软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.005942]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

轻量级神经网络架构综述*

葛道辉[1], 李洪升[2], 张亮[2,4], 刘如意[1], 沈沛意[2], 苗启广[1,3]

[1](西安市大数据与视觉智能关键技术重点实验室(西安电子科技大学),陕西 西安 710071)

[2](嵌入式技术与视觉处理研究中心(西安电子科技大学),陕西 西安 710071)

[3](陕西省区块链与安全计算重点实验室(西安电子科技大学),陕西 西安 710071)

[4](上海宽带技术及应用工程研究中心,上海 200436)

通讯作者: 李洪升, E-mail: hsli@stu.xidian.edu.cn

摘 要: 深度神经网络已经被证明可以有效的解决图像、自然语言等不同领域的问题.同时伴随着移动互联网技术的不断发展,便携式设备得到了迅速的普及,用户提出了越来越多的需求.因此,如何设计高效、高性能的轻量级神经网络是解决问题的关键.本文详细阐述了三种构建轻量级神经网络的方法,分别是人工设计轻量级神经网络、神经网络模型压缩算法和基于神经网络架构搜索的自动化神经网络架构设计,同时简要总结和分析了每种方法的特点,并重点介绍了典型的构建轻量级神经网络的算法.最后,总结现有的方法,并给出了未来发展的前景.

关键词: 轻量级神经网络,便携式设备,神经网络模型压缩,神经网络架构搜索,自动机器学习

Survey of Lightweight Neural Network

GE Dao-Hui[1], LI Hong-Sheng[2], ZHANG Liang[2,4], LIU Ru-Yi[1], SHEN Pei-Yi[2], MIAO Qi-Guang[1,3]

[1](Xi'an Key Laboratory of Big data and Intelligent Vision (Xidian University), Xi'an 710071, China)

[2](Embedded technology & vision processing Research Center (Xidian University), Xi'an 710071, China)

[3](Shaanxi Key Laboratory of Blockchain and Secure Computingr (Xidian University), Xi'an 710071, China)

[4](Shanghai Broadband Network technology and Application Engineering Research Center, Shanghai 200436, China)

Abstract: Deep neural network has been proved to be effective in solving problems in different fields such as image, natural language and so on. At the same time, with the continuous development of mobile Internet technology, portable devices have been rapidly popularized, and users have put forward more and more demands. Therefore, how to design an efficient and high performance lightweight neural network is the key to solve the problem. In this paper, three methods of constructing lightweight neural network are described in detail, which are artificial design of lightweight neural network, compression algorithm of neural network model and automatic neural network architecture design based on searching of neural network architecture. The characteristics of each method are summarized and analyzed briefly, and the typical algorithms of constructing lightweight neural network are introduced emphatically. Finally, the existing methods are summarized and the prospects for future development are given.

Key words: Lightweight neural network, Mobile device, Compression of neural network, Neural network architecture searching, Auto Machine Learning.

* 基金项目: 国家重点研发计划(2018YFC0807500);国家重点研发计划项目(238);国家自然科学基金(61772396, 61472302,61772392, 61902296);西安市大数据与视觉智能关键技术重点实验室(201805053ZD4CG37);中央高校基本科研业务费专项资金(JBF180301);陕西省重点研发计划项目(2018ZDXM-GY-036)

Foundation item: the National Key R&D Program of China (2018YFC0807500); the National Key Research and Development Program (238); the National Natural Science Foundations of China (61772396, 61472302, 61772392); the Fundamental Research Funds for the Central Universities (JBF180301); and Xi'an Key Laboratory of Big Data and Intelligent Vision (201805053ZD4CG37); Shaanxi Province Key Research and Development Program (2018ZDXM-GY-036)

收稿时间: 2019-7-1; 修改时间: 2019-8-18; 采用时间: 2019-11-2; jos 在线出版时间:2019-12-5

CNKI 网络优先出版: 2019-12-05 14:55:05, http://kns.cnki.net/kcms/detail/11.2560.TP.20191205.1454.005.html

深度学习与传统手工设计目标特征的方法不同,深度学习以端到端的方式训练深度神经网络,实现自动提取目标的深度特征,避免人为设计的干扰,同时,深度特征与传统特征相比,深度特征可以多层次表示目标,从浅层的局部特征到深层的全局特征,具有更强的鲁棒性和表达能力.基于此,深度学习近些年来已经引起了学术界和工业界的广泛关注,并且深度学习已经成功的应用于解决多种类型的任务中,如目标检测、图像检索、语言识别、智能问答等领域.

移动互联网时代智能手机和其他便携设备的普及给人们之间的日常交流、信息获取、学习和工作等生活的各个方面均带来了极大的便利.伴随着移动互联网的迅速发展,推出了如人脸识别、视频直播、美颜相机、拍照识图,自动驾驶等多种不同类型的移动式应用,丰富了人们的日常生活,其中产生的数据绝大部分为静态图像和动态视频数据.目前,传统的深度神经网络[1-3]通过设计非常深的神经网络结构用于提取表达能力更强的深度特征,这对存储设备和计算资源的要求非常高,常用的便携式设备无法满足该需求,这严重限制了深度神经网络在便携式设备上的发展与应用.

为了提高便携式设备处理图像和视频数据的效率和能力,同时需要满足存储空间和功耗的限制,设计适用于便携式设备的轻量化深度神经网络架构是解决该问题的关键.近些年来,轻量级神经网络架构的设计得到了学术界和工业界的广泛关注,提出了一些典型的方法[4-9],主要包括三个不同的方向,分别是(1)人工设计轻量化神经网络模型；(2)基于神经网络架构搜索(Neural Architecture Search, NAS)的自动化神经网络架构设计；(3)神经网络模型的压缩.轻量级神经网络架构的设计已经取得了一定的成果,谷歌通过深度可分离卷积代替标准卷积提出了轻量级网络架构 MobileNet V1[8],Face++通过逐点群卷积核通道混洗技术提出了ShuffleNet V1[9]；MnasNet[7]和 NasNet[6]通过强化学习方法学习神经网络架构搜索策略,实现便携式设备上轻量化神经网络的自动化构建,不同的是 NasNet[6]设计了基于块的搜索空间,大大加快搜索速度； Deep compression[4]通过剪枝、权值共享和权值量化、哈夫曼编码实现卷积神经网络模型的压缩；AMC(AutoML for Model Compression, AMC)[5]利用强化学习方法自动学习模型压缩策略,具有更高的压缩比,可以更好的保持神经网络的性能.

目前,人工设计轻量级神经网络的主要思想在于设计更高效的网络计算方式,主要是针对卷积的计算方法.现有的深度卷积神经网络为了能够取得更好的性能,通过设置大规模的特征通道数、卷积核大小的数量,但是往往存在大量的冗余.人工设计轻量级神经网络通过合理的减少卷积核的数量,减少目标特征的通道数,结合设计更高效的卷积操作等方式,从而构造更加有效的神经网络结构,可以在保持神经网络性能的前提下,显著的减少网络的参数和计算量,实现在便携式设备上训练和应用深度神经网络.

MobileNet[8,10]、ThunderNet[11]、ShuffleNet[9,12]、SqueezeNet[13]等人工设计的神经网络虽然已经取得了令人瞩目的成绩,但是设计高性能的轻量级神经网络需要设计者具有丰富的专业知识和领域知识,并且需要大量重复的实验,导致研究成本和时间成本极高,严重限制了轻量级神经网络在便携式设备上的发展与应用.为了减少人为因素的干扰,通过给定所有候选神经网络架构的集合作为搜索空间,使用学习到的搜索策略从搜索空间中构建最优的神经网络架构,利用性能评估策略度量网络架构的性能,并在训练阶段,作为奖励指导搜索策略的学习,通过反复的迭代,从而得到解决特定任务的最优神经网络架构,实现深度神经网络模型的自动搜索.神经网络架构搜索方法与超参数优化[14]和元学习[15]有显著的重叠.神经网络架构搜索方法主要由三部分组成:搜索空间,搜索策略和性能评估策略.

除人工设计轻量化神经网络模型外,学者和工业界也在不断的探索如何进一步的通过压缩神经网络模型的规模,进一步的降低对存储设备和计算资源的需求,实现在便携式设备上应用深度神经网络.根据神经网络中每层的冗余程度,通过将网络权重的全精度的浮点数进行裁剪、对网络中间的特征输出进一步进行量化,以及剪枝、权值共享、低秩分解、知识蒸馏等方法实现神经网络模型的压缩.通过压缩神经网络模型,降低了占用的存储空间,满足功耗的限制,嵌入到便携式设备的芯片上,实现实时的运行.

人工设计的神经网络模型压缩技术依赖启发式和基于规则的策略,算法设计者需要探索较大的设计空间,以及在模型大小、速度和准确率之间权衡,而这通常是次优的且耗时的.自动机器学习(Automated Machine Learning, AutoML)通过结合强化学习等方法将特征提取、模型选择、超参优化等步骤实现自动化的学习,提高模型的压缩比,在更好地保持神经网络模型性能的同时减少人为因素的干扰.

随着轻量级神经网络的不断发展,提出了越来越多的设计方法,已经被成功的用于解决各类问题,在图像分类问题上已经超过了人工设计的方法,并广泛应用于目标检测、图像分割、智能问答等不同领域.本文首先概括性介绍轻量级神经网络,第一章介绍人工设计轻量级神经网络方法,第二章详细介绍了基于规则和基于自动机器学习的神经网络模型压缩算法,第三章总结了基于神经网络架构搜索的自动化轻量级网络设计,最后展开讨论并对未来的研究方向给出建议.

1 人工设计的轻量级神经网络模型

在本节中,我们将介绍人工设计的高效的轻量化网络结构及其背后的原理.随着模型参数量的增加,模型的准确率和复杂度也逐步增加,但同时,神经网络中存在较大冗余.设计更加高效的网络结构对于压缩模型大小、加快运行速度以及减轻训练难度都有重要的作用.本节将从减少卷积核的冗余、减少输入特征的通道数,设计更加高效的模块三个方面介绍如何设计轻量级的神经网络.

1.1 使用小卷积核代替大卷积

利用多层小卷积核替代一层大卷积核可以有效的减少网络的参数,在 Simonyan 等人提出的 VGG[1]网络中,使用 3×3 卷积核代替 5×5 和 7×7 大小的大卷积核,对于一个大小 5×5 的感受野,可以通过两层 3×3 大小的卷积实现.对于一个 7×7 的卷积核可通过三层 3×3 卷积实现,如图 1 左.

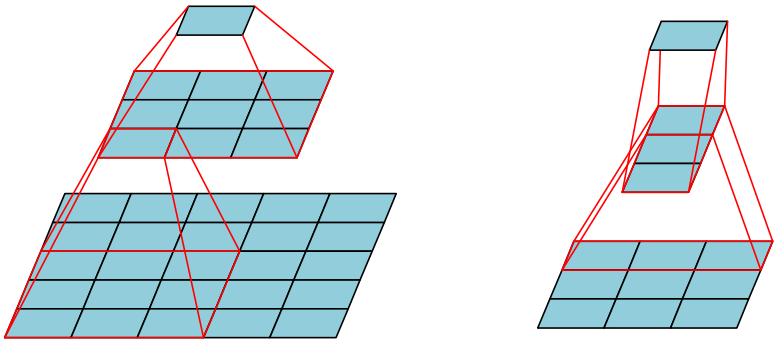


图 1 左图为 5×5 卷积分解为两层 3×3 卷积,右图为 3×3 卷积分解为连续的 1×3 和 3×1 的卷积

在参数量上,5×5 大小的卷积核参数量为 25,而两层 3×3 的卷积核参数量为 18,相比减少了 28%的参数量.从浮点运算数(FLOPs)角度,对于输入大小为 $H\times W\times C_{in}$ 的特征,输出为 $H\times W\times C_{out}$ 大小的特征图时,$FLOPs_{5\times5}=H\times W\times C_{in}\times C_{out}\times5^2$,而两层卷积的 $FLOPs_{(3\times3)\times2}=2\times H\times W\times C_{in}\times C_{out}\times3$,计算量也相应的减少.并且两层 3×3 卷积可以合并两层非线性层,比一层大卷积核更能增加非线性能力.

类似的,Jaderberg[16]等人提出将一个 3×3 的卷积操作分解为连续的 1×3 和 3×1 的卷积,如图 1 右,分解之后参数量和 FLOPs 都下降了 33%.Chao[17]提出使用大的 $k\times k$ 卷积核可以增加感受野,同时为了减少计算量,可以使用两层 $1\times k$ 和 $k\times1$ 的卷积代替,这样参数量和 FLOPs 都变为之前的 $2/k$.

1.2 限制中间特征的通道数量

对于标准的不带 bias 的卷积操作,$FLOPs=H\times W\times C_{in}\times C_{out}\times k^2$,运算量受到输入通道数 C_{in} 和卷积核数量 C_{out} 的影响,一般来说卷积核数量代表提取的特征数量,减少会影响网络的准确率,因此可以选择降低输入通道数 C_{in} 来减少运算量.

Iandola 等人在 SqueezeNet[13]中提出了 Fire module,如图 2 所示,在保证准确率的同时减少运算量.Fire module 包含两部分,压缩层(squeeze)层和扩张层(expand)层,通过减少 squeeze 层的通道数量来减少整个模型需要的计算量.与 AlexNet[18]相比,在保证了相同的性能的条件下,模型大小压缩了近 50 倍.

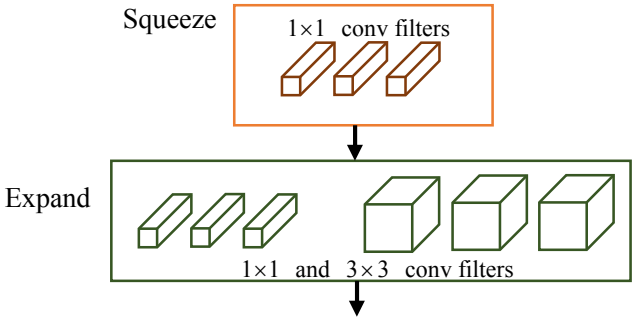


图 2 SqueezeNet 的 Fire Module,可以分为 squeeze 层和 expand 层

1.3 分解卷积运算

标准的卷积操作是将一个卷积核用在输入特征的所有通道上.一方面模型的参数量较大,另一方面合并所有通道的卷积运算存在很大冗余.通过分组卷积,ResNeXt[19]将多个分支合并为一个分支,与 ResNet[2]同等计算消耗的情况下,有着更高的准确率.

Howard 等人提出了 MobileNet V1[8]利用深度可分离卷积(Depthwise Separable Convolution)对标准的卷积进行了分解,如图 3 所示.深度可分离卷积(Depthwise Separable Convolution)可分为深度卷积(Depthwise Convolution)和逐点卷积(Pointwise Convolution)两个操作.深度卷积(Depthwise Convolution)对于每个输入通道采用不同的卷积核,即一个通道对应一个卷积核,卷积操作是按照通道进行分解的；逐点卷积(Pointwise Convolution)是卷积核大小为1×1的标准卷积,作用在输入的所有通道上,将来自不同通道的特征进行融合.

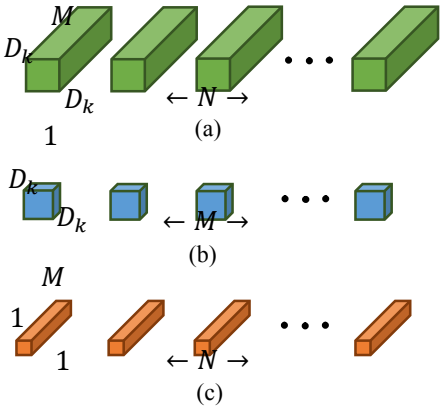


图 3 (a)为标准的卷积操作,(b)为深度卷积(Depthwise Convolution),(c)为逐点卷积(Pointwise Convolution),(b)和(c)合起来组成深度可分离卷积(Depthwise Separable Convolution)

标准卷积的计算量为 $D_k \times D_k \times M \times N \times H \times W$,而两步的深度可分离卷积(Depthwise Separable Convolution)卷积的计算量为 $D_k \times D_k \times M \times H \times W + M \times N \times H \times W$,与标准卷积的运算量相比, $\frac{D_k \times D_k \times M \times D_F \times D_F + M \times N \times D_F \times D_F}{D_k \times D_k \times M \times N \times D_F \times D_F} = \frac{1}{N} + \frac{1}{D_k^2}$,计算量大致变为原来的1/9至1/8.最终,MobileNet[8]与VGG-16[1]相比,在准确率接近一致的情况下,模型大小是 VGG-16 的1/32,计算消耗量是其1/27.

一般常用的卷积核大小为3×3,即一个深度可分离卷积(Depthwise Separable Convolution)的 FLOPs 为 $3 \times 3 \times M \times H \times W + 1 \times 1 \times M \times N \times H \times W$, M 为输入特征通道数, H,W 为输入特征高和宽, N 为当前层卷积核数量,一般来说 N 要远大于 9,此时深度卷积(Depthwise Convolution)的主要计算量集中在1×1的逐点卷积(Pointwise Convolution)上,为了解决这个问题,ShuffleNet[9]提出了 Group 逐点卷积(Pointwise Convolution),将

逐点卷积(Pointwise Convolution)进行分组卷积操作,以降低在逐点卷积(Pointwise Convolution)操作方面的计算消耗,如图 4 所示.

ShuffleNet[9]首先将输入通道分组,在每个组中进行单独的卷积运算,如果单纯的堆叠分组卷积操作,会造成各个组之间的信息无法流通,影响网络的表达能力.为了让分组卷积能够得到其他分组产生的特征,ShuffleNet[9]提出了混洗(Shuffle) 操作,将来自不同组的特征重新进行排列,使得新的分组中包含来自之前各个组的特征,保证了各个组之间的信息流通.

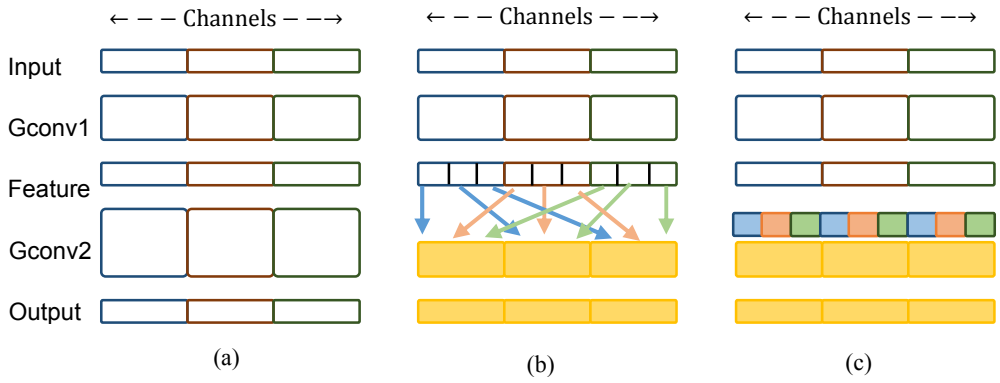


图 4 (a)为一般分组卷积(分组卷积(Group Convolution)),造成分组之间信息无法流通,(b)和(c)为通道混洗操作(Channel shuffle)

ShuffleNet[9]利用混洗(Shuffle)操作和分组逐点卷积(Group Pointwise Convolution)组合成 ShuffleNet Unit.当输入特征大小为 $H\times W\times C_1$ ，Bottlenect 的通道数为 C_2 时 .ResNet unit 需要 $2\times H\times W\times C_1\times C_2+9\times H\times W\times C_2^2$ 的 FLOPs,ResNeXt[19]需要 $2\times H\times W\times C_1\times C_2+9\times H\times W\times C_2$ 的 FLOPs,而 ShuffleNet Unit 只需要 $\frac{2\times H\times W\times C_1\times C_2}{g}+9\times H\times W\times C_2$ 的 FLOPs,进一步减少了计算量.最终 ShuffleNet[9]在与 AlexNet[18]相同准确率的情况下比后者快 13 倍.

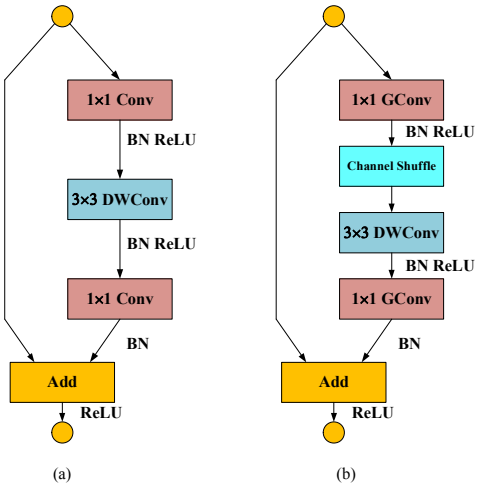
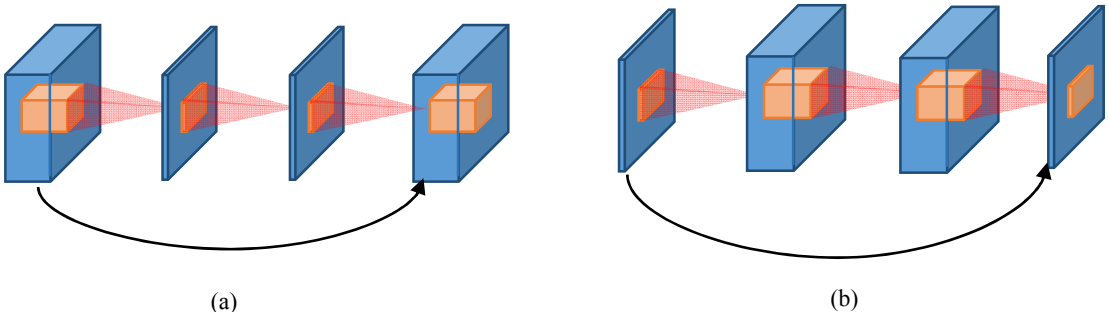


图 5 (a)是深度分离卷积(Depthwise Separable Convolution,(b)为 ShuffleNet Unit

为了在保持网络规模的前提下进一步提高网络的性能,在后来的 MobileNet V2[10]中,引入了残差结构,并在残差结构上进行改造.首先,由于深度卷积(Depthwise Convolution)本身不能改变特征的通道数,并且在输入通道较少的情况下无法较好的提取特征,因此在残差分支中使用逐点卷积(Pointwise Convolution)来先增加后

减少特征通道数,使得深度卷积(Depthwise Convolution)层工作在高维的特征之中,与之前的残差分支正好相反.其次,激活函数在高维的空间中能够有效的增加非线性能力,但是在低维空间中会破坏特征的表达能力,因为第二个逐点卷积(Pointwise Convolution)起到降维的作用,所以去掉第二个逐点卷积(Pointwise Convolution)之后的激活层,也就是使用了线性的Bottleneck,如图6所示.由于提升了网络的表达能力,MobileNet V2[10]只需要 MobileNet V1[8]的 2 / 5 的计算量就能达到相同的性能.



(a) (b)

图 6 (a)为正常的 Residual Block,残差分支通道数小于主分支通道数,(b)为反残差结构(Inverted residual block),残差分支通道数大于主分支通道数.

后来,Ma[12]等人发现,FLOPs 作为一个间接的指标,只能大致地估测运行速度,与速度并不完全相符.主要原因有两方面,一是很多操作无法被算入 FLOPs 中,例如内存访问成本(Memory Access Cost, MAC)、并行化程度等;二是受到计算平台的影响.Ma[12]等人提出了四种轻量级网络的指导方案:1、相同的输入输出通道数能够减少内存访问成本(MAC);2、过多的分组卷积(Group Convolution)会增加 MAC;3、网络的碎片化程度会减少并行化程度;4、不能忽略元素级操作.

根据以上四条方案,在 ShuffleNet V1[9]的基础上进行改进,如图 7 所示:

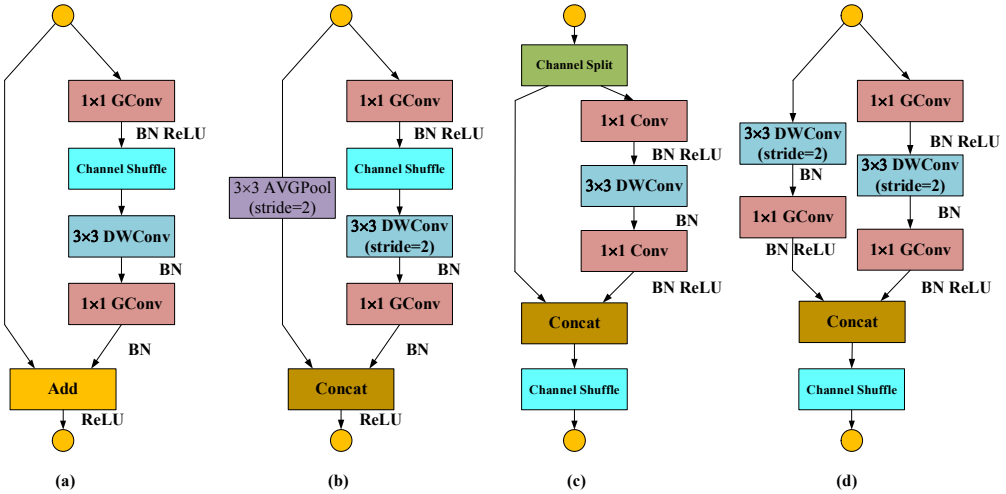


图 7 (a)(b)分别为空间尺度不变的ShuffleNet V1 Unit和空间尺度变小的ShuffleNet V1 Unit,(c)(d)分别为空间尺度不变的 ShuffleNet V2 Unit 和空间尺度变小的 ShuffleNet V2 Unit

(a)(b)为原始的 ShuffleNet V1[9] Unit,(c)(d)为改进后的 ShuffleNet V2[12] Unit.(c)利用通道分割(Channel Split)的方法,将输入分割成两部分,一半作为直接连接分支,一半作为残差分支,并且残差分支中$1*1$卷积的输入输出通道数相等(符合方案 1),并且不使用分组卷积(Group Convolution)(符合方案 2).最后进行 Concat,避免了特征相加的操作(符合方案 4).(d)去掉了通道分割(Channel Split)操作,从而在减少了特征图的空间尺寸同时,通过 Concat 增加了通道数.ShuffleNet V2 通过提出新的指标(MAC),并且在 MAC 上优化,加快了运行速度同时也提升了准确率.新的 ShuffleNet V2 比 MobileNet V1 快 58%,比 ShuffleNet V1 快 63%,比 Xception 快 25%.

在 Zhang 提出的 IGCNets[20]中同样的利用到了分组卷积(Group Convolution),深度卷积(Depthwise

Convolution)可以看作是分组卷积(Group Convolution)的一种极限情况,即将每个通道分为一组.在 IGCNets[20] 中,使用了两级分组卷积(Group Convolution),第一级是空间卷积,一般大小为3×3,第二级是1×1的分组卷积(Group Convolution).为了防止各个组的特征之间无法进行信息交互,采取了不同的分组数量,即第一级卷积分为 2 组,然后对特征进行重新排列,第二级卷积分为 3 组,并且每一组都包含之前所有组的一部分特征,如图 8 所示.

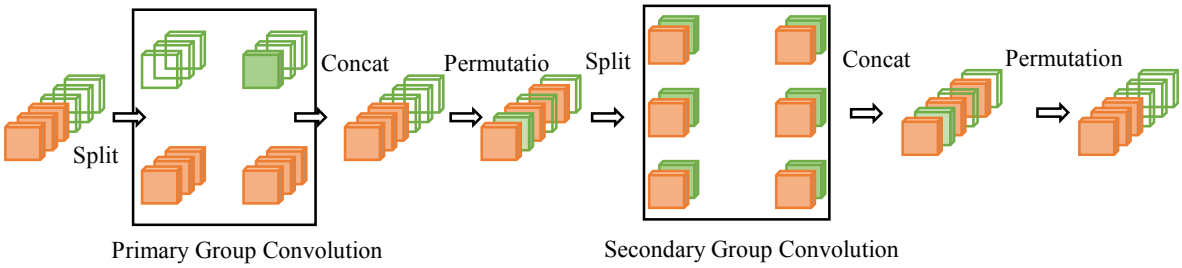


图 8 交错组卷积结构(IGCV1)

而 IGCV2[21]更进一步,通过使用3×3的深度卷积(Depthwise Convolution)的卷积和 L 个1×1的分组卷积(Group Convolution)来提升卷积效率.并且要求这一系列的卷积都满足互补条件(complementary condition),即最终的输出的特征中的一个通道存在且仅存在一条路径连接到输入特征的任意一个通道.最终确保由多个稀疏的卷积核(分组卷积)组成的卷积核矩阵是稠密的,如图 9 所示.

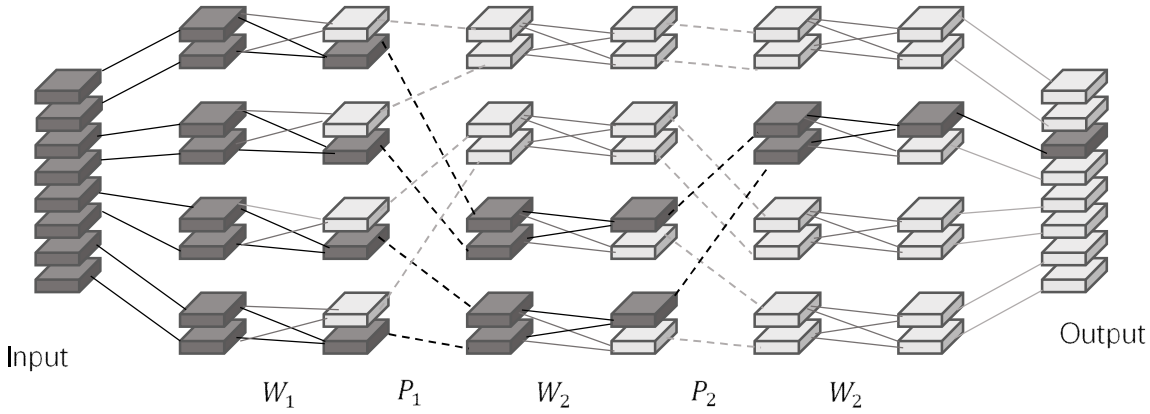


图 9 满足互补条件(complement condition)的交错分组卷积结构(IGCV2)

Mehta 等人提出了 EspNet[22],将标准卷积分解为1×1的逐点卷积(Pointwise Convolution)和空间金字塔扩张卷积(spatial pyramid of dilated convolutions).在减少了参数量和计算量的同时,增加感受野.在随后的 EspNet V2[23]中,将原来的 dilated convolution 也采用了深度卷积(Depthwise Convolution).同时将1×1的逐点卷积(Pointwise Convolution)更改为1×1的分组卷积(Group Convolution).

人工设计轻量化神经网络,不同于对现有的网络进行压缩,而是重新设计更加高效的网络单元(主要针对卷积运算),从而在不损失网络性能的条件下,减少网络的参数量,提升网络运行速度.以上介绍的三种主要方式,主要从卷积核的空间尺度、卷积的输入通道数以及卷积的通道之间的稀疏连接等方面对标准卷积操作进行改进.在这一方向,模型压缩和加速都已经取得了较好的成果.

2 卷积神经网络的压缩算法

2.1 基于规则的神经网络模型压缩

除人工设计轻量化神经网络模型外,学者们也在探究如何将已有的网络进行压缩,从而实现在便携式设

备上运行.根据神经网络在不同方面的冗余,分别使用网络的分支裁剪,减少网络权重占用的比特数,对卷积核进行低秩分解,知识蒸馏等方法,对已有的网络模型进行压缩,降低对空间和计算能力的需求,实现在便携设备上的实时运行.

权重裁剪:权重裁剪是指对网络结构中的权重进行删减从而达到压缩网络的目的.权重裁剪基于一个假设,在网络中很多参数都是不需要的或者不重要的,通过裁剪的方法可以将这些参数移除.通过权重裁剪的方法能够带来两点好处:首先参数量的减少所需要的存储空间也会相应的减少,其次,由于参数的减少导致网络运行时计算量的减少,能够减少对计算能力的需求,同时增加运行速度.

Li 等人提出基于权重 L1 范数的逐层裁剪策略,该策略为对每层的滤波器的 L1 范数进行排序,然后删除范数最小的滤波器,再对裁剪后的网络进行重新训练以恢复精度[24]; LIU 等人提出了一种针对卷积神经网络中的批量归一化(batchnorm)层进行 L1 稀疏化的压缩算法,该算法通过迫使批量归一化(batchnorm)层中一些 scale 参数接近于 0,从而可以安全的删除这些节点而不会对网络的精度造成巨大的影响,并通过多次使用该策略使得网络达到很高的压缩比[25];Hu 等人发现卷积神经网络中有大量的神经元的输出结果接近于 0,于是提出了一种 Apoz 的通道重要性衡量方法,以单个神经元通道的输出结果中值接近为 0 的占比来衡量该神经元的重要性[26].Tian 等人通过 LDA 方法来获取与分类最相关的权重,而裁减掉与分类相关度不大的权重,在人脸检测网络中得到了很高的压缩比[27]; Molchanov 等人提出了一种通过衡量裁剪前后权重误差的泰勒展开式来衡量通道重要性的策略来逐步裁剪卷积神经网络[28]; Luo 等人提出了一种方法,通过最小化裁剪后通道数在下一层的误差的方法来去掉不重要的神经元[29],得到了非常鲁棒的裁剪效果; He 等人为每个通道增加一个掩膜,然后利用稀疏优化的策略来迭代优化掩膜和权重,从而最小化裁剪后的网络与原始网络之间的误差[30].Wen 等人提出一种在滤波器级别,通道级别以及层级别通过组 Lasso 稀疏化进行网络裁剪的方法,可以在不同层面上对卷积神经进行裁剪[31]; Yu 等人提出一种将权重的重要性从后往前逐层反传的方法,可以快速有效地得到整体网络的权重重要性[32].

权重量化:在权重量化层面,Gupta 发现,使用 16 位的定点数作为权重,足够在 MNIST 上训练一个神经网络[33].此外,Dettmers 研究了 8 位定点量化,加快并行训练中深度网络的收敛速度[34].Han 等人提出了结合权重剪枝,量化和霍夫编码的策略, 可以得到非常高的压缩比,但是这种方法需要专门的运算策略来实现.

Matthieu Courbariaux 等人提出了二值权重网络[35](Binary Weight Network, BWN)即对于网络的权重而言,只有 1 或-1 两个值.BWN 采用了一种混合策略(BinaryConnect)对于网络的中间层特征,保留其原始精度,只将网络权重进行二值化,将网络前向传播与反向传播时的乘法操作变为加法操作.在网络的训练过程中,二值化的权重应用于前向传播与反向传播的梯度计算,而在更新权重时,采用全精度的权重,当全精度的权重越过阈值时,其对应的二值化后的权重就会发生改变.在测试时,只保留和使用二值化之后的权重,每个权重只占用一个 bit 的空间,对于 32 位或者 64 位 bit 的浮点数,有 32~64 倍的压缩倍率,并且由于用累加代替了乘法运算,使得网络的运行效率也大幅提升.

二值权重网络的成功表明对权重进行量化是可行的,随后 Matthieu Courbariaux 等人再次提出了二值神经网络(Binary Neural Network, BNN)[36],进一步增加了二值神经网络的量化程度,将需要的计算量压缩至极低的水平.二值神经网络与二值权重网络相比,不仅在权重上是二值化的,并且在每一层的激活值上也进行了二值化.BNN 将每一层的激活值和权重的 1 和-1 分别用 1 和 0 代替,原来用 32 bit 存储的权重和激活值只需 1 bit 就能存储.同时,由于激活值和权重都为 1 或-1,在计算的过程中产生的数值仍然是 1 或-1,将两个浮点数的乘法计算变成了 1 bit 的位运算.从而从参数的占用空间和计算量两个方面减少了对承载设备能力的需求,使得模型能够更加容易的移植到嵌入式设备上.

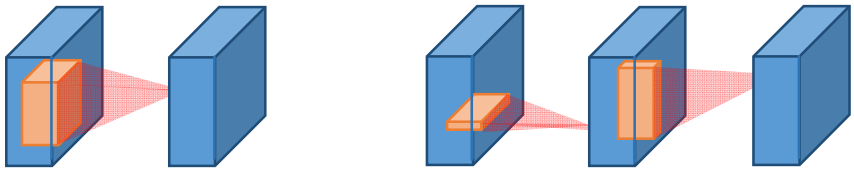
虽然二值神经网络极大地减少了计算量和模型大小,但是其精度有待提高.由于二值权重网络的权重只有 1 和-1 两种状态,而一般神经网络训练得到的权重往往是均值为 0 的正态分布,导致原始精度的权重与量化后的权重存在较大的误差.Fengfu Li 和 Bo Zhang 等人为了解决这个问题,在二值权重网络(BWN)的基础上提出了三值权重网络(TWN)[37].与二值权重网络相比,三只权重网络增加了 0 这种状态,即权重存在-1,0 和 1 三种量化状态.同时,由于状态的增加,模型的表达能力得到增强,以常用的$3*3$大小的卷积核为例,二值权重网络的 3×3 卷积核存在 $2^{3\times3}=512$ 种状态,而三值权重网络存在 $3^{3\times3}=19683$ 种状态; 在模型的压缩方面,由于

存在三值权重网络存在三种状态,因此压缩能力是二值权重网络的一半,但是从压缩模型角度来看已经足够;在网络的计算上,新增加的 0 状态不需要额外的计算,因此三值权重网络与二值权重网络所需的计算量相同.从以上三个方面来看,三值权重网络在与二值权重网络在运算量基本相同的条件下能够显著提高了网络的准确率.

Leng 等人将网络的低比特量化问题建模为离散约束优化问题,再将原问题分解为若干子问题,并通过交替求解子问题对网络进行量化,并取得了很好的效果[38]; Hu 等人提出了一种基于哈希来训练二值神经网络的算法,通过哈希算法来训练二值网络[39]; Wang 等人提出一种在激活函数和权重上先后量化网络的方法[40]; Frederick 等人提出一种同时进行剪枝和量化的网络裁剪架构,充分利用了两类方法的优势[41].

低秩分解:卷积层的卷积核 $W \in R^{w\times h\times c\times n}$ 是一个 4D 的张量,低秩分解的方法主要是将卷积核进行分解以减少冗余,低秩分解的关键在于如何对卷积核的参数进行排列,以及要在哪个维度上进行秩的约束.

Emily Denton 研究了多种低秩分解策略,例如二维矩阵的 SVD 分解,三维张量的二维展开,以及基于聚类分解等[42].Zhang 采用了广义奇异值分解策略,在分类精度轻微下降的情况下实现四倍的加速效果[43].将一个四维的卷积核形状转换成 $(w\times h\times c)\times n$,然后将其分解为两部分,如图 10 所示,一部分为 d 个大小为 $w\times h\times c$ 的卷积核,另一部分为 n 个 $1\times 1\times d$ 的卷积核,其中 $d<n$,在卷积核的数量上施加低质约束,原来的 n 个卷积核可以看作是 d 个大小为 $w\times h\times c$ 的卷积核的线性组合.用这种方法,在 VGG[1]上实验,理论上可以获得三倍的速度提升,top-5 error 仅仅增加 1.66%.类似的,也可以在特征的输入通道数 c 上施加低秩约束,将卷积核形状变为 $c\times(w\times h\times n)$,然后分解为 c 个 $1\times 1\times d$ 的卷积核和 d 个大小为 $w\times h\times n$ 的卷积核.两种方法只是施加约束的维度不同,是对称的分解操作.Lebedev.V.等人提出了一种基于 CP 分解对卷积网络中的卷积核进行压缩的方法[44]. Kim 等人提出一种基于 tucker 分解的卷积核分解方法,同样可以有很好的压缩效果[45].



(a) (b)

图 10 将卷积核进行低秩分解操作

知识蒸馏:不同于其他的压缩或加速的方法,知识蒸馏是创建一个“学生”网络,然后让其去拟合“教师”网络分布的方法,使得小型网络能够学习到大型网络的知识.一般来说,“教师”网络是一个较大的网络,而“学生”网络是一个轻量级的网络.利用网络之间的知识传递,而不是直接训练一个轻量级网络.

Hinton 首先提出了知识蒸馏的概念,让小网络在学习分类目标的同时,也尽量拟合大网络对不同类别的软分类结果[46],如图 11; Romero 等人在 Hinton 的基础提出 Fitnet[47],网络的深度比宽度更加重要,因此,在 Fitnet 中使用了更深但是更窄的网络当作“学生”网络,并且不单只考虑最终分类层的知识提取,也考虑中间层特征的拟合,基于这个概念甚至能够取得比原始网络更好的分类结果.Zagoruyko 等人提出了 Attention Transfer(AT)扩展了 FitNet 的约束,让“学生”网络学习“教师”网络的注意力图(attention map)而不是中间层的特征.取得了比 FitNet 更好的效果[48].

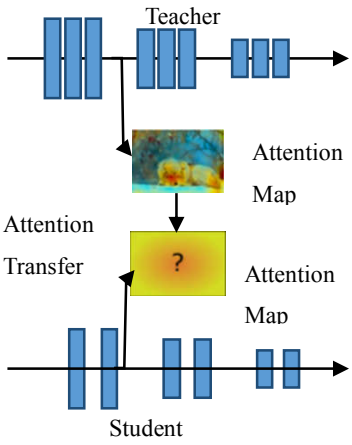


图 11 知识蒸馏(Teacher-Student Network)

在模型压缩方面,主要是对已有的网络进行参数量或者网络规模上的压缩.权重裁剪主要是去掉对网络贡献小的参数,最终的效果依赖于剪枝的策略；权重量化使用更少的比特数来表示权重,能够极大的对网络进行压缩,但是对网络性能影响较大；低秩分解主要是从稀疏矩阵的冗余上对其进行分解,在模型压缩与加速上都取得了一定的成果,但是实际应用中,分解操作需要的计算量较大；知识蒸馏是使用小网络来模拟大网络的性能,但是其应用场景受限于“教师”的应用场景.

2.2 基于自动机器学习的自动神经网络模型压缩

模型压缩技术的核心是决定每一层的压缩策略,传统的模型压缩模型主要包括剪枝、权值共享、低秩分解和知识蒸馏.一般而言,深度神经网络中各层的冗余程度不同,浅层具有较少的冗余,全连接层具有较大的冗余,使用这些方法不仅需要算法设计者具有丰富的专业知识和领域知识,探索大规模的参数空间,权衡模型大小、运算效率和模型性能.同时还需要反复的实验以获得最佳的模型参数,非常的耗时.然而,由于深度神经网络中的各层不是相互孤立的,基于规则的压缩策略往往并不是最优的.深度神经网络需要大量的训练数据学习针对特定问题的模型参数,导致了传统的模型压缩算法不能直接从一个模型迁移到另外一个模型.这些影响因素严重限制了深度神经网络模型压缩算法的发展与应用.

近些年来,自动机器学习算法(AutoML)受到学术界和工业界的广泛关注,可以看作是设计一系列的控制器操作机器学习模型,使得模型可以自动的学习到合适的参数和网络架构而不需人工的干预,减少对专业知识和领域知识的要求,从而降低设计机器学习算法的难度.

随着深度神经网络越来越深,设计空间的复杂性以指数级的速度爆炸式的增加,因此传统的基于规则的模型压缩算法不能满足模型压缩的需求.西安交通大学和谷歌联合提出了自动模型压缩算法 AMC[5],利用强化学习方法学习压缩策略,学习到的压缩策略优于传统的基于规则的压缩策略,具有更高的压缩比,并且能更好的保持模型的性能,大大的减少人为因素的干扰.

AMC[5]算法提出压缩精度对每层的稀疏性非常敏感,需要细粒度的动作空间.与传统的在离散空间上进行搜索不同,AMC[5]算法采用 DDPC[49]代理的连续压缩比控制策略,通过优化损失函数学习模型压缩策略.特别地,DDPC[49]代理以分层的方式处理神经网络,对于每一层 L_t ,代理接受编码该层的有用特征 S_t ,然后输出精确压缩比.第 L_t 层压缩完成后,代理移动到下一层 L_{t+1} ,在没有微调的情况下,对压缩后的神经网络中的所有层的精度进行评估.这种简单的近似可以避免不必要的重新训练,并提供高质量的搜索结果.不同的基于强化学习的轻量级模型方法的比较如表 1 所示.AMC[5]与其他方法 N2N[50],NT[51],NAS[6]的区别在于无需微调,仅需要设计一个简单的控制器,实现用较少的 GPU 时间实现快速压缩,同时也支持连续的动作空间.

表 1. 基于强化学习的轻量级模型方法的比较

	NAS	NT	N2N	AMC

Optimize for accuracy	√	√	√	√
Optimize for latency				√
Simple, non-RNN controller				√
Fast exploration with few GPUs		√	√	√
Continuous action space				√

针对不同的应用场景,AMC 提出了两种压缩策略搜索协议,对时效性要求较高的场景,例如移动应用程序、自动驾驶和广告排名等,通过限制搜索空间,即在模型大小和滤波器的数量等方面进行压缩,以获得最大的硬件资源；对性能要求较高的场景,例如拍照识图和谷歌图像等,通过研究模型的精度和规模设计奖励函数,实现模型的压缩并保持模型的精度.图 12 表示了 AMC 模型压缩算法流程图.

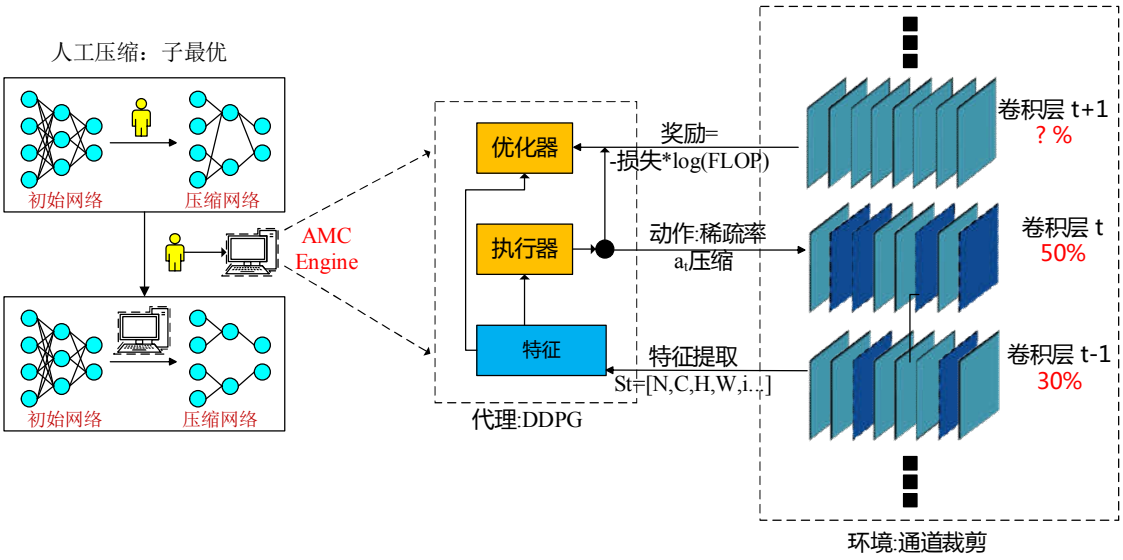


图 12 AMC 模型压缩算法结构图

AMC 模型通过强化学习算法训练代理,实现动作的预测并计算稀疏性,执行压缩.同时快速评估压缩后的模型性能,通过优化损失函数,鼓励更小,更精确的结果更新代理.AMC 在 ResNet-50 上评估了压缩的性能,并测试了压缩模型从分类到目标检测的泛化能力.实验表明,AMC 提供了比手工设计的启发式策略更的性能.对于 ResNet-50,压缩比从 3.4 倍提高到了 5 倍而不损失准确率,如图 13 所示.

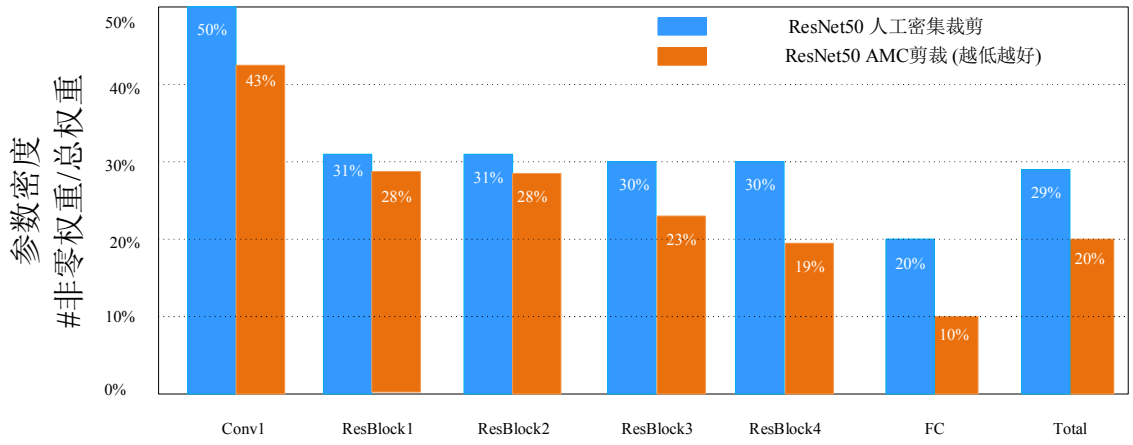


图 13 AMC 与人工设计压缩算法的比较

此外,AMC 将 MobileNet 的 FLOPS 减少了 2 倍,使最高准确率达到了 70.2,在 Titan XP 和安卓手机上,分别实现了 1.53 倍和 1.95 倍的加速,如表 2 所示.

表 2. 基于学习的压缩 VS 基于规则的压缩

	Policy	FLOPs	△Accuracy%
VGG-16	FP(handcraft)[24]	20%	-14.6
	RNP(handcraft)[52]		-3.58
	SPP(handcraft)[53]		-2.3
	CP(handcraft)[30]		-1.7
	AMC(ours)[5]		-1.4
MobileNet	Uniform(0.75-224)[8]	56%	-2.5
	AMC(ours)	50%	-0.4
	Uniform(0.75-192)[8]	41%	-3.7
	AMC(ours)	40%	-1.7
MobileNet-v2	Uniform(0.75-224)[10]	70%	-2.0
	AMC(ours)		-1.0

3 基于神经网络架构搜索的自动化轻量级神经网络设计

人工设计轻量级神经网络需要考虑层间的连接方式、网络的深度、卷积计算方式等大量的影响因素,同时需要根据模型的性能不断的调整网络的结构.不仅需要设计者具有丰富的专业知识和领域知识,同时非常依赖设计者的经验,需要耗费大量的训练时间和计算资源评估神经网络的性能.因此人工设计轻量级神经网络需要大量的时间、人力和物力,这些影响因素严重限制了轻量级神经网络在便携式设备上的应用与发展.

为了减轻对设计者的要求,神经网络架构搜索(Neural Architecture Search, NAS)是指根据某种搜索策略,在特定的搜索空间内,自动设计出解决特定任务的高性能神经网络架构.到目前为止,NAS 在图像分类[54]、语义分割[55]等视觉任务上的性能已经超过了手工设计的神经网络架构.NAS 由三部分构成,分别是搜索空间,搜索策略和性能评估策略.其中搜索空间定义了构建神经网络的基本架构单元,将适合特定任务的典型网络架构作为先验知识,可以有效的减少搜索空间的大小.但是,该过程往往会引入人为因素的干扰,可能会影响不同类型的神经网络架构的搜索.搜索策略决定了如何在通常为指数级甚至是无限大的搜索空间内搜索用于解决特定任务的神经网络架构,并决定网络中不同层/模块的连接方式和参数等.一方面需要快速找到性能良好的神经网络架构,另一方面需要避免过早的收敛到次优架构.性能评估策略是指准确、高效的度量神经网络的性能.传统的性能评价方法通过在针对特定任务构建的训练数据集上,进行完整训练和测试度量神经网络的性能.这种方法的计算成本和时间复杂度高,对计算资源的需求大.因此,最近的主要研究方向集中在如何设计高效率的性能评估策略.图 14 表示 NAS 算法的流程.搜索策略从预定义的搜索空间 Θ 内选择神经网络架构 A,通过性能评估策略度量神经网络架构的性能,作为奖励反馈回搜索策略,搜索策略通过接受到的奖励调整神经网络的架构,通过反复的迭代,最终得到最优的神经网络架构.

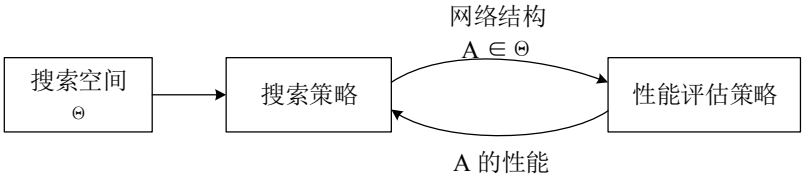


图 14 神经网络搜索(NAS)的流程图

3.1 搜索空间

搜索空间定义了 NAS 中构建神经网络的基本架构单元,是神经网络模型通用定义的一个子空间.可以根据特定任务的先验知识引入各种约束,限制基本架构单元的类型和搜索空间的大小.搜索空间根据基本架构的类型分为两种主要类型:第一种是直接搜索整个神经网络架构的全局搜索空间,包括链式架构搜索空间和多分支架构搜索空间;第二种是通过重复某些特定结构构建神经网络架构的局部搜索空间,代表性方法是基

于块的搜索空间.

3.1.1 全局搜索空间

图 15 表示了链式和多分支架构搜索空间,图中不同节点表示神经网络中不同类型的层,采用不同的颜色表示,图中的有向边指示了输入和输出的关系.其中最简单的类型是链式架构搜索空间,如图 11(a)所示.链式架构搜索空间通过连续的连接 n 层构建神经网络,其中第 L_i 层以第 L_{i-1} 层的输出作为输入,并且第 L_i 的输出作为第 L_{i+1} 层的输入.链式架构搜索空间的参数包括:(1)神经网络的(最大)层数 n ;(2)神经网络中每层的类型,包括池化、卷积或其他高级操作,例如深度可分离卷积[56]、空洞卷积[57]等;(3)神经网络的超参数,包括卷积核的数量、核大小、卷积步长[58-60]和全连接层的数量[61]等.需要注意的是,不同的参数之间存在关联关系,例如神经网络的超参数受到卷积核的数量和核大小的约束,因此搜索空间的参数不是固定长度.

Baker 等人[58]的工作研究了链式架构搜索空间,确定了一组可被搜索的操作,包括卷积、池化和线性变化等,以及设置了不同的超参数,包括滤波器的数量、卷积核的大小、卷积步长等.进一步的,该方法同时考虑了一些额外的限制,排除一部分显著不正确的神经网络架构,例如神经网络的第一层为池化层的网络架构,将具有高分辨率的特征作为全连接层的输入等.Zoph 等人[62]提出了松弛链式架构搜索空间,允许链式架构间的任意按序节点存在残差连接,可以实现搜索得到更加广泛、更加不同的神经网络架构.

根据人工设计深度卷积神经网络的经验和教训,链式架构搜索空间具有结构简单,计算复杂度低的特点.但是在训练链式神经网络的过程中,容易发生梯度消失的问题,并且神经网络模型的规模小,学习能力差.为解决上述问题,最近提出的 NAS 方法受到了近些年来典型人工设计的深度卷积神经网络模型(ResNet[2],Inception[63])的等启发,建立复杂的、具有多分支架构的搜索空间,如图 15(b)所示.

多分支架构搜索空间中第 L_i 层通过联合第 L_{i-1} 层的全部输出作为输入 $g_i(L_0,...,L_{i-1})$,根据神经网络的结构分类,主要存在三种多分支连接方式:(1)链式连接方式, $g_i(L_{i-1},...,L_0)=L_{i-1}$,即第 L_i 层以第 L_{i-1} 层的输出作为输入,并且第 L_i 的输出作为第 L_{i+1} 层的输入.因此,链式架构搜索空间可以被解释为是多分支架构搜索空间的特例;(2)残差连接方式[2],对前一层的所有输出求和,即 $g_i(L_{i-1},...,L_0)=L_{i-1}+L_k, k<i-1$;(3)密集连接方式[3],串联前一层的所有输出,即 $g_i(L_{i-1},...,L_0)=concat(L_{i-1},...,L_0)$.多分支架构搜索空间可以更好的探索神经网络中层之间不同的连接方式,丰富了神经网络结构的类型,增加神经网络模型的容量,提高神经网络的学习能力,提升解决特定任务的性能.

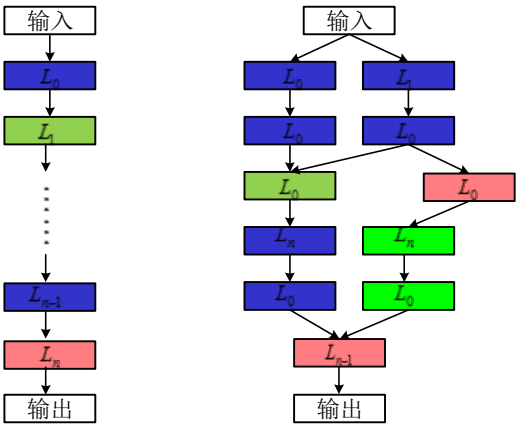


图 15 链式和多分支架构搜索空间

3.1.2 局部搜索空间

虽然链式和多分支架构搜索空间可以以更灵活的方式构建不同类型的神经网络结构,但是面临着搜索空间大,参数规模大,需要大量的计算资源实现神经网络的搜索.因此,往往收敛至次优或局部极小的神经网络架构,不能有效的解决特定任务.为了解决上述问题,学者们受到人工设计的深度神经网络模型中存在大量重复的块结构的启发,提出了重复堆叠块结构,而不是通过搜索单一层构建神经网络.这种块结构通常由更小的块构成,通过堆叠块构建更大的神经网络架构.这种设计不仅可以保证神经网络架构的性能,而且通过简单修

改神经网络的参数,可以很容易的将搜索到的神经网络推广到其他的数据集和任务中.

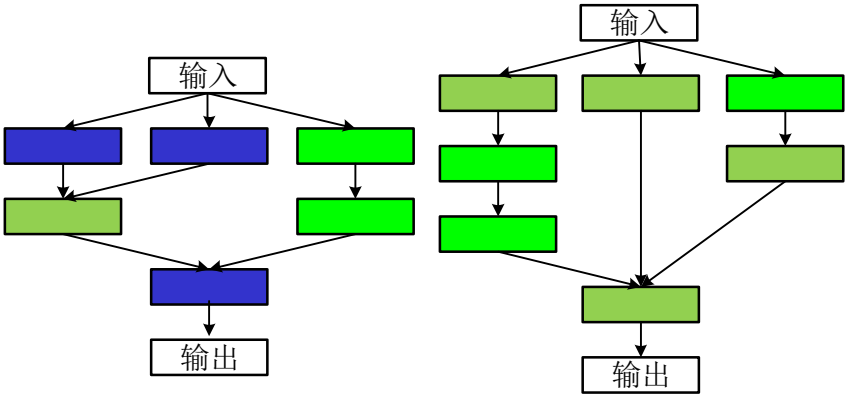


图 16 不同类型的块结构

通过引入针对特定问题的先验知识,设计具有不同特定架构的块,构成基于块的搜索空间,而不仅仅包括不同类型的单一层.近些年来,学者们成功设计了一些基于块的搜索空间[54,64-69],NasNet 模型[6]是第一个提出了基于块的搜索空间,该方法设计了两种类型的块结构,分别是正常块(normal cell)和降维块(Reduction cell),如图 16 所示.正常块内的卷积步长设置为 1,可以保持输入特征的维度不变,降维块内的卷积步长设置为 2,降低输入特征的空间维度.同时,Cai 等人[67]将如 DenseNet[3]等经典手工设计的深度神经网络模型直接应用在基于块的搜索空间中；Dong 等人[70]通过设置搜索没有分支结构并交替使用具有固定结构的块,提出了满足模型参数少、推理时间快的快速搜索模型,如图 17 所示.原则上,不同的块之间可以采用任意的连接方式,多分支架构搜索空间内的所有连接方式均可以使用.在理想情况下,块结构和整体的神经网络架构应进行联合优化,而不是单独优化某一部分.

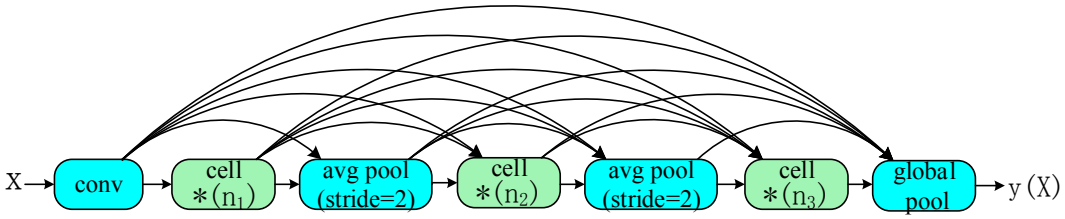


图 17 密集连接的移动搜索空间

3.1.3 全局搜索空间和局部搜索空间的比较

搜索空间的类型和大小决定了网络架构搜索的难度.需要注意的是,不管采用哪种类型的搜索空间,都具有不连续性和相对高维的问题.与链式和多分支等的全局搜索空间相比,基于块局部搜索空间具有以下三个优点:(1)显著降低了搜索空间的规模.Zoph 等人提出的 NasNet[6]方法的效率比 NASRL 方法[62]快了 7 倍；(2)通过堆叠块结构创建神经网络架构已经被证明是一个非常有效的设计原则,例如在递归神经网络中重复 LSTM 模块或 ResNet[2]网络中堆叠残差结构；(3)通过简单的改变块结构中卷积核的数量和大小,采用块结构创建的神经网络架构更容易的迁移到其他任务或数据集中,最后通过某种搜索策略不断的堆叠不同的块结构构建最终的网络架构,如图 18 所示.

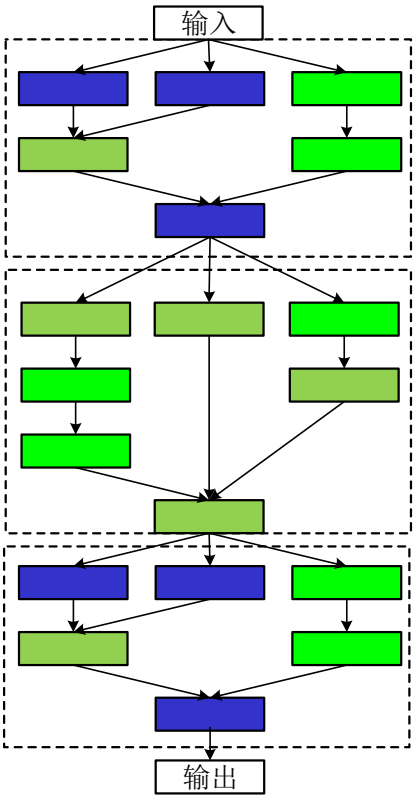


图 18 基于块结构的神经网络架构

到目前为止,学者们提出的基于块的搜索空间的方法,特别是 NasNet 搜索空间,是其他后续神经网络架构搜索方法的基础.大多数研究搜索空间的工作都支持局部搜索空间方法,因为引入了关于特定任务的先验知识,实验结果[65]同时表明了基于块的搜索空间可以获得更好的结果.同时,基于块的搜索空间通过简单改变块的结构,即改变滤波器的数量和卷积核的大小等,使得神经网络架构非常容易的泛化到其他的数据集或任务中.一般来说,全局搜索空间没有上述优点,但是,在特定的应用环境下,是非常有用的.例如,当改变输入数据的结构和类型时,仅仅改变块的结构是不可行.Tan 等人[7]在搜索移动架构时采用全局搜索策略,该方法认为层的多样性对于移动设备中实现高精度和低延迟都是至关重要的,并且这些不能由基于块的搜索空间提供的.

3.2 搜索策略

当完成构建搜索空间后,搜索策略决定了如何在搜索空间内搜索针对特定任务的基本架构单元,并确定神经网络内部的连接方式.一方面需要快速找到性能良好的神经网络架构,另一方面需要避免过早的收敛到次优的神经网络架构.首先我们给出搜索策略的定义, D 表示训练数据, M 表示构建的神经网络模型, S 表示搜索空间,则搜索策略过程被形式化为 Λ :

$$\Lambda : D \times S \to M$$

训练搜索策略不仅仅需要确定神经网络架构的拓扑结构,同时需要预测优化器、正则约束和其他超参数等.给定训练数据 $d \in D$,被划分为训练集 d_{train} 和验证集 d_{valid} ,搜索策略的目的是找到最优的网络架构,使得在验证集 d_{valid} 上的训练误差最小,即

$$\Lambda(A,d) = \arg\min L(m_\theta, d_{valid}) + R(\theta)$$

其中 L 表示损失函数,用于度量神经网络架构的性能(3.3 节介绍), R 表示正则约束.优化损失函数 L 是全局优化问题,主流的优化策略包括基于强化学习的优化策略和基于进化算法的优化策略.

3.2.1 基于强化学习的优化策略

强化学习通过代理与环境进行交互,其目标是获得最大化的未来奖励,常用于顺序决策任务中.在每次迭代过程中,一方面代理执行动作,观察环境的状态变化并接收奖励.另一方面,环境根据代理发出的动作,转换

到下一个状态,并且根据新的状态产生奖励,其过程如图 19 所示.强化学习通常被定义为一个四元组 (S,A,ρ,f),其中:

(1) S 表示所有环境状态的集合. $s_t \in S$ 表示代理在 t 时刻所处的状态;

(2) A 为代理可执行动作的集合. $a_t \in A$ 表示代理在 t 时刻所采取的动作;

(3) $\rho: S \times A \to R$ 为反馈函数. $r_i \sim \rho(s_t,a_t)$ 表示代理在状态 s_t 执行动作 a_t 获得的实时反馈.

(4) $f: S \times A \times S \to [0,1]$ 为状态转移概率分布函数. $s_{t+1} \sim f(s_t,a_t)$ 表示代理在状态 s_t 执行动作 a_t 转移到下一状态 s_{t+1} 的概率.

强化学习算法非常适用于解决 NAS 问题,其中代理(搜索策略)决定了系统的状态,即在搜索空间内自动构建神经网络模型;环境(性能评价策略)评估神经网络架构的性能,并产生相应的奖励;代理接收到奖励并根据环境当前的状态,预测下一个动作(即搜索).通过反复迭代,最终得到对解决特定任务性能最高的神经网络模型.

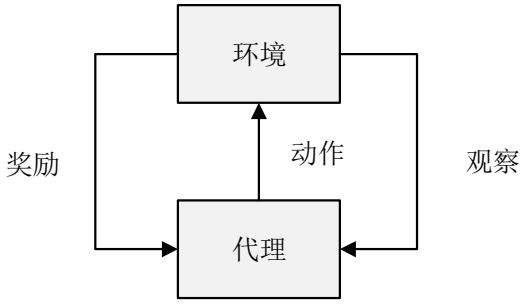


图 19 强化学习框架

Q 学习优化算法,最早的在线强化学习算法,同时也是强化学习最重要的算法之一.该算法的主要思路是通过定义 Q 函数,将在线观测到的数据代入到下面的更新公式中对 Q 函数进行迭代学习,得到精确解:

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha_t \delta_t$$

$$\delta_t = r_{t+1} + \gamma \max Q_t(s_{t+1},a_k) - Q_t(s_t,a_t)$$

其中 t 是当前时刻, α_t 是学习率, δ_t 表示时间差分误差, a_k 是状态 s_{t+1} 能够执行的动作, r 是平衡因子. Q 学习使用一个合理的策略使代理预测下一时刻的动作,根据该动作与环境的交互所得到的下一个状态和反馈来学习得到另一个最优的 Q 函数.由于 Q 学习很重视通过与环境的交互改变代理的动作,因此在最优控制和游戏上有许多的应用.

Baker 等人[58]第一次提出了基于强化学习的神经网络架构搜索策略,结合了 Q 学习优化, ε – 贪婪策略和经验回放方法.该方法将代理的动作定义为选择不同类型的层添加到神经网络架构中,同时决定是否终止构建新的神经网络架构;状态是构建的神经网络架构,不同时刻产生的模型被分别被训练以评估其解决特定任务的性能; Q 函数通过经验回放技术进行适当的更新.为了权衡探索与利用,该方法采用 ε – 贪婪策略,以概率 ε 在搜索空间内随机采样,但是不能保证搜索的架构是最优的.与该方法不同的是,Zhang 等人[71]在基于块的搜索空间上提出了基于 Q 学习算法的搜索网络架构.

策略梯度优化算法是一种通过不断逼近最优值,最终得到最优策略的优化方法.该方法又可以分为确定策略梯度算法和随机策略梯度算法.在确定策略梯度算法中,动作以概率 1 被执行.在随机策略梯度算法中,动作以某一概率被执行.近些年来,确定策略梯度算法逐渐受到了学者们的关注.假设需要逼近的策略是 $\pi(s,a;\theta)$,而且该策略对参数 θ 可导,则可定义目标函数和值函数:

$$J(\pi_\theta) = E\sum \gamma_{t-1} r_t \mid s_0, \pi_0$$

$$Q_{\pi_\theta}(s,a) = E\sum_{k=1}^{\infty} \gamma_{t-1} r_{t+k} \mid s_t = s, a_t = a, \pi_\theta$$

其中 J 表示在策略 J 下的决策, E 表示期望.

Zoph 等人[62]第一次提出了基于策略梯度算法实现神经网络架构搜索,通过直接控制模拟器实现神经网络架构的预测.由 θ 参数化的自动回归控制器定义了随机策略 $\pi_\theta(\alpha|s)$,该控制器根据先前的动作预测后续的动作,并使用递归神经网络(Recurrent Neural Network, RNN)对其进行建模,如图 20 所示.

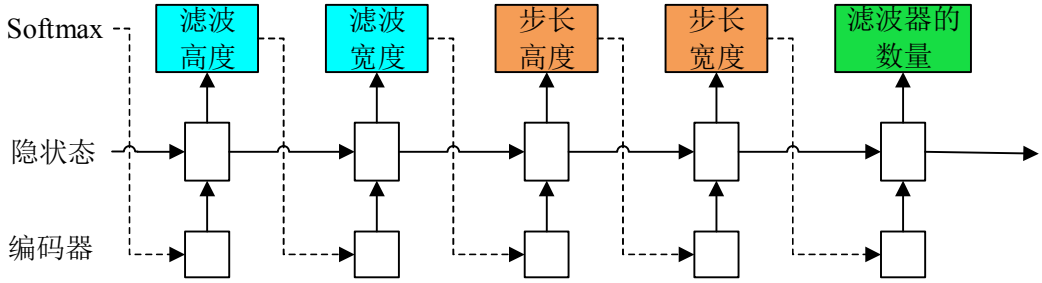


图 20 自动回归控制结构

在每次迭代中,每个动作以服从某一概率分布在搜索空间中采样,按序连接并传输到下一次迭代中.通过性能评估策略(第 3.3 节中介绍)度量预测神经网络架构的性能,并根据强化更新规则更新控制器的参数.

$$E_\pi\left[\sum_{t=0}^{T}\nabla_\theta \ln \pi_\theta(\alpha_t|s_t)G_t\right]$$

学者们同样研究了基于块的搜索空间下的策略梯度控制器.NasNet [6]定义了递归网络控制器,该控制器顺序的确定搜索空间内预定义数量的块结构的输入和动作.Cai 等人[60]提出了不同动作空间的控制器,并采用强化学习算法训练.该方法需要一个初始框架,通过 Chen 等人[72]提出的方法加宽或加深神经网络,虽然神经网络架构发生了变化,但是它学习到的功能仍然保持不变.

3.2.2 基于进化算法的优化策略

进化算法(Evolutionary Algorithms, EA)是基于种群的全局优化算法,包括初始化、父选择[73]、重组和突变、种群选择等基本组件.初始化定义了如何生成初始种群,初始化后,优化器通过反复迭代以下步骤直到种群收敛.图 21 表示了进化算法的基本流程.

1. 从种群中选择父节点进行迭代
2. 应用重组和变异操作来创建新的个体
3. 评估新个体的适应度
4. 选择合适的个体

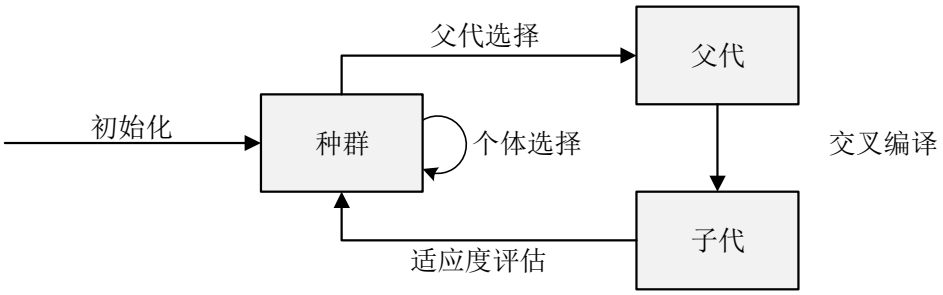


图 21 进化算法框架

在进化算法中,变异、重组和父节点的选择决定了整个搜索过程,通过变异和重组权衡种群的多样性和相似性,适应性函数反映了优化目标,并且个体选择保证了种群中不同个体间的竞争.针对 NAS 问题,种群由一组基本网络架构单元构成,通过在种群内选择一个基本架构单元作为父节点用于变异和重组,即产生新的神经网络架构.一般而言,在 NAS 中,只使用变异算子,大量的实验表示两个高度适应的个体通过重组操作不能得到相似或更好的子代.

NasNet[6]第一个提出了使用进化算法搜索深度神经网络架构进行图像分类.该方法从 1000 份最简单的神经网络架构的副本开始,即一个全局池化层后面连接输出层.在父节点选择的步骤中,提出了对成对的神经网络架构进行采样以便进一步处理.当两个神经网络架构中较好的一个连同权重被复制和变异,以批量大小为 50 的规格训练了 25600 步,并添加到种群中,而另一个神经网络架构被从种群中删除.这相当于使用 $k=2$

和 $p=1$ 的锦标赛选择算法.这组突变包括一系列简单的操作,例如在全局池化层之前的任意位置添加卷积层,以及卷积核的大小、通道数、卷积步长、学习速率、添加或删除残差连接、删除卷积、重置网络权重以及对应于无变化的突变.由于不再强制执行域约束,因此可以使用此方法对具有冗余组件的神经网络架构进行隐式的采样,如图 22 所示.

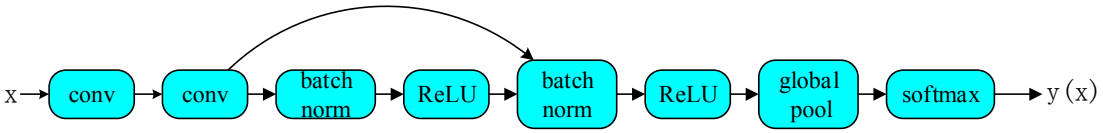


图 22 可能存在的网络架构

与 NasNet[6]方法不同,MnasNet[7]在更结构化的搜索空间上设计了一种基于遗传算法搜索策略,该搜索空间包括具有可能并行卷积操作的一系列片段.该方法认为网络架构由三个片段组成,每个片段由多个卷积层组成,并且每个片段由邻接矩阵描述,该邻接矩阵定义为片段内的每层上的有向无环图.该搜索策略从搜索空间中的 20 个随机基本架构单元开始.在父节点选择步骤,所有样本对 a_i, a_{i+1} ,其中 $i \bmod 2 = 1$,在交叉重组操作中被考虑.交叉重组操作以概率 p 在两个所选的神经网络架构之间交换片段.然后,所有在前一步中未经过修改的个体都考虑进行突变操作.MnasNet[7]算法中,突变是在定义片段的邻接矩阵上的随机翻转操作.最后,从头开始训练获得的后代并在计算其适应度.个体的适应度被定义为其在测试数据上的准确性与种群内所有个体中最低准确性之间的差异.由于 MnasNet[7]中通过设定阈值,根据适应度选择个体,确保性能最差的网络架构以零概率存活.值得一提的是,这是首次证明在较小数据集上自动发现的神经网络架构成功转移到较大数据集的工作之一,即从 CIFAR-10 到 ImageNet [18].

Suganuma 等人[59]提出了另一个基于遗传算法的优化器,与上述方法不同的是,考虑在其搜索空间中定义更广泛的操作集,包括卷积和池化以及向量的求和等.该方法将整个神经网络架构编码为由三元组表示的块序列,该三元组定义了块的操作和输入.通过连接块编码获得的串构成了基因型.在这种基因型定义中可以接受非活动部分,例如某些特定操作或断开的块的未使用输入.不活跃的部分不在表型中具体化.图 23 表示了遗传编码,其中基因型的灰色阴影部分表示无效.节点 3 中的非活动部分不存在于表现型中,由节点 4 引入的额外最大池化层是基因型中未明确编码的操作如何出现在表现型中的示例.该方法提出 $(1+\lambda)-$ 进化策略[74]方法来指导进化,其中突变和选择算子在循环中使用直到终止进化.从对应于父节点的基因开始(首次迭代是随机的),通过强制修改基因型中的活性部分来生成后代.突变也适用于其基因型的一个非活性部分的父节点.然而,这不会导致表现型的任何改变,因此不会增加整体的训练成本,即使在这一代的父节点将再次成为下一代的父节点的情况下,确保了整个搜索的进程.训练下一代网络架构,度量其准确度,并使用精英选择策略选择下一次迭代的父节点.

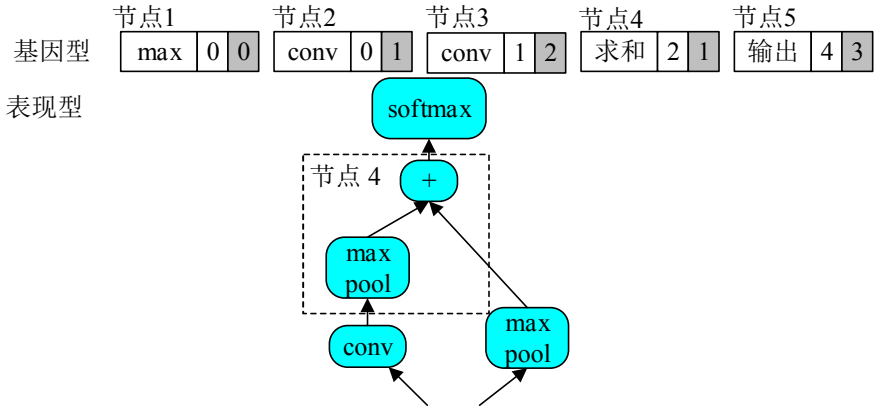


图 23 遗传编码

Liu 等人[75]提出了另一种基于进化算法的搜索策略,设计了分层的搜索空间.在每次迭代中,突变操作都

会选择要修改的层次以及对该层次的修改.该种群由 200 个小的基因型初始化,这些基因型通过应用 1000 个随机突变增加种群的多样性.此外,对父代中的个体采用利用锦标赛算法(5%的种群数量)进行选择,并且在进化过程中不会移除任何个体.该方法中,突变可以执行添加,改变和删除基因型等操作.

Real 等人[54]的后续工作是在使用进化算法进行神经网络架构搜索方面最重要的工作之一.该方法主要用于发现 AmoebaNet-B 和 AmoebaNet-C 神经网络架构,它在具有不同规模大小的 CIFAR-10 和 ImageNet 数据集上的图像分类任务创造了新的记录,超过了人工设计的深度神经网络模型的性能.然而,在该方法的搜索过程共使用了 3150 个 GPU 小时,并且可以在 NasNet 搜索空间上执行,并使用锦标赛选择算法在每次迭代中选择父代个体.所选择的个体在正常块或降维块的操作或连接中随机改变而发生突变,并训练 25 个周期.与之前的工作相反,该算法不仅仅依赖于性能评估策略的准确性,而且还包括了个体存在的时间.这有助于抑制重复选择表现良好的突变模型并为种群引入多样性,其等价于为目标函数添加了一个正则化约束,确保搜索到的神经网络架构不仅能够一次性能评估具有高精度,而且每次评估都具有高性能.

之前讨论的基于进化算法的搜索策略主要关注点的是找到一个忽略 GPU 预算限制的良好性能架构.这些方法中使用的最短搜索时间仍然需要 17 个 GPU 天.接下来讨论的两项工作更关注搜索效率,将搜索时间缩短到 1 个 GPU 天以内,并提供了可比较的结果.实现这一目标的关键是不同突变的组合,这些突变是功能保持转换和更具侵略性的贪婪策略.

Elsken 等人[76]提出了一种简单而有效的进化算法实现搜索,其灵感来自于功能保持变换.该方法采用精英选择策略来选择父代,用功能保持突变产生八个不同的后代,并且每个后代训练 17 个周期.初始的神经网络架构包括三个卷积层和两个最大池化层.如前所述功能保持转换操作显著的减少了每个神经网络架构的训练时间,从而缩短了整个搜索过程的持续时间.

在类似的工作中,Wistuba 等人[77]同样利用了功能保持突变策略,但使用了基于块的搜索空间.该方法提出的搜索策略从一个简单的神经网络架构开始,并执行功能保持突变以增加种群的数量和多样性,即通过从基础的神经网络生成 15 个额外的子代使得初始种群变得多样化.在执行进化算法的过程中,采用锦标赛选择算法选择父代的个体,并在对后代进行适应性评价前,对其进行 15 个周期的训练.与 Liu 等人[75]的方法类似,所有的个体都在进化过程中存活以确保种群的多样性.如图 24 中提供了一个搜索过程的示例,可以发现具有任意神经网络结构的块.

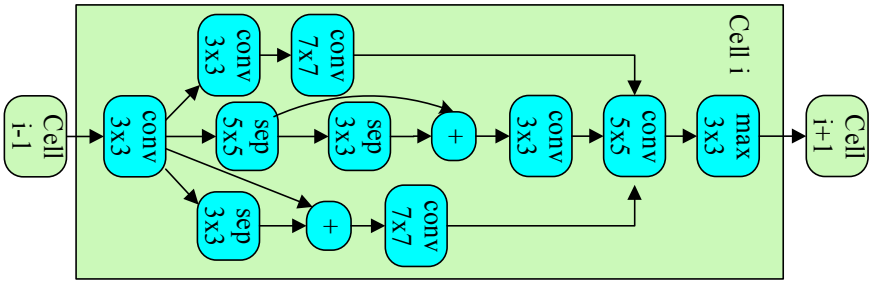


图 24 基于进化算法实现神经网络架构搜索

3.2.3 不同搜索策略的比较

表 3 总结了不同的搜索策略在 CIFAR-10 基准数据集上分类的结果.然而,不同的方法在搜索空间、搜索时间和数据扩充等方面有着很大的差异,缺乏统一的可比较的基准.虽然 NAS 可以被视为优化超参数的一种特殊方法,但大多数相关工作都被忽略了.NasNet 搜索空间是最常用的,这是因为引入的先验知识有利用发现性能良好的神经网络架构.类似地,参数重用策略、功能保持策略在提高搜索效率方面是有效的.

表 3. 不同搜索策略在 CIFAR-10 数据集上的分类结果和搜索时间比较

Reference	Error (%)	Params(Millions)	GPU Days
Baker et al. (2017)[58]	6.92	11.18	100
Zoph and Le(2017)[62]	3.65	37.4	22400

Cai et al.(2018)[60]	4.23	23.4	10
Zoph et al.(2018)[6]	3.41	3.3	2000
Zoph et al.(2018)+Cutout	2.65	3.3	2000
Zhang et al.(2018)[71]	3.54	39.8	96
Cai et al.(2018)[67]	2.99	5.7	200
Cai et al.(2018)+Cutout	2.49	5.7	200
Real et al.(2017)[78]	5.40	5.4	2600
Xie and Yuille (2017)[79]	5.39	N/A	17
Suganuma et al(2017)[59]	5.98	1.7	14.9
Liu et al.(2018)[75]	3.75	15.7	300
Real et al.(2019)[54]	3.34	3.2	3150
Elsken et al.(2018)[76]	5.2	19.7	1
Wistuba(2018)+Cutout[77]	3.57	5.8	0.5
Kandasamy et al.(2018)[80]	8.69	N/A	1.7
Luo et al.(2018)[81]	3.18	10.6	200
Pham et al.(2018)[65]	3.54	4.6	0.5
Pham et al.(2018)+Cutout	2.89	4.6	0.5
Bender et al.(2018)[82]	4.00	5.0	N/A
Brock et al.(2018)[83]	4.03	16.0	3
Zhang et al.(2019)[84]	4.30	5.1	0.4
Random(Luo et al. 2018)[81]	3.92	3.9	0.3

3.2.4 轻量级神经网络搜索

MnasNet 算法[7]提出了一种用于移动神经网络模型设计的自动神经结构搜索方法,其中与以前方法的主要区别是延迟感知多目标奖励和新的搜索空间.MnasNet[7]算法基于两个主要思想:(1)将搜索问题表示为一个多目标优化问题,同时考虑神经网络模型的准确性和推理延迟；(2)与以前使用 FLOPS 来近似推断延迟的工作[62,68,85]不同,直接通过在实际移动设备上执行模型来测量实际的延迟.该想法的灵感来源于搜索的失败往往是由一个不准确的代理导致的:例如,MobileNet[8]和 NASNet[6]有类似的失败现象(575M vs 564M),但是他们的延迟明显不同(113ms 与 183ms).其次,观察到以前的自动化方法主要是寻找一些类型的块,然后重复相同的块结构构建神经网络.这简化了搜索过程,但也排除了对计算效率很重要的层多样性.为了解决这个问题,该方法提出了一种新的分解层次搜索空间,它允许层在架构上不同,但仍然在灵活性和搜索空间大小之间取得适当的平衡.

本文设计了一种新的分解层次搜索空间,它将 CNN 模型分解为独特的块,然后分别搜索每个块的操作和连接,从而允许在不同块中使用不同的层结构.图 25 表示了分解层次搜索空间,根据神经网络中每层的输入特征分辨率和滤波器的大小,神经网络中的层被分组成许多预定义的架构,称为块.每个块包含数量可变的重复相同的层,如果输入/输出的分辨率不同,但所有其他层都具有跨距 1,则只有第一层具有跨距 2.对于每个块,该方法搜索单个层的操作和连接以及层数 n ,然后同一层重复 n 次(例如,层 4-1 到 $4-N_4$ 是相同的).不同块(如 2-1 层和 4-1 层)的层可以不同.

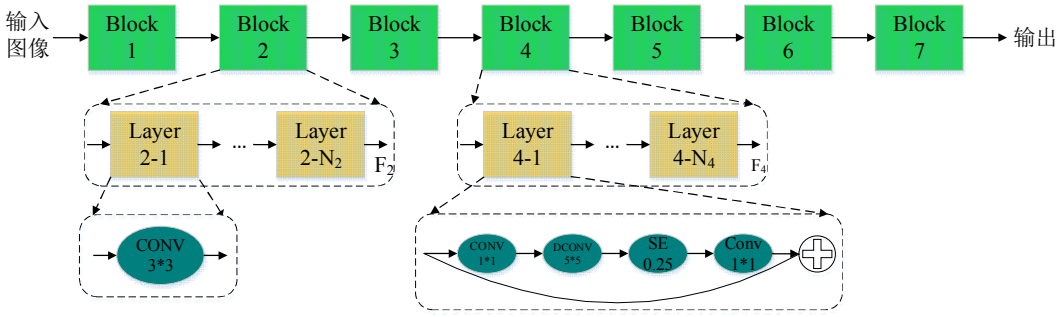


图 25 分解层次搜索空间

该方法使用强化学习方法为多目标搜索问题找到 Pareto 最优解.将搜索空间中的每个神经网络模型映射到令牌列表.这些令牌是由参数为 θ 的代理生成的一串动作 $a_{1:t}$ 决定的.目标是使预期的奖励最大化.

$$J = E_{p(a_{1:T};\theta)}[R(m)]$$

其中由动作 $a_{1:t}$ 采样得到的, $R(m)$ 是定义的多目标函数.

图 26 表示了 MnasNet 算法的整体流程,搜索策略由三个部分组成:基于递归神经网络(RNN)的控制器、获得模型精度的训练器和用于测量延迟的基于移动电话的推理引擎.按照众所周知的样本-评估-更新策略来训练控制器.在每个步骤中,控制器首先使用其当前参数对一批模型进行采样,方法是根据其 RNN 模型中的 SoftMax 预测令牌的序列.训练每一个样本模型,并获得在特定任务上的准确度 $ACC(m)$,并在真实手机上运行它来获得它的推理延迟 $LAT(m)$.然后计算奖励值 $R(m)$.在每一步结束时,通过使用近端策略优化使预期奖励最大化来更新控制器的参数.不断迭代样本-评估-更新策略,达到最大迭代步数或参数收敛为止.

MnasNet 模型应用于图像分类[86]和 COCO 对象检测[87].与 MobileNetV2[10]相比,MnasNet[7]模型在谷歌像素手机上以相同的延迟将 ImageNet 的精度提高了 3.0%.另一方面,如果限制目标精度,那么 MnasNet 模型比 MobileNetV2 快 1.8 倍,比 NASNet[6]快 2.3 倍,并具有更好的精度.与广泛使用的 ResNet-50[2]相比,MnasNet 模型在减少 4.8 倍的参数和 10 倍乘法加法操作的情况下,精度略有提升(76.7%).

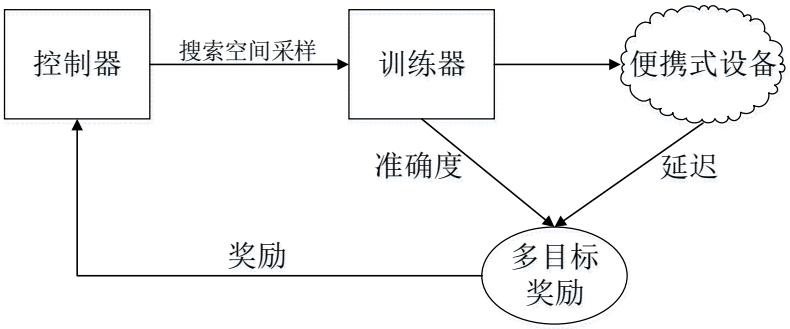


图 26 MnasNet 算法的整体流程

3.3 性能评估策略

通过学习到的搜索策略找到解决特定任务的神经网络架构 M ,为了更好的学习搜索策略,需要评估神经网络架构 M 的性能,最简单的方法是在原始训练数据集上训练神经网络架构 M ,并在验证集上测试其解决特定任务的性能.但是,需要对每个神经网络架构 M 从头开始训练,然后进行测试,这个过程非常耗时,并且需要大量的 GPU 计算资源.因此,设计高效合理的性能评估策略是非常重要的.近些年,如何高效、准确的度量神经网络模型的性能是研究的热点.

影响神经网络训练效率的因素非常多,包括训练数据集的大小、图像的分辨率、训练迭代次数等.为了加快搜索过程,评估神经网络架构性能的主要挑战在于采用较少的计算资源,高效准确的估计神经网络架构的性能.最近,一些工作通过减少训练次数[6,88]、采用训练数据的子集[89]、低分辨率图像[90]、更少的滤波器和更少的块结构[6,54]等方式,以低保真度的方式近似神经网络的真实性能.虽然低保真评估方法降低了计算成本,同时也会在估计中引入偏差,神经网络的性能通常被低估.但是当搜索策略只依赖不同构架间性能的排序,当

性能排序相对稳定时,引入的误差对学习搜索策略没有任何的影响.

但是最近的研究结果表明,低保真评估的性能近似与真实的神经网络性能之间的差异太大时,不同神经网络架构间性能的相对排序可能会发生显著的变化[88].为了解决该问题,学者们提出在学习曲线的基础上,推断不同神经网络架构的性能.Domhan 等人[90]提出从初始学习曲线出发,终止预测表现不佳的学习曲线以加速搜索过程.另外,一些方法同时考虑架构超参数预测最优的部分学习曲线[91-94].加速性能评估的另一种方法是基于已有的神经网络架构初始化新的神经网络架构.Wei 等人[94]提出了网络态射的方法,在修改神经网络架构的同时保持神经网络学习到的功能.通过该方法可以持续不断的增加神经网络架构的容量并保持神经网络的性能,而无需重新训练[96].

一次架构搜索(one-shot architecture search)将所有可能的神经网络架构视为超图的不同子图,并在具有边连接的架构之间共享权重.图 27 表示一次架构搜索框架,由一个输入节点(节点 0),三个隐藏结合(节点 1,2,3)和一个输出节点(节点 4)构成的简单神经网络.一次架构搜索模型(左图)包含对每个节点的所有可能的候选操作,即 3×3 卷积、 5×5 卷积和最大池化等操作,每个可能的神经网络架构是一次架构搜索模型的子图(右图),并共享权重.ENAS[65]学习递归神经网络控制器,从搜索空间中采样,并通过强化学习近似梯度实现一次架构搜索框架的训练.DARTS[68]通过联合优化一次架构模型的全部参数和搜索空间的连续松弛变量,实现异构架构搜索算法的训练.

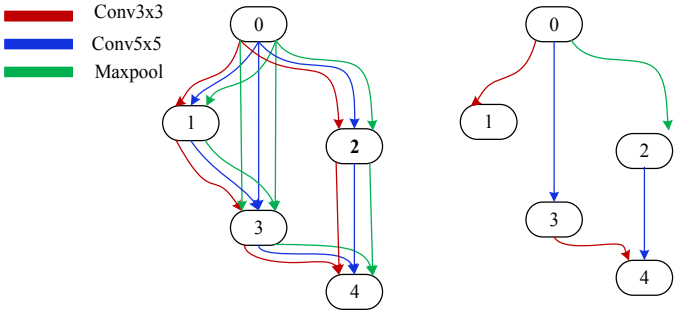


图 27 一次架构搜索框架

不同性能评估策略通过不同的方法实现高效、准确的度量神经网络架构的性能,低保真度近似方法通过减少训练时间、采用训练子集等方式提高效率,保持神经网络性能的相对排序；学习曲线推理方法根据训练过程中的学习曲线推理神经网络架构的性能,提前终止性能差的搜索；网络态射方法通过从已有的神经网络架构初始化新的神经网络架构,可以有效的保持神经网络架构的性能,显著的减少了训练所需的时间；一次架构搜索模型仅仅需要一次训练,不同的神经网络架构之间通过共享权重加快性能评估的效率.表 4 表示了不同性能评估策略的比较.

表 4. NAS 性能评估策略的不同方法的比较

加速策略	加速方法	参考文献
低保真评估	通过减少训练次数、训练数据集的规模、缩小模型的参数量等方式降低训练时间,实现加速.	Li et al. (2017)[95] Zoph et al. (2018)[6] Runge et al. (2019)[96]
学习曲线推断	通过少量的训练次数的学习曲线推断模型的性能,实现加速.	Swersky et al. (2014)[91] Domhan et al. (2015)[90]
网络态射	通过继承已有模型的权重训练网络参数,避免从头训练,加速训练.	Real et al. (2017)[78] Elsken et al. (2019)[66] Cai et al. (2018)[60]
一次架构搜索	只需训练一次模型.该模型的所有子图共享权重,实现加速.	Pham et al. (2018)[65] Bender et al. (2018)[82]

		Liu et al. (2019)[68] Xie et al. (2019)[97]

4 总结与展望

目前,存在三种构建轻量级神经网络的主流方法,分别是人工设计轻量级神经网络、神经网络模型压缩算法和基于神经网络架构搜索的自动化神经网络架构设计.人工设计的轻量级神经网络已经取得了显著的成果,主要有减少卷积核的数量,减少特征的通道数以及设计更高效的卷积操作等关键技术.但是非常依赖设计者的经验,如何有效的将针对特定问题的先验知识加入到模型构建过程中是未来研究的重点方向.通过网络剪枝、权重压缩和低秩分解是对已有的网络进行压缩,但是压缩算法需要设计者探索较大的设计空间以及在模型大小、速度和准确率之间权衡.为了减少人为因素的干扰,自动机器学习技术是未来研究的热点,联合优化深度神经网络流程的所有模型参数.神经网络架构搜索的研究主要集中在深度神经网络上,许多搜索架构都源自 NASNet[6]搜索空间,通过各种搜索算法在定义的搜索空间内自动生成的,广泛应用于解决图像识别、图像分割和语言建模等任务[6,7,98,99],但是只能针对某一特定或同一类型的数据集,如何使用跨不同数据集的知识来加速优化过程是未来研究的热点.其他的挑战是联合优化深度神经网络流程的所有模型参数.到目前为止,深度神经网络的通用自动化仍处于起步阶段,许多问题尚未得到解决.然而,这仍然是一个令人兴奋的领域,并且未来的工作的方向需要强调其突出的实用性.

轻量级模型的发展使得神经网络更加高效,从而能够广泛的应用到各种场景任务中.一方面,轻量级神经网络有更小的体积和计算量,降低了对设备存储能力和计算能力的需求,既可以装配到传统家电中使其更加智能化,也可以将深度学习系统应用在虚拟现实、增强现实、智能安防和智能可穿戴设备等新兴技术中；另一方面,轻量级神经网络具有更快的运行速度和更短的延时,能够对任务进行实时处理,对于在线学习、增量学习和分布式学习有重大意义,另外,实时处理的神经网络能够满足自动驾驶技术的需求,提高自动驾驶的安全性.轻量级神经网络将对于人工智能技术的普及、建立智能化城市起不可或缺的作用.

References:

[1]. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, 2015.

[2]. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016: 770-778.

[3]. Huang G, Liu Z, Van Der Maaten L, Q. Weinberger K. Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017: 4700-4708.

[4]. Han S, Mao H, Dally W J. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. In: 4th International Conference on Learning Representations, 2016.

[5]. He Y, Lin J, Liu Z, Wang H, Li L, Han S. Amc: Automl for model compression and acceleration on mobile devices. In: Proceedings of the European Conference on Computer Vision, 2018: 784-800.

[6]. Zoph B, Vasudevan V, Shlens J, V. Le Q. Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018: 8697-8710.

[7]. Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, V. Le Q. Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019: 2820-2828.

[8]. Howard A G, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017: 432-445.

[9]. Zhang X, Zhou X, Lin M, Sun J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 6848-6856.

[10]. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 4510-4520.

[11]. Qin Z, Li Z, Zhang Z, Bao Y, Yu G, Peng Y, Sun J. ThunderNet: Towards Real-Time Generic Object Detection on Mobile Devices. In: Proceedings of the IEEE International Conference on Computer Vision. 2019: 6718-6727.

[12]. Ma N, Zhang X, Zheng H, Sun J. ShuffleNet V2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European Conference on Computer Vision, 2018: 116-131.

[13]. Landola F N, Han S, Moskewicz M W, Ashraf K, Han S, Dally W J, Keutzer K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. In: 5th International Conference on Learning Representations, 2017.

[14]. Bergstra J S, Bardenet R, Bengio Y, Kégl B. Algorithms for hyper-parameter optimization. In: Advances in neural information processing systems, 2011: 2546-2554.

[15]. Yoon J, Kim T, Dia O, Kim S, Bengio Y, Ahn S. Bayesian model-agnostic meta-learning. In: Advances in Neural Information Processing Systems. 2018: 7332-7342.

[16]. Jaderberg M, Vedaldi A, Zisserman A. Speeding up Convolutional Neural Networks with Low Rank Expansions, In: British Machine Vision Conference, 2014.

[17]. Peng C, Zhang X, Yu G, Luo G, Sun J. Large Kernel Matters--Improve Semantic Segmentation by Global Convolutional Network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017: 4353-4361.

[18]. rizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, 2012: 1097-1105.

[19]. Xie S, Girshick R, Dollár P, Tu Z, He K. Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1492-1500.

[20]. Zhang T, Qi G J, Xiao B, Wang J. Interleaved group convolutions. In: Proceedings of the IEEE International Conference on Computer Vision. 2017: 4373-4382.

[21]. Xie G, Wang J, Zhang T, Lai J, Hong R, Qi G-J. Interleaved structured sparse convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 8847-8856.

[22]. Mehta S, Rastegari M, Caspi A, Shapiro L, Hajishirzi H. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In: Proceedings of the European Conference on Computer Vision, 2018: 552-568.

[23]. Mehta S, Rastegari M, Shapiro L, Hajishirzi H. Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 9190-9200.

[24]. Li H, Kadav A, Durdanovic I, Samet H, Graf H P. Pruning filters for efficient convnets. In: 5th International Conference on Learning Representations, 2017.

[25]. Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C. Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE International Conference on Computer Vision, 2017: 2736-2744.

[26]. Hu H, Peng R, Tai Y-W, Tang C-K. Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures. In: 5th International Conference on Learning Representations, 2015.

[27]. Tian Q, Arbel T, Clark J J. Deep lda-pruned nets for efficient facial gender classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017: 10-19.

[28]. Molchanov P, Tyree S, Karras T, Aila T, Kautz J. Pruning convolutional neural networks for resource efficient inference. In: 5th International Conference on Learning Representations, 2018.

[29]. Luo J-H, Wu J, Lin W. Thinet: A filter level pruning method for deep neural network compression. In: Proceedings of the IEEE international conference on computer vision, 2017: 5058-5066.

[30]. He Y, Zhang X, Sun J. Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision, 2017: 1389-1397.

[31]. Wen W, Wu C, Wang Y, Chen Y, Li H. Learning structured sparsity in deep neural networks. In: Advances in neural information processing systems, 2016: 2074-2082.

[32]. Yu R, Li A, Chen C-F, Lai J H, Morariu V I, Han X, Davis L S. Nisp: Pruning networks using neuron importance score propagation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 9194-9203.

[33]. Gupta S, Agrawal A, Gopalakrishnan K, Barayanan P. Deep learning with limited numerical precision. In: International Conference on Machine Learning, 2015: 1737-1746.

[34]. Dettmers T. 8-bit approximations for parallelism in deep learning. In: 4th International Conference on Learning Representations, 2016.

[35]. Courbariaux M, Bengio Y, David J-P. Binaryconnect: Training deep neural networks with binary weights during propagations. In: Advances in neural information processing systems, 2015: 3123-3131.

[36]. Hubara I, Courbariaux M, Soudry D, Yaniv R, Bengio Y. Binarized Neural Networks. In: Advances in Neural Information Processing Systems 29. 2016:4107-4115.

[37]. Li F, Zhang B, Liu B. Ternary weight networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016.

[38]. Leng C, Dou Z, Li H, Zhu S, Jin R. Extremely low bit neural network: Squeeze the last bit out with admm. In: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[39]. Hu Q, Wang P, Cheng J. From hashing to cnns: Training binary weight networks via hashing. In: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[40]. Wang P, Hu Q, Zhang Y, Zhang C, Liu Y, Cheng J. Two-step quantization for low-bit neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 4376-4384.

[41]. Tung F, Mori G. CLIP-Q: Deep network compression learning by in-parallel pruning-quantization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 7873-7882.

[42]. Denton E L, Zaremba W, Bruna J, LeCun Y, Fergus R. Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in neural information processing systems, 2014: 1269-1277.

[43]. Zhang X, Zou J, He K, Sun J. Accelerating very deep convolutional networks for classification and detection. IEEE transactions on pattern analysis and machine intelligence, 2015, 38(10): 1943-1955.

[44]. Lebedev V, Ganin Y, Rakhuba M, Oseledets I, Lempitsky V. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In: 3rd International Conference on Learning Representations, 2015.

[45]. Kim Y D, Park E, Yoo S, Choi T, Yang L, Shin D. Compression of deep convolutional neural networks for fast and low power mobile applications. In: 4th International Conference on Learning Representations, 2016.

[46]. Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network. In: Advances in Neural Information Processing Systems 27 Workshop, 2014.

[47]. Romero A, Ballas N, Kahou S E, Chassang A, Gatta C, Bengio Y. Fitnets: Hints for thin deep nets. In: 3rd International Conference on Learning Representations, 2015.

[48]. Zagoruyko S, Komodakis N. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In: 5th International Conference on Learning Representations, 2017.

[49]. Lillicrap T P, Hunt J J, Pritzel A, Heess N, Erez T, Tassa, Y, Wierstra D. Continuous control with deep reinforcement learning. In: 4th International Conference on Learning Representations, 2016.

[50]. Ashok A, Rhinehart N, Beainy F, Kitani K M. N2n learning: Network to network compression via policy gradient reinforcement learning. In: 6th International Conference on Learning Representations, 2018.

[51]. Wong C, Houlsby N, Lu Y, Gesmundo A. Transfer Learning with Neural AutoML. In: Advances in Neural Information Processing Systems 31, 2018:8366-8375.

[52]. Lin J, Rao Y, Lu J, Zhou J. Runtime neural pruning. In: Advances in Neural Information Processing Systems, 2017: 2181-2191.

[53]. Wang H, Zhang Q, Wang Y, Hu H. Structured probabilistic pruning for convolutional neural network acceleration. In: British Machine Vision Conference, 2018.

[54]. Real E, Aggarwal A, Huang Y, Le Q V. Regularized Evolution for Image Classifier Architecture Search. In: AAAI Conference on Artificial Intelligence, 2019.

[55]. Chen L-C, Collins M, Zhu Y, Papandreou G, Zoph B, Schroff F, Adam H, Shlens J. Searching for efficient multi-scale architectures

for dense image prediction. In: Advances in Neural Information Processing Systems, 2018: 8699-8710.

[56]. Chollet F. Xception: Deep learning with depth-wise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017: 1251-1258.

[57]. Yu F, Koltun V. Multi-scale context aggregation by dilated convolutions. In: 4th International Conference on Learning Representations, 2016.

[58]. Baker B, Gupta O, Naik N, Raskar R. Bowen Baker, Otkrist Gupta, Nikhil Naik, Ramesh Raskar. In: 5th International Conference on Learning Representations, 2017.

[59]. Suganuma M, Shirakawa S, Nagao T. A genetic programming approach to designing convolutional neural network architectures. In: Proceedings of the Genetic and Evolutionary Computation Conference, 2017: 497-504.

[60]. Cai H, Chen T, Zhang W, Yu Y, Wang J. Efficient architecture search by network transformation. In: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[61]. Mendoza H, Klein A, Feurer M, Springenberg J, Hutter F. Towards automatically-tuned neural networks. In: Workshop on Automatic Machine Learning, 2016: 58-65.

[62]. Zoph B, Le Q V. Neural architecture search with reinforcement learning. In: 5th International Conference on Learning Representations, 2017.

[63]. Szegedy C, Liu W, Jia Y, Sermanet P, Reed R E, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015: 1-9.

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

[65]. Pham H, Guan M Y, Zoph B, Le Q V, Dean J. Efficient Neural Architecture Search via Parameter Sharing. In: Proceedings of the 35th International Conference on Machine Learning, 2018: 4092-4101.

[66]. Elsken T, Metzen J H, Hutter F. Efficient multi-objective neural architecture search via lamarckian evolution. In: 7th International Conference on Learning Representations, 2019.

[67]. Cai H, Yang J, Zhang W, Han S, Yu Y. Path-Level Network Transformation for Efficient Architecture Search. In: Proceedings of the 35th International Conference on Machine Learning, 2018: 677-686.

[68]. Liu H, Simonyan K, Yang Y. DARTS: Differentiable Architecture Search. In: 7th International Conference on Learning Representations, 2019.

[69]. Zhong Z, Yan J, Wu W, Shao J, Liu C-L. Practical block-wise neural network architecture generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 2423-2432.

[70]. Dong J-D, Cheng A-C, Juan D-C, Wei W, Sun M. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. In: Proceedings of the European Conference on Computer Vision, 2018: 517-531.

[71]. Zhong Z, Yan J, Wu W, Shao J, Liu C-L. Practical block-wise neural network architecture generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 2423-2432.

[72]. Chen T, Goodfellow I, Shlens J. Net2net: Accelerating learning via knowledge transfer. In: 4th International Conference on Learning Representations, 2016.

[73]. Goldberg D E, Deb K: A comparative analysis of selection schemes used in genetic algorithms, Foundations of genetic algorithms: Elsevier, 1991: 69-93.

[74]. Cubuk E D, Zoph B, Schoenholz S S, Le Q V. Intriguing Properties of Adversarial Examples. In: 6th International Conference on Learning Representations Workshop, 2018.

[75]. Liu H, Simonyan K, Vinyals O, Fernando C, Kavukcuoglu K. Hierarchical Representations for Efficient Architecture Search. In: 6th International Conference on Learning Representations, 2018.

[76]. Elsken T, Metzen J H, Hutter F. Simple and efficient architecture search for Convolutional Neural Networks. In: 6th International Conference on Learning Representations, workshop, 2018.

[77]. Wistuba M. Deep learning architecture search by neuro-cell-based evolution with function-preserving mutations. In: Joint European

Conference on Machine Learning and Knowledge Discovery in Databases, 2018: 243-258.

[78]. Real E, Moore S, Selle A, Saxena S, Suematsu Y L, Tan J, Le Q V, Kurakin A. Large-scale evolution of image classifiers. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70, 2017: 2902-2911.

[79]. Xie L, Yuille A. Genetic cnn. In: Proceedings of the IEEE International Conference on Computer Vision, 2017: 1379-1388.

[80]. Kandasamy K, Neiswanger W, Schneider J, Póczos B, Xing E P. Neural architecture search with bayesian optimisation and optimal transport. In: Advances in Neural Information Processing Systems, 2018: 2016-2025.

[81]. Luo R, Tian F, Qin T, Chen E, Liu T-Y. Neural architecture optimization. In: Advances in neural information processing systems, 2018: 7816-7827.

[82]. Bender G, Kindermans P-J, Zoph B, Vasudenvan V, Le Q V. Understanding and simplifying one-shot architecture search. In: International Conference on Machine Learning, 2018: 549-558.

[83]. Brock A, Lim T, Ritchie J M, Weston N. SMASH: One-Shot Model Architecture Search through HyperNetworks. In: 6th International Conference on Learning Representations, 2018.

[84]. Zhang C, Ren M, Urtasun R. Graph HyperNetworks for Neural Architecture Search. In: 7th International Conference on Learning Representations, 2019.

[85]. Real E, Aggarwal A, Huang Y, Le Q V. Regularized evolution for image classifier architecture search. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33: 4780-4789.

[86]. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M S, Berg A C, Li F-F. Imagenet large scale visual recognition challenge. International journal of computer vision, 2015, 115(3): 211-252.

[87]. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick C L. Microsoft coco: Common objects in context. In: European conference on computer vision, 2014: 740-755.

[88]. Zela A, Klein A, Falkner S, Hutter Frank. Towards Automated Deep Learning: Efficient Joint Neural Architecture and Hyperparameter Search. In: Proceedings of the 21th International Conference on Artificial Intelligence and Statistics Workshop. 2018.

[89]. Klein A, Falkner S, Bartels S, Hennig P, Hutter F. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. 2017:528-536.

[90]. Domhan T, Springenberg J T, Hutter F. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[91]. Snoek J, Rippel O, Swersky K, Kiros R, Satish N, Sundaram N, Patwary M, Prabhat, Adams R. Scalable Bayesian Optimization Using Deep Neural Networks. In: Proceedings of the 32nd International Conference on Machine Learning, 2015:2171-2180

[92]. Klein A, Falkner S, Springenberg J T, Hutter F. Learning curve prediction with Bayesian neural networks. In: 5th International Conference on Learning Representations, 2017.

[93]. Baker B, Gupta O, Raskar R, Naik N. Accelerating neural architecture search using performance prediction. In: 6th International Conference on Learning Representations, Workshop, 2018.

[94]. Wei T, Wang C, Rui Y, Chen C. Network morphism. In: Proceedings of the 33nd International Conference on Machine Learning, 2016: 564-572.

[95]. Li L, Jamieson K, DeSalvo G, Rostamizadeh A, Talwalkar A. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. Journal of Machine Learning Research, 2017, 18(185):1-52.

[96]. Runge F, Stoll D, Falkner S, Hutter F. Learning to Design RNA. In: 7th International Conference on Learning Representations, 2019.

[97]. Xie S, Zheng H, Liu C, Lin L. SNAS: stochastic neural architecture search. In: 7th International Conference on Learning Representations, 2019.

[98]. Weng Y, Zhou T, Li Y, Qiu X. NAS-Unet: Neural Architecture Search for Medical Image Segmentation. IEEE Access, 2019, 7: 44247-44257.

[99]. Liu C, Chen L C, Schroff F, Adam H, Hua W, Yuille A, Li F-F. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 82-92.