的目标是卷积层输出通道，如无特指，本文中的卷积层通道即指卷积层输出通道（卷积核）。

（2） 卷积层的第 i 个卷积核表示为 k_i，其尺寸为 (c_{in}, kh, kw)，其中 c_{in} 表示卷积层输入通道，kh 和 kw 分别表示卷积核的高和宽。

（3） 输入特征图表示为 X，其尺寸为 $(bs, c_{in}, h_{in}, w_{in})$，其中 bs 表示 batch size，$c_{in}$ 表示与卷积层输入通道数相对应的输入特征图通道数，h_{in} 和 w_{in} 分别表示输入特征图高和宽。

（4） 输出特征图表示为 Y，其尺寸为 $(bs, c_{out}, h_{out}, w_{out})$，$c_{out}$ 表示与卷积层输出通道数相对应的输出特征图通道数，h_{out} 和 w_{out} 分别表示输出特征图高和宽。

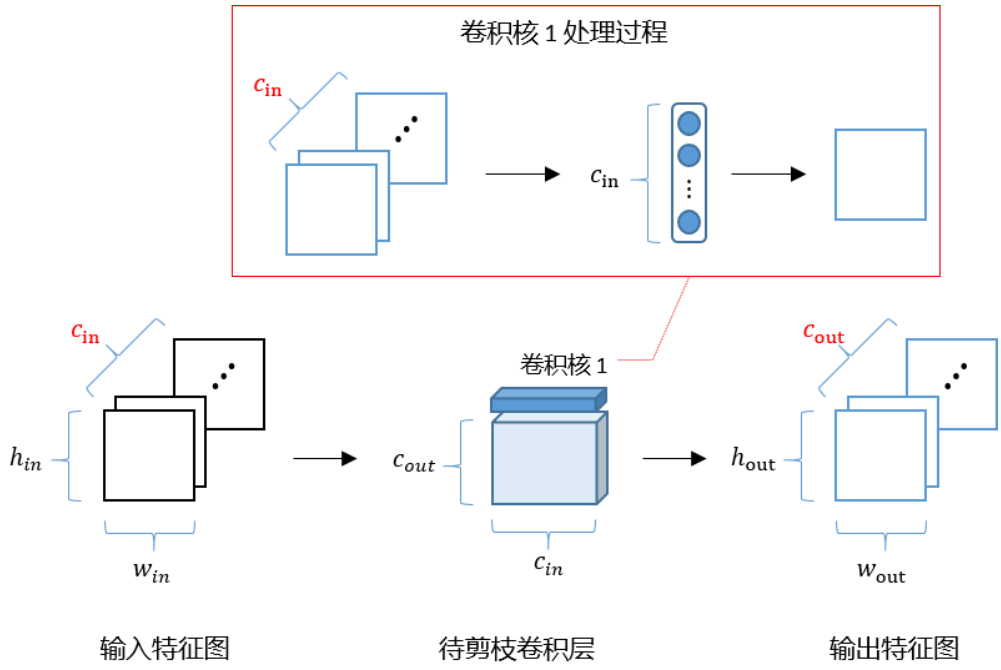


图 3-2 卷积层处理过程

对于一个待剪枝卷积层 K，输入特征图 X 和输出特征图 Y，输出特征图的每个通道都是由卷积层对应通道综合输入特征图所有通道后生成的，如图 3-2 所示，图中 c_{in} 和 c_{out} 分别表示卷积层输入通道数和输出通道数，h_{in} 和 w_{in} 分别表示输入特征图高和宽，h_{out} 和 w_{out} 分别表示输出特征图高和宽，红色框内展示了一个卷积层一个通道的处理过程。将输出特征图 Y 和待剪枝卷积层的卷积核集合 K 按照输出通道数拆分为式（3-3）所示的矩阵形式。

$$\begin{aligned} Y &= (y_1, y_2, \cdots, y_{c_{out}})^T \\ K &= (k_1, k_2, \cdots, k_{c_{out}})^T \end{aligned} \tag{3-3}$$

则卷积层处理输入特征图并生成输出特征图的过程可表示为式（3-4）。

$$(y_1, y_2, \cdots, y_{c_{out}})^T = (k_1, k_2, \cdots, k_{c_{out}})^T \oplus X \tag{3-4}$$

式中 $\oplus$ 表示卷积核的操作，其中第 i 个卷积核处理过程表示为式（3-5）。

$$y_i = k_i \oplus X \tag{3-5}$$

在可解释机器学习的特征重要性分析中，打乱测试集某项特征取值分布是为了破坏该项特征与模型在信息上的相关关系。模型经过充分训练后，其权重分布与训练集样本各项特征所包含的信息相对应，若模型在打乱测试集上能得到与在原测试集上相同水平的分数，则被打乱的特征对当前任务而言就是不重要的甚至是不相关的。反过来，在保持测试集样本特征不发生变化的基础上，将模型的某一卷积层的某一卷积核权重打乱，使该通道包含的任务信息无效化，若模型能够在测试集上得到与原模型相似或相同的输出，则被打乱的通道对当前任务而言是不重要的。

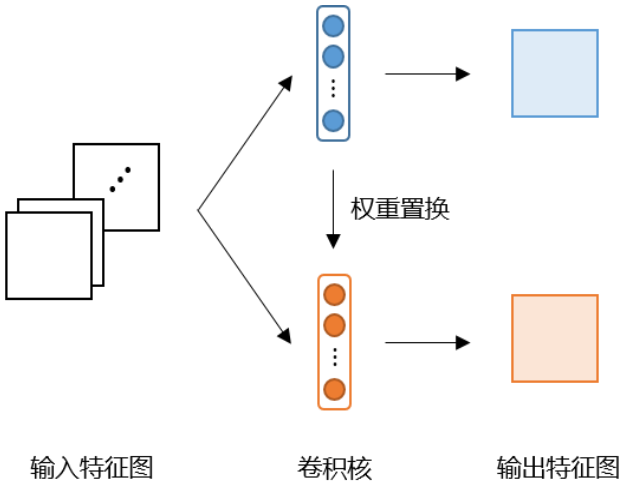


图 3-3　权重置换

特征图以图像的形式存储目标的特征信息，它的每个通道是由各卷积核对输入特征图的信息综合抽象后生成的，如式（3-5），它不仅反映了输入特征图的信息，也反映了卷积核的信息。因此，当卷积核 k_i 的权重发生变化时，输出特征图的相应通道 y_i 也将发生变化，如图 3-3。在模型经过充分训练后，其权重趋于稳定，此时对卷积层中某个卷积核的权重进行置换引起的输出特征图变化程度与该卷积核对卷积层输出的影响程度是正相关的，因此输出特征图在通道权重置换前后的变化可用作衡量通道重要性的标准，通道权重置换前后输出特征图的变化越大则通道重要性越高，反之通道重要性越低。在本文中将特征图的变化 D（Deviation）定义为其像素点差值的总和：

$$D = \sum_{C} \sum_{pixel} \left(val_{pixel} - val_{pixel}' \right)^2 \tag{3-6}$$

式中 C 代表输出特征图的通道，pixel 代表特征图像素点，val_{pixel} 和 val_{pixel}' 分别代表通道权重置换前后的特征图像素值，对一个 n 通道卷积层

的计算复杂度为$O(n)$。此处必须强调，式（3-6）中像素点差值需要采用平方（或绝对值）计算以避免因正负值相抵消导致的通道重要性误判。

当待剪枝卷积层的所有通道都独立地进行一次权重置换和输出特征图变化比较后，即可得到所有通道的重要性，在剪枝过程中根据使用者设定的剪枝率将其中重要性最低的若干个通道选作裁剪的对象。

3.2.3 算法实现

对于一个待剪枝卷积层，基于通道重要性的剪枝过程如算法 3-1 所示。基于权重置换的通道剪枝算法接收三个输入：待剪枝卷积核集合 K、输入特征图 X 和使用者设定的超参数剪枝率 pr。首先（1）根据 K 和 X 生成原输出特征图 Y，如式（3-5），然后（2）将 K 中的卷积核挨个地进行权重置换并通过输出特征图变化计算通道重要性，如式（3-6），接着（3）根据剪枝率确定剪枝的通道重要性阈值，最后（4）将通道重要性低于阈值的通道移除并输出剪枝后的卷积层。如此处理待剪枝网络的每个卷积层后即可得到剪枝后网络。算法 3-1 展示了基于权重置换的通道剪枝算法过程，在实际应用中还有一些细节，比如权重置换的具体方法。

算法 3-1：基于权重置换的通道剪枝 prune

输入：待剪枝卷积核集合 K，输入特征图 X，剪枝率 pr
输出：剪枝后的卷积核集合 K′

初始化空的通道重要性列表 DList;
K′ ← K; //复制原卷积核集合（剪枝在此复制集合中进行）
KBackup ← K; //生成待剪枝卷积核集合副本
Y ← K(X); //生成标准（原始）输出特征图
for each $k_i \in$ K **do**
    对 k_i 进行权重置换;
    Y′ ← $K(X)$; //生成权重置换后的输出特征图
    $D_i \leftarrow (Y - Y')^2$; //计算通道重要性
    DList ← DList ∪ D_i; //将 D_i 加入 DList
    K ← KBackup; //恢复待剪枝卷积核集合
end for
idx ← DList.length() × pr; //通道重要性阈值的序号
找到 DList 中第 idx 小的值 $thre$; //获取通道重要性阈值
for each $k_i \in$ K **do**
    if $D_i < thre$ **then**
        K′ ← K′ − {K_i}; //移除 K_i
end for
return K′;

（1） 权重置换

权重置换的目的在于破坏待剪枝卷积核通道中经训练得到的目标信息，

使输出特征图的相应通道不再反映出相应抽象级别的空间特征，在实现中可以通过以下几种方法达到这样的效果：

1）基于置零的权重置换。置零将选定通道的权重全部置为 0，使该通道不再对输入特征图进行任何处理。这种方法起到了预剪枝的作用，即令待裁剪通道无效化，但仍然将通道留在卷积核中，原本是用于测试剪枝后对的网络精度，对单一通道进行预剪枝可根据输出的变化评估通道重要性。

2）基于重初始化的权重置换。随机重初始化将选定通道的权重像模型初始化那样重新随机地赋值，使得该通道恢复到模型训练前的状态，此时的通道权重不包含任何与目标相关的信息，在对输入特征图进行计算后生成的输出特征图对任务不再具有意义。

3）基于随机排序的权重置换，如算法 3-2 所示。随机排序将选定通道的权重随机地进行重新排序，打乱该通道原有的权重分布，此时相应的输出特征图通道表达的特征是无意义的，对目标的预测将是无效的。在基于随机排序的权重置换中需要保证通道的任意权重在重排序后出现在任意个位置的概率是相等的。本章采用 “shuffle” 算法对选定通道的权重进行重排序，首先将选定通道的权重矩阵转换成向量形式，然后将向量中每一个元素与其后（包括本身）的随机一个元素调换位置，最后将已打乱的向量重新转换成矩阵形式并替换原通道权重矩阵从而完成权重的随机重排序。可以证明该算法满足打乱后任意元素出现在任意位置概率相等的条件：对于一个拥有 n 个元素的向量 $\boldsymbol{v}=(a_1,a_2,\cdots,a_n)$，任意元素 a_k（$k\in\{1,2,\cdots,n\}$）出现在位置 i 的概率 $P_{k,i}$ 可计算为式（3-7）。

$$P_{k,i}=\left(1-\frac{1}{n}\right)\left(1-\frac{1}{n-1}\right)\cdots\left(1-\frac{1}{n-i+1}\right)\frac{1}{n-i}=\frac{1}{n} \qquad (3-7)$$

算法 3-2：基于随机排序的权重置换 wReorder

输入：待置换通道 k_i

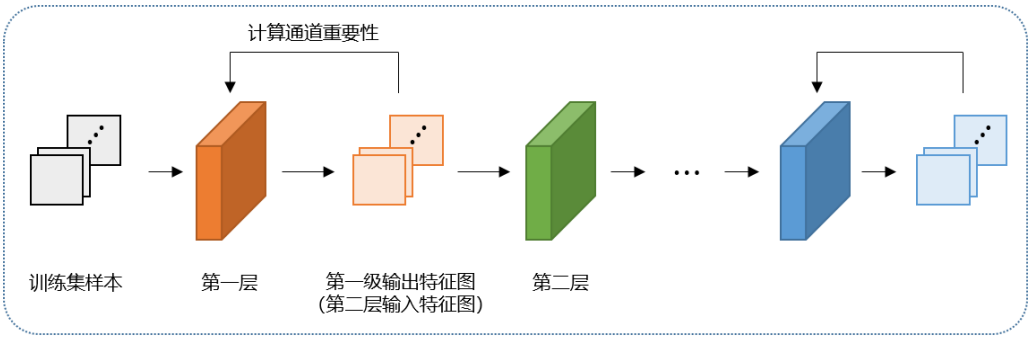
输出：置换后通道 k_i'

```
1.  v ← k_i.reshape(-1); //将待置换通道的权重矩阵转为向量
2.  for each p ∈ {0,1,⋯,v.length} do
3.      q ← randomInteger(p, v.length); //随机确定一个元素
4.      swap(v[p], v[q]); //交换位置
5.  end for
6.  k_i' ← v.reshape(k_i.shape); //将重排序后的向量转为矩阵
7.  return k_i';
```

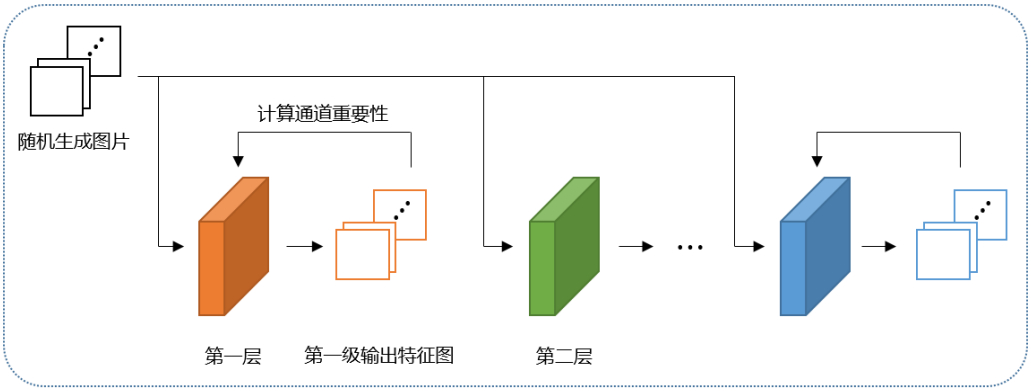
（2） 输入特征图

基于权重置换的通道重要性计算算法依赖于卷积层输出特征图的变化，影响输出特征图的除了通道权重还有输入特征图，因此需要将上一抽象级的特征图作为卷积核的输入参与通道重要性计算，如式（3-5）所示。鉴于卷积

核的输出特征图与输入特征图的紧密关系，在计算通道重要性时，不同的输入特征图会使卷积层计算出不同的结果，这种不同可能对剪枝时的通道选择产生影响，比如同一通道的取舍情况不同。在算法实现中，输入特征图可以选择训练集样本或者随机生成的图片。



(a) 训练集样本



(b) 随机生成图片

图 3-4 不同输入在模型前向传播中的实现

1）训练集样本，如图 3-4（a）所示。网络权重所包含的信息是从训练集的样本中学习获得的，对于待评估通道而言，其权重分布与训练集样本的信息分布是对应的，因此采用训练集的样本作为输入计算权重置换前后的输出特征图变化，可以直观地反映通道变化对其所属卷积层的输出特征图的影响。这一过程的逻辑与排列重要性的原理相仿。在算法的实现中，由于既定策略是将有意义的特征图作为卷积层输入，对任意待剪枝卷积层，其输入特征图的抽象级都应该与之对应。因此，对一个待剪枝网络，从第一层开始输入训练集样本，得出的第一级输出特征图用于计算第一层通道重要性，并且用作第二层的输入特征图，以此类推从而完成所有卷积层通道的重要性评估。

2）随机生成图片，如图 3-4（b）所示。基于权重置换的通道剪枝算法是依据卷积层输出特征图变化量评估通道重要性的，鉴于本文的通道重要性度量算法式（3-6）与特征图所含的空间信息无关，因此它并不关注特征图包

含的信息是否与目标的抽象特征相关。对于一个待剪枝卷积层，当输入数据固定时，每一层的输出特征图变化量只与通道权重分布的变化相关，可以用随机生成的图片作为每个卷积层的输入，如此即可忽略“上一层输出特征图作为下一层输入特征图”的传递关系，使算法更易于实现。

在神经网络中讨论输入输出就必须考虑 batch size 的设置问题，因此本文对通道重要性计算公式做出调整：

$$\mathrm{D}=\frac{1}{bs}\sum_{bs}\sum_{C}\sum_{pixel}\left(val_{pixel}-val_{pixel}'\right)^2 \qquad (3-8)$$

式中 bs 表示批尺寸（batch size）。基于输出特征图变化计算的通道重要性，理论上讲只需一张特征图即可（batch size=1），但在以输出作为评估依据的方法中，无论输入与目标是否相关，评估效果都在很大程度上受输入独特性的影响，进而可能导致同一通道在不同输入特征图下出现取舍判定的矛盾。为消除这种影响，本文将多个特征图的平均结果作为通道重要性，如式（3-8）所示。显然 bs 取值越大则输入特征图的独特性对通道重要性计算结果的影响越小（当 bs 与数据集样本量相等时可忽略样本独特性带来的影响），但由此引入的计算量也随之增大，因此在实际应用中应当结合需求和设备资源选取合适的 bs 从而取得计算结果稳定性和计算成本的平衡。

3.2.4 算法计算复杂度与可解释性分析

本节沿用 3.2.2 节中的符号定义。基于权重置换的通道剪枝算法计算复杂度主要分为三个方面：

（1） 单个通道随机重排序的计算复杂度为 $\mathrm{O}(c_{in}\times kh\times kw)$。

（2） 单个卷积层的通道重要性计算复杂度为 $\mathrm{O}(c_{out})$。

（3） 对于拥有 L 个卷积层的待剪枝模型 $M=\{K_1,K_2,\cdots,K_L\}$，通道重要性的计算复杂度为 $\mathrm{O}(L)$。

综合以上分析，对模型 M 的剪枝算法计算复杂度可表示为 $\mathrm{O}(L\times c_{out}\times c_{in}\times kh\times kw)$。注意到每个通道的重要性是由式（3-8）计算的，由此可以估计基于权重置换的通道剪枝算法的计算量 C，如式（3-9）所示。

$$C=\sum_{l=1}^{L}\left[\left(bs\times h_{out,l}\times w_{out,l}\times c_{out,l}\right)+\left(c_{out,l}\times c_{in,l}\times kh_l\times kw_l\right)\right] \qquad (3-9)$$

可解释性是深度学习领域的一个热点问题，一个具备良好解释性的模型可以帮助使用者建立对模型预测结果的信任，在通道剪枝中，算法的可解释性同样可以帮助使用者建立对剪枝过程和剪枝结果的信任。本章算法的可解

释性主要在于两个方面：

（1） 将特征图变化量作为通道重要性的衡量标准的可解释性。对于特征图的一个通道，其承载的目标特征信息表达为一个灰度图，其像素值的大小表示了相应特征信息的强弱，这与该特征图在后续处理中的影响程度是正相关的。当特征图像素值发生变化时，变化越不明显的特征对模型预测结果产生的影响越小，反之则特征对模型预测结果产生的影响越大。因此，将权重置换前后引起的特征图变化量作为通道重要性的衡量标准，可以反映出通道对其所属卷积层输出的影响程度。

（2） 利用权重置换引发特征图变化的可解释性。采用特征图变化衡量通道重要性必须保证变化后的特征图对目标预测是没有意义的，即与目标特征信息不相符。对于经过充分训练的模型，其权重取值和排列与目标特征信息是直接相关的，破坏通道权重与目标特征信息的相关关系即可实现对输出特征图与目标特征对应关系的破坏。本章采用的权重置换方法保证重排序后的权重排列是随机的，如 3.2.3 节中所述。

上述分析从特征图变化和通道对输出的影响程度说明了本章算法的可解释性，这一理论在排列重要性中也可得到支持。

3.3 实验验证

本节将对基于权重置换的通道剪枝算法进行实验验证，测试它在不同数据集和不同模型上的剪枝表现以及与其它剪枝算法的对比。另外，在 3.2.3 节中提及三种权重置换方法（置零、重初始化和随机排序）和两种输入特征图（数据集样本和随机生成图片），本章将以随机排序和数据集样本作为基准测试不同权重置换方法和不同输入特征图对通道重要性计算结果的影响。

3.3.1 数据集介绍

本章实验将采用的数据集包括 CIFAR10、CIFAR100 和 STL10。

CIFAR10 是由 Alex Krizhevsky 和 Ilya Sutskever 整理的一个用于识别普适物体的小型数据集。数据集由 60000 张 32×32 的彩色（RGB）图片组成，包含 10 个类别，每个类别有 6000 张图片。这些类别是完全互斥的，即一个类别中的图片不会出现在其它类别中。数据集被分为 5 个训练块和 1 个测试块，每个块包含 10000 张图片。图 3-5 展示部分 CIFAR10 中的样本。

CIFAR100 数据集包含 100 个类别的图片，每个类别有 600 张 32×32 的彩色（RGB）图片，其中 500 张属于训练集，100 张属于测试集。CIFAR100

中的图片类别是有层级的，每张图片都对应一个细粒度标签（fine labels）和一个粗粒度标签（coarse labels），分别对应类别（classes）和超类（superclass），如表 3-1 所示。

图 3-5 CIFAR10 类别标签与部分图片

表 3-1 CIFAR100 超类与对应类别

超类	类别
水生哺乳动物	海狸，海豚，水獭，海豹，鲸
鱼	观赏鱼，比目鱼，鳐，鲨鱼，鲑鳟鱼
花	兰花，罂粟，玫瑰，向日葵，郁金香
食物容器	瓶子，碗，罐头，杯子，盘子
水果和蔬菜	苹果，蘑菇，橘子，梨子，甜椒
家用电子设备	钟，电脑键盘，台灯，电话，电视
家具	床，椅子，沙发，桌子，衣柜
昆虫	蜂，甲壳虫，蝴蝶，毛虫，蟑螂
大型食肉动物	熊，豹，狮子，老虎，狼
大型户外人造设施	桥，城堡，房子，公路，摩天大楼
大型自然户外场景	云，森林，山，平原，海
大型杂食动物和食草动物	骆驼，牛，黑猩猩，大象，袋鼠
中型哺乳动物	狐狸，豪猪，负鼠，浣熊，臭鼬
非昆虫类无脊椎动物	蟹，龙虾，蜗牛，蜘蛛，蠕虫
人	婴儿，男孩，女孩，男人，女人
爬行动物	鳄鱼，恐龙，蜥蜴，蛇，乌龟
小型哺乳动物	仓鼠，老鼠，兔子，鼩鼱，松鼠
树	枫树，橡树，棕榈，松树，柳树
交通工具 1	自行车，公交车，摩托车，皮卡货车，火车
交通工具 2	割草机，火箭，有轨电车，坦克，拖拉机

STL10 是一个用于开发无监督特征学习、深度学习和自监督学习算法的图像识别数据集，其模仿 CIFAR10，但每个类别的标签图片数量更少，并且

提供大量的无标签图片。该数据集包含 113000 张 96×96 的来自 ImageNet 的彩色（RGB）图片，其中 5000 张属于训练集，8000 张属于测试集，另有 100000 张无标签图片。

3.3.2 实验一：基于权重置换的通道剪枝结果

本实验在不同数据集（CIFAR10、STL10 和 CIFAR100）上分别训练了 VGG16、ResNet50 和 GoogLeNet，训练轮次为 300，优化器为 SGD，学习率变化函数为 StepLR。本文观察它们在不同剪枝率下（剪枝率取值为 $\{0.1,0.2,\cdots,0.9\}$）的浮点数计算量和网络参数量的压缩比例以及剪枝后模型精度相比于基准值的变化。表 3-2 展示了理想结果（剪枝后网络精度高于原网络精度且剪枝率最高的结果）和可接受结果（剪枝后网络精度损失在 1% 内且剪枝率最高的结果）。由于 VGG16 和 GoogLeNet 在 CIFAR100 上无法得到高于原网络精度的结果，表中将剪枝率最低时的结果展示出来。另外，各网络后的数字表示其剪枝率，精度变化后括号内数字表示其测试集精度。

表 3-2 VGG16、ResNet50 和 GoogLeNet 在 CIFAR10、STL10 和 CIFAR100 上的剪枝结果

数据集	网络	精度变化 %	计算量 ↓%	参数量 ↓%
CIFAR10	VGG16 – 0.2	+0.35 (89.43)	43.4	24.6
	VGG16 – 0.6	-0.39 (88.69)	86.9	64.8
	ResNet50 – 0.6	+0.22 (88.76)	85.5	86.2
	ResNet50 – 0.7	-0.70 (87.84)	91.7	92.4
	GoogLeNet – 0.1	+0.17 (91.23)	27.3	22.4
	GoogLeNet – 0.4	-0.98 (90.08)	73.9	68.8
STL10	VGG16 – 0.5	+0.71 (86.90)	75.0	47.7
	VGG16 – 0.6	-0.95 (85.24)	83.2	56.3
	ResNet50 – 0.7	+0.27 (92.89)	91.7	92.4
	ResNet50 – 0.9	-0.67 (91.95)	99.0	99.5
	GoogLeNet – 0.1	+0.13 (91.97)	27.3	22.4
	GoogLeNet – 0.4	-0.97 (90.87)	73.9	68.8
CIFAR100	VGG16 – 0.1	-0.02 (60.95)	23.9	14.4
	VGG16 – 0.4	-0.94 (60.04)	69.7	46.1
	ResNet50 – 0.2	+0.22 (83.59)	56.5	48.8
	ResNet50 – 0.3	-0.93 (82.50)	69.4	63.1
	GoogLeNet – 0.1	-0.29 (82.24)	27.3	22.4
	GoogLeNet – 0.2	-1.31 (81.22)	47.3	41.0

VGG16 在 CIFAR10 和 STL10 上能够在得到比原网络更高精度的同时使计算量分别下降 43.4% 和 75.0%，参数量分别下降 24.6% 和 47.7%，并且在计算量下降超过 80% 的基础上保证网络精度损失少于 1%。在 CIFAR100 上，原网络的精度仅达 60.97%，但剪枝网络仍然在计算量减少

69.7%，参数量减少 46.1% 的情况下保证精度损失少于 1%。

ResNet50 在 CIFAR10 和 STL10 上可在计算量和参数量压缩率超过 90% 的情况下保证网络精度损失少于 1%。对于 CIFAR100，剪枝后的 ResNet50 可在计算量和参数量分别下降 56.5% 和 48.8% 时得到比原网络精度高的结果，并且在计算量和参数量分别下降 69.4% 和 63.1% 的情况下保持精度损失少于 1%。

GoogLeNet 在 CIFAR10 和 STL10 上的剪枝后模型，可在获得比原网络更高精度的基础上分别减少 27.3% 的计算量和 22.4% 的参数量，并且保持剪枝后网络精度损失少于 1% 的基础上达到 73.9% 的计算量下降和 68.8% 的参数量下降。在 CIFAR100 上，当网络压缩超过 40% 后已无法保证剪枝后网络精度损失少于 1%。

不同网络的特征表达能力是不同的，因此它们在同一数据集和不同数据集上会得到不同的剪枝效果。以上分析出现了通道剪枝使网络精度提升的现象，这是因为原模型因特征表达能力富余而出现的过拟合现象在通道剪枝后得到一定的抑制，而随着剪枝率继续提升，网络特征表达能力逐渐被削弱至无法满足任务需要的水平，此时剪枝后的网络精度开始下降。鉴于剪枝的目标在于以可接受的精度损失换取计算成本的下降，本文假定剪枝后的模型精度损失在 1% 以内都是可接受的。

3.3.3 实验二：不同权重置换方法对比

本实验沿用实验一中在 CIFAR10 上训练的 VGG16，对比其同一卷积层在不同权重置换方法下（输入为数据集样本）的通道重要性计算结果。

对于通道重要性的计算结果，本文关注的并非其幅值，而是各通道按其重要性的排序。在一个待剪枝卷积层中，不管通道重要性幅值的分布区间如何，被移除的总是重要性最小的若干个通道，因此通道的重要性排序才是影响剪枝结果的根本因素。图 3-5 上展示了 VGG16 features.0（第一个卷积层）在三种权重置换方法下的通道重要性计算结果，由于通道重要性数值分布区间较大，本文对其取对数进行展示。通道重要性的排序反映在图中曲线的变化趋势上，可以看到表示不同权重置换方法的曲线的变化趋势基本相同，这意味着各通道按照重要性排序的结果是基本相同的。为了进行更加细致的分析，本文从 VGG16 features.0 的 64 个通道中选择第 21 至第 30 个通道进行展示，如图 3-6 下，尽管同一通道重要性在三种权重置换方法下的幅值迥异，但各通道按照重要性排序的结果是相同的，因此在同一剪枝率的指导下

被选中的待剪枝通道也应当是相同的。基于上述分析和图 3-6 展示的结果，本文在此处预言基于三种权重置换的通道剪枝将得到相似的结果。

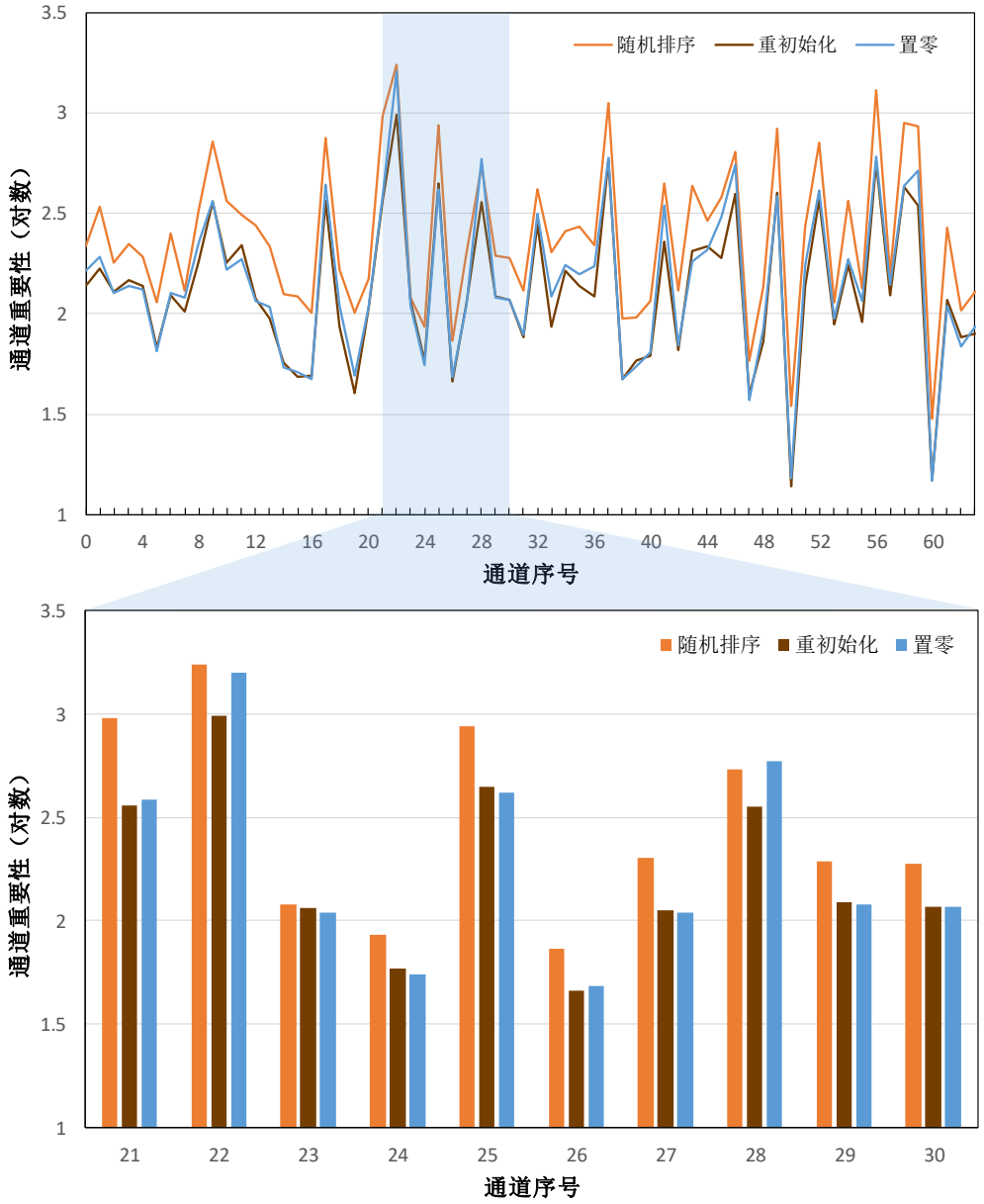


图 3-6　VGG16 features.0 在不同权重置换方法下的通道重要性计算结果

表 3-3 展示了采用不同权重置换方法计算通道重要性的 VGG16 剪枝后精度相较于基准值的变化，剪枝率取值为 {0.2,0.4,0.6,0.8}，各模型精度变化后括号内数字表示该模型精度。在相同剪枝率下三种方法的剪枝后模型精度略有不同，但考虑到模型精度可能存在的波动，本章认为表 3-3 中展现出来的微小精度差别是可接受的，因此可认为表 3-3 展示的剪枝结果与前文中的预言相符。预言的成功说明在实际应用中三种方法都可用于计算通道重要性，但根据本章所阐述的通道重要性计算原理，基于随机排序的权重置换方

法仍是本文的首选。

表 3-3 VGG16 在 CIFAR10 上于不同权重置换方法下的剪枝结果

剪枝率	精度变化 %		
	置零	重初始化	随机排序
0.2	+1.00 (90.08)	+0.90 (89.98)	+0.93 (90.01)
0.4	+0.62 (89.70)	+0.74 (89.82)	+0.71 (89.79)
0.6	-0.07 (89.01)	-0.13 (88.95)	-0.16 (88.92)
0.8	-1.76 (87.32)	-1.71 (87.37)	-1.67 (87.41)

3.3.4 实验三：不同输入特征图对比

本实验再次沿用实验一中在 CIFAR10 上训练的 VGG16，对比不同输入特征图（权重置换方法为随机重排序）下通道重要性的计算结果。

图 3-7 展示了 VGG16 features.0 在两种输入下的重要性计算（对数）结果。本实验关注不同通道间的重要性对比情况（曲线的变化趋势），如橙色曲线表示的数据集样本和蓝色曲线表示的随机生成图片，图中红色圆点标注了部分重要性变化趋势明显不同的通道。由于存在计算结果偏差较大的通道，这将导致在剪枝过程中同一通道剪枝与否的判定不同。

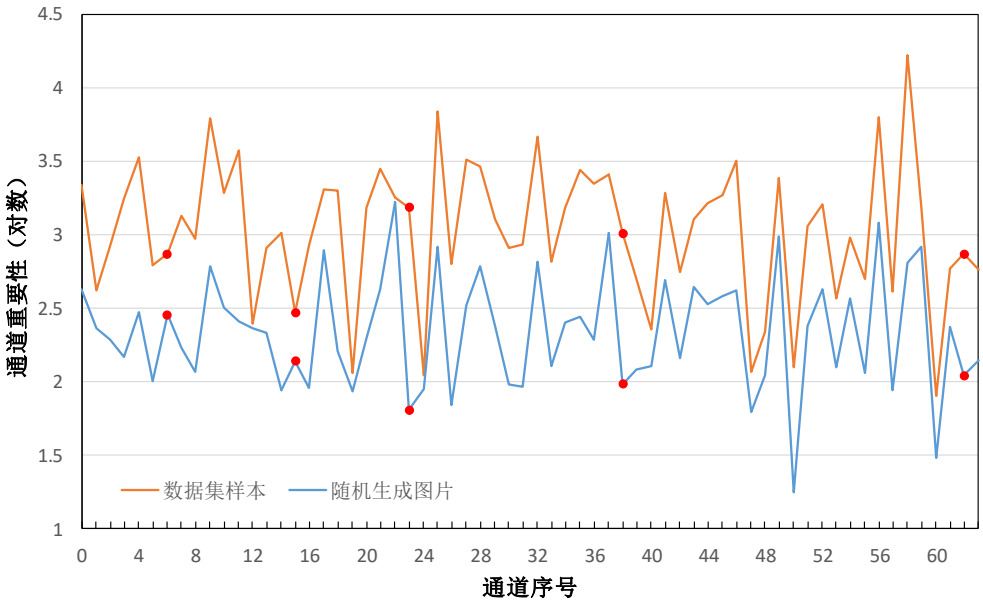


图 3-7 VGG16 features.0 在不同输入下的通道重要性计算结果

表 3-4 展示了采用随机生成图片何数据集样本作为输入的 VGG16 剪枝后精度相较于基准值的变化，剪枝率取值为 {0.2,0.4,0.6,0.8}，各模型精度变化后括号内的数字表示该模型精度，图中红点标出两条曲线趋势明显不同的通道。在同一剪枝率下，采用随机生成图片作为输入的剪枝后模型精度都

比采用数据集样本的剪枝结果更低，并且在剪枝率合理（剪枝后模型精度与原模型精度处于同一水平）时这一差距更明显。表 3-4 展示的剪枝结果同时说明两个问题：（1）基于随机生成图片的通道重要性计算结果不合理。基于随机生成图片的通道重要性计算结果与通道对输出的实际影响程度有偏差，这使得其剪枝结果精度更低。在实际应用中，应采用数据集样本（以及它们不同抽象级别的特征图）计算各卷积层的通道重要性。（2）基于权重置换的通道重要性计算结果是可靠的。由于不同输入下计算的通道重要性不相符，所以同一剪枝率下被移除的通道是不同的，结合图 3-7 展示的通道重要性计算结果，本文证明了采用权重置换前后特征图变化程度衡量通道重要性是正确的。

表 3-4 VGG16 在 CIFAR10 上于不同输入下的剪枝结果

剪枝率	精度变化 %	
	随机生成图片	数据集样本
0.2	+0.46 (89.54)	+0.93 (90.01)
0.4	+0.15 (89.23)	+0.71 (89.79)
0.6	-0.51 (88.57)	-0.16 (88.92)
0.8	-1.87 (87.21)	-1.67 (87.41)

3.3.5 实验四：与经典算法的对比

本实验在 CIFAR10 上训练了 ResNet56，训练轮次为 350，优化器为 SGD，学习率变化函数为 StepLR。本文在训练后的 ResNet56 上应用本章提出的基于权重置换的通道剪枝算法，并用之与一些基于度量和基于误差重建的通道剪枝算法进行比较，观察剪枝后模型精度相比于基准值的变化，以及浮点数计算量和参数量的压缩情况。

表 3-5 ResNet56 在 CIFAR10 上于不同剪枝算法下的剪枝结果

		精度变化 %	计算量 ↓%	参数量 ↓%
基于度量	PFEC[40]	+0.02	27.6	17.7
	SFP[140]	-0.24	52.6	-
	FPGM[127]	-0.10	52.6	-
基于误差重建	CP[42]	-0.10	50.0	-
	NISP[136]	+0.03	43.6	42.6
本章算法 – 0.30		+0.24 (91.30)	49.6	45.5
本章算法 – 0.35		-0.04 (91.02)	56.2	51.7

在实验中本文尝试了多个剪枝率，并从中挑选实际压缩率与其它算法具有可比性的结果，如表 3-5 所示。当剪枝率为 30% 时，模型浮点数计算量

下降约 49.6%，参数量下降约 45.5%，测试集精度相比于基准值上升 0.24%，这组数据是在模型剪枝后得到更高精度的前提下达到最高压缩率的结果。当剪枝率达到 35% 时，本章提出的通道剪枝算法以 0.04% 的精度损失换取超过 50% 的计算量和参数量下降，这比其它算法更具优势。

3.4 小结

本章提出了基于权重置换的通道剪枝算法。本文介绍了通道对模型的影响并将这种影响程度定义为通道重要性。权重置换是本章提出用于计算通道重要性的方法，它通过破坏模型权重与目标特征之间的相关性，引起输出特征图变化，从而展现出通道对卷积层输出的影响程度，权重置换尝试以排列重要性的思想解决通道剪枝问题。特征图像素点差值的平方和用于计算通道权重置换前后的特征图变化程度，它直接反映了通道的重要性，这种衡量指标比各种模型权重的幅值更具可解释性。另外，本章对比了三种权重置换方法和两种用于计算通道重要性的输入，实验数据表明不同权重置换方法对最终剪枝结果造成的差异是可以接受的，而随机生成图片会对剪枝后模型带来较明显的精度损失。最后，本文在不同模型和不同数据集上测试了本章提出的通道剪枝方法，并与一些同类算法进行比较，验证了基于权重置换的通道剪枝算法的通用性和有效性。这种结合特征图变化的通道重要性度量算法为第四章中的基于累计信息贡献率的全自动化通道剪枝算法提供了基础。

4 基于累计信息贡献率的全自动化通道剪枝算法

4.1 引言

通道剪枝的另一个关键问题在于找到适合当前任务的剪枝率。通道剪枝的基本前提是模型特征表达能力对当前任务而言存在冗余，而剪枝率代表着模型能力削减的比例，若剪枝率过低则模型仍然处于冗余状态，若剪枝率过高则会导致模型能力削弱过多以至于无法满足任务所需。现有相关研究中剪枝率的设置方法主要分为全局共享式[36]和逐层定制式[40,118]：（1）**全局共享式**将同一剪枝率应用于所有卷积层，实现简单，但忽略了不同卷积层对剪枝敏感度的差异性[40,118]，使得剪枝结果难以达到最优解；（2）**逐层定制式**为每个卷积层设置一个合适的剪枝率，解决了全局共享式的不合理问题，但它要求使用者进行繁杂的敏感度分析[40,118]，提高了通道剪枝的使用成本。

为了简化通道剪枝的使用，一些研究者将注意力放到自动化剪枝上，如 He 等人利用强化学习来制定模型压缩策略实现剪枝自动化[33]。NAS 也是一种有效的模型压缩方法，并且它是确定能够找到最优解的，于是一些研究尝试将 NAS 与剪枝结合起来实现自动化寻找最优解，如 Luo 等人提出的 ThiNet 通过遍历每个卷积层寻找一个输出能够与原输出等效的通道子集从而判定剪枝后保留下来的通道[41]，Yu 等人提出的 AutoSlim 则是训练一个可瘦身模型并通过贪心法逐步进行剪枝[130]，Dong 等人提出的 TAS 则采用了一种可微分算法寻找小型网络，并将原模型信息迁移至这个网络[141]。现有的相关工作已经提出很多有效的自动化剪枝算法，在保证最优解的同时极大解放了人力，但 NAS 也将其原生的时间成本高的问题带入了基于其思想的自动化通道剪枝中。基于 NAS 的通道剪枝，本质上是遍历待剪枝模型所有卷积层的通道组合并从中选择最优解，对于一个复杂的模型，算法需要做大量的探查，比如 VGG16、GoogLeNet 和 ResNet152 分别有 2^{104}、2^{448} 和 2^{1182} 种通道组合[142]，这意味高昂的时间代价和严苛的算力需求，这种成本在实际应用中几乎是不可接受的。一些研究通过设置新的超参数定义更小的搜索空间，从而降低搜索的复杂度，如 ABCPruner 为模型设置超参数 α 用于限定每个卷积层最大保留比例[142]。然而这些超参数要求使用者设置合适的值才能得到理想剪枝结果，降低了算法的自动化程度，违背了算法减少人工参与的初衷。传统通道剪枝算法为保证剪枝的合理性，应当为待剪枝模型的每个卷积层设置一个剪枝率。这些剪枝率形成一个组合，搜索这个组合的最优解是自动化通道剪枝的核心任务。对于一个深度模型，算法针对剪枝率组

合的搜索空间是高维的，对最优剪枝结果的搜索是多元搜索，这是自动化通道剪枝算法计算成本高的原因。

本章在第三章的研究之上提出一种基于累计信息贡献率的全自动化通道剪枝算法。本文引入信息贡献率的概念，将待剪枝模型通道组合的搜索过程转换为累计信息贡献率的搜索过程，并采用二分搜索策略寻找最优的模型信息保留率，大大降低自动化剪枝的时间成本。每个通道对其所属卷积层都具有一定的贡献率，本文评估每个通道的贡献率并选择贡献率最大的若干个通道进行保留，使得这些通道的累计贡献率达到预设的超参。通道的贡献率与其所含信息量直接相关，因此基于累计信息贡献率的剪枝算法可以实现在全局共享式超参数的指导下逐层定制通道保留数量，省去了繁杂的敏感度分析，兼顾全局共享式的简单性和逐层定制式的合理性。在自动化通道剪枝过程中，算法在累计信息贡献率定义域上的一元搜索取代了在剪枝率组合上的多元搜索，其搜索空间由高维降至一维，大幅降低了搜索成本。另外，鉴于模型特征表达能力与通道数量的正相关关系，本文在累计信息贡献率的定义域上采用二分搜索策略寻找当前任务下最优解，通过缩小搜索空间进一步降低自动化剪枝的时间成本。

4.2 算法原理与实现

4.2.1 基于累计信息贡献率的通道保留数算法

传统的通道保留数计算是基于剪枝率的。在给定剪枝率 pr，一个 n 通道卷积层的通道保留数 n_{to_remain} 计算为式（4-1）。

$$n_{to\_remain} = n(1 - pr) \qquad (4-1)$$

现有剪枝率的设置方法可分为两类：

（1） **全局共享式**，这类方法将同一剪枝率应用至待剪枝模型的所有卷积层中，它有两个优点：①剪枝率作为全局共享式超参数，实现简单，用户只需在剪枝率的定义域上进行搜索即可得到最优的剪枝结果；②当模型深度和宽度发生变化时，其搜索空间总是剪枝率的定义域 [0,1]，对剪枝结果的搜索成本不会改变。然而，不同卷积层对剪枝的敏感度是不同的，因此全局共享式的剪枝率可能导致敏感度高的卷积层剪枝过度而敏感度低的层剪枝不足，如图 4-1（a）所示，图中 pr 表示剪枝率。这种剪枝率设置方法是不合理的，它难以达到模型表现和计算资源的最优平衡。

（2） **逐层定制式**，这类方法将不同卷积层对剪枝敏感度的差异纳入考虑，

为每个卷积层设置一个合适的剪枝率，解决了全局共享式中的不合理问题，如图 4-1（b）所示，图中 pr_i （$i \in \{1,2,\cdots,L\}$）表示第 i 层的剪枝率。然而这类方法要求使用者对待剪枝模型进行剪枝敏感度分析，这为使用者引入了人工成本和技术要求，使通道剪枝的使用变得繁琐。

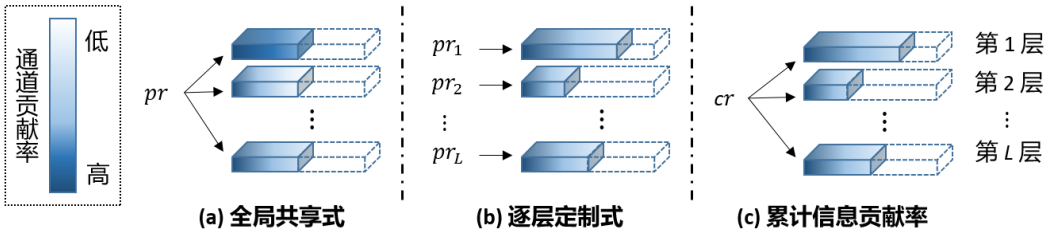


图 4-1　通道保留数算法与剪枝结果

低秩压缩是一种基于线性代数的通道剪枝算法，它通过寻找一个低维权重矩阵替换原始权重矩阵完成剪枝目的，其中低维矩阵的维度就是通道保留数[143]。对于存在冗余的卷积层 l，其权重矩阵 W_l 是由某个极大线性无关组 W_l^* 张成的线性相关矩阵，并且 W_l^* 的信息量与 W_l 的信息熵相等，因此权重矩阵 W_l 的秩 r_{W_l} 可用作卷积层 l 的通道保留数。由于卷积层权重矩阵往往是满秩的，将权重矩阵的秩作为通道保留数会导致剪枝过程中无通道可剪，但低秩压缩的思想昭示着卷积层通道保留数的计算应当基于其信息量保留率。剪枝率实际表示的是模型特征表达能力的削减率，它忽略了不同通道在卷积层信息总量上的占比差异，本文认为这是导致传统通道保留数计算遇到瓶颈的原因。

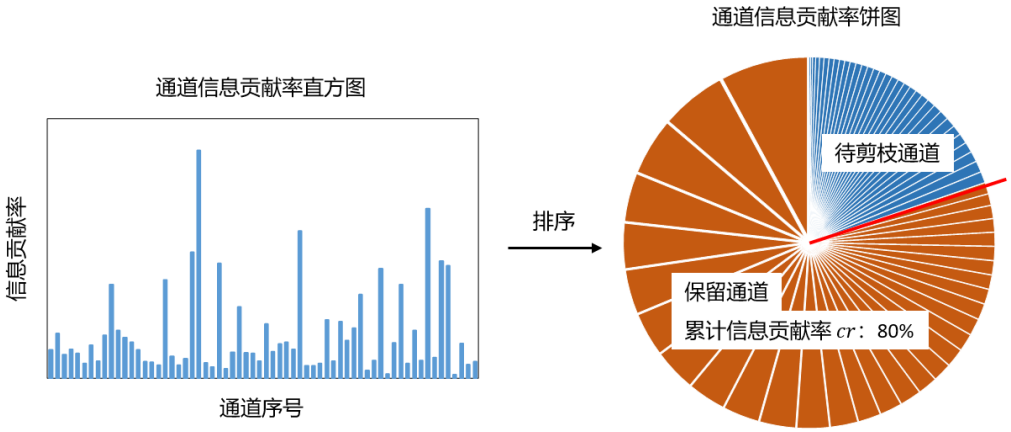


图 4-2　基于累计信息贡献率的通道剪枝算法

基于上述分析，本章在此定义信息贡献率：在卷积层 l 中，通道 i 的信息贡献率是通道 i 所含信息量在 l 所含信息量中的占比。另外，本章设计一个名为累计信息贡献率的超参数 cr，它表示剪枝过程中卷积层的信息量保留

率，由此通道剪枝中待剪枝通道的选择转换为待保留通道的选择。如图 4-2 所示，对于给定 cr，按信息贡献率为待剪枝卷积层的通道排序，并保留信息贡献率最高的若干个通道，使这些通道的信息贡献率和达到 cr，如式（4-2）所示，从而完成通道剪枝。

$$\sum_{R} \varepsilon_i \geq cr \tag{4-2}$$

式中 R 表示被保留的通道集合，ε_i 表示 R 中第 i 个通道的信息贡献率。对于待剪枝卷积层，基于累计信息贡献率计算的通道保留数将恰好足够承载 cr 所规定的信息量，而不同卷积层在剪枝敏感度上的差异将不会对此产生影响。也就是说，在同一个累计信息贡献率的指导下，每个卷积层的通道保留数将自行适应卷积层本身，使用者只需在累计信息贡献率的定义域上进行搜索，即可得到最优的剪枝结果。

本文在 3.2.2 节中将通道权重置换前后输出特征图的变化程度作为通道重要性的衡量指标，若待剪枝卷积层 l 全部通道同时置换，则输出特征图将完全变为另一个特征图，而每个通道都对这一变化做出一定程度的贡献，且每个通道的贡献率表示为式（4-3）。

$$P_i = \frac{D_i}{\sum_l D} \tag{4-3}$$

式中 P_i 表示卷积层 l 中第 i 个通道的贡献率，D_i 表示卷积层 l 中第 i 个通道权重置换引发的输出特征图变化程度，$\sum_l D$ 表示卷积层 l 所有通道权重同时置换引发的特征图变化程度。对于卷积层 l，所有通道权重同时置换后的特征图变化 $\sum_l D$ 是一定的，结合 3.3.2 节中对通道重要性幅值和排序结果的分析，可以认为通道贡献率 P_i 等效于通道重要性 D_i。本章对 3.2.2 节中的通道重要性定义做出调整：对于卷积层 l，在全部通道权重同时置换引发的特征图变化量中，由通道 i 权重置换所引起的特征图变化量的比例为通道 i 的重要性。需要说明的是，虽然通道权重置换引起的特征图变化量并不是通道所含的信息量，但是通道所含信息量越大则其权重分布变化后引发的特征图变化量也就越大，因此前者可以视作后者的一种反映。

有了调整后的通道重要性定义，结合式（4-2）即可在一个给定的累计信息贡献率 cr 下，得到通道剪枝结果，如图 4-1（c）所示，第三章中提出的基于权重置换的通道剪枝算法的最大特点也在此体现。

4.2.2 全自动化通道剪枝算法

虽然基于累计信息贡献率的通道保留数计算消除了模型压缩程度设定中简单性与合理性的矛盾，但累计信息贡献率无法让模型压缩自动适配当前任

务。卷积神经网络的特征表达能力是难以量化的，并且一项具体任务对模型特征表达能力的需求同样难以评估，这意味着在通道剪枝的实际应用中，模型相对于任务的冗余程度是难以评估的。因此，使用者在特定任务下的模型压缩仍然需要尝试不同的累计信息贡献率，才能得到模型精度与计算成本的最佳平衡，这对剪枝应用提出了一定程度的人工成本。

现有的自动化通道剪枝通常以遍历待剪枝模型子结构空间的方式寻找最优解。这类算法的搜索复杂度为 $O(w^d)$，其中 w 表示卷积层通道数量，d 表示待剪枝模型卷积层的数量。对深度模型而言，这样的搜索成本是难以接受的。另一类算法是在剪枝率的定义域上进行遍历，其搜索复杂度为 $O(n)$。然而基于剪枝率计算的保留通道数无法自动适应剪枝敏感性不同的卷积层，因此在剪枝率定义域上的搜索仍然要求在不同卷积层上单独进行，这与子结构空间的搜索在本质上是相同的。基于累计信息贡献率的通道保留数计算可以解决这一问题，因为它能够在同一个累计信息贡献率下实现逐层定制保留通道数，实现多元搜索到一元搜索的转换，极大地压缩了搜索空间。由此，自动化剪枝即可在累计信息贡献率的定义域中搜索最优解。另外，本章注意到模型压缩率与剪枝后模型精度的近似负相关关系，二分搜索将应用于最优剪枝结果的搜索过程，通过减少探查次数进一步降低自动化剪枝成本。

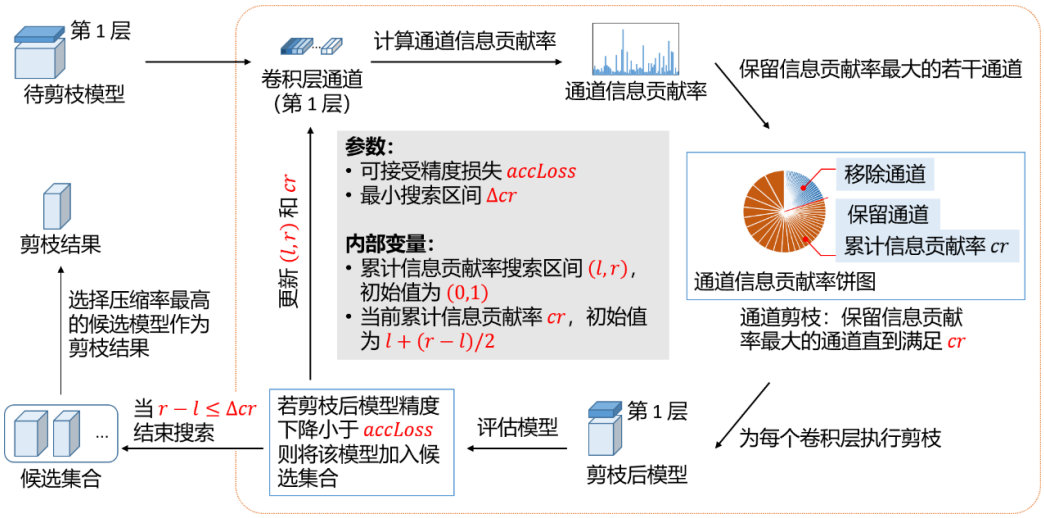


图 4-3　基于累计信息贡献率的全自动化剪枝框架

基于累计信息贡献率和二分搜索的全自动化剪枝算法如图 4-3 所示，图中 $accLoss$ 和 Δcr 分别表示可接受精度损失和最小搜索区间，它们是算法的参数；cr 表示当前探查的累计信息贡献率，l 和 r 则分别表示当前累计信息贡献率搜索区间的左右边界。首先，计算待剪枝卷积层的通道信息贡献率；其次，结合当前探查的累计信息贡献率，保留信息贡献率最大的若干通

道，使得这些通道的信息贡献率之和满足式（4-2）；再次，为待剪枝模型的每个卷积层执行剪枝操作，得到剪枝后模型；然后，对剪枝后模型进行评估，若其相对于原模型的精度下降在可接受范围，则将这个剪枝后模型保留下来作为候选模型之一；最后，更新搜索空间边界和累计信息贡献率，当搜索空间小于给定的最小搜索区间时算法结束，并从候选模型集合中选择压缩率最高的作为自动化剪枝的结果。

在本章提出的全自动化剪枝算法中，象征模型压缩率的累计信息贡献率不再是一个超参数，而是作为搜索目标的内部变量。对于一个经过充分训练的待剪枝模型，任务信息都包含在模型的通道中，使用者只需要设置可接受精度损失 $accLoss$ 和最小搜索区间 Δcr 就可以在无人工参与的情况下得到满足精度要求的高压缩率剪枝模型，摆脱了对剪枝率等传统超参数的依赖，实现了通道剪枝的全自动化。作为算法输入的可接受精度损失和最小搜索区间，只需要使用者结合任务需求考虑即可，对包括非相关专业人士在内的使用者十分友好。

4.2.3 算法实现

将通道重要性转换为信息贡献率的过程如算法 4-1 所示。具体而言，对于一个待剪枝卷积层，通道信息贡献率计算方法接收其通道重要性列表作为输入，并计算所有通道重要性的总和，然后计算每个通道的重要性占比作为其信息贡献率。可以看到，在基于权重置换的通道剪枝中，每个通道的信息贡献率实质是其权重置换所引发的特征图变化量在所有（同卷积层的）通道中的占比。

算法 4-1：计算通道信息贡献率 $getChannelInfoContributionRatio$

输入：待剪枝卷积层通道重要性列表 $DList$

输出：通道信息贡献率列表 $DContribution$

1. $\sum \leftarrow sum(DList)$; //计算当前卷积层通道重要性总和
2. **for** each $D_i \in DList$ **do**
3. $DContribution \leftarrow DContribution \cup \frac{D_i}{\sum}$; //计算 $i-th$ 通道的信息贡献率
4. **end for**
5. **return** $DContribution$;

得到卷积层的通道信息贡献率列表后即可开始进行剪枝，其过程如算法 4-2 所示。$prune$ 方法接受待剪枝卷积核集合 K、输入特征图 X 和累计信息贡献率 cr 作为输入，首先通过权重置换的方法计算出 K 中每个通道的重要性，然后调用算法 4-1 计算通道信息贡献率列表，最后保留 K 中信息贡献率最大的若干通道，并保证这些通道的信息贡献率和不小于设定的累计信

息贡献率 cr。算法 4-2 与算法 3-1 的主要有三个区别：1）选取剪枝对象的标准从通道重要性转变为信息贡献率，2）通道的移除过程变为保留过程，和 3）计算通道保留数或者信息贡献率阈值等过程不再作为一个显式的步骤，而是融合在式 4-2 的指导下的保留通道选择过程中。

算法 4-2：基于累计信息贡献率的通道剪枝 $prune$

输入：待剪枝卷积核集合 K，输入特征图 X，累计信息贡献率 cr

输出：剪枝后的卷积核集合 K'

初始化空的通道重要性列表 $DList$;
$K' \leftarrow \emptyset$; //初始化空卷积核集合
$Y \leftarrow K(X)$; //生成标准（原始）输出特征图
for each $K_i \in K$ **do**
    $wReorder(K_i)$; //权重置换
    $Y' \leftarrow K(X)$; //生成权重置换后的输出特征图
    $D_i \leftarrow (Y - Y')^2$; //计算通道重要性
    $DList \leftarrow DList \cup D_i$; //将 D_i 加入 $DList$
end for
$DContribution \leftarrow getChannelInfoContributionRatio(DList)$;
$curCr \leftarrow 0$; //初始化当前累计信息贡献率
while $curCr < cr$ **do**
    找到 $DContribution$ 中贡献率最大的通道 K_i
    $K' \leftarrow K' \cup K_i$; //保留 K_i
    $curCr \leftarrow curCr + DContribution_i$;
    $DContribution \leftarrow DContribution - \{K_i\}$;
end while
return K';

算法 4-3：基于累计信息贡献率的全自动化通道剪枝 $autoPrune$

输入：待剪枝模型 M，可接受精度损失 $accLoss$，输入特征图 X，最小搜索区间 Δcr

输出：候选剪枝后模型集合 M'

计算原模型的精度 $originalAcc$;
$l \leftarrow 0$, $r \leftarrow 1$; //初始化搜索边界
while $r - l > \Delta cr$ **do**
    $cr = l + (r - l)/2$; //基于二分法搜索计算当前累计信息贡献率
    $C \leftarrow \emptyset$; //初始化剪枝后的层集合（通道组合）
    for each $K \in M$ **do**
        $K' = prune\ (K, X, R)$; //基于权重置换的通道剪枝（信息贡献率）
        $C \leftarrow C \cup K'$; //将剪枝后的层加入层集合
    end for
    $tempM \leftarrow C$; //根据剪枝后层集合构建剪枝后模型
    $curAcc \leftarrow evaluate(tempM)$; //评估剪枝后模型精度
    if $curAcc + accLoss \geq originalAcc$ **then**
        $M'Set \leftarrow M'Set \cup tempM$; //若剪枝后模型精度损失可接受则保留
        $r \leftarrow cr$; //更新搜索边界
    else
        $l \leftarrow cr$; //更新搜索边界
    end if
end while
在 $M'Set$ 中寻找压缩率最高的模型 M';
return M';

基于累计信息贡献率的全自动化通道剪枝算法如算法 4-3 所示。算法接收待剪枝模型M、可接受精度损失 $accLoss$、输入特征图 X 和最小搜索区间 Δcr。首先，评估 M 在测试集上的精度作为基准；其次，通过二分搜索的策略在当前搜索区间内选取累计信息贡献率 cr，并且调用算法 4-2 对带剪枝模型进行剪枝从而得到一个剪枝后模型 M'；然后，评估 M' 在测试集上的精度，若其精度损失在可接受范围内则将它放入候选集合中，否则将它丢弃；最后，根据 M' 的保留与否更新当前搜索区间的边界，直到搜索区间小于最小搜索区间 Δcr 停止算法，并输出候选模型集合中压缩率最高的剪枝模型。

事实上，无论累计信息贡献率如何设置，每个通道在其所属卷积层中的贡献率是一定的（只跟输入特征图和通道权重置换方法相关），在实际应用中应该考虑在搜索开始前先计算出各卷积层通道的信息贡献率，从而免去重复计算带来的成本。

4.3 实验验证

本节将对基于累计信息贡献率的全自动化通道剪枝算法进行实验验证。本节将分别测试基于累计信息贡献率的通道保留数计算对不同卷积层的自适应性和基于二分搜索的自动化剪枝对不同任务（数据集）的自适应性，并与一些现有的经典自动化通道剪枝算法进行对比。

4.3.1 数据集介绍

本章实验采用的数据集包括 CIFAR10、CIFAR100、STL10 和 ILSVRC-2012。ILSVRC-2012 是 2012 年 ImageNet 大规模视觉识别挑战赛（ImageNet Large Scale Visual Recognition Challenge）所采用的数据集，被应用于图像分类与目标定位、目标检测、视频目标检测和场景分类。ILSVRC-2012 由 ImageNet 数据集中的部分图片组成，包含 1000 个类别， 1281167 张训练集图片，50000 张验证集图片和 100000 张测试集图片。本文基于以下原因在对比实验中采用该数据集：（1）ILSVRC-2012 是 ImageNet 数据集的子集，其图片类别和样本丰富性能够满足本文的需要；（2）现有研究大都采用了 ILSVRC-2012，该数据集的采用便于本文与其它研究成果进行对比。

4.3.2 实验一：通道保留数计算结果

本实验在 CIFAR10 上训练了 VGG16 和 ResNet34，训练轮次为 300，优化器为 SGD，学习率变化函数为 StepLR，观察模型不同卷积层在相同累

计信息贡献率下（累计信息贡献率取值为 {0.2,0.4,0.6,0.8}）的通道保留数。

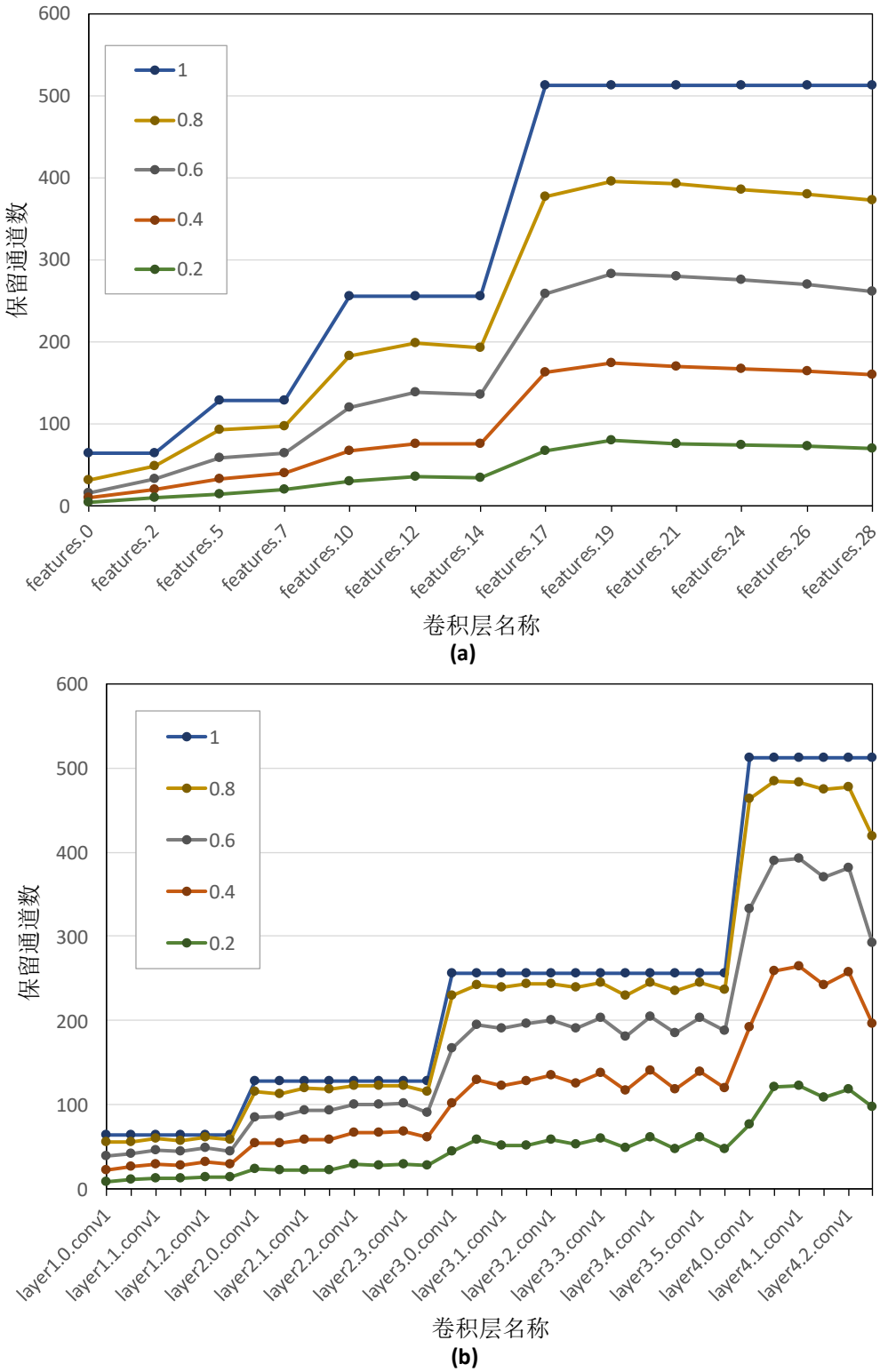


图 4-4 VGG16（a）和 ResNet34（b）各卷积层通道保留数计算结果

通道保留数的计算结果如图 4-4 所示，图中不同颜色的曲线表示在不同累计信息贡献率下的计算结果（如图例所示），其中累计信息贡献率为 1 的

曲线展示了待剪枝模型各卷积层的原始通道数。重点关注那些原始通道数相同的卷积层，如图 4-4（a）中 VGG16 的 features.17 到 features.28 和图 4-4（b）中 ResNet34 的 layer3.0.conv1 到 layer3.5.conv2，这些卷积层的原始通道数是相等的，但是不同卷积层对剪枝的敏感度不同决定了它们不同的通道保留数。卷积层对剪枝的敏感度实质上反映的是其所含任务相关的信息量，一个卷积层所含信息量越多，则通道剪枝对其信息量的折损越明显，这一规律的感性表达就是该卷积层对通道剪枝较敏感。基于累计信息贡献率的通道保留数计算不再拘泥于通道数量上的保留，而是着眼于卷积层所含信息量的保留，因此它能够在同一个超参数下为待剪枝模型的每个卷积层定制合适的通道保留数，从而实现传统剪枝率中全局共享式超参数（全局共享式）和逐层分析（逐层定制式）的融合，这就体现了基于累计信息贡献率计算通道保留数对不同卷积层的自适应性。

图 4-4 中还可以看出，在不同累计信息贡献率指导下计算得到的各卷积层通道保留数虽然在数值上不同，但曲线的变化趋势基本是一致的。在模型经过充分训练后，每个卷积层所含任务相关的信息量是确定的，因此在不同累计信息贡献率指导下计算得到的各卷积层通道保留数应当是近乎等比例地上升或下降，也就是说表示不同累计信息贡献率的曲线变化趋势相似是合理的，这也从侧面证明了基于累计信息贡献率计算卷积层通道保留数的合理性。

表 4-1 展示了 VGG16 和 ResNet34 在全局共享式和累计信息贡献率指导下的部分剪枝结果（剪枝后精度最高的结果以及 3.3.1 中定义的理想结果与可接受结果），表中“全局共享式”和“累计信息贡献率”后面的数字分别表示全局共享式的剪枝率和需要保留的累计信息贡献率，各模型后括号内的数字表示该模型精度。

表 4-1 VGG16 和 ResNet34 在全局共享式和累计信息贡献率指导下的剪枝结果

		精度变化 %	计算量 ↓%	参数量 ↓%
VGG16	全局共享式 – 0.2	+0.50 (89.58)	31.3	19.2
	全局共享式 – 0.7	+0.01 (89.09)	88.2	63.5
	全局共享式 – 0.8	-1.11 (87.97)	93.5	71.8
	累计信息贡献率 – 0.8	+0.96 (90.04)	43.4	24.6
	累计信息贡献率 – 0.4	+0.27 (89.35)	86.9	62.0
	累计信息贡献率 – 0.3	-0.65 (88.43)	92.2	69.5
ResNet34	全局共享式 – 0.1	+0.12 (90.22)	17.0	18.9
	全局共享式 – 0.6	+0.05 (90.15)	78.3	83.7
	全局共享式 – 0.7	-0.89 (89.21)	86.0	90.6
	累计信息贡献率 – 0.8	+0.89 (90.99)	34.2	37.6
	累计信息贡献率 – 0.4	+0.11 (90.21)	79.8	84.9
	累计信息贡献率 – 0.3	-0.54 (89.56)	87.0	91.6

VGG16。全局共享式的剪枝后模型精度相比于基准值最高提升 0.50%，此时浮点数计算量和参数量分别下降 31.3% 和 19.2%，而基于累计信息贡献率的剪枝后模型精度最高提升了 0.43%，浮点数计算量和参数量下降分别达到 58.3% 和 35.2%，其压缩率比基于全局共享式剪枝结果高约 1.8 倍。在理想结果与可接受结果中，两种剪枝后模型压缩率基本处于同一水平，然而全局共享式的剪枝结果（剪枝率为 0.8）精度损失已经超出 1%。

ResNet34。全局共享式的剪枝后模型精度相比于基准值最高仅提升 0.12%，浮点数计算量和参数量下降分别为 17.0% 和 18.9%，而基于累计信息贡献率的剪枝后模型精度最高提升了 0.89%，且模型压缩率比前者高约 2 倍。其余两组数据，基于累计信息贡献率的剪枝结果都能在模型压缩率更高的情况下取得比全局共享式剪枝结果更高的精度。

以上分析说明两个问题：（1）无论是剪枝率还是通道保留数，逐层定制式比全局共享式更合理；（2）基于累计信息贡献率的通道剪枝可以在单一超参数下为每个卷积层制定合适的通道保留数，兼顾了全局共享式的实现简单性与逐层定制式的剪枝合理性。

4.3.3 实验二：全自动化通道剪枝结果

本实验分为两组，分别验证本章提出的全自动化剪枝算法对不同模型和任务的自适应性。

（1） 在 CIFAR10 上分别训练了 AlexNet、GoogLeNet、VGG16、ResNet34、ResNet50 和 DenseNet121，并采用本章提出的基于累计信息贡献率的全自动化通道剪枝算法对这些模型进行自动化剪枝，其中可接受精度损失为 1%，最小搜索区间为 0.01，观察它们剪枝后相较于原模型在精度、浮点数计算量和参数量上的变化。剪枝结果如表 4-2 所示，各模型精度变化后括号内的数字表示其剪枝后模型的测试集精度。相比于原模型，在精度损失小于 1% 的基础上每个模型的浮点数计算量和参数量下降比例不同，这是因为这些模型原始的特征表达能力不同。如具备强大特征表达能力的 ResNet50，在精度损失低于 1% 时可以达到 95% 以上的压缩率，而早期的 AlexNet 在精度损失为 0.82% 时的浮点数计算量和参数量下降仅达 46.9% 和 27.8%。

（2） 在 CIFAR10、CIFAR100、STL10 和 ILSVRC-2012 上分别训练了 ResNet50，并采用本章提出的自动化算法进行通道剪枝，其中可接受精度损失为 1%，最小搜索区间为 0.01，结果如表 4-3 所示，表中各

数据集精度变化后括号内的数字表示 ResNet50 剪枝后精度。不同任务（数据集）对模型特征表达能力的要求是不同的，诸如 CIFAR10 和 STL10 等 10 分类任务，ResNet50 在压缩率达到 90% 以上时仍然能够维持小于 1% 的精度损失，而对于 ILSVRC-2012 等复杂的任务而言，ResNet50 在保持精度损失小于 1% 的基础上只能达到 35% 左右的压缩率。

表 4-2 不同模型在 CIFAR10 上的自动化剪枝结果

	精度变化 %	计算量 ↓%	参数量 ↓%
AlexNet	-0.82 (84.93)	46.9	27.8
GoogLeNet	-0.87 (90.19)	66.2	49.9
VGG16	-0.90 (88.18)	90.2	68.7
ResNet34	-0.73 (89.37)	86.4	91.2
ResNet50	-0.94 (87.60)	95.1	95.7
DenseNet121	-0.33 (86.81)	66.2	49.9

表 4-3 ResNet50 在不同数据集上的自动化剪枝结果

	精度变化 %	计算量 ↓%	参数量 ↓%
CIFAR10	-0.94 (87.60)	95.1	95.7
CIFAR100	-0.95 (82.48)	71.1	68.4
STL10	-0.79 (91.83)	99.2	99.7
ILSVRC-2012	-0.94 (74.88)	35.4	35.2

以上两组实验中，除了可接受精度损失 1% 和累计信息贡献率的最小搜索区间 0.01，本文没有为自动化通道剪枝过程施加任何人工干涉，即得到如表 4-2 和表 4-3 所示的剪枝结果，这验证了本章提出的基于累计信息贡献率的全自动化通道剪枝算法对不同模型和不同任务的自适应性。另外，算法对包括非专业人士在内的使用者十分友好，因为它仅有的参数设置对使用者的技术要求十分容易满足：**（1）可接受精度损失 $accLoss$** 是剪枝后模型相比于原模型的精度下降。使用者在设置该参数时只需要结合任务需要及自身期望即可，比如对精度要求较高则可设置较低的可接受精度损失。本文认为剪枝后模型精度损失在 1% 都是可接受的，如 3.3.1 节所述，因此本实验中可接受精度损失设为 1%。**（2）最小搜索区间 Δcr** 是最小的累计信息贡献率搜索区间间隔，是自动化剪枝算法的终止条件。这个参数与算法的搜索成本是相关的，比如当 $\Delta cr = 0.1$ 时，自动化剪枝算法共进行 4 次探查，而当 $\Delta cr = 0.01$ 时算法共进行 7 次探查。本文发现对于现有的经典卷积神经网络，两个累计信息贡献率相差小于 0.01 的剪枝结果在精度和压缩率上是十分接近的，因此本实验中将最小搜索区间设为 0.01。

4.3.4 实验三：与经典算法的对比

本实验沿用实验二中在 ILSVRC-2012 上训练的 ResNet50，并对其应用本章提出的基于累计信息贡献率的自动化通道剪枝，其中可接受精度损失为 1%，最小搜索区间为 0.01，并与一些经典的自动化剪枝算法比较，观察剪枝后模型精度、浮点数计算量和参数量的变化以及搜索成本。

表 4-4 展示了 ResNet50 在不同自动化剪枝算法下的结果。比起 ThiNet[41]、CP[42] 和 metaPruning[144]，本章提出的基于累计信息贡献率的全自动化通道剪枝算法可以在更短的时间内得到精度更高的剪枝结果。相比于 ABCPruner[142]，本章提出的算法在搜索成本上多 42 epochs，而剪枝后模型在精度上高 0.23%，在浮点数计算量减少上低 2.8%。必须说明的是，ABCPruner 为待剪枝模型设定了表示卷积层通道数最大保留比例的超参数 α，用以限定搜索空间从而控制搜索代价，这种设计可以有效降低搜索成本，然而其 α 的本质等同于全局共享式剪枝率，需要使用者尝试不同的值才能找到适合所有层的通道保留数比例上限，从而避免某些层在自动化剪枝开始阶段就处于剪枝过度的隐患。

表 4-4 不同算法对 ResNet50 在 ILSVRC-2012 上的剪枝结果及搜索成本

	精度变化 %	计算量 ↓%	搜索成本（epochs）
ThiNet-50	-4.90 (70.92)	58.7	244
CP	-3.71 (72.11)	34.1	206
metaPruning	-1.52 (74.30)	29.5	160
ABCPruner	-1.17 (74.65)	38.2	102
本章算法	-0.94 (74.88)	35.4	144

除了具有竞争性的自动化剪枝结果，本章提出的基于累计信息贡献率的全自动化通道剪枝算法还有一个突出特点：不需要实验性或经验性的超参数。算法仅有的参数为可接受精度损失和最小搜索区间，如 4.2.2 中所述，它们的设定对使用者几乎毫无技术要求，因此本章的算法对包括非专业人士在内的使用者十分地友好。

4.4 小结

本章提出了基于累计信息贡献率的全自动化通道剪枝算法。本章分析了基于剪枝率计算卷积层通道保留数在简单性与合理性上的矛盾，指出了通过剪枝率控制模型压缩的不合理性，并提出了基于累计信息贡献率的通道保留数算法，实现在全局共享式超参数的指导下逐层定制通道保留数。本章将累

计信息贡献率和二分搜索应用到自动化剪枝算法中，在累计信息贡献率的定义域上搜索最优剪枝结果，前者将自动化通道剪枝的高维搜索空间降低至一维，后者则通过减少探查次数进一步降低自动化剪枝的成本。基于累计信息贡献率的全自动化通道剪枝算法将控制模型压缩率的超参数转变为内部变量，避免了人工调参的过程，使用者只需设置可接受精度损失和最小搜索区间即可启动算法，实现了通道剪枝的全自动化，对包括非专业人士在内的使用者十分友好。实验结果表明，基于累计信息贡献率的全自动化通道剪枝算法在不同模型和数据集上具有通用性，相比于其它经典算法，本章算法可以在几乎零人工参与的情况下，获得相近甚至更好的剪枝结果，并且算法所需搜索成本平均降低 36%。

5 结论

针对当前深度模型在边缘计算场景应用中存在的设备资源受限与模型对硬件资源需求高的矛盾，本文从通道剪枝的角度对模型压缩方法进行研究，提出了一种高效可解释的全自动化通道剪枝算法：本文提出了基于权重置换的通道重要性度量算法，通过卷积层通道权重置换引发的输出特征图变化衡量通道的重要性，兼顾了模型权重自身所含的信息与训练集样本的信息，融合了基于度量的通道剪枝和基于误差重建的通道剪枝。本文分析了基于剪枝率的通道保留数计算方法，阐述了这种方法陷入简单性（全局共享式）与合理性（逐层定制式）难以兼容之窘境的原因是其超参数（剪枝率）在关注模型特征表达能力削减的同时忽略了模型所含信息量，并且说明了卷积层对通道剪枝的敏感度是其所含任务相关信息量的“感性”表现。本文提出了基于累计信息贡献率的通道保留数算法，说明指导模型压缩率的超参数应当对应模型信息量削减而非表达能力削减，在此基础上对通道剪枝过程做出调整，将传统的待剪枝通道的选择转换为待保留通道的选择。这种通道保留数计算方法，可以自动适配剪枝敏感度不同的卷积层，无需人工进行繁杂的剪枝敏感度实验。本文在基于累计信息贡献率的通道剪枝算法上实现了高效可解释的全自动化通道剪枝算法，将累计信息贡献率的定义域定义为搜索空间，基于模型压缩率与其精度的近似负相关关系采用二分搜索策略，实现了通道剪枝的自动化。由于本文提出的全自动化通道剪枝算法的输入对使用者几乎没有技术要求，算法对包括非专业人士在内的使用者十分友好。另外，本文使用不同模型在不同数据集上对提出的全自动化通道剪枝算法进行了实验与分析，结果表明本文对基于权重置换的通道重要性度量算法、基于累计信息贡献率的通道保留数算法和基于累计信息贡献率的全自动化通道剪枝算法的设计是正确且有效的。

本文提出的全自动化通道剪枝算法并不是完美的，算法在设计与实现上仍然存在改进空间：

（1） 在基于权重置换的通道重要性度量算法中，权重置换是基于随机重排序实现的，这种随机性将导致计算结果的不稳定。通过提高输入批尺寸（batch size）的方法可以在一定程度上缓解这个问题，但这种做法存在结果稳定性与计算简单性难以兼容的矛盾。

（2） 本文采用特征图像素点差值衡量两幅特征图的变化量，这种计算方式较为简单直接，可以考虑采用更具理论性的特征图相似度衡量方法，如传统的余弦相似度、SSIM[145] 等和基于深度学习的方法[146,147]。

（3） 在基于二分搜索的自动化剪枝中，搜索空间的调整和下一次探查依赖于当前剪枝结果。由于模型微调后的测试集精度存在波动性，当剪枝模型的精度损失接近可接受阈值时，可能出现二分决策的偏差。

前人已就模型压缩做了大量研究，细化出权重量化、网络剪枝、知识蒸馏、NAS 和轻量化模型设计等诸多方向以及许多具体的方法，本文认为未来相关研究中可以着眼于几个方向：（1）多种压缩方法的融合。结合 NAS 的网络剪枝昭示了不同方法相融合是未来模型压缩研究中的一个重要方向，比如在结合 NAS 的网络剪枝中融入权重量化，从模型权重存储格式的角度，在寻得最优模型结构的基础上进一步压缩模型。（2）降低模型压缩过程中的人工参与度和超参数依赖。现有方法大多需要在合适的超参数下获得理想压缩效果，如权重量化中的量化精度和网络剪枝中的剪枝率等，这些直接影响算法效果的超参数提高了算法使用成本，降低算法对超参数的依赖可以极大地提高它们的自动化程度。

参考文献

[1] 韦冬东. 变转速工况下的行星齿轮箱智能故障诊断方法研究 [D].

[2] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural Networks [J]. Communications of the ACM, 2017, 60(6): 84–90.

[3] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition [C]. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[4] Huang G, Liu Z, Van Der Maaten L, et al. Densely Connected Convolutional Networks [C]. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[5] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition [J]. International Conference on Learning Representations, 2014.

[6] Szegedy C, Liu W, Jia Y, et al. Going deeper with Convolutions [C]. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[7] 王兆丰年, 欧中洪, 宋美娜. 基于剪枝的卷积神经网络压缩算法综述 [Z](2022).

[8] Hinton G E, Salakhutdinov R R. Reducing the Dimensionality of Data with Neural Networks [J]. Science, 2006, 313(5786): 504–507.

[9] Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document Recognition [J]. Proceedings of the IEEE, 1998, 86(11): 2278–2324.

[10] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector [M]. Cham: Springer International Publishing, 2016: 21–37.

[11] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection [C]. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[12] Redmon J, Farhadi A. YOLO9000: Better, Faster, Stronger [C]. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[13] Nie Y, Han X, Guo S, et al. Total3DUnderstanding: Joint Layout, Object Pose and Mesh Reconstruction for Indoor Scenes From a Single Image [C]. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[14] Yang H, Carlone L. In Perfect Shape: Certifiably Optimal 3D Shape Reconstruction From 2D Landmarks [C]. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[15] He Y, Sun W, Huang H, et al. PVN3D: A Deep Point-Wise 3D Keypoints Voting Network for 6DoF Pose Estimation [C]. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[16] Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space [J]. International Conference on Learning Representations, 2013.

[17] Cho K, Van Merrienboer B, Gulcehre C, et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation [C]. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.

[18] Jordan M I. Serial Order: A Parallel Distributed Processing Approach [M]. Elsevier, 1997: 471–495.

[19] Vaswani A, Shazeer N M, Parmar N, et al. Attention is All you Need [J]. NIPS, 2017.

[20] Devlin J, Chang M-W, Lee K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [J]. North American Chapter of the Association for Computational Linguistics, 2019.

[21] Rastegari M, Ordonez V, Redmon J, et al. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks [M]. Cham: Springer International Publishing, 2016: 525–542.

[22] Lin X, Zhao C, Pan W. Towards Accurate Binary Convolutional Neural Network [J]. NIPS, 2017.

[23] Zhang Y, Xiang T, Hospedales T M, et al. Deep Mutual Learning [C]. ., 2018.

[24] Zhang L, Song J, Gao A, et al. Be your own teacher: Improve the performance of convolutional neural networks via self Distillation [C]. ., 2019: 3713--3722.

[25] Ji M, Shin S, Hwang S, et al. Refine myself by teaching myself: Feature refinement via self-knowledge Distillation [C]. ., 2021: 10664--10673.

[26] Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network [Z](2015).

[27] Zoph B, Le Q V. Neural Architecture Search with Reinforcement Learning [J]. International Conference on Learning Representations, 2016.

[28] Real E, Moore S, Selle A, et al. Large-Scale Evolution of Image Classifiers [J]. International Conference on Machine Learning, 2017.

[29] Luo R, Tian F, Qin T, et al. Neural Architecture Optimization [J]. Neural Information Processing Systems, 2018.

[30] LeCun Y, Denker J, Solla S. Optimal Brain Damage [J]. NIPS, 1989.

[31] Hassibi B, Stork D. Second Order Derivatives for Network Pruning: Optimal Brain Surgeon [J]. NIPS, 1992.

[32] Liu Z, Li J, Shen Z, et al. Learning Efficient Convolutional Networks through Network Slimming [C]. 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

[33] He Y, Lin J, Liu Z, et al. AMC: AutoML for Model Compression and Acceleration on Mobile Devices [M]. Cham: Springer International Publishing, 2018: 815–832.

[34] Ding X, Ding G, Guo Y, et al. Centripetal SGD for Pruning Very Deep Convolutional Networks With Complicated Structure [C]. 2019

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[35] Iandola F, Moskewicz M, Ashraf K, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model Size [J]. ArXiv, 2016.

[36] Howard A, Zhu M, Chen B, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [J]. ArXiv, 2017.

[37] Sandler M, Howard A, Zhu M, et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks [C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[38] Ma N, Zhang X, Zheng H-T, et al. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design [M]. Cham: Springer International Publishing, 2018: 122–138.

[39] Han K, Wang Y, Tian Q, et al. GhostNet: More Features From Cheap Operations [C]. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[40] Li H, Kadav A, Durdanovic I, et al. Pruning Filters for Efficient ConvNets [J]. International Conference on Learning Representations, 2016.

[41] Luo J-H, Wu J, Lin W. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression [C]. 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

[42] He Y, Zhang X, Sun J. Channel Pruning for Accelerating Very Deep Neural Networks [C]. 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

[43] Wen W, Wu C, Wang Y, et al. Learning Structured Sparsity in Deep Neural Networks [J]. NIPS, 2016.

[44] Zhang T, Ye S, Zhang K, et al. A Systematic DNN Weight Pruning Framework Using Alternating Direction Method of Multipliers [M]. Cham: Springer International Publishing, 2018: 191–207.

[45] Liu Z, Sun M, Zhou T, et al. Rethinking the Value of Network Pruning [J]. ICLR, 2018.

[46] Mittal D, Bhardwaj S, Khapra M M, et al. Recovering from Random Pruning: On the Plasticity of Deep Convolutional Neural Networks [C]. 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), 2018.

[47] McCulloch W S, Pitts W. A logical calculus of the ideas immanent in nervous Activity [J]. Bulletin of Mathematical Biology, 1990, 52(1–2): 99–115.

[48] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation [M]. Cham: Springer International Publishing, 2015: 234–241.

[49] Lin T-Y, Dollar P, Girshick R, et al. Feature Pyramid Networks for Object Detection [C]. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[50] Cheng H-T, Koc L, Harmsen J, et al. Wide & Deep Learning for Recommender Systems [C]. Proceedings of the 1st Workshop on Deep

Learning for Recommender Systems, 2016.

[51] Covington P, Adams J, Sargin E. Deep Neural Networks for YouTube Recommendations [C]. Proceedings of the 10th ACM Conference on Recommender Systems, 2016.

[52] Barkan O, Koenigstein N. ITEM2VEC: Neural item embedding for collaborative Filtering [C]. 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP), 2016.

[53] Jia Y, Zhang Y, Weiss R J, et al. Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis [J]. Neural Information Processing Systems, 2018.

[54] Kumar K, Kumar R, T. Boissière, et al. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis [J]. Neural Information Processing Systems, 2019.

[55] Ren Y, Ruan Y, Tan X, et al. FastSpeech: Fast, Robust and Controllable Text to Speech [J]. Neural Information Processing Systems, 2019.

[56] Ren Y, Hu C, Tan X, et al. FastSpeech 2: Fast and High-Quality End-to-End Text to Speech [J]. International Conference on Learning Representations, 2020.

[57] Finn C, Abbeel P, Levine S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks [J]. International Conference on Machine Learning, 2017.

[58] Dabney W, Ostrovski G, Silver D, et al. Implicit Quantile Networks for Distributional Reinforcement Learning [J]. International Conference on Machine Learning, 2018.

[59] Girshick R. Fast R-CNN [C]. 2015 IEEE International Conference on Computer Vision (ICCV), 2015.

[60] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(6): 1137–1149.

[61] Archer E, Park I-S, Pillow J W. Bayesian entropy estimation for binary spike train data using parametric prior Knowledge [J]. NIPS, 2013.

[62] Shen Y, Qin J, Huang L, et al. Invertible Zero-Shot Recognition Flows [M]. Cham: Springer International Publishing, 2020: 614–631.

[63] Stratos K, Collins M, Hsu D. Unsupervised Part-Of-Speech Tagging with Anchor Hidden Markov Models [J]. Transactions of the Association for Computational Linguistics, 2016, 4: 245–257.

[64] Chiu J, Rush A. Scaling Hidden Markov Language Models [C]. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020.

[65] Li Y, Shetty P, Liu L, et al. BERTifying the Hidden Markov Model for Multi-Source Weakly Supervised Named Entity Recognition [C]. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021.

[66] Burg G V D, Groenen P. GenSVM: A Generalized Multiclass Support Vector Machine [J]. Journal of machine learning research, 2016.

[67] Chen Y, Huang W, Nguyen L M, et al. On the Equivalence between Neural Network and Support Vector Machine [J]. Neural Information Processing Systems, 2021.

[68] Rigatti S J. Random Forest [J]. Journal of Insurance Medicine, 2017, 47(1): 31–39.

[69] Correia A H C, Peharz R, De Campos C P. Joints in Random Forests [J]. Neural Information Processing Systems, 2020.

[70] Defazio A, Bach F, Lacoste-Julien S. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives [J]. NIPS, 2014.

[71] Siahkamari A, Gangrade A, Kulis B, et al. Piecewise Linear Regression via a Difference of Convex Functions [J]. International Conference on Machine Learning, 2020.

[72] Li W, Zhou K, Qi L, et al. LAPAR: Linearly-Assembled Pixel-Adaptive Regression Network for Single Image Super-resolution and Beyond [J]. Neural Information Processing Systems, 2021.

[73] Baevski A, Schneider S, Auli M. Vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations [J]. International Conference on Learning Representations, 2019.

[74] Caron M, Bojanowski P, Joulin A, et al. Deep Clustering for Unsupervised Learning of Visual Features [M]. Cham: Springer International Publishing, 2018: 139–156.

[75] He Z, Kan M, Shan S. EigenGAN: Layer-Wise Eigen-Learning for GANs [C]. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 2021.

[76] Keller M, Zuffi S, Black M J, et al. OSSO: Obtaining Skeletal Shape from Outside [C]. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022.

[77] Gemp I, McWilliams B, Vernade C, et al. EigenGame Unloaded: When playing games is better than Optimizing [J]. International Conference on Learning Representations, 2021.

[78] Acuna D, Law M, Zhang G, et al. Domain Adversarial Training: A Game Perspective [J]. International Conference on Learning Representations, 2022.

[79] Gilles T, Sabatini S, Tsishkou D, et al. THOMAS: Trajectory Heatmap Output with learned Multi-Agent Sampling [J]. International Conference on Learning Representations, 2021.

[80] Lin M, Chen Q, Yan S. Network In Network [J]. International Conference on Learning Representations, 2013.

[81] He K, Zhang X, Ren S, et al. Identity Mappings in Deep Residual Networks [M]. Cham: Springer International Publishing, 2016: 630–645.

[82] 刘翔. 基于注意力机制和特征融合的甲状腺结节辅助诊断研究 [D].

[83] 张洋. 基于雷达传感网的信息融合研究 [D].

[84] 朱浩. 基于深度学习的少样本目标检测方法研究 [D].

[85] 周永庆. 基于一维并行多通道卷积神经网络的轴承故障研究 [D].

[86] 刘俊宏. 基于图像处理和小波卷积神经网络的气液两相流流型识别方法 [D].

[87] 张俊男. 无人机目标检测硬件加速设计与实现 [D].

[88] 荣昕萌. 基于非局部注意力与通道注意力的图像去噪方法研究 [D].

[89] 贾洁茹. 跨摄像机行人再识别中度量学习算法研究 [D].

[90] Zhang C-L, Luo J-H, Wei X-S, et al. In Defense of Fully Connected Layers in Visual Representation Transfer [M]. Cham: Springer International Publishing, 2018: 807–817.

[91] Hubara I, Courbariaux M, Soudry D, et al. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations [J]. Journal of machine learning research, 2016.

[92] Zhou Y, Moosavi-Dezfooli S-M, Cheung N-M, et al. Adaptive Quantization for Deep Neural Network [J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2018, 32(1).

[93] Liu Z, Wu B, Luo W, et al. Bi-Real Net: Enhancing the Performance of 1-Bit CNNs with Improved Representational Capability and Advanced Training Algorithm [M]. Cham: Springer International Publishing, 2018: 747–763.

[94] 李博闻. 深度神经网络量化及其硬件加速研究 [D].

[95] 顾逸枫. 基于深度学习的水稻病害边缘智能预警研究与实现 [D].

[96] Courbariaux M, Bengio Y, David J. BinaryConnect: Training Deep Neural Networks with binary weights during Propagations [J]. NIPS, 2015.

[97] Zhu C, Han S, Mao H, et al. Trained Ternary Quantization [J]. International Conference on Learning Representations, 2016.

[98] Gou J, Yu B, Maybank S J, et al. Knowledge distillation: A Survey [Z](2021).

[99] 郭亚光. 面向高分辨率图像的目标检测算法的研究 [D].

[100] Passalis N, Tefas A. Learning Deep Representations with Probabilistic Knowledge Transfer [M]. Cham: Springer International Publishing, 2018: 283–299.

[101] Lee S H, Kim D H, Song B C. Self-supervised Knowledge Distillation Using Singular Value Decomposition [M]. Cham: Springer International Publishing, 2018: 339–354.

[102] Passalis N, Tzelepi M, Tefas A. Heterogeneous Knowledge Distillation Using Information Flow Modeling [C]. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[103] Liu Y, Cao J, Li B, et al. Knowledge Distillation via Instance Relationship Graph [C]. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[104] Baker B, Gupta O, Naik N, et al. Designing Neural Network Architectures using Reinforcement Learning [J]. International Conference on Learning Representations, 2016.

[105] Liu H, Simonyan K, Yang Y. DARTS: Differentiable Architecture Search [J]. International Conference on Learning Representations, 2018.

[106] Cortes C, Gonzalvo X, Kuznetsov V, et al. AdaNet: Adaptive Structural Learning of Artificial Neural Networks [J]. International Conference on Machine Learning, 2016.

[107] Kandasamy K, Neiswanger W, Schneider J, et al. Neural Architecture Search with Bayesian Optimisation and Optimal Transport [J]. Neural Information Processing Systems, 2018.

[108] Zoph B, Vasudevan V, Shlens J, et al. Learning Transferable Architectures for Scalable Image Recognition [C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[109] Liu H, Simonyan K, Vinyals O, et al. Hierarchical Representations for Efficient Architecture Search [J]. International Conference on Learning Representations, 2017.

[110] Elsken T, Metzen J H, Hutter F. Simple And Efficient Architecture Search for Convolutional Neural Networks [J]. International Conference on Learning Representations, 2017.

[111] Pham H, Guan M, Zoph B, et al. Efficient Neural Architecture Search via Parameter Sharing [J]. International Conference on Machine Learning, 2018.

[112] Zhang X, Huang Z, Wang N, et al. You Only Search Once: Single Shot Neural Architecture Search via Direct Sparse Optimization [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021, 43(9): 2891–2904.

[113] Liu C, Zoph B, Neumann M, et al. Progressive Neural Architecture Search [M]. Cham: Springer International Publishing, 2018: 19–35.

[114] Klein A, Falkner S, Springenberg J T, et al. Learning Curve Prediction with Bayesian Neural Networks [J]. International Conference on Learning Representations, 2016.

[115] Bello I, Zoph B, Vasudevan V, et al. Neural Optimizer Search with Reinforcement Learning [J]. International Conference on Machine Learning, 2017.

[116] 夏磊. 基于元学习和通道剪枝的轻量级遥感图像目标检测方法研究 [D].

[117] Hanson S, Pratt L. Comparing Biases for Minimal Network Construction with Back-Propagation [J]. NIPS, 1988.

[118] Han S, Pool J, Tran J, et al. Learning both Weights and Connections for Efficient Neural Network [J]. NIPS, 2015.

[119] Han S, Liu X, Mao H, et al. EIE [J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 243–254.

[120] Han S, Mao H, Dally W. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding [J]. International Conference on Learning Representations, 2015.

[121] Chen S, Zhao Q. Shallowing Deep Networks: Layer-Wise Pruning Based on Feature Representations [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019, 41(12): 3048–3056.

[122] Yang T-J, Chen Y-H, Sze V. Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning [C]. 2017 IEEE

Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[123] Yang T-J, Howard A, Chen B, et al. NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications [M]. Cham: Springer International Publishing, 2018: 289–304.

[124] Ye J, Lu X, Lin Z, et al. Rethinking the Smaller-Norm-Less-Informative Assumption in Channel Pruning of Convolution Layers [J]. ICLR, 2018.

[125] Molchanov P, Tyree S, Karras T, et al. Pruning Convolutional Neural Networks for Resource Efficient Inference [J]. International Conference on Learning Representations, 2016.

[126] Lee N, Ajanthan T, Torr P H S. SNIP: Single-shot Network Pruning based on Connection Sensitivity [J]. International Conference on Learning Representations, 2018.

[127] He Y, Liu P, Wang Z, et al. Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration [C]. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[128] Ashok A, Rhinehart N, Beainy F, et al. N2N Learning: Network to Network Compression via Policy Gradient Reinforcement Learning [J]. ICLR, 2017.

[129] Peng H, Wu J, Chen S, et al. Collaborative Channel Pruning for Deep Networks [J]. ICML, 2019.

[130] Yu J, Huang T. AutoSlim: Towards One-Shot Architecture Search for Channel Numbers [J]. ., 2019.

[131] Cai H, Gan C, Han S. Once for All: Train One Network and Specialize it for Efficient Deployment [J]. International Conference on Learning Representations, 2019.

[132] Zhang X, Zhou X, Lin M, et al. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices [C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[133] Howard A, Sandler M, Chen B, et al. Searching for MobileNetV3 [C]. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[134] Luo J-H, Wu J. An Entropy-based Pruning Method for CNN Compression [J]. ArXiv, 2017.

[135] Molchanov P, Mallya A, Tyree S, et al. Importance Estimation for Neural Network Pruning [C]. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[136] Yu R, Li A, Chen C-F, et al. NISP: Pruning Networks Using Neuron Importance Score Propagation [C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[137] Huang Z, Wang N. Data-Driven Sparse Structure Selection for Deep Neural Networks [M]. Cham: Springer International Publishing, 2018: 317–334.

[138] Breiman L. Random Forests [J]. Machine-mediated learning, 2001.

[139] Fisher A J, Rudin C, Dominici F. Model Class Reliance: Variable Importance Measures for any Machine Learning Model Class, from the “Rashomon” Perspective [J]. ., 2018.

[140] He Y, Kang G, Dong X, et al. Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks [C]. Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, 2018.

[141] Dong X, Yang Y. Network Pruning via Transformable Architecture Search [J]. Neural Information Processing Systems, 2019.

[142] Lin M, Ji R, Zhang Y, et al. Channel Pruning via Automatic Structure Search [C]. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, 2020.

[143] Idelbayev Y, Carreira-Perpinan M A. Low-Rank Compression of Neural Nets: Learning the Rank of Each Layer [C]. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[144] Liu Z, Mu H, Zhang X, et al. MetaPruning: Meta Learning for Automatic Neural Network Channel Pruning [C]. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[145] Wang Z, Bovik A C, Sheikh H R, et al. Image Quality Assessment: From Error Visibility to Structural Similarity [J]. IEEE Transactions on Image Processing, 2004, 13(4): 600–612.

[146] Zagoruyko S, Komodakis N. Learning to compare image patches via convolutional neural Networks [C]. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[147] Zhang R, Isola P, Efros A A, et al. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric [C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

作者简历及在学研究成果

一、 作者入学前简历

起止年月	学习或工作单位	备注
2016 年 9 月至 2020 年 6 月	在北京科技大学计算机科学与技术专业攻读学士学位	

二、 在学期间从事的科研工作

2020 年 9 月至 2022 年 11 月，作为算法研究员参与实验室的快速信号处理系统开发及模型压缩研究项目。

三、 在学期间所获的科研奖励

四、 在学期间发表的论文

[1] [第一作者] 一种高效可解释的全自动化通道剪枝方法[P]. 北京市：202310566213X,2023-05-19.（已受理）

[2] [第五作者] 推理速度优先的层-通道联合剪枝方法[P]. 北京市：2023105662110,2023-05-19.（已受理）

独创性说明

本人郑重声明：所呈交的论文是我个人在导师指导下进行的研究工作及取得研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写的研究成果，也不包含为获得北京科技大学或其他教育机构的学位或证书所使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

签名：_____ 日期：_____

关于论文使用授权的说明

本人完全了解北京科技大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后应遵循此规定）

签名：_____ 导师签名：_____ 日期：_____

学位论文数据集

关键词*	**密级***	**中图分类号***	**UDC**	**论文资助**
深度学习，卷积神经网络，模型压缩，通道剪枝，神经网络结构搜索	公开	TP391		
学位授予单位名称*		**学位授予单位代码***	**学位类别***	**学位级别***
北京科技大学		10008	工学	硕士
论文题名*		**并列题名**		**论文语种***
一种高效可解释的全自动化通道剪枝算法				中文
作者姓名*	黄源翔		**学号***	G20208807
培养单位名称*		**培养单位代码***	**培养单位地址**	**邮编**
北京科技大学		10008	北京市海淀区学院路 30 号	100083
学科专业*		**研究方向***	**学制***	**学位授予年***
计算机技术		深度学习	3 年	2023
论文提交日期*	2023 年 5 月 30 日			
导师姓名*	齐悦		**职称***	副教授
评阅人	**答辩委员会主席***		**答辩委员会成员**	
	张德政		刘宏岚 阿孜古丽	
电子版论文提交格式 文本（ ） 图像（ ） 视频（ ） 音频（ ） 多媒体（ ） 其他（ ） **推荐格式：**application/msword；application/pdf				
电子版论文出版（发布）者		**电子版论文出版（发布）地**		**权限声明**
论文总页数*	71			
共 33 项，其中带*为必填数据，为 22 项。				