

中国科学技术大学

# 专业硕士学位论文

(专业学位类型)



## 面向嵌入式设备下 Resnet 卷积神经网 络尺寸压缩与加速方法研究

作者姓名：翟文杰

专业领域：软件工程

校内导师：朱宗卫 教授

企业导师：朱宗卫 高级工程师

完成时间：二〇二一年十一月二日



University of Science and Technology of China  
A dissertation for master's degree  
(Professional degree type)



**Research on Optimization and  
Acceleration Methods of  
Convolutional Neural Networks on  
Embedded Platforms**

Author: WenJie Zhai

Speciality: Software Engineering

Supervisor: Prof. Chao Wang

Advisor: Senior Engineer Zongwei Zhu

Finished time: November 2, 2021



## 中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文，是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名: \_\_\_\_\_

签字日期: \_\_\_\_\_

## 中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅，可以将学位论文编入《中国学位论文全文数据库》等有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

公开  保密 (\_\_\_\_ 年)

作者签名: \_\_\_\_\_

导师签名: \_\_\_\_\_

签字日期: \_\_\_\_\_

签字日期: \_\_\_\_\_



## 摘要

深度卷积神经网络在不同任务上有着远超其他方法的优秀性能，但其计算开销和内存开销限制了其在嵌入式设备上的部署和使用。我们研究如何在保持神经网络精度的情况下，同时减小网络计算开销和运行时内存，以在资源受限(计算延迟、内存占用)的情况下进行部署和使用。在本文中，我们首先广泛调研目前的神经网络压缩和优化研究现状，总结了该领域现有的几类主流研究方向：模型修剪、矩阵/张量分解、模型参数量化、高效网络设计、知识蒸馏，对各类算法和研究进行了大量的文献阅读和探索性的实验，进行一定的总结和综述。其次，我们利用网络参数对于不同输入的重要性的不同，学习不同类型任务在现有网络中的有效连接(即子网络)，并提出一种在运行时预测任务类型和子网络动态选择的新方法——任务感知的子网络切换。该方法在现有网络模型的基础上引入一个辅助分类器和通道选择结构。与传统减枝相比，它根据输入进入不同子网络，动态跳过不重要的参数和连接，在保证计算精度的同时加速卷积，并且通过子网络感知动态加载模型参数，能够显著减小运行时内存，使得深度卷积网络能够更好地部署在内存和算力受限的嵌入式硬件设备上。然后，我们针对多分支网络的内存低效和低并行度结构，提出部署时多分支算子融合策略，提出采用重参数化技术对多分支网络进行融合，减少网络分支情况，将多分支网络合并成为一个类 VGG 的单路网络，提高网络部署运行时的内存效率和并行度，节省网络资源消耗，加快网络推理速度。

**关键词：**卷积神经网络；嵌入式；部署；知识蒸馏；重参数化；网络优化

## ABSTRACT

Deep Convolutional Neural Network has far superior performance in different tasks than other methods, but its computational and memory overhead limits its deployment and use on embedded devices. We study how to reduce network computing overhead and runtime memory while maintaining the precision of the neural network, so as to deploy and use it under the condition of resource constraints (computation latency, memory footprint). In this article, we first compression and extensive research current neural network optimization research present situation, summarizes the existing several kind of mainstream research direction in this field: model pruning, matrix/tensor decomposition, model parameter quantitative distillation, efficient network design, knowledge, research on all kinds of algorithms and do a lot of literature reading and exploratory experiments, a summarized and reviewed in this paper. Secondly, we take advantage of the different importance of network parameters to different inputs to learn the effective connections of different types of tasks in the existing network (i.e., subnetworks), and propose a new method to predict the dynamic selection of task types and subnetworks at runtime – task-aware subnetwork handoff. This method introduces an auxiliary classifier and channel selection structure on the basis of the existing network model. Compared with traditional cut branches, it is based on the input into different sub networks, dynamic skip important parameters and no connection, in guarantee the calculation accuracy and speed up convolution, and dynamic load model parameters through sub network awareness, can significantly reduce the runtime memory, make the depth of the convolution network can be better deployed in memory and calculate the force limited embedded hardware devices. Then, our memory is inefficient for many branch network and low degree of parallel structure, proposes a multiple branch operator fusion strategy deployment, parametric technology is presented in this paper to fusion of multiple branch network, reduce network branch, several branch network will become a kind of VGG single road network, improve the network deployment runtime memory efficiency, and parallelism, save the network resource consumption, speed up the network reasoning.

**Key Words:** Convolutional Neural Networks; Embedded; Deployment, Knowledge Distillation; Re-parameterization; Networks Optimization

## 目 录

第 1 章 绪论 ······	1
1.1 引言 ······	1
1.2 研究背景及意义 ······	3
1.3 国内外研究现状 ······	6
1.4 本文的主要研究内容 ······	7
1.5 本文的结构安排 ······	8
第 2 章 相关背景知识及理论基础 ······	11
2.1 引言 ······	11
2.2 深度学习概述 ······	11
2.3 卷积神经网络架构 ······	13
2.4 数据集 ······	15
2.5 实验环境说明 ······	16
2.5.1 深度学习框架说明 ······	16
2.6 模型压缩与优化方法 ······	18
2.6.1 模型修剪 ······	18
2.6.2 矩阵/张量分解 ······	19
2.6.3 参数量化 ······	20
2.6.4 高效网络设计 ······	21
2.6.5 知识蒸馏 ······	22
2.7 本章小结 ······	23
第 3 章 任务感知的动态子网络交换 ······	24
3.1 引言 ······	24
3.2 动态网络 ······	25
3.3 卷积层通道激活特性的分析 ······	27
3.4 任务感知的子网络划分 ······	28
3.5 通道选择因子 ······	29
3.6 关键子网络激活 ······	30
3.7 任务感知的子网络交换 ······	32
3.8 子网方案实现成本 ······	32
3.9 实验结果与分析 ······	33
3.9.1 实验设置 ······	33

3.9.2 实验结果	33
3.10 本章小结	35
第 4 章 针对残差结构的分支融合方法	37
4.1 引言	37
4.2 重参数化与辅助训练	38
4.3 线性网络与多分支结构网络	39
4.3.1 线性网络	39
4.3.2 残差网络与 Shortcut 连接	40
4.3.3 多分支 inception 模块	40
4.3.4 线性网络与多分支网络对比	41
4.4 基于重参数化的残差模块融合	42
4.4.1 卷积层和批归一化层合并	43
4.4.2 卷积层纵向融合	45
4.4.3 卷积层横向融合	46
4.5 残差模块融合收益分析与通道调整	48
4.5.1 BasicBlock 残差模块	49
4.5.2 Bottleneck 残差模块	50
4.6 实验结果与分析	50
4.6.1 实验设置	50
4.6.2 实验结果	51
4.7 本章小结	52
第 5 章 总结与展望	53
5.1 全文总结	53
5.2 后续工作展望	54
参考文献	55
致谢	61
在读期间发表的学术论文与取得的研究成果	62
.1 硕士期间发表的论文	62
.2 硕士期间参与的项目	62

# 第1章 绪 论

## 1.1 引言

自上世纪以来，实现人工智能 (Artificial Intelligence, AI) 是许多学者的梦想。在许多来自不同领域的学者们的共同不懈地努力下，人工智能至今已经取得了长足的发展。当下，人工智能已经能够完成多样复杂的任务，如识别复杂物体<sup>[1]</sup>、识别语音内容<sup>[2]</sup>、处理自然语言<sup>[3]</sup>、过滤社交网络信息、机器翻译<sup>[4]</sup>、设计药物、分析医疗图像<sup>[5]</sup> 和材料审查等任务。因此，人工智能被广泛地用于人类社会中的各个领域中，以帮助人类更快的解决完成处理信息问题。但纵观人工智能的发展历史，其并不是一路向上，而是经历了 60 年的沉浮，并且其深刻地受到了计算机硬件技术发展的影响。总的来说，人工智能的发展经历了三次高峰。人工智能的第一次高峰是 20 世纪 40 年代-60 年代的控制论。彼时的学者认为人工智能起源于数理逻辑，希望用数理逻辑描述生物智能行为。符号主义是纯数学上的逻辑推理的，即将输入数据转换成逻辑表达，用符号演算的形式化方法进行逻辑推理，尚无训练的概念。1956 年 Simon 开发出一个启发程序“LT 逻辑理论家”，能够证明 38 条数学定理。随后符号主义学派发展了启发式算法、专家系统、知识工程理论等。但符号主义学派框架下的人工智能需要足够多的先验知识，人工智能并不具备自主学习的能力，这一缺陷使得人工智能只能处理相对简单的任务。此后符号主义发展缓慢，人工智能进入瓶颈期。人工智能的第二次高峰是 20 世纪 80 年代-90 年代的连接主义。自 1946 年第一台通用计算机 ENIAC 诞生和随后几十年计算机算力的飞速发展，人工智能的研究思路从数理逻辑的符号主义，逐步切换到连结主义和概率统计模型的新方法。连接主义认为人工智能起源于仿生学，其首次诞生于 1943 年，神经生理学家 Warren McCulloch 和数学家 Walter Pitts 对生物神经元建模，提出了一种神经元 M-P 模型。在 M-P 模型基础之上，1957 年 Frank Rosenblatt 将多个 M-P 模型组合叠加，提出了感知机模型 (Perceptron)。受限于彼时计算机算力的不足，Frank Rosenblatt 在 1960 年用硬件电路实现了第一个神经网络。随后，Paul Werbos 在 1974 年提出了反向传播算法 (Backpropagation) 成功的让感知机具备了学习线性不可分函数的能力。人工智能进入到一个高速发展的黄金时期。但在 20 世纪 90 年代中期，Vapnik 等人提出了比神经网络更高效，无须调参，泛化性能优异的支持向量机 (SVM)，它同样可以解决线性不可分的问题，支持向量机的出现迅速打败了神经网络。同时，受限于当时计算机算力的制约和数据量，人工智能未能走出实验室进行大规模的应用。人工智能的第三次高峰是 2006 年至今的深度学习。2006 年，Hinton 提出贪婪逐层的训练方法，这个技术能够有效训练深度神经网络，但彼时的算力并不支持该

技术的实施。后在 2012 年, Alex 等人借助于分布式 GPU 的计算能力, 成功训练出深度神经网络 AlexNet 斩获 Imagenet 比赛冠军。同时, 各类移动计算设备的计算能力也快速提升。2017 年苹果发布第一款人工智能芯片 A11, 该芯片首次搭载了神经网络引擎处理单元 (NPU)。此后人工智能开始飞速发展, 硬件算力的提升加上多种新方法的涌现, 人工智能走向市场, 并被广泛地应用于无人驾驶、无人机、智能手机、智能眼镜、机器人等应用。

在所有深度学习方法中, 卷积神经网络 (Convolutional Neural Networks, CNN) 以其优异的性能受到了大量学者的广泛关注。卷积神经网络是一类以卷积运算为主、有较深层次结构、具有表征学习能力 (representation learning) 的前馈神经网络。卷积神经网络的首次提出是在 1998 年, Yann LeCun 等人提出一个小型卷积网络 LeNet, 用于手写字符识别任务<sup>[6]</sup>。该项工作首次定义了 CNN 网络的结构的基本框架: 卷积层 + 池化层 + 全连接层。随后, 在 2012 年举办的大规模图像识别竞赛 (ImageNet Large Scale Visual Recognition Challenge, ILSVRC) 中, Krizhevsky 等人提出的深度卷积神经网络 AlexNet<sup>[7]</sup> 以超过第二名 10.9 个百分点的识别准确度斩获 ImageNet 比赛的冠军。该工作有力地证明了卷积神经网络的学习能力。此后, 随着 GPU 算力的不断提升, 卷积神经网络的层数也不断加深。如 2014 年提出的有 19 层深度的 VGGNet<sup>[8]</sup> 和 2015 年提出的有 22 层深度的 GoogLeNet<sup>[9]</sup>。在以往的研究工作中, 学者们普遍认为卷积神经网络的拟合能力由模型层数决定, 即模型层数越多, 网络的拟合能力越强。但事实上, 随着网络层数的增多, 卷积神经网络会出现退化问题 (Degrade Problem), 即卷积神经网络层数到达一定深度后, 拟合能力会出现退化, 在测试集上的误差反倒会上升。2015 年的 ResNet<sup>[11]</sup> 提出使用跳层连接 (shortcut) 来减少梯度消失的问题。该工作成功的解决了神经网络层数增多后出现的拟合能力退化的问题, 并首次将神经网络层数增加到了 152 层, 并将 imagenet 数据集上识别准确率提高到 96.5%。同时, 在其他各个领域学者们的合力推动下, 卷积神经网络在目标检测、语义分割、人脸识别等其他计算机视觉任务也取得了巨大的成功。

目前, 随着各类移动计算设备的算力和存储空间的飞速提升, 卷积神经网络已成为许多智能化系统中的基本工具。但现有的卷积神经网络模型需求的计算资源和硬件实际能提供的计算资源供需之间仍然存在不匹配: 1) 神经网络模型尺寸远远大于计算设备可提供 DRAM 空间; 2) 神经网络模型的计算量远远大于计算设备可提供的计算资源。其中, 第一个挑战决定了是否可以将神经网络部署在移动计算设备上进行处理。第二个挑战则决定了神经网络是否可以满足某些特定实时场景下的推理时间约束。因此, 对于神经网络模型的尺寸压缩和加速已经成为目前深度学习研究的一个热门话题。本文以深度卷积神经网络为研究对象, 从不同的角度探索研究深度卷积神经网络在嵌入式平台下部署的优化与加

速算法，在有限的计算资源的前提下尽可能的发挥出深度学习的性能优势，摆脱硬件和功耗的束缚，扩展深度学习的应用场景。

## 1.2 研究背景及意义

近年来，深度卷积神经网络因其有效性和优异的泛化性，得到了各领域学者的广泛关注。但模型的准确度和泛化性取决于模型的参数量和层数多少，即越复杂、层数越深、参数量越多的模型往往拥有更优异的准确度表现和泛化性能。表 1.1展示了历届 ImageNet 大规模视觉识别挑战赛 (ILSVRC) 中最有代表性的五个模型的 Top-5 错误率、层数、参数量、FLOPs(浮点运算次数, floating point operations, FLOPs)。从表1.1中可以明显的观察到，随着模型层数的加深和 FLOPs 的增多，卷积神经网络模型在 Imagenet 上的 Top-5 错误率呈明显下降的趋势。对比于 2012 年提出的 AlexNet，SENet 的 FLOPs 增加了惊人的 29 倍，同时它的图片分类错误率 Top-5 错误率也下降到了 2.25%。在卷积神经网络层数和参数量飞速增加的另一面，虽然硬件的计算性能也在逐年增长，却总是无法与模型所需 FLOPs 保持持平。以英伟达公司在 2012 年发布的顶级 GPU 处理器 Tesla K40 为例，其理论双精度每秒浮点运算次数是 1.43TFLOPS (每秒浮点运算次数, tera float point operations per second, TFLOPS)。而 2019 年英伟达公司发布的 Tesla v100 其理论双精度每秒浮点运算次数是 7.066TFLOPS，7 年时间性能只提升了 4.9 倍。而卷积神经网络模型的 FLOPs 却在 5 年时间飞速增长了 29 倍。这显示出了，硬件所可以提供计算能力和模型所需要计算力之间的严重的供需不匹配关系。即便是使用当年最顶级的 NVIDIA K40 GPU 在 ImageNet 数据集上训练 AlexNet 也需要 2-3 天时间。此外，除了卷积神经网络模型 FLOPs 的显著增长之外，卷积神经网络模型的架构设计也越来越复杂。更复杂的计算流，进一步对底层硬件提出了更高的性能要求。如表1.1所示，有越来越明显的趋势表明，更复杂的网络架构模型被设计出来用于提供图片分类准确率。与 vgg 和 alexnet 等基于线性拓扑结构的模型相比，google Net 采用了精心设计的多分支架构，而 resnet 提出了双分支跳层连接架构，senet 在 resnet 中引入了通道注意模块。多分支架构的 googlenet、resnet 和 senet 的性能优于线性拓扑结构 alexnet 和 vgg。原因是随着网络层数的增加，线性拓扑结构无法避免梯度消失问题。因此，线性拓扑模型很难达到与多分支体系结构相当的精度。

在卷积神经网络和硬件计算力供需严重不匹配的另一面，是移动和嵌入式场景下对卷积神经网络的强烈需求。在早期，为了满足深度学习的计算需求，许多学者提出了云计算的概念。即将数据从网络边缘的数据源(从智能手机到物联网 (IoT) 传感器) 传输到云服务平台，让云服务平台进行计算后再将结果返回到

边缘设备端。但同时这种将数据从源迁移到云的解决方案带来了几个问题：a) 延迟：实时推理对于许多应用程序都是至关重要的。例如，自动驾驶汽车上的摄像机帧需要快速处理以检测和避免障碍物，或者基于语音的辅助应用程序需要快速解析和理解用户的查询并返回响应。但是，将数据发送到云中进行推理或训练可能会导致额外的排队和网络的传输延迟，无法满足这些有着端到端、低延迟需求的交互式应用场景。例如，实验表明，将摄像机帧 offload 到 Amazon Web 服务服务器并执行计算机视觉任务需要 200 ms 以上的端到端时间。b) 隐私性：在将数据传输至云端进行推理时存在隐私暴露的问题。近年来，随着无人机、智能手机、智能眼镜等嵌入式和移动设备计算能力的长足进步，使得在这些设备上直接部署深度神经网络成为可能。同时也催生出了许多应用场景如，手机人脸识别、自动驾驶、手机语音助手、智能停车系统、巡检无人机、智能菜品识别等。但这些场景下仍面临着卷积神经网络推理速度慢和模型体积过大的挑战。

综上，巨大的计算复杂性在实际应用中的 CNN 部署中引入了两个问题。一种是随着计算复杂度的增加，CNN 推理阶段变慢。这使得很难在实时应用程序中部署 CNN。另一个问题是，CNN 固有的密集计算将消耗大量的电池电量。CNN 的大量参数会消耗相当多的存储空间和运行时内存，而这在嵌入式设备上是非常有限的。在嵌入式设备上，包括存储和计算单元以及电池电量等仍然非常有限，这对于在低成本环境中加速现代深度神经网络模型构成了真正的挑战。

表 1.1 *alexnet*, *vgg*, *google net* 和 *resnet* 的 top-5 错误率，模型大小和计算量。

	AlexNet	VGG	Google Net	ResNet	SENet
年份	2012	2014	2014	2015	2017
层数	8	19	22	152	152
Top-5 错误率	15.4%	7.3%	6.7%	3.57%	2.25%
模型尺寸	233MB	548MB	6.7%	230MB	440MB
FLOPs	727M	20G	6.7%	11G	21G

考虑到更深更加复杂的神经网络，减少网络的存储和计算成本是非常指的关注的问题，尤其是对一些于实时性强的应用而言。高效的深度学习方法可能会对分布式系统，嵌入式设备和用于人工智能的 FPGA(可编程门阵列, Field Programmable Gate Array) 产生重大影响。例如，具有 50 个卷积层的 ResNet-50<sup>[1]</sup> 在处理一张图片时需要超过 95MB 的存储空间和 38 亿次浮点运算，在丢弃一些冗余权重后，网络节省超过 50% 的计算时间和 75% 的参数空间占用却仍然可以正常工作，这对于仅有有限计算和存储资源的手机和 FPGA 等设备，如何压缩和优化其上使用的模型也很重要。

因此，当前的一个关键问题是如何在不显著降低性能的情况下在资源有限

的硬件部署和运行深度神经网络模型。为了解决这个问题，在过去的几年中，已经研究了许多算法方面的巨大进步和方法，联合多学科，包括但不限于机器学习，优化理论，计算机体系结构，信号处理和硬件设计等。其中，对于硬件方面已经提出了多种基于 FPGA / ASIC 的加速器，用于嵌入式和移动应用；一些工作着重于模型本身的压缩和优化，而另一些工作着重于部署端的运行加速或降低功耗。

目前，深度学习技术已经在我们日常生活中开始应用，比如：

- 1) 智能安防：智能安防是深度学习在计算机视觉领域最早的应用和落地案例之一，使用区分性强，应用方便的人脸和指纹图像信息作为身份特征，广泛应用于各种城市公安系统和安防系统中。越来越多的企业如依图科技、商汤科技和 Face++ 等纷纷采用深度卷积神经网络算法来进行人脸识别系统和安防监控系统的开发和部署，通过公共摄像头拍摄的画面，自动筛查犯罪嫌疑人和可疑车辆，并自动或辅助人工进行对犯罪行为的发现和追踪。随着人工智能技术的发展和智慧城市构建的需要，基于深度学习的计算机视觉技术将会有更加广泛的应用。
- 2) 智能机器人：智能机器人是深度学习在人工智能领域的重要应用场景之一。智能机器人通过多种传感器融合对环境进行感知或进行人机交互，根据环境和接收指令进行后续的运动控制和规划决策，在教育、医疗、服务等行业都有一定程度的应用，基于深度学习的自动驾驶和无人机自动勘探也是近年来逐渐火热的应用场景。
- 3) 移动设备：移动设备有大量的应用可以通过人工智能进行优化，在在线购物、游戏娱乐，摄像摄影等手机功能中都能带来更好的使用体验。在智能手机、平板电脑等嵌入式移动设备上进行深度学习算法的部署和应用也随着近年来手机处理芯片的不断提升而成为可能。

在以上这些场景中，硬件的存储容量、计算能力和功耗限制都格外严苛，同时也对应用的实时性有着较高的要求，而深度学习的模型大小动辄上百兆，并且运行时的巨量的内存消耗和算力要求也是资源受限的嵌入式平台所无法忍受的，其他一些嵌入式设备如智能电视、可穿戴设备和摄影设备等的性能更低，对深度卷积神经网络的限制也更加大。为了能更好的将深度学习技术利用在日常生活的落地与应用，网络模型的压缩有优化以及在嵌入式平台上的部署与加速成为比模型性能更为重要也更为关键的问题。

### 1.3 国内外研究现状

近年来，深度神经网络在许多领域，尤其是视觉识别任务中取得了巨大的成功。然而，现有的深度神经网络模型计算成本高，内存密集，阻碍了它们在内存资源较低的设备或在有严格延迟要求的应用中部署。

尽管面临挑战和限制，研究者们在硬件和软件两方面着手，积极进行研究和攻关。硬件方面，高效的深度学习专用硬件快速发展，设计者为神经网络专门设计了定制的硬件加速器<sup>[12-19]</sup>。由于专门化，这些加速器根据深度学习的计算模式定制了硬件架构，相比cpu和gpu实现了更高的效率。第一波加速器有效地实现了神经网络的计算原语<sup>[12,14,20]</sup>。研究人员随后意识到内存访问更昂贵，急需优化，因此第二波加速器有效地优化了内存传输和数据移动<sup>[15-19]</sup>。这两代加速器在提高DNNs运行速度和能源效率方面取得了可喜的进展。

然而，仅仅只是专注于优化硬件架构，提升硬件执行效率只是一方面。事实上，现存的使用传统硬件架构的嵌入式设备已经普遍的应用和服务，并不具备大规模硬件替换和升级的可能性。为了将深度学习技术应用和部署起来，更加通用和可行的方案是通过优化算法和模型，在接触硬件之前，对深度神经网络模型进行压缩和优化。研究发现，神经网络通常是过度参数化的，深度学习模型存在显著的冗余<sup>[21]</sup>，这导致了计算和内存的浪费。研究者在多个方向进行了研究和探索，对模型进行压缩和优化，基于他们的性质，将这些优化方法分为五类：模型修剪、矩阵/张量分解、模型参数量化、高效网络设计。

- 模型修剪：包括结构化减枝和非结构化减枝，关注探索模型中不重要的参数以及冗余结构，并尝试删除以降低模型复杂度；
- 矩阵/张量分解：基于矩阵低秩分解原理，对模型参数矩阵进行低秩分解以得到每层参数的有效信息，能够有效地压缩和加速模型；
- 模型参数量化：通过聚类算法，减少表示每个参数的比特位数来减少参数的存储开销，其中定点量化使用更少比特如1比特，8比特来存储参数，能显著降低存储和运算开销；
- 高效网络设计：直接手动设计或自动搜索层数浅、参数少的高效网络结构，以更少的参数和计算量得到不错的效果；
- 知识蒸馏：使用预训练的大网络（教师网络）来训练一个小网络（学生网络），将知识从繁琐的模型转移到更是一个部署的小模型。

早期的方法是最优神经损失<sup>[22]</sup>和最优神经手术<sup>[21]</sup>，这减少了基于损失功能的连接数量。Denton等人<sup>[23]</sup>通过寻找合适的低秩近似来减少参数的数量，利用了神经网络的线性结构。奇异值分解(SVD)和Tucker分解也可以减少权值的数量<sup>[24]</sup>。为了减少神经网络参数的数量，已经出现了一些架构上的创新，比如用卷

积层代替全连接层，或者用全局平均池化代替全连接层。NIN<sup>[25]</sup> 和 GoogleNet<sup>[9]</sup> 采用了这一思想，取得了最先进的成果。也有人建议减少精度和位宽。Vanhoucke 等人<sup>[26]</sup> 探索了一个用 8 位整数 (vs 32 位浮点) 激活的定点实现。Abwer 等人<sup>[27]</sup> 使用 L2 误差最小化量化了神经网络。Hwang 等<sup>[28]</sup> 提出了一种三元权值和 3 位激活的神经网络优化方法。Gong 等人<sup>[29]</sup> 使用矢量量化压缩深度神经网络。HashedNets<sup>[30]</sup> 通过使用哈希函数将连接权值随机分组到哈希表中，从而减少了参数的位宽。通过上述技术，可以降低权值的精度，并且每个权值可以用更少的比特来表示。Loss 近似泰勒展开<sup>[31]</sup> 提出了用于剪枝的基于梯度的重要性度量。Anwar 等人<sup>[32]</sup> 通过数百次随机评估选择剪枝候选。Yang 等<sup>[33]</sup> 根据能量消耗加权选择剪枝候选。He 等<sup>[34]</sup> 探索了基于 LASSO 回归的信道选择，在 ResNet 和 Xception 上实现了 2 倍的理论加速，精度损失分别为 1.4 % 和 1.0 %。早期剪枝<sup>[35]</sup> 和动态剪枝<sup>[36]</sup> 探讨了如何更好地将剪枝和再训练结合起来，节省再训练时间。最近的研究也讨论了不同粒度的结构化剪枝，例如降低后在行和列的粒度上进行剪枝<sup>[37]</sup>，在通道的粒度上进行剪枝，而不是在单个权重上进行剪枝<sup>[31,38-40]</sup>。Anwar 等<sup>[41]</sup> 研究了不同尺度下的结构稀疏度，包括通道稀疏度、核稀疏度和核内跨步稀疏度。Mao 等人<sup>[42]</sup> 对不同的剪枝粒度 (元素、行、核和通道) 进行了详细的设计空间探索和权衡分析。在量化方面，更激进的压缩将比特宽度推得更窄，使用 2 位权值<sup>[43]</sup>、三元权值<sup>[86]</sup> 或二进制权值<sup>[44]</sup>。也有努力量化重量和激活到一个较低的精度<sup>[45-46]</sup>。当权重和激活都被二值化时，乘法和减法可以被 XNOR 和 pop 计数代替。另外，Miyashita 等人<sup>[47]</sup> 实验了对量化权值，其中乘法可以用代价更低的移位代替。同时在部署方面，NVIDIA 在 TensorRT 上通过层融合与分支融合技术，同时进行并行化计算和调度的优化，在无损失的情况下有效的加速模型的运行时间。丁等人提出 ACNet<sup>[48]</sup>、RepVGG<sup>[49]</sup>、DBB<sup>[50]</sup>，使用重参数化技术无损进行模型分支，提高计算并行度，有效减少模型部署内存消耗和加速模型运行速度。

## 1.4 本文的主要研究内容

由于在嵌入式设备和边缘设备计算和存储资源有限，现有多数卷积神经网络在该场景下的部署和应用局限性比较大。本文主要针对嵌入式环境下计算和存储资源受限的问题，对卷积神经网络在嵌入式设备上的部署加速和优化进行研究。首先广泛调研目前的神经网络压缩和优化研究现状，探讨了如何在低算力嵌入式设备上适配神经网络或者紧凑的网络设计，即考虑通过合理选择神经网络架构和规模压缩方案，来有效地实现约束型硬件资源上的模型部署和推理。本文提出了采用模型压缩技术精简模型以减小计算量和存储需求，同时设计一种

部署友好的网络分支融合方案，提高模型性能，减小模型运行时峰值内存开销，并最终在嵌入式端进行部署和实验。

主要创新点和贡献如下：

- 1) 对模型压缩和优化加速的向相关算法进行了广泛的研究，总结了该领域现有的几类主流研究方向：模型修剪、矩阵/张量分解、模型参数量化、高效网络设计、知识蒸馏、分支融合与重参数化，对各类算法和研究进行了大量的文献阅读和探索性的实验，进行一定的总结和综述。
- 2) 由于神经网络各个卷积层有不同的分工功能，所以不同卷积层在进行裁剪时对整体性能的影响程度也会有所不同。因此，考虑迭代剪枝时的性能下降，我们改进了传统的顺序逐层剪枝的方法，首先定义我们的卷积层敏感度评价标准，基于此提出了一种基于硬性卷积层敏感度的剪枝算法，在不同剪枝率下的剪枝次序将会沿着当前状态下的低敏感度卷积层到高敏感度卷积层进行，这种算法能够最小化迭代剪枝对网络的破坏，避免造成不可逆的损失，而且，能够迅速提高剪枝率。并且提出任务感知的子网络切换优化部署时内存加载过程，减少网络模型运行时内存消耗。
- 3) 针对多分支网络的内存低效和低并行度结构，提出部署时多分支算子融合策略，提出采用重参数化技术对多分支网络进行融合，减少网络分支情况，对于无法直接合并的分支使用知识蒸馏技术进行辅助训练，将多分支网络合并成为一个类 VGG 的单路网络，提高网络部署运行时的内存效率和并行度，节省网络资源消耗，加快网络推理速度。

## 1.5 本文的结构安排

全文一共分为五个章节，第一章介绍了研究背景、研究现状和意义，并概括了主要研究内容。第二章介绍了我们在论文中所接触到的深度神经网络、数据集、训练系统和硬件平台的背景，此外还综述了模型压缩和优化方面的工作。第三章提出了一种基于卷积通道敏感度评价的子网络划分方法，并提出与之相配合的运行时任务感知子网络切换手段，在不降低精准度的前提下减少网络模型运行时内存占用同时提升模型推理速度。第四章提出了一种分支融合的方法，使用重参数化和知识蒸馏方法将多分支网络合并线性网络模型，提升网络并行度加速推理速度同时减少运行时内存占用。第五章是对全文的总结以及对未来工作的展望。

本文的各章节结构安排如下：

第一章为绪论部分，首先介绍了神经网络的发展和应用发展趋势，随后介绍了深度神经网络模型的压缩加速相关研究背景和意义。深度学习的飞速发展和

硬件设备计算能力的快速提升，使得神经网络被广泛应用与嵌入式设备上成为可能。然而深度神经网络庞大的参数与模型使其具有高存储占用和多浮点运算的特点，难以部署在资源受限的嵌入式设备上，因此非常有必要对神经网络进行压缩和加速以部署在嵌入式设备上。最后，概述了本文的主要研究内容和论文中每章的内容安排。

第二章的主要内容是相关背景知识说明，主要包含两个方面。一方面，简要说明我们在论文中所使用的深度神经网络、数据集、训练系统和硬件平台，选取了常用的几个卷积神经网络模型进行详细说明，同时介绍文章研究中所使用的训练数据集 MNIST、CIFAR10、ImageNet，以及实验训练和部署所使用的深度学习框架和硬件平台说明。另一方面，调研目前深度学习模型压缩与优化领域的相关研究，并对近几年的网络压缩与优化算法发展进行了综述，重点介绍了模型压缩和优化领域的几大主流研究方向：模型修剪、矩阵/张量分解、模型参数量化、高效网络设计、知识蒸馏、分支融合与重参数化，及其各自的优缺点。

第三章提出了基于卷积通道敏感度的子网络划分与任务感知的子网络切换。在这一章我们首先介绍了深度学习在嵌入式设备上的发展，引出模型压缩与优化的必要性。其次对比了减枝方法中静态剪枝与动态减枝的区别，突出动态减枝算法的优越性。随后我们分析了卷积核中通道激活与输入数据的关联，进而提出基于卷积通道敏感度的子网络划分方法。最后我们提出了任务感知的子网络切换方法，用以减少部署时模型的内存占用以及加速模型推理过程，并展示实验效果，验证方法有效性。

第四章为基于卷积通道敏感度的子网络划分与任务感知的子网络切换的相关研究。在这一章我们首先介绍了深度学习在嵌入式设备上的发展，引出模型压缩与优化的必要性。其次对比了减枝方法中静态剪枝与动态减枝的区别，突出动态减枝算法的优越性。随后我们分析了卷积核中通道激活与输入数据的关联，进而提出基于卷积通道敏感度的子网络划分方法。最后我们提出了任务感知的子网络切换方法，用以减少部署时模型的内存占用以及加速模型推理过程，并展示实验效果，验证方法有效性。

第四章提出了一种分支融合的方法，在部署阶段融合多分支卷积核，尽量减少网络分支结构来提高网络的内存使用效率和并行度。在这一章里我们首先介绍和分析了多分支网络模型的优势和劣势，针对多分支网络运行效率不高的问题进行重点研究。随后分析了网络模型中能够直接无损合并的分支结构，提出采用重参数化方法进行分支合并和参数融合，对于无法直接合并的分支结构，提出在分支合并之后借助知识蒸馏进行辅助训练以恢复模型性能。最后我们通过分支融合将多分支网络转换为纯线性网络进行部署，并展示实验效果，验证方法有效性。

第五章为总结与展望，主要对整篇文章的研究内容进行总结，分析研究方法中的优势和不足，并对未来可能的研究方向和预期会得到的成果进行展望。

最后是致谢部分，以及本人在研究生阶段的研究成果。

## 第2章 相关背景知识及理论基础

### 2.1 引言

在本章中，我们首先介绍什么是深度学习，它是如何工作的，以及它的应用。然后我们介绍了我们实验中涉及到的神经网络架构，数据集，以及我们用来训练数据集架构的框架。在此之后，我们将对近期在网络模型压缩与优化领域的相关工作进行综述。

### 2.2 深度学习概述

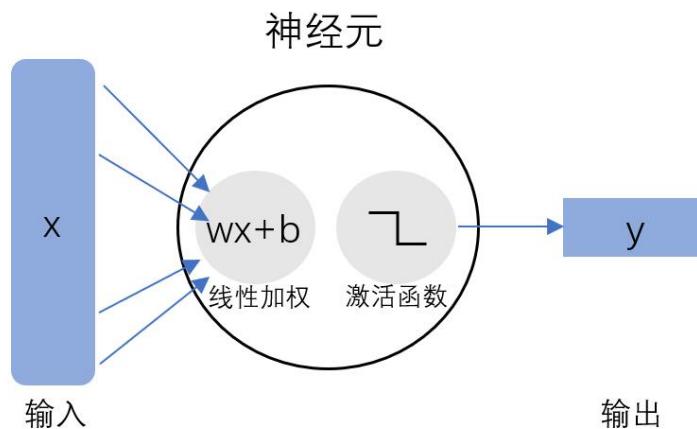


图 2.1 神经元示意图

深度学习使用深度神经网络来解决机器学习任务。神经网络由神经元和连接组成，如图 2.1 所示，一个神经元包含多个输入参数和一个输出参数，输入参数经过加权和计算之后经过一个激活函数为输出，激活函数通常是非线性函数。其数学表达式如下所示：

$$y = \sigma(\vec{w}\vec{x} + b) \quad (2.1)$$

在神经网络中，神经元被组织成层状，同一层的神经元之间没有连接，没有前驱的神经元称为输入神经元，没有后继的神经元称为输出神经元。如果输入神经元和输出神经元之间的层数较大，则称为深度神经网络。深度神经网络没有严格的定义，但一般来说，超过 8 层的被认为是“深”的，现代深度神经网络可以有数百层。神经元之间通过连接连接起来，每个连接都将一个神经元  $i$  的输出传输到另一个神经元  $j$  的输入。每个连接都有一个权值  $w_{ij}$ ，它将与激活次数相乘，从而增加或减少信号，这个权重会在学习过程中进行调整，这个过程叫

做训练。其中，梯度下降法是深度神经网络最常用的训练方法，它是一阶优化方法，通过计算损失函数在参数的梯度，并在梯度的负方向调整参数，其步长与梯度的绝对值成正比，步长与梯度绝对值的比值称为学习速率  $\alpha$ 。计算梯度是梯度下降的关键步骤，它是基于反向传播算法。为了通过反向传播来计算梯度，首先需要通过从输入神经元到输出神经元的前馈传递来计算每一层的激活。这种前向传递也称为推理，推理的输出可以是回归问题中的连续值，也可以是分类问题中的离散值。推理结果可以是正确的，也可以是错误的，这是由损失函数定量地进行衡量的；接下来，计算每个神经元和每个权重参数的损失函数的梯度，根据链式法则，从输出层到输入层迭代计算每层的参数梯度，然后我们用梯度下降  $w_{i,j}^{t+1} = w_{i,j}^t - \alpha \frac{\partial L}{\partial w_{i,j}}$  更新权值。这种前馈、反向传播和权值更新构成了一次训练迭代。训练一个深度神经网络通常需要成百上千次的迭代。例如，在 ImageNet 上训练 ResNet-50 需要 450,450 次迭代。

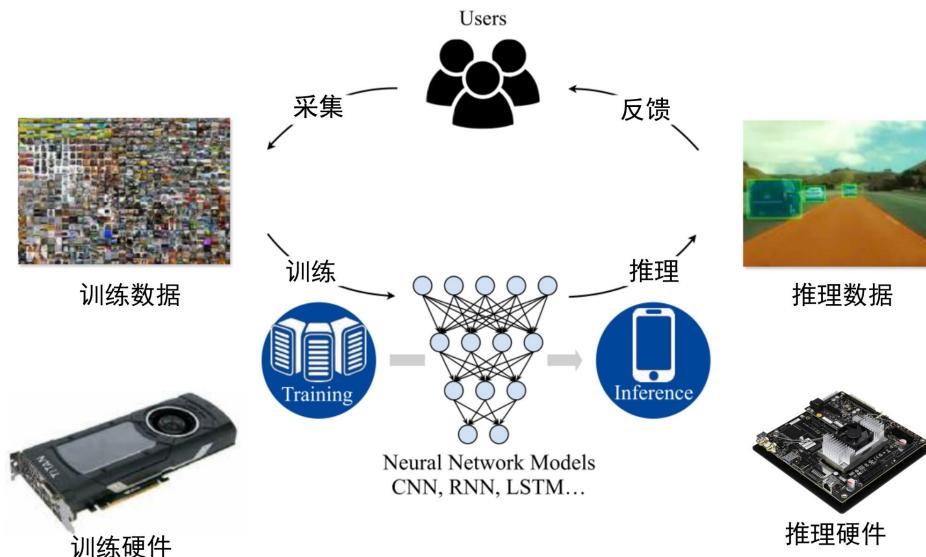


图 2.2 深度神经网络的训练与部署

图 2.2 总结了深度学习训练和部署的整个流程。在图的左边是训练，右边是推理，中间是模型。用户模型、数据模型和神经网络模型之间存在良性循环。更多的用户将采集更多的训练数据（可能是图像、语音、搜索历史或驾驶动作），DNNs 的性能随训练数据量的增加而增加，有了更多的训练数据，我们可以训练更大的模型而不需要过度拟合，从而得到更高的推理精度。更好的准确性将吸引更多的用户，从而产生更多的数据……这是一种正反馈，形成良性循环，硬件在这个循环中扮演着重要的角色。在培训时，高效的硬件可以提高新模型设计的生产率，从而方便深度学习研究者可以快速迭代不同的模型架构。在推理时，高效的硬件可以通过减少延迟和实现实时推理来提高用户体验和降低成本。

### 2.3 卷积神经网络架构

在本节中，我们将概述接下来本文中涉及到的卷积神经网络，卷积神经网络利用了输入信号（如图像）的空间局部性，并在空间中共享权值，这使得它对输入的翻译不变。这种权值共享使得权值的数量比完全连接的层在相同的输入/输出维度下要小得多。因为卷积核可以在不同的地方重用，从硬件的角度来看 CNN 体系结构具有良好的数据局部性。在本文下面的实验中，将使用下面几种目前比较常见的几种卷积神经网络，包括 LeNet<sup>[6]</sup>、AlexNet<sup>[7]</sup>、VGGNet<sup>[8]</sup>、GoogleNet<sup>[9]</sup>、ResNet<sup>[1]</sup> 来对论文中提出的方法进行评估。

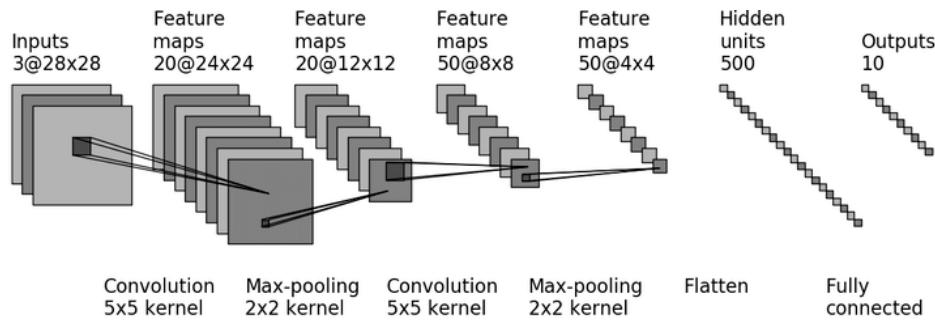


图 2.3 LeNet

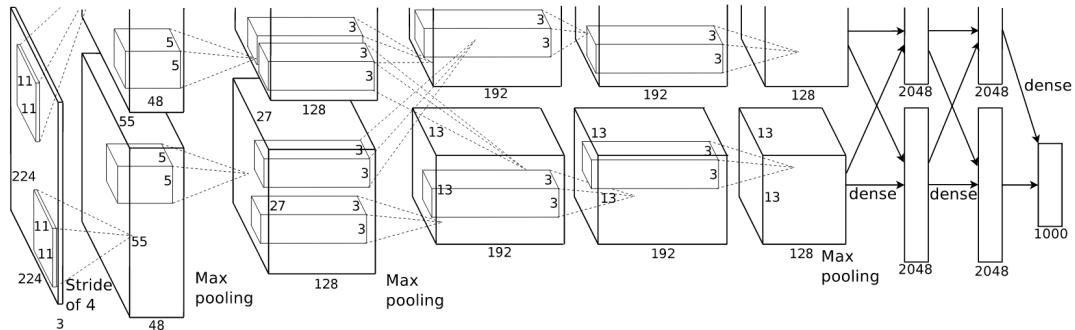


图 2.4 AlexNet

- LeNet: LeNet-5 是一个提出的卷积神经网络，是 Yann LeCun 在 1998 年为手写数字识别而设计的。如图 2.3 所示，它是一个 5 层的卷积神经网络（将卷积核池化当成一个层），包括 1 个输入层，2 个卷积层和 3 个全连接层，两个卷积层的卷积核尺寸都为 5\*5，大约包含 60,000 参数，随着网络越来越深，特征图的尺寸在缩小，与此同时，其通道一直在增加，输入为 32\*32 的手写图片，输出为 0-9 的十个数字类别。
- AlexNet: AlexNet 是 Alex 等人与 2012 年 ImageNet 大赛上提出的，成为 ImageNet 大赛上图像分类的冠军。与以往基于传统的手工特征的图像分类方法相比，它的图片分类识别准确率显提升，在 ImageNet 数据集上获得了 57.2 % 的 Top1 精度和 80.3 % 的 Top5 精度。如图 2.4 所示，AlexNet 有 5

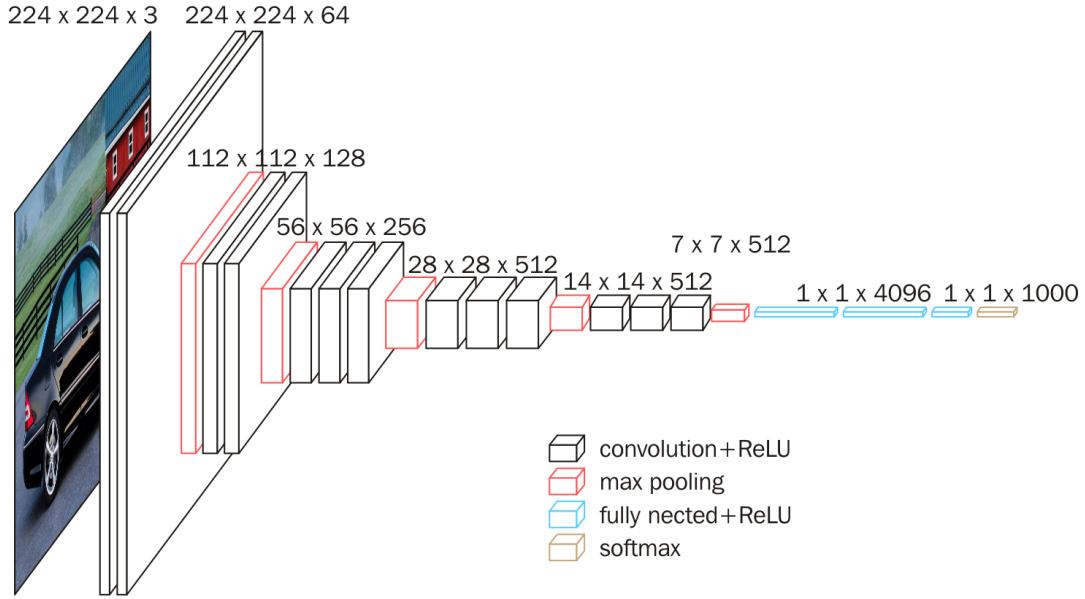


图 2.5 VGG-16

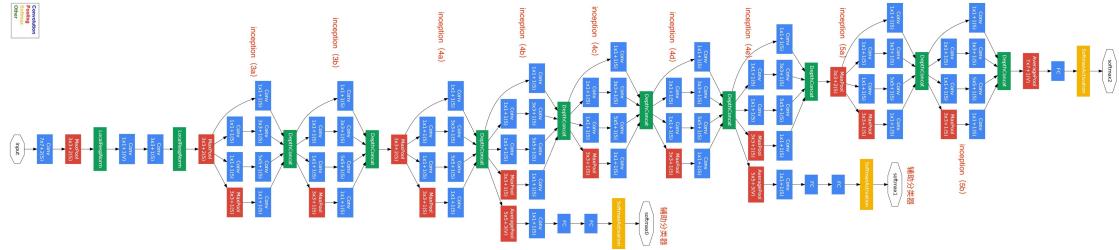


图 2.6 GoogleNet

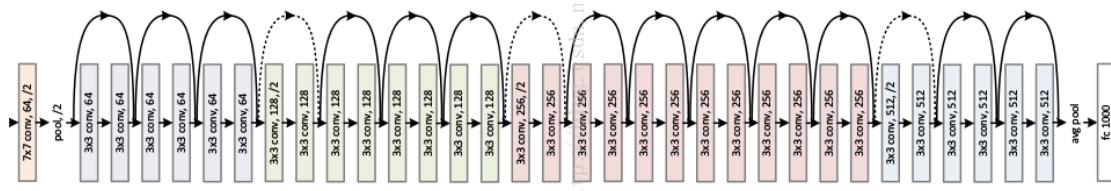


图 2.7 ResNet

一个卷积层（包含三种不同的内核大小:  $11 \times 11$ ,  $5 \times 5$  和  $3 \times 3$ ）和 3 个全连接层，总共包含 6100 万个参数。

- VGGNet: VGGNet 是 ILSVRC 2014 上的相关工作，取得了当年图像定位的冠军，在 ImageNet 数据集上获得了 68.5 % 的 Top1 精度和 88.7 % 的 Top5 精度。其主要工作是证明了增加卷积神经网络的深度能有效提高模型精度，并且多次使用小卷积核可以获得更大的感受野，等效于直接使用大卷积核，如 2 个  $3 \times 3$  卷积核相当于 1 个  $5 \times 5$  卷积核、3 个  $3 \times 3$  卷积核相当于 1 个  $7 \times 7$  卷积核，同时能够有效减少参数。如图 2.5 所示，VGG 包含 5 层卷积结构、3 层全连接层、softmax 输出层，层与层之间使用 max-pooling（最大池化）缩小尺寸，采用 ReLU 激活函数。VGG 有两种结构，分别是 VGG-16 和

VGG-19，两者除了网络深度不一样并没有本质上的区别。

- **GoogleNet:** GoogleNet 也叫 Inception-V1，具有非常高效的网络结构，它只有 700 万个参数，就在 ILSVRC 2014 取得图像分类的冠军，在 ImageNet 数据及上获得了 68.9 % 的 Top1 精度和 89.0 % 的 Top5 精度。如图 2.6 所示，它以 inception 单元为基本结构，具有 9 个 inception 模块，每个 inception 模块由四个分支组成，分别是  $1 \times 1$ 、 $3 \times 3$ 、 $5 \times 5$  卷积和 down-sampling，具有多尺度感受野。inception 结构的主要贡献有两个：一是使用  $1 \times 1$  的卷积来进行升降维；二是在多个尺寸上同时进行卷积再聚合。训练时使用两个辅助损耗层从中间层注入 Loss，防止梯度消失，而在推理时，删除辅助层。
- **ResNet:** ResNet（残差神经网络）是由微软研究院的何恺明等人提出的，在 ILSVRC 2015 中取得了冠军，在 ImageNet 数据及上获得了 76.1 % 的 Top1 精度和 92.9 % 的 Top5 精度。残差神经网络的主要贡献是发现了“退化现象（Degradation）”，并针对退化现象发明了 Shortcut，提出了带有旁路层的残差模块，帮助梯度在层与层之间能够更加容易的传递，解决了深度过大的神经网络训练困难的问题，使得神经网络的“深度”首次突破了 100 层，最大甚至可以超过 1000 层。

## 2.4 数据集

在本节中，将介绍本文实验中使用的不同数据集。本文使用多种数据及进行各种机器学习任务，用以测试提出的模型压缩和优化方案的性能。实验中使用的数据集包括用于图像分类的 MNIST、Cifar-10 和 ImageNet。

- **MNIST:** 如图 2.8 所示，MNIST 数据集 (Mixed National Institute of Standards and Technology database) 是美国国家标准与技术研究院收集整理的大型手写数字数据库 [lecun1998mnist]，用于进行手写数字识别，一共 10 各类表示 10 个数字，图片大小为  $28 \times 28$ ，一共包含 60,000 个示例的训练集以及 10,000 个示例的测试集。这个数据集相对来说比较简单和小，只需要几分钟的时间来训练模型。我们只使用 MNIST 进行原型设计，并使用更大的数据集进一步验证每个想法。
- **CIFAR-10:** 如图 2.9 所示，CIFAR-10 数据集由 60000 个  $32 \times 32$  彩色图像组成，其中有 50000 个训练图像和 10000 个测试图像，包含互不相关的 10 个类<sup>[51]</sup>，每个类都有 6000 个图像示例，。数据集比 MNIST 稍难一些，但模型仍然只需要几个小时的训练。当需要多次重复一组类似的实验时，我们使用 Cifar-10 进行消融研究。
- **ImageNet:** 如图 2.10 所示，ImageNet 是一个针对 ILSVRC 挑战的大规模数



图 2.8 MNIST

据集<sup>[7]</sup>。其中，训练数据集包含 1000 个类别和 120 万幅图像，验证数据集包含 50,000 张图像，每个类 50 张。使用 Top1 和 Top5 的准确率表示分类性能的结果，Top1 准确率衡量的比例正确标记的图像，而 Top5 表示如果五个概率最大的标签中有一个是正确的标签，那么这张图像就被认为是一个正确的标签。我们使用 ImageNet 数据集来测量模型压缩和优化的性能。

## 2.5 实验环境说明

### 2.5.1 深度学习框架说明

直接编程一个多核 CPU 或 GPU 程序是比较困难的，但幸运的是，目前深度学习相关开源框架的生态非常完美，国内外都有多个开源框架可供研究者和开发者使用，光是为人熟知的就有 Caffe，Tensorflow，Pytorch，Keras，Mxnet，NCNN 等，它们将神经网络的计算抽象成一些基本的操作，如卷积，矩阵乘法等，并进行高度优化。用户只需要关注高级神经网络架构而不是低级实现，因此程序员只需要写一个计算描述，然后深度学习框架就可以高效地将计算映射到高性能硬件上。在本文中，主要是通过 Pytorch 对神经网络进行训练和验证，使

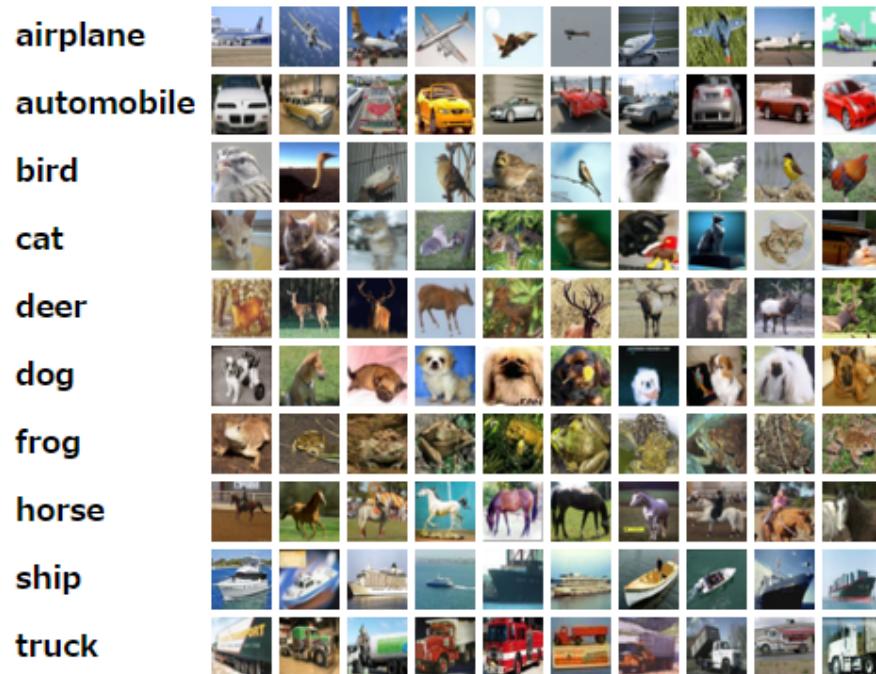


图 2.9 Cifar-10

## ImageNet Dataset



Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). [Imagenet large scale visual recognition challenge](#). *arXiv preprint arXiv:1409.0575*. [\[web\]](#)

3

图 2.10 ImageNet

用 NCNN 和 TensorRT 在嵌入式平台上进行部署和推理验证。

- Pytorch: PyTorch<sup>[52]</sup> 是有 Facebook 推出的一个最新的、更灵活的深度学习框架，专门针对 GPU 加速的深度神经网络（DNN）编程，支持动态图。我们使用 PyTorch 进行了网络模型的训练和压缩优化算法实现。
- NCNN: NCNN<sup>[53]</sup> 是一个为嵌入式设备极致优化的高性能神经网络推理框架，只支持部署和推理。我们使用 NCNN 将网络模型部署到嵌入式设备上查看资源使用情况和性能测试。

- **TensorRT:** TensorRT<sup>[54]</sup> 是 NVIDIA 的一款高性能深度学习推理框架，包含深度学习推理优化器和运行环境，可为深度学习推理应用提供低延迟和高吞吐量。我们使用 NCNN 将网络模型部署到嵌入式设备上查看资源使用情况和性能测试。

## 2.6 模型压缩与优化方法

近年来，深层卷积神经网络（CNN）因为其强大的表示能力，越来越广泛应用于各个领域。并且随着 CNN 的巨大成功，在实际应用中部署深度网络的需求不断增加。但巨大的计算复杂性在实际应用中的 CNN 部署中引入了两个问题。一种是随着计算复杂度的增加，CNN 推理阶段变慢。这使得很难在实时应用程序中部署 CNN。另一个问题是，CNN 固有的密集计算将消耗大量的电池电量，而这仅限于移动设备。同时，CNN 的大量参数会消耗相当多的存储空间和运行时内存，而这在嵌入式设备上是非常有限的。因此，减少 CNN 的存储和计算成本变得至关重要，网络的优化和加速已经成为目前深度学习研究的一个热门话题。

近年来深度神经网络压缩和加速领域的研究取得多方面的进展。根据它们的性质，我们将这些优化方法分为五类：模型修剪、矩阵/张量分解、模型参数量化、高效网络设计。这些方法大多数是相互正交且相互补充的，可以同时应用。比如说，为了达到更高的压缩比，高效卷积网络和参数修剪与量化可以一起部署。同样的，模型量化也可以与低秩分解一起使用，以实现更好的压缩和加速效果。我们将在以下各节中分别描述它们的详细特性以及优点和缺点的分析。

### 2.6.1 模型修剪

早期的剪枝研究最佳脑损伤<sup>[22]</sup> 和最佳脑外科<sup>[21]</sup> 方法基于损失函数的黑森性减少了连接数。他们的研究表明，这种剪枝比基于大小的剪枝（如重量衰减法）具有更高的准确性。

在这个研究方向上，后面的趋势是在预先训练好的 DNN 模型中修剪冗余的、无信息的权重。如 Srinivas 和 Babu<sup>[55]</sup> 对神经元之间的冗余进行了研究，提出了一种去除冗余神经元的无数据剪枝方法。Han 等人<sup>[56]</sup> 提出减少整个网络的参数总数和操作总数。Chen 等人<sup>[30]</sup> 提出了一种哈希网模型，该模型使用低成本哈希函数将权重分组到哈希桶中以共享参数。<sup>[57]</sup> 中的深度压缩方法去除冗余连接并对权重进行量化，然后利用 Huffman 编码对量化后的权重进行编码。在<sup>[58]</sup> 中，提出了一种基于软权重共享的简单正则化方法，将量化和修剪都包含在一个简单的（再）训练过程中。上述剪枝方案通常在 DNNs 中产生连接剪枝。

人们对训练具有稀疏性约束的紧凑 DNN 也越来越感兴趣。这些稀疏性约束通常作为低范数或 L1 范数正则化引入优化问题中。<sup>[59]</sup> 中的工作对卷积滤波器施加了组稀疏性约束，以实现结构性脑损伤，即以组方式对卷积核的条目进行修剪。在<sup>[60]</sup> 中，在训练阶段在神经元上引入了一组稀疏正则化器来学习具有简化滤波器的紧凑 CNN。Wen 等人<sup>[37]</sup> 为了减少冗余的滤波器、通道甚至层，在网络的每一层上增加了一个正则化器来引导结构化稀疏。在层级剪枝中，以上所有的工作都使用了 L2 范数正则化器。<sup>[38]</sup> 中的作品使用 L1-Norm 来选择和修剪不重要的过滤器。

值得注意的是，使用网络修剪的需要注意一些问题。首先，相对来说，使用 L2 正则化来进行剪枝更加难以收敛，往往需要更多次的迭代；此外，所有的修剪条件都需要手动对参数进行微调，设置层的敏感性，对于一些应用程序来说可能会比较麻烦；最后，网络剪枝通常能够减少模型规模，但不能显著提高效率（训练或推理时间）。

### 2.6.2 矩阵/张量分解

利用低秩分解来加速卷积已经有很长一段时间了，例如，高维离散余弦变换（DCT）和由一维 DCT 变换和一维小波分别构造张量积的小波系统。Rigamonti 等人<sup>[61]</sup> 用字典学习的方法介绍了学习可分离 1D 滤波器。对于一些简单的网络模型，在<sup>[23]</sup> 中使用了一些卷积核的低秩近似和聚类方案仅仅付出分类精度下降 1 % 的微小代价就实现单个卷积层上的 2 倍加速。<sup>[62]</sup> 的工作提出了使用不同的张量分解方案，在文本识别的工作中仅仅损失 1 % 的准确性的前提下得到了 4.5× 的速度提升。

经典的压缩 3D 卷积层的方法通常是低秩近似逐层进行，完成后固定一层的参数，并根据重构误差准则对以上各层进行微调，如图 2.11 所示。在此基础上，<sup>[63]</sup> 使用非线性最小二乘来计算 CP 分解对核张量进行了正则多元分解。<sup>[64]</sup> 则提出了一种新的低秩张量分解算法，它使用批处理标准化（BN）来转换内部隐藏单元的激活，用于从零开始训练低秩约束 CNN。一般来说，<sup>[64]</sup>（BN 低秩）中的 CP 和 BN 分解方案都可以从零开始训练 CNN。然而，它们之间几乎没有区别。例如，在 CP 分解中寻找最佳低秩近似是一个不适定问题，因为最佳 K 秩数近似有时可能不存在。而对于 BN 格式，分解总是存在的。如表 2.1 所示，本文对这两种方法进行了简单比较，通过实际的加速比和压缩率来衡量他们的性能。

值得注意的是，虽然基于低秩近似的矩阵分解方法可以直接用于模型压缩和加速，但是因为它涉及到分解操作，具有较高的计算复杂度，并不容易有高效实现；同时，鉴于不同的层包含不同的信息不能进行全局参数压缩，目前的方法是逐层进行低秩近似；最后，因式分解后得模型需要较多的重训以达到与原始模

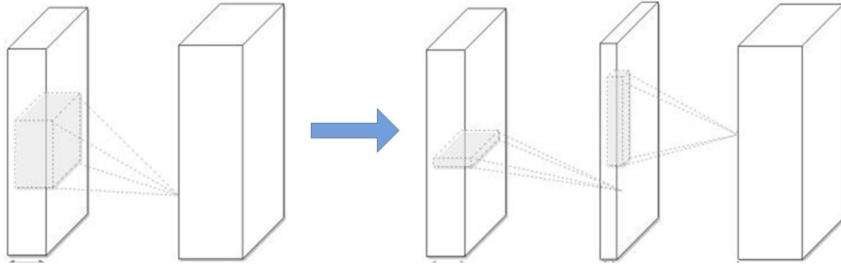


图 2.11 低秩正则化方法的一个典型框架。左边是原卷积层，右边是秩为  $k$  的低秩约束卷积层

表 2.1 ILSVRC-2012 上不同低秩模型及其基线的比较

模型	Top-5 准确率	加速比	压缩率
AlexNet	80.03%	1.	1.
BN Low-rank	80.56%	1.09	4.94
CP Low-rank	79.66%	1.82	5.
VGG-16	90.60%	1.	1.
BN Low-rank	90.47%	1.53	2.72
CP Low-rank	90.31%	2.05	2.75
GoogleNet	92.21%	1.	1.
BN Low-rank	91.88%	1.08	2.79
CP Low-rank	91.79%	1.20	2.84

型相比的准确度。

### 2.6.3 参数量化

网络参数量化通过减少表示每个权值所需的比特数来压缩原始网络。Gong 等人的<sup>[29]</sup> 和 Wu 等人的<sup>[65]</sup> 对参数值进行 k-means 标量量化。Vanhoucke 等人<sup>[26]</sup> 表明，参数的 8 位量化可以在精度损失最小的情况下导致显著的加速。<sup>[66]</sup> 在基于随机四舍五入的 CNN 训练中使用了 16 位定点表示，显著减少了内存使用量和浮点运算，分类精度几乎没有损失。

<sup>[57]</sup> 中提出的方法采用权值共享的方式对连接权值进行量化，然后对量化后的权值和码本进行 Huffman 编码，进一步降低速率。如图 2.12 所示，它首先通过正常的网络训练学习连接，然后修剪小权重连接。最后，对网络进行再训练微调，以恢复网络模型的能力与精度。在所有基于量化的方法中，这项工作达到了最先进的性能。在<sup>[67]</sup> 中，证明了使用 Hessian 权值可以度量网络参数的重要性，并提出了最小化 Hessian 权值对聚类参数的平均量化误差。

量化是一种非常有效的模型压缩和加速方法。在极端情况下，甚至可以使

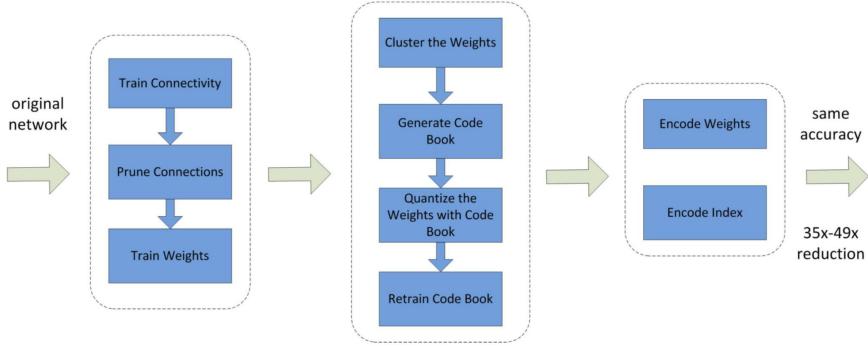


图 2.12 Han 提出的三阶段压缩方法: 剪枝、量化和 huffman 编码。输入是原始模型, 输出是压缩后的模型。

用 1 位表示每个权重的值, 即二进制权值神经网络, 采用 01 来表示模型的权重参数和激活值。有几种方法可以直接用二进制权值训练卷积神经网络, 例如 BinaryConnect<sup>[68]</sup>, BinaryNet<sup>[69]</sup> 和 XNOR<sup>[45]</sup>。<sup>[70]</sup> 的一项系统研究表明, 使用反向传播训练可以恢复量化网络(包括二进制权值)的性能。

但是, 二进制网络的准确性在面对 GoogleNet 这样的大型网络时显著降低。同时, 现有的二值化方案基于简单的矩阵近似, 表达能力不够, 精度的损失较大。为了解决这个问题, 在<sup>[71]</sup>的工作提出了一个近牛顿算法与对角 Hessian 近似, 直接最大限度地减少与二进制权值有关的损失。<sup>[72]</sup> 通过随机二值化权值, 将隐藏状态计算中的乘法转化为显著变化, 减少了训练阶段浮点乘法运算的时间。Zhao 等人提出了半波高斯量化来学习低精度网络, 取得了令人满意的结果。

#### 2.6.4 高效网络设计

卷积网络加速和压缩的目标是优化给定深度神经网络的执行和存储框架, 对网络加速和压缩的另一个平行探索是设计更高效但低成本的网络架构本身。

在<sup>[25]</sup>中, 作者提出了 NIN 架构, 使用  $1 \times 1$  卷积在保持整体计算复杂度较小的前提下增加网络容量, 为了减少 CNN 模型的存储需求, 他们还提出去除完全连接层, 并使用全局平均池。这些策略也被许多最先进的 CNN 模型使用, 如 GoogleNet<sup>[9]</sup> 和 ResNet<sup>[1]</sup>。分组卷积是降低网络复杂度的另一种常用策略, GoogleNet<sup>[9]</sup> 的研究中也探讨了这一点。<sup>[73]</sup> 中提出的 SqueezeNet 通过大量使用  $1 \times 1$  卷积和分支策略, 在于 AlexNet 精度相当的前提下实现了约 50 倍的参数压缩。通过分支结构,<sup>[74]</sup> 的 ResNeXt 工作可以在相同的计算预算下获得比 ResNet 更高的精度。MobileNet<sup>[75]</sup> 中提出的深度可分离卷积将分组卷积发挥到极致, 即分组的数量等于输入/输出通道的数量, 由此产生的 MobileNet 可以比 VGG16 模型小 32 倍, 快 27 倍, 它们在 ImageNet 上具有相似的图像分类精度。文献<sup>[76]</sup> 中提出的 shuffle 网引入了信道 shuffle 操作, 增加了多组内的信息变化, 显著提高

了网络的表示能力。他们的方法在 AlexNet 上实现了大约 13 倍的实际加速，而且精度相当。

然而手动设计高效模型仍然是一项挑战，因为要考虑太多因素且需要研究人员大量的相关经验和知识。因此另一个研究致力于使用启发式和基于规则的策略进行自动的网络架构搜索<sup>[77]</sup>。

### 2.6.5 知识蒸馏

最早利用知识转移 (knowledge transfer, KT) 来压缩模型的是由 Caruana 等人<sup>[78]</sup> 提出的，他们使用一个大网络的输出来训练了一个带有伪数据标记的强分类器压缩/集成模型，但这项工作仅限于肤浅的模型。这个想法最近在<sup>[79]</sup> 中被采用，将使用复杂模型指导和训练简单模型的方法乘坐知识蒸馏 (knowledge distillation, KD)，基于 KD 方法的主要思想是通过柔性标签分布输出，将知识从大型教师网络模型转换为小型学生网络模型。

[80] 的工作提出了 KD 压缩框架，它通过遵循学生-教师范式来简化深度网络的训练，在这种范式中，学生根据教师输出的柔性标签进行惩罚。该框架将教师网络压缩成深度相似的学生网络。训练学生预测输出和分类标签。尽管它很简单，KD 在各种图像分类任务中展示了有希望的结果。<sup>[81]</sup> 的提出了一种利用深度神经网络来解决网络压缩问题的方法，称为 FitNets，通过训练一个薄而深的网络来压缩宽而浅 (但仍然深) 的网络。该方法扩展了思想，允许更薄和更深的学生模型。为了从教师网络的中间表示中学习，FitNet 让学生模拟教师的全部特征图。然而，这样的假设太严格了，因为老师和学生的能力可能有很大的不同。

沿着这个方向，蒸馏知识有几个扩展。<sup>[82]</sup> 的工作是通过蒙特卡洛近似法来用一个预训练的教师模型训练学生模型。所提出的框架使用在线训练，并使用深度神经网络的学生模型。不同于以往的用柔标概率表示知识的工作，<sup>[83]</sup> 是利用较高隐层的神经元来表示知识，它保留了与标签概率一样多的信息，但更加紧凑。<sup>[84]</sup> 的工作通过瞬间将知识从以前的网络转移到每一个新的更深或更宽的网络，加速了实验过程。这些技术是基于神经网络规范之间的函数保持变换的概念。Zagoruyko 等人<sup>[85]</sup> 提出注意转移 (AT) 来放宽 FitNet 的假设。

值得注意的是，基于 KD 的方法虽然可以使更深层次的模型变得更浅，并帮助显著降低计算成本，但是也有一些缺点。其中之一是 KD 的应用有局限性，只能应用于以 softmax 为损失函数的任务；另一个缺点是，与其他类型的方法相比，基于 KD 的方法的压缩性能并不突出。

## 2.7 本章小结

在本章，我们介绍了论文的实验环境，也重点介绍了深度学习与模型压缩加速的基本理论知识。首先引入了机器学习领域中发展最为迅速的研究方向——深度学习，介绍其原理；然后从五个方面介绍了深度神经网络的压缩与优化方法，并对其优点与缺点进行了讨论和分析。本章旨在为读者提供一个本文工作的背景知识，为读者建立对深度学习和卷积神经网络的整体概念，同时对深度神经网络模型压缩与优化领域的研究做个基础的理论铺垫，以便读者更好地理解后面章节的研究和实验工作。

## 第3章 任务感知的动态子网络交换

### 3.1 引言

在深度卷积网络的压缩与优化领域中，减小网络参数所占用的内存空间是一个关键的问题和优化目标。正如前文所提到的，深度卷积神经网络的巨大的参数量不仅仅让设备间的存储和共享变得困难，并且在网络进行推理的时候会占用大量的内存，同时会产生大量访存和计算，使得推理速度和推理能耗变得不能接受。因此，卷积神经网络的模型参数大小成为了制约其应用拓展的一个巨大的瓶颈，也是卷积神经网络压缩与优化问题的一个关键点。

近年来，为了提高深度神经网络 (DNNs) 的推理效率，人们在降低内存和计算成本方面做了大量的工作。现有的方法包括网络剪枝 [1]、[2]、轻量级架构设计 [3]、[4] 和自适应推理 [5]。其中，网络剪枝和自适应推理因其有效性而受到越来越多的关注。

信道剪枝以一种粗粒度的方式从网络中去除不太重要的信道，从而产生小而密集的模型。尽管已经有许多修剪技术来减少深度学习模型的尺寸，但因为修剪会降低模型的精度，在内存约束的嵌入式设备中部署一个修剪的深度神经网络是非常具有挑战性的。例如，具有 1.96 % 尺寸缩减的修剪模型的 AlexNet 模型在使用 ImageNet[6] 时的精度损失为 0.97 %。He 等 [2] 对信道进行了基于最大保留权值的结构化剪枝。通过对不重要的连接进行修剪，Han 等人的 [1] 极大地减少了 AlexNet 的参数数量，而非结构化修剪的精度损失很小。基于剪枝的解决方案虽然可以减少 DNN 的内存占用，但在精度要求下所节省的内存资源是有限的。为了保证得到理想的推理结果，不允许对网络进行过度修剪。最近，一种用于在有限 GPU 全局内存 [7] 上运行大模型的交换技术被提出。然而，这种方法依赖于大量的 CPU DRAM，这在边缘嵌入式设备中是不可用的。

在物联网 (IoT) 时代，图像分类和识别是边缘计算环境的常见任务，如工厂自动化管道、高速公路和电梯内部等。在嵌入式设备中部署深度学习模型，可以在边缘分析第一手数据，实现广泛的人工智能应用。嵌入式设备的内存通常比较小，这对网络参数的大小提出了更高的要求。为了部署深度学习推理模型，满足嵌入式设备的资源约束是最具挑战性的问题之一。以往的研究建议设计专用硬件，使用专门为深度神经网络优化的具有高能效和高性能的专用硬件平台来进行部署与推理 [9], [10]。然而，这些优化需要特殊的硬件比如专用 FPGA 或 ASIC，这将模型部署限制在某些硬件平台上。最近的工作旨在在具有 DRAM 约束的硬件设备上部署和优化深度学习推理 [11], [12], Lin[13] 等人提出联合设计神经网络体系结构和深度学习推理库，使基于微处理器的物联网设备上的视

觉和音频唤醒词任务成为可能。但是，上述的工作仍然受到 DRAM 不足的瓶颈，只适用于深度学习功能有限的简单任务。

在保持网络性能的前提下，持续缩小网络参数并不是一个容易的目标。因此，退而求其次，追求网络运行时内存开销也是一个优化的方向。因为卷及网络结构的特性，计算总是一层一层的进行推进，不少嵌入式推理框架都会采用懒加载的方式放弃层层推理中产生的中间结果以降低运行时内存开销。

传统的交换技术通过将冷数据交换到外部存储器(例如 NAND 闪存设备)并按需获取所需的数据，从逻辑上扩展了 DRAM 空间 [14]、[15]。然而，在当前的交换技术下，简单地传输运行时数据可能会在发生缓存丢失时产生 I/O 损失，因为在 DRAM 中丢失的数据必须从闪存中读取，并且在推理过程中产生不可接受的 I/O 时间。最近的一项研究提议通过在图形处理单元 [16] 中交换主机 CPU 内存来扩展 GPU 内存。针对深度学习训练，提出了 Capuchin[7]，通过 GPU 和 CPU 空间之间的张量提取/预取来减少内存占用。然而，这些研究只关注内存空间之间的数据交换，很少注意在推理过程中适应嵌入式系统的内存约束。由于 DRAM 丢失造成巨大 I/O 损失，很少有工作致力于集成 DRAM 和闪存之间的交换解决方案。

综上所述，针对减小网络推理运行时内存开销也是一个可以工作的方向，旨在减少目标检测推理过程中对主存空间的需求，以满足在资源受限的嵌入式设备上部署深度学习应用。在这一思路的基础上，从神经网络卷积核通道激活和输入数据的相关性出发，以在运行时动态跳过无关卷积计算为目的，而不是静态的对网络进行减枝和轻量化，提出了基于任务感知的动态子网络算法，同时辅以任务感知的子网络交换，减小运行时网络参数冗余，以达到降低运行时内存开销的目的。同时本方法是正交于现有的模型优化方案，它可以作为现有解决方案的附加模块来实现。

鉴于本研究是基于动态网络理论开展的，本章首先介绍网络压缩与优化算法中的动态网络，之后进行卷积神经网络中卷积核通道激活特性的分析，进而引出本章提出的模型压缩方法，之后介绍任务感知的子网络训练方法和子网络切换策略，进行实验结果的展示和分析，最后对本章进行总结。

## 3.2 动态网络

深度神经网络已经广泛应用在计算机视觉、自然语言处理等领域，并且取得了较大的成功。近年来，越来越强大、越来越高效的神经网络模型设计(如 AlexNet[1]，VGG[2]，GoogleNet[3]，ResNet[4]，DenseNet[5] 以及 SENet[] 等)层出不穷，而近几年开始发展的 NAS 网络架构搜索技术 [7],[8] 也在不断寻找强大

高效的网络结构。然而，大多数当前流行的深度网络是静态的，具有相同的静态推理范式：一旦模型完成训练，网络的结构和参数在测试阶段都始终保持不变，这一定程度上限制了网络模型的表征能力、推理效率和可解释性 [9],[10],[11],[12]。

如前所述，动态网络则可以在推理阶段根据输入样本的不同，自适应地调节自身的结构或参数，从而拥有诸多静态网络无法享有的良好特性。动态网络的优势有：

- 1) 推理效率：很多动态网络可以通过选择性地激活模型中的模块（如网络层 [10]，卷积通道 [13] 或子网络 [14]）来实现计算资源的按需分配，从而在更容易识别的样本上或信息量更少的区域位置（时间或空间）上节约冗余计算，提升推理效率。
  - 2) 表达能力：通过动态地调节网络的结构或参数，网络模型可以拥有更大的参数空间以及更强的表达能力。例如，通过基于特征的动态集成多组卷积层权重 [11],[15]，可以在只增加极小计算开销的前提下显著提升模型参数容量。同时，动态网络可以集成常用的注意力机制，动态的评估因为输入的不同，根据不同通道 [16]、空间区域 [17] 和时间位置 [18] 在推理阶段不同响应，对参数进行动态地重新加权。
  - 3) 自适应性：相较于静态网络的固定计算量，动态网络可以在不同环境下，如不同的硬件平台，实现模型精度与效率的动态平衡。
  - 4) 兼容性：动态网络并非“另起炉灶”，而是正交于现有的模型优化方案，可以与现有模型优化的其他先进技术相兼容，从而可以借助这些先进技术达到 SOTA(技术发展水平，state of the art) 性能。动态网络可以直接基于现有轻量化模型进行构建 [19]，也可以利用 NAS 技术 [7],[8]。另外，动态网络也可以利用模型剪枝技术 [20] 和模型量化技术 [21] 等进一步提升运算效率。
- 通用性：动态网络已经被应用于多任务中，其中有图像分类 [10],[22]、目标检测 [23]、语义分割 [24] 等。另外，许多视觉任务中发展起来的技术可以被拓展至 NLP 领域 [25],[26]，反之亦然。
  - 可解释性：利用动态网络，人们可以分析在处理一个样本时，网络激活了哪些模块 [22]，也可以分析输入中的哪些部分对最终的预测起决定性作用 [29]。有研究指出了大脑处理信息的动态机制 [27],[28]，未来关于动态网络的研究可能会帮助人们更好的探索深度网络模型和大脑的内在的运行机制之间。

### 3.3 卷积层通道激活特性的分析

以前关于通道减枝的研究 [10]，根据评估卷积核通道显著性来对通道进行修剪，并从不重要的通道中删除所有输入和输出连接从而生成更小的密集模型。说明了通道对于不同数据的响应各不相同，其显著性不是静态的，而是和输入数据有一定的关系。基于此，我们对卷积层通道激活特性进行分析，力图发现相关规律。

然而，基于显著性的剪枝方法有三个缺点。首先，通过删除频道，CNN的能力将永久丧失，而对于被删除频道负责的困难输入，由此产生的CNN可能永远无法恢复其准确性。其次，尽管信道剪裁可能会大幅缩小模型尺寸，但如果设计不精，一个CNN中不能有效地减少计算资源而不损害其准确性。最后，神经元的显著性不是静态的，这可以通过图1的特征可视化来说明。这里，CNN显示一组输入图像，卷积输出中的某些通道神经元可能会非常兴奋，而另一组图像从相同的通道中却几乎没有反应。

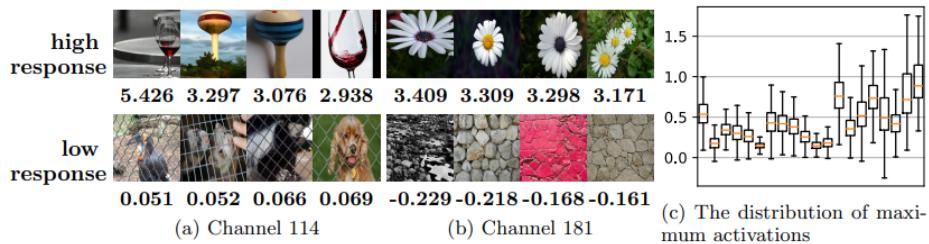


图 3.1 ResNet18 模型在 ImageNet 图片上部分卷积核的通道响应情况 [10]

图 3.1 为预训练的 ResNet18 模型在 ImageNet 验证数据集上的通道响应输出情况，其中 (a) 和 (b) 中最上面的一行分别对 3b/conv2 层块的 114 和 181 通道响应情况，(c) 显示了前 20 个通道中观察到的最大激活的分布，每个图像下面的数字表示在通道添加快捷方式和激活前观察到的最大值。如图所示，卷积核之间的通道数输出出现了很大差异，某些通道的神经元可能会非常兴奋，而另一组图像从相同的通道中却几乎没有反应。

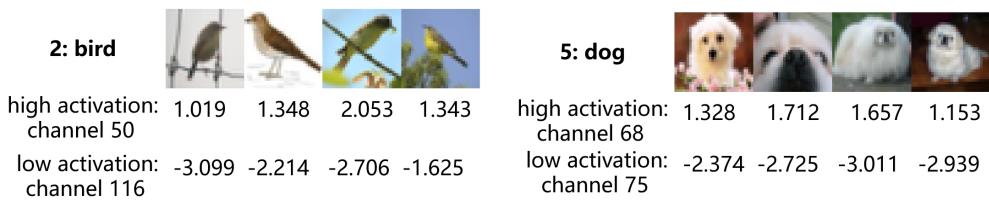


图 3.2 ResNet18 模型在 CIFAR-10 图片上部分卷积核的通道响应情况

如图 3.2 所示，在 ResNet-18 预训练模型的基础上，我们对 CIFAR-10 的每个图像类别进行不同模型层的激活强度评估。在 CIFAR-10 中，我们发现层块

2a/conv2 的输出在不同的图像类上差异很大，而在相似的图像上输出更一致。如图 1 所示。例如，鸟类的图像极大地激活了第 50 通道的神经元，而在第 116 通道却很少引起激活，而狗的图像在第 68 通道的激活程度很高，在第 75 通道的激活程度较低。

### 3.4 任务感知的子网络划分

我们发现相似的图像具有相似的通道显著性(见图 3.2)，进一步的，我们将激活强度按通道的顺序排列成向量，使用向量  $\vec{v}_c$  来表示  $c$  类通道的显著性，其中  $a_{ij}$  为  $c$  类在第  $i$  个卷积层和第  $j$  个通道的平均激活值， $N$  为卷积层数， $C_i$  为第  $i$  个卷积层。如此  $\vec{v}_c$  表示如下：

$$\vec{v}_c = [a_{00}, a_{01}, a_{02}, \dots, a_{NC_{i-1}}, a_{NC_i}] \quad (3.1)$$

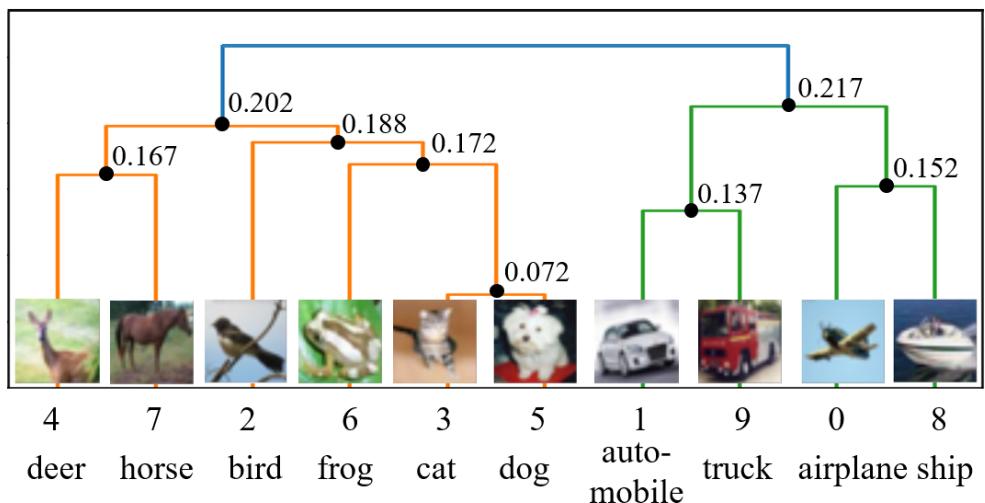


图 3.3 对 CIFAR-10 的 10 个图像类的聚类结果。结点上的数字表示两个分支之间的余弦距离，横轴表示它们的类数。

为了探究通道激活和输入数据的特性，我们对不同类别的  $\vec{v}_c$  进行聚类，在本文中，我们使用余弦距离来表示不同  $\vec{v}_c$  之间的相似性距离，通过层次聚类来得到图像类之间的关系。图 3.3 为 ResNet-18 网络和 CIFAR-10 数据集上的聚类结果。如图 3.3 所示，第 1、9、0、8 类图像对于 ResNet-18 模型的每一层都具有类似的激活强度。换句话说，ResNet-18 认为这些图像类是非常相似的图像。因此，可以使用动态网络将根据聚类结果划分子网，同一个子网对这四个类进行后续的高维特征提取和分类。

根据聚类结果，可将其划分为 2 个子任务。为了获得一个简单和有效的子网分类器，我们推荐更少的子任务划分。每个子网单独负责其对应的数据的推理工

作，此时其他子网参数并不参与计算。如图3.4所示，为子网络工作说明，根据输入类别从整个网络中推导出相应的子网，并动态选择关键子网进行推理，而不是使用整个网络。

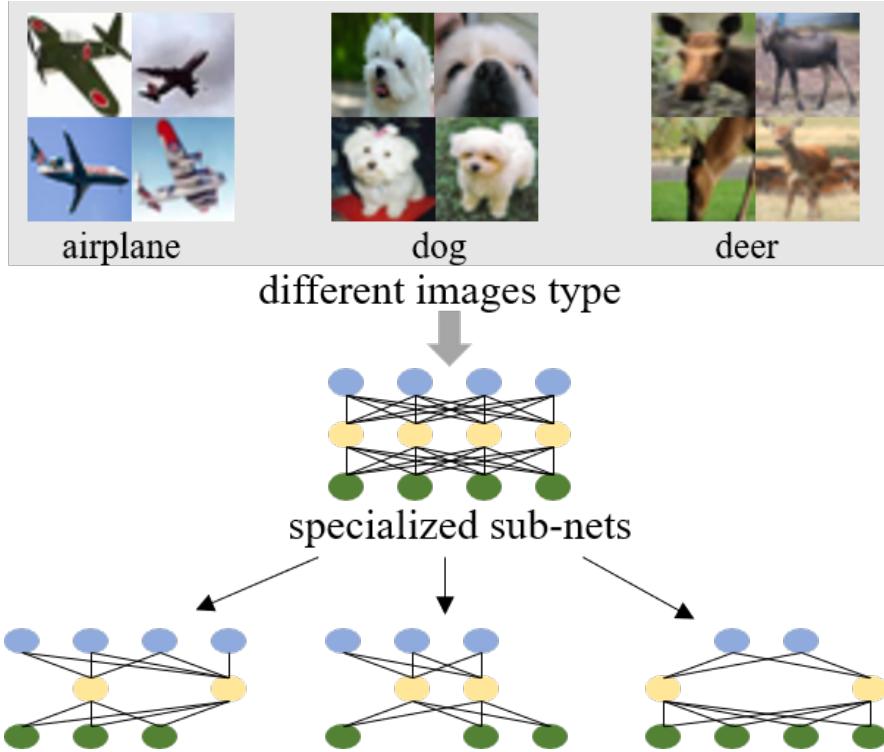


图3.4 子网络和相应图像类的说明。根据输入类别从整个网络中推导出相应的子网，并动态选择关键子网进行推理，而不是使用整个网络。

### 3.5 通道选择因子

子任务划分完成后，将创建子网络，子网络的参数由整个网络共享。由于通道激活值在不同任务类型之间的变化，在一定程度上它也显示了通道的重要性。因此，我们使用集合  $S = \{s_1, s_2, \dots, s_n\}$  描述了整个网络的连接方法，其中  $s$  表示卷积层每个通道的选择因子，其反映了通道的重要性。然后对网络权值和选择因子进行训练，对选择因子进行稀疏正则化。训练后，我们直接将小于选择阈值的  $s$  设为 0。之后，基于  $S$  构建子网。在训练中，损失函数为：

$$Loss_{new} = Loss_{origin} + \lambda \cdot \sum_{s \in S} ||s||_1 \quad (3.2)$$

其中  $Loss_{origin}$  是一个 CNN 的正常训练损失， $|| \cdot ||_{L1}$  是一个 L1 范数的稀疏选择因子， $\lambda$  目的是平衡正则程度。

如图4所示，在卷积层中，每个通道都有一个选择因子。通过在训练过程中对这些因子进行稀疏正则化，以此训练其可以自动识别不重要的通道。再训练

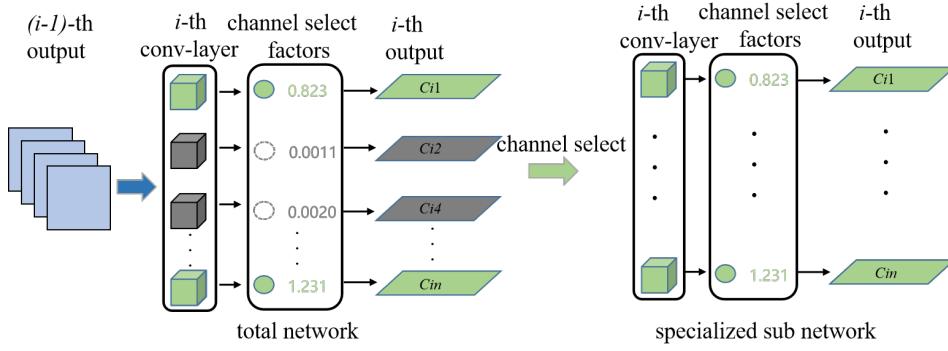


图 3.5 子网络划分过程

中，通过将小选择因子设置为 0，我们获得了一个专门的子网 (右侧)，然后对其进行微调，以达到与整个网络相当的精度。将小的选择因子设置为 0，我们就可以得到一个合适的子网。

在这个过程中，对于每个子网，只添加非常少量的参数用以表示其信道选择，如图 3.7。在部署时，对于非零选择因子，在子网加载时纳入批数层。我们知道批归一化从层 (BN 层) 的计算公式为：

$$z_{out} = \gamma \cdot \frac{z_{in} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \quad (3.3)$$

我们使用  $\gamma'$  and  $\beta'$  替换上式中的  $\gamma$  and  $\beta$ ：

$$\gamma' = s \cdot \gamma; \beta' = s \cdot \beta \quad (3.4)$$

### 3.6 关键子网络激活

通过任务感知的动态子网络，能够根据输出动态的跳过无关的参数计算，从而加快整个网络的推理速度，但是这并不能降低网络的运行时内存需求，甚至因为要存储额外的参数和额外的子网分类器，还会增加少许内存占用。在此，我们提出了一个关键的子网激活方案来减少内存消耗，并协助推理框架更好地做出交换决策。在本研究中，关键子网是指当前运行的推理任务中不可缺少的子网。该方案能够准确地获取关键子网的数据而不是不相关子网。

由于每一组子网有效地工作于特定的任务类型，我们需要在子网激活前预测进入图像的类别。如图3.6(a) 所示，在推理开始使用快速任务分类器，利用之前计算的卷积结果为推理任务选择相应的子网。其中选择子网络之后，需要根据子网络的选择因子参数 (如图 3.7) 来激活相应的子网络参数，假设子任务是通过类的通道显著性特征聚类产生的 (如公式3.1所示)，这些子任务的数量较少，因此只需要一个小规模网络就可以对子任务进行精确的分类。基于子网数量和分

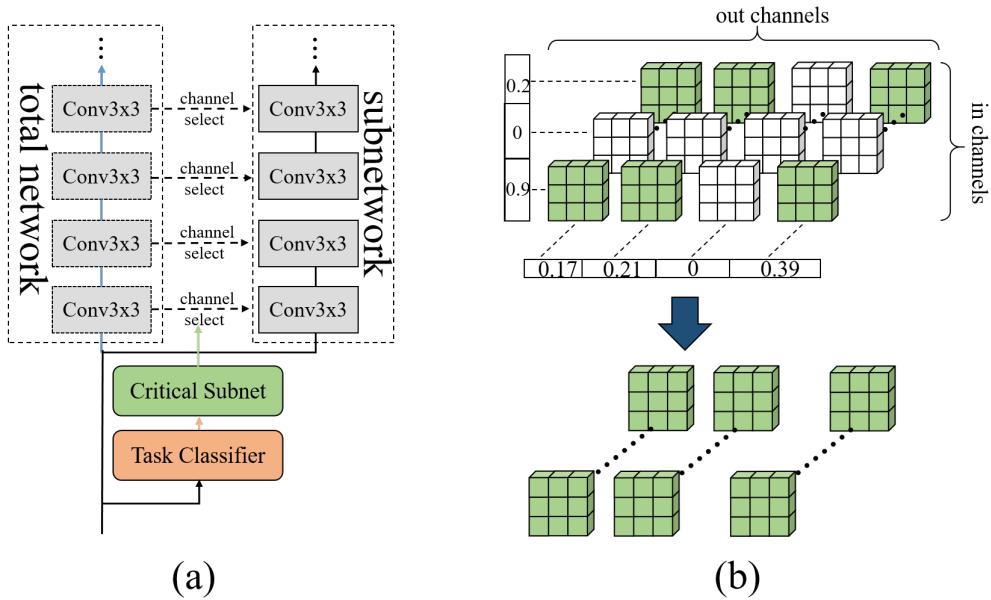


图 3.6 (a) 表示快速图像类识别的任务分类器。它共享整个网络的前两层，识别输入的任务类型，然后通过信道选择因子激活关键子网。(b) 利用通道选择因子选择性加载关键子网参数的方法。

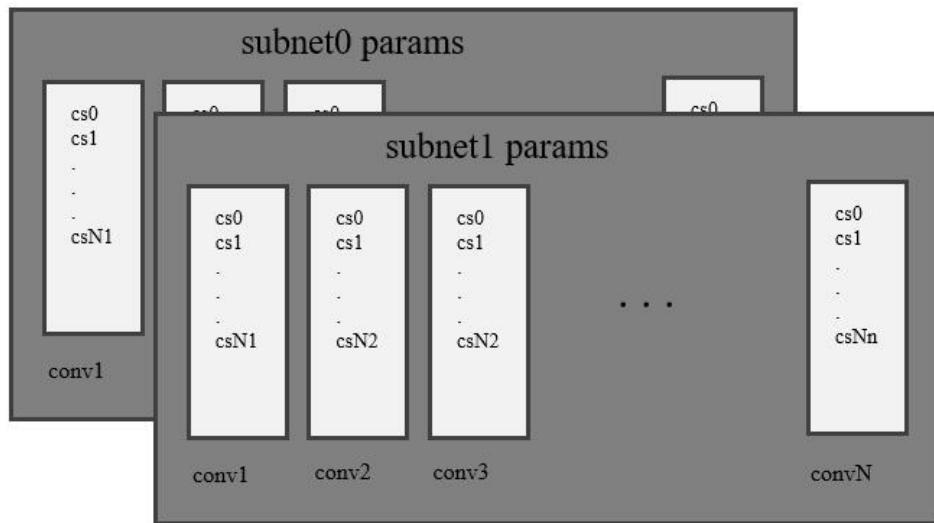


图 3.7 子网络选择因子

类要求，快速任务分类器可以简洁地进行分类，提高分类效率。

图图3.6(b)展示了子网络激活过程。在加载子网参数时，通过信道选择因子选择性地加载当前层的权值参数。这种修剪后的小尺度网络模型可以降低内存消耗和计算成本。注意，与以往的剪枝研究不同，我们没有抛弃任何网络或参数；相反，关键子网的必要参数将被选择性地加载到当前任务中，而剩余的参数可以被激活到以后的任务中。

### 3.7 任务感知的子网络交换

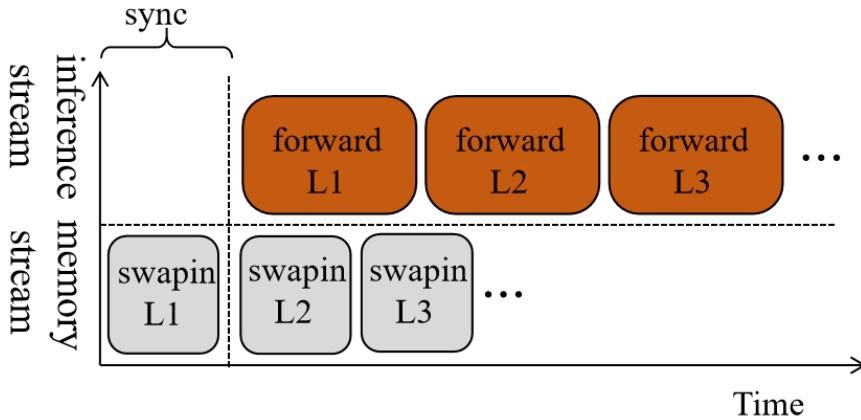


图 3.8 参数的交换方法的说明

在此基础上,我们提出了一种新的任务感知交换 (Task-Aware Swapping, TAS) 算法, 在推理框架层扩展内存空间。其基本思想是将未使用子网的冷数据交换到闪存中, 并确保关键子网的数据在访问之前提前加载于内存之中。为了最小化权重参数加载延迟, 我们使用 pipeline 进行参数加载和推力计算, 不需要等待所有权重参数加载完成, 在第一层加载完成后就可以开始进行推理。

任务感知交换的工作过程如下。首先, 传入的图像被缓冲在内存中。其次, 基于快速任务分类器识别的图像类激活关键子网。读取已调优的子网有助于避免从无关子网中不必要地交换冗余数据。最后, 将关键子网的数据, 如参数和模型, 预取到内存中。如图 3.8 所示, 为了提高交换效率, 模型参数的交换操作可以通过正向推理的执行实现流水线化, 从而隐藏 I/O 开销。正向推理时间通常大于交换时间。在本例中, 数据同步 (sync) 后, 第 1 层 (L1) 和第 2 层 (L2) 的正向推断时间可以与第 2 层 (L2) 和第 3 层 (L3) 的交换时间重叠。使用这种方法, 几乎所有层 (除了第一层) 的交换时间都可以成功地与正向推理时间进行流水线化。这样, 就可以将任务无关参数压缩在闪存中, 从而节省了运行时内存。使用较大的顺序读取来获取这些数据, 可以避免对数据碎片进行昂贵的块读取。注意, 由于推理任务的模型和参数是只读的, 它们在内存中永远不会变脏, 所以交换只需要在推理过程中处理交换操作。

### 3.8 子网方案实现成本

为了实现关键子网络激活, 引入了任务分类器在推理开始时快速识别图像类, 从而产生额外的推理时间。根据不同图像的通道显著性相似度对子任务进行划分。由于嵌入式识别系统的图像类有限, 简单的任务分类器只需要一个小规模

的网络,这限制了时间开销。注意,在极端情况下,如果聚类结果甚至不在图像类之间(如手写数字图像),则不可能进一步划分网络。在这种情况下,只能导出一个子网,该子网将回归到传统的剪枝方法。

## 3.9 实验结果与分析

在本节中,我们在实际的算力较低的嵌入式设备 TX2 上进行了部署推理实验,将我们的 TAS 方法与传统减枝方法进行对比。其中,剪枝率为剪枝后的信道数与总信道数之比。

### 3.9.1 实验设置

在本次实验对比中,我们将所提出的任务感知交换 (TAS) 与原始 DNN 模型和剪枝方法 [20](**Prune**) 进行了比较。注意, TAS 是在 Prune 的基础上实现的, 并在此基础上集成了任务感知的子网交换策略。在实验中, 我们使用 NCNN[21] 作为神经网络推理计算框架。在 CIFAR-10 数据集 [8] 上对 DNN 模型 VGG16BN[22] 进行训练和验证。

#### 1. 硬件平台说明

我们的实验基于 Nvidia TX2 嵌入式平台, 搭载四核 ARM®Cortex®-A57 MP-Core, 8GB 256 位 LPDDR4 内存, 32GB eMMC flash 存储设备, 运行 Ubuntu 18.04。其配置如表 4.3 所示。

表 3.1 Nvidia TX2 配置表

OS	Ubuntu 18.04 bionic
CPU	HMP Dual Denver + Quad ARM A57
GPU	Nvidia Pascal 256 CUDA cores
RAM	8GB 128-bit LPDDR4
DISK	32GB eMMC

### 3.9.2 实验结果

本小节提出推论延迟, 准确性和内存需求的结果。使用 TAS 将网络划分为两个子网 (subnetwork0 和 subnetwork1), 如上文分析。

- 1) 性能与精度: 图 3.11(a) 和 (b) 显示了推理延迟和准确性。使用传统的修剪方法 (**Prune**), 当修剪比率为 0.7 时, 产生的加速效果最高, 相比原模型(修剪比例是 0)推断延迟可以显著减少从 151.5 ms 到 85.4 ms。另一方面, 与修剪比例大于 0.6 时, 模型精度开始快速下降。这表明, 过度修剪可能导

model	dataset	DRAM reduction	delay reduction	subnetwork num.
VGG16BN	CIFAR-10	76.4%	35.9%	2
	SVHN	72.6%	33.4%	3
ResNet18	CIFAR-10	87.4%	24.8%	2
	SVHN	78.5%	22.9%	3
ResNet50	CIFAR-10	89.8%	26.1%	2
	SVHN	79.6%	23.7%	3

图 3.9 不同模型和数据集的 DRAM 和推理延迟减少的结果。

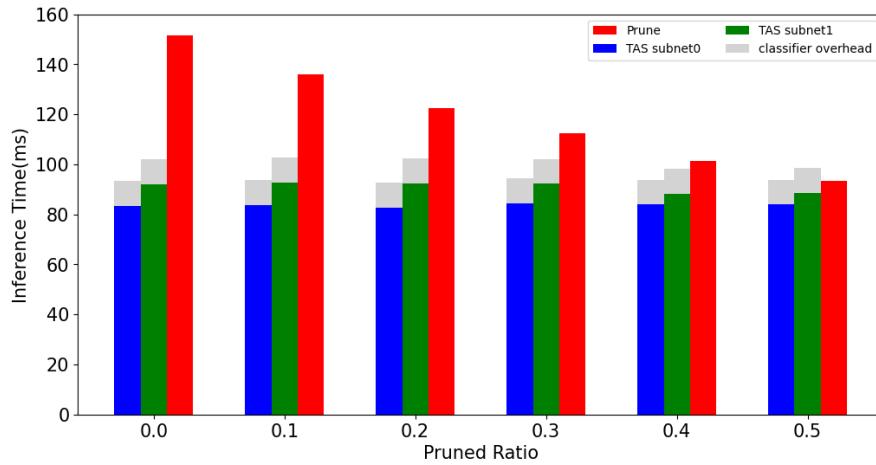


图 3.10 分类器的时间开销和不同修剪程度下的总推理时间。

致明显的精度下降。相比之下，提出的 TAS 方法在不同图像类别的性能和精度方面都优于 Prune。Prune 的精度甚至高于 TAS，因为划分的子网 (subnetwork0 和 subnetwork1) 具有更小的输入图像类和更少的通道，从而消除了无关卷积核产生的干扰噪声。因此，TAS 进一步降低了网络复杂性，提高了推理精度。最重要的是，TAS 保证了全模型结构的精度，同时减少了推理时间。另一方面，由于最小的子网数据读取量和流水线方案的高效率，交换开销只占总推理时间的 0.3 %。由于层的深度通常比较大，例如 VGG16BN 模型有 16 层，所以第一层的加载时间对整体任务的影响有限。

- 2) 内存缩减: 图 3.11(c) 显示了运行时内存消耗。对于 TAS，它总是为 subnetwork0 和 subnetwork1 任务保持较低的内存消耗水平 (大约 20 MB)。例如，当整合到未修剪网络 (修剪比为 0) 时，subnetwork0 和 subnetwork1 任务对 TAS 的 DRAM 消耗分别为 21.7 MB 和 20.7 MB，而未修剪网络的内存消耗则高达 92.0 MB。这是因为 TAS 精确地按照实际推理要求激活了子网，避免了无关网络对内存的不必要使用。注意，当修剪率为 0.5 时，修剪达到了精度要求下的修剪极限。在这一点上，与 Prune 相比，TAS 仍然实现了

34.6% 的 DRAM 减少。图 3.9 为不同 DNN 模型下 TAS 降低 DRAM 和推理延迟的结果。结果表明，在评估的数据集下，由于优化的临界子网有效地加快了推理过程，并最大限度地减少了 DRAM 消耗，因此 TAS 优于所有 DNN 模型。

- 3) 快速任务分类的开销：为了快速识别输入图像类，我们引入了一种快速任务分类器。图 3.10 显示了不同修剪程度下分类器的时间开销和总推理时间。对于 subnetwork0 任务，分类器的推理时间平均为总推理时间的 10.6%。在 subnetwork1 任务中也观察到了类似的结果。原因在于只有两个子网来激活粗粒度分类，而粗粒度分类只需要一个小规模的网络来进行分类。结果表明，分类器的推理时间相对于整体推理时间来说相对较小。

### 3.10 本章小结

借助动态网络的思想和运行时内存交换技术，本章提出了任务感知的子网络切换算法，解决了在内存约束的嵌入式设备上部署 DNN 模型时内存不足的问题。首先，它通过考虑用于识别不同图像类的推理任务的特征来识别和激活关键子网。其次，提出了一种任务感知交换算法来获取关键子网络的必要数据。本研究是对现有网络剪枝方法的补充。与传统的剪枝方法相比，该方法可以进一步减少运行时推理任务的内存需求，而不影响其准确性。

本方法的出发点是卷积神经网络中卷积核通道激活情况与输入数据强相关性，这是网络中存在大量冗余参数的体现，通过动态网络的思想为不同类的数据训练一个专属子网，并在运行时根据输入数据动态切换子网进行推理，可以有效消除这种冗余，加快推理速度，并有效减少运行时内存需求。

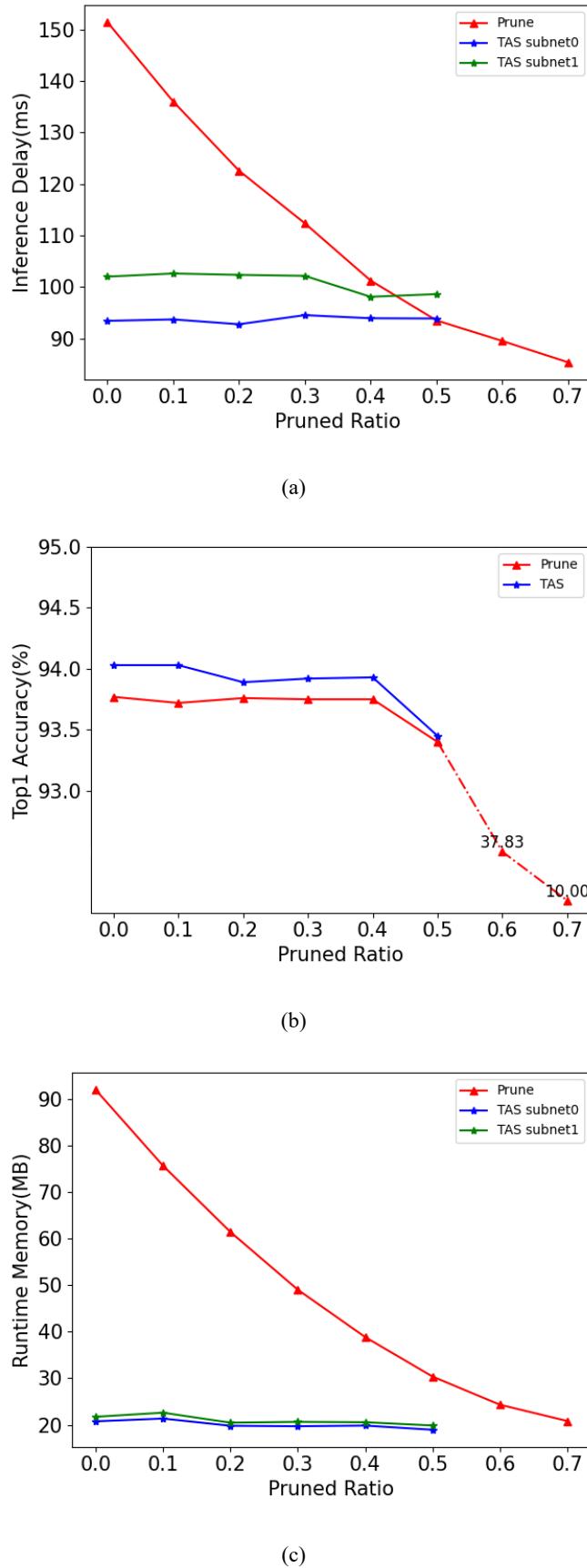


图 3.11 (a) 推理延迟 (b) 推理准确性 (c) 运行时内存消耗的结果。

## 第4章 针对残差结构的分支融合方法

### 4.1 引言

近年来，深度学习快速发展，已在许多任务中达到了优异的性能表现，使其越来越多的应用在众多生活与工业领域。但这些深度神经网络模型优异的性能都依赖于具有数百万甚至数十亿个参数的深度网络模型，需要消耗大量的计算资源和存储资源，需要高性能的硬件作为支撑，这对实际部署、运行和应用的环境提出了较高的要求。因此，深度神经网络的压缩与优化一直都是一个非常关键的问题，特别是对于计算资源有限的嵌入式移动设备来说，如此高的计算资源要求是不可接受的。另一个关键的挑战是能源消耗：由于移动设备的电池有限，繁重的计算会很快耗尽电池。然而大多数的模型压缩算法与高效的网络结构对于计算硬件并不友好，虽然能够显著模型参数，但是并没有很好的利用硬件性能，达到理论的上的加速比。

在任意深度学习的应用场景落地一个模型/算法时，需要经历两个基本步骤：

- 1) 根据目标数据和任务进行模型训练的训练步骤；
- 2) 将模型和应用部署到目标设备上执行服务的推理步骤。

训练步骤目前基本由 Tensorflow<sup>[86]</sup>、PyTorch<sup>[52]</sup>、MXNet<sup>[87]</sup>等主流框架主导，但是在部署步骤，基本是处在“百家争鸣”的状态，不同的部署框架对于不同的硬件架构有着专门的手动优化，在相应平台能够实现更快的运行速度，比如 NCNN<sup>[53]</sup> 针对手机移动端做了专门的优化，TensorRT<sup>[54]</sup> 针对 NVIDIA GPU 架构，特别是其嵌入式深度学习设备进行专门的优化和部署，运行效率和资源消耗都远远低于 Pytorch 等通用框架，同时高通和苹果也都发布了针对自家平台优化的深度学习部署框架。

一般而言，模型训练要求模型精度，训练速度；而模型部署则要求模型在部署端的硬件高效性、内存高效性和运行效率。目前大多数的模型压缩与轻量化研究，力图用更少的参数和更快的计算得到与大规模网络模型一致或差不多的模型精度，并没有考虑到模型训练时和部署时的差异性。目前大多数模型优化和压缩并没有考虑到模型训练和部署阶段的异同，专注于模型本身，在保持网络性能的前提下，持续缩小网络参数并不是一个容易的目标，目前大多数的研究也正是将其作为研究目标，比如模型裁剪，模型量化，矩阵/张量低秩分解等等。而将部署时的模型优化工作则一般是独立于模型结构进行研究的，在保持模型参数和计算规模不变的前提下，提高模型推理精度和推理速度。现阶段大多数模型部署优化方面的研究都集中在优化深度学习框架和模型部署工具方面，通过优化计算图，提高模型并行度，减少参数访存和模型在特定硬件的计算效率等加快模

型推理速度，如贾等人提出 metaflow<sup>[88]</sup> 通过放松图替换算法寻找最佳计算图优化以提升网络推理效率，丁、韩等人提出 IOS<sup>[89]</sup> 通过一种动态规划算法自动调度多个算子并行执行加速网络运行。但是这些研究都仅仅是立足于模型部署侧，并没有结合网络结构进行协同设计，存在一定的局限性。

最近开始有相关研究开始研究结合部署阶段特点的网络优化<sup>[48-50]</sup>，针对训练和部署时的不同要求，设计在训练和部署阶段不同行为了网络结构，并在部署时进行网络融合。本章也从这个角度入手，提出针对残差结构的分支融合部署方法，针对类<sup>[1]</sup> 网络，通过融合残差结构结构，在不损失精度的前提下进行结构替换和融合，将残差结构融合为一个卷积，避免了网络的多分支结构带来了额外的内存消耗，同时减少了网络深度，提高模型部署时的运行效率。

本章首先介绍相关新兴研究方向，之后分析残差结构和线性网络结构的特点，进而引出本章提出的分支融合方法，之后详细介绍采用重参数化和知识蒸馏进行分支融合的原理与方法，最后对本章进行总结。

## 4.2 重参数化与辅助训练

重参数化是指，模型训练阶段和模型推理阶段分别使用不同的网络结构和模型参数，在模型训练完成之后通过结构和参数变换，将训练阶段的模型参数和结构转变成推理阶段使用的参数和结构，即模型的训练阶段和模型的推理阶段的行为并不完全一致。其中，模型结构和参数变换的约束条件是它们所表示的模型功能在数学上是等价的。一般来说，进行重参数化是为了在模型训练阶段能够达到更高的精度，在推理阶段能够更加高效的运行。即训练阶段倾向于易于优化，推理阶段倾向于高效部署。从这个角度来看，Nvidia TensorRT<sup>[54]</sup> 在部署时采用的网络融合也是重参数化的应用，近期丁等人提出 ACNet<sup>[48]</sup>，使用 1\*3 和 3\*1 非对称卷积在训练时提升 3\*3 卷积的性能，在部署时将其融合为 1 个 3\*3 卷积，实现无代价的性能提升，RepVGG<sup>[49]</sup> 提出在原始 VGG 网络基础上引入 1x1 卷积和 identity 辅助分支来解决模型优化问题，最终在部署时融合为纯 3\*3 卷积的线性网络，获得在速度和精度两方面的性能提升。Diverse Branch Block<sup>[50]</sup> 提出了一个通用的类似于 Inception Block 的结构来替换普通的卷积，极大地丰富模型的微观结构，提升多种架构的性能，这样一个复杂的 block 可以在训练结束后等价转换为一个卷积，因此模型的最终大小和速度（相对于使用普通卷积的模型）完全不变。

辅助监督在训练时使用额外的辅助损失函数来得到更高的性能，辅助损失函数只在训练时存在，在部署时舍弃。LabelEnc<sup>[90]</sup> 提出在训练时采用额外结构 LabelEncoding Function 对网络中间特征进行监督，利用类似模型蒸馏的方法完

整训练检测模型，并在推理时舍弃额外结构实现无痛涨点。

重参数化和辅助监督这两个思路的共同特性是只是训练阶段使用的技巧而不增加推理阶段的开销，实现模型推理精度的提升，因此它们在实际模型训练过程和部署过程是非常有利的。

## 4.3 线性网络与多分支结构网络

### 4.3.1 线性网络

卷积神经网络最早由 LeCun 提出，首个卷积神经网络 LeNet[] 和早期卷积神经网络 AlexNet<sup>[7]</sup>、VGGNet<sup>[8]</sup> 等都是朴素的线性结构，即层与层之间以此前后连接起来，没有跳层连接和分支，一个典型的线性网络代表就是 VGGNet，形如图 4.1。VGGNet 是 ILSVRC 2014 上的相关工作，取得了当年图像定位的冠军，在 ImageNet 数据集上获得了 68.5 % 的 Top1 精度和 88.7 % 的 Top5 精度。其主要工作是证明了增加卷积神经网络的深度能有效提高模型精度，并且多次使用小卷积核可以获得更大的感受野，等效于直接使用大卷积核，如 2 个 3\*3 卷积核相当于 1 个 5\*5 卷积核、3 个 3\*3 卷积核相当于 1 个 7\*7 卷积核，同时能够有效减少参数。如图 4.1 所示，VGG 包含 5 类卷积结构共 13 个卷积层、3 层全连接层、softmax 输出层，层与层之间使用 max-pooling（最大池化）缩小尺寸，采用 ReLU 激活函数。VGG 有多种结构，分别是 VGG-11、VGG-13、VGG-16 和 VGG-19，它们除了网络深度不一样并没有本质上的区别。

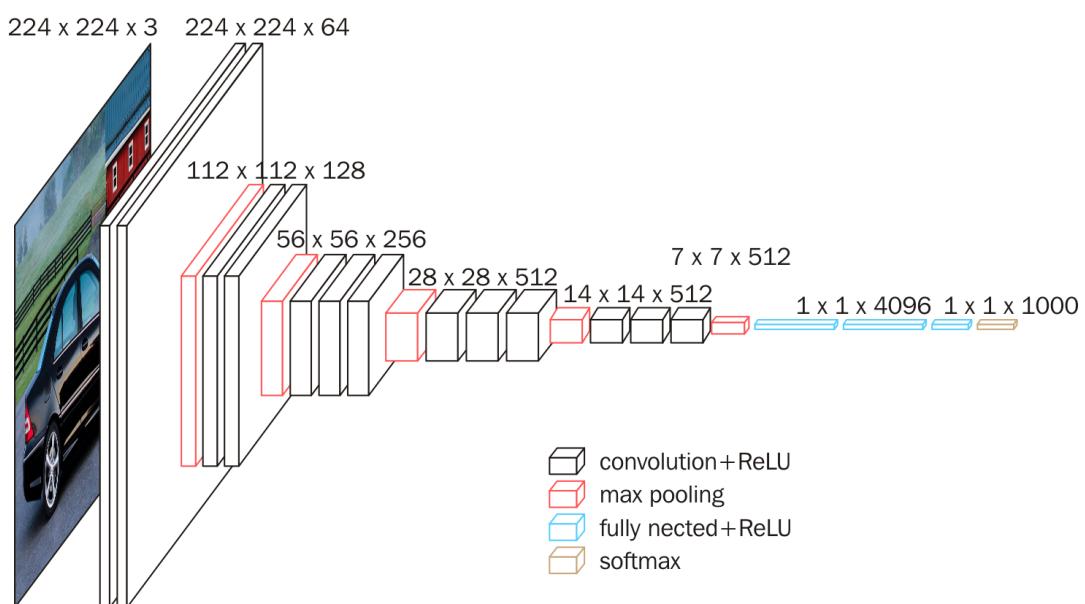


图 4.1 VGG-16，线性网络结构

### 4.3.2 残差网络与 Shortcut 连接

残差网络与 Shortcut 连接由 ResNet<sup>[1]</sup> 提出，使用 Shortcut 跳层连接，可以训练更深的神经网络。ResNet 的结构如图 4.2 所示。ResNet（残差神经网络）是由微软研究院的何恺明等人提出的。残差神经网络的主要贡献是发现了“退化现象（Degradation）”，并针对退化现象发明了 Shortcut 连接，提出了带有旁路层的残差模块，帮助梯度在层与层之间能够更加容易的传递，解决了深度过大的神经网络训练困难的问题，使得神经网络的“深度”首次突破了 100 层，最大甚至可以超过 1000 层。

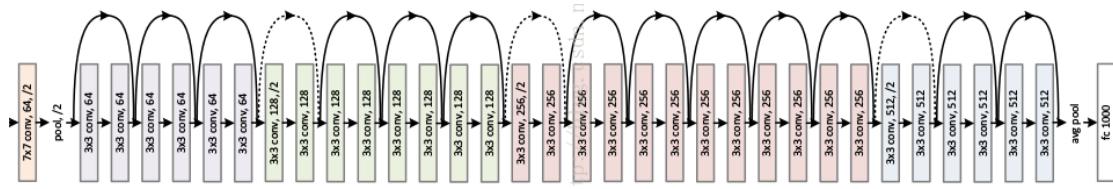


图 4.2 ResNet，残差网络结构

ResNet 使用两种不同的残差单元，如图 4.3 所示。左图对应的是浅层网络的残差结构，而右图对应的是深层网络的残差结构。对于 Shortcut 连接，当输入和输出维度一致时，可以直接将输入加到输出上。但是当维度不一致时，这就不能直接相加。有两种策略：(1) 采用 zero-padding 增加维度，此时一般要先做一个 downsample，可以采用 stride=2 的 pooling，这样不会增加参数；(2) 一般采用 1x1 的卷积（projection shortcut），这样会增加参数，也会增加计算量。短路连接除了直接使用恒等映射，也可以采用 1x1 卷积。

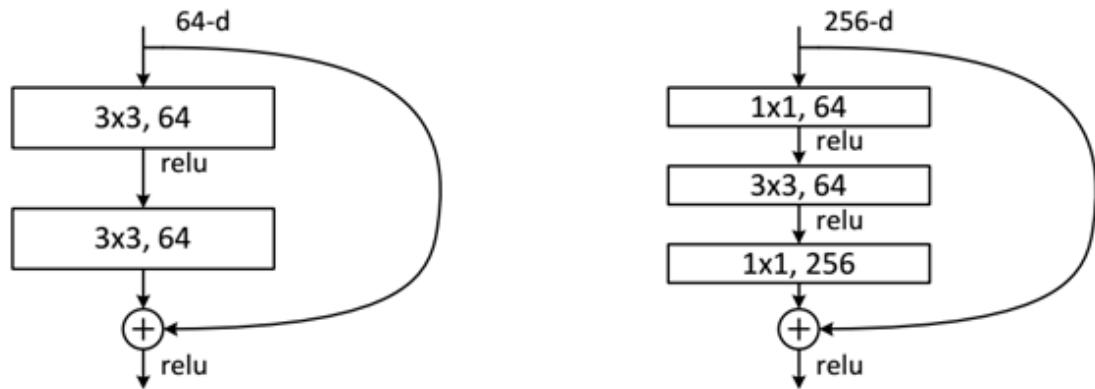


图 4.3 两种的残差结构

### 4.3.3 多分支 inception 模块

GoogleNet 使用 inception 模块作为其卷积的基本单元，inception 模块是一种多分支和多尺度的卷积核拼接而成。如图 4.4 所示，GoogleNet 即 Inception-V1，它以 inception 单元为基本结构，具有 9 个 inception 模块，每个 inception 模块由

四个分支组成，分别是  $1 \times 1$ 、 $3 \times 3$ 、 $5 \times 5$  卷积和 down-sampling，具有多尺度感受野。inception 结构的主要贡献有两个：一是使用  $1 \times 1$  的卷积来进行升降维；二是使用多尺度的卷积核进行特征提取。训练时使用两个辅助损耗层从中间层注入 Loss，防止梯度消失，而在推理时，删除辅助层。

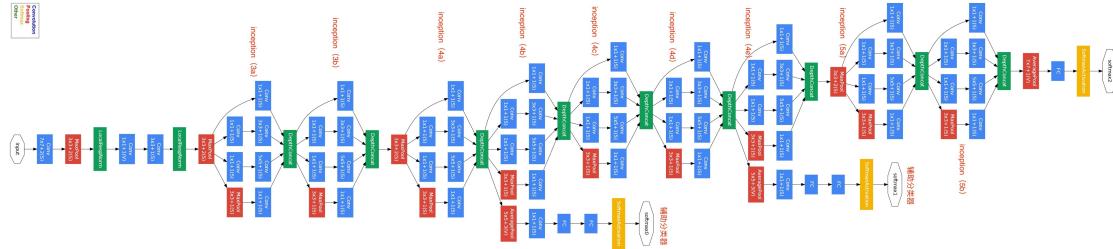


图 4.4 GoogleNet，多分支 inception 模块

残差结构经过发展，有多种变形，但其主要思想就是通过在多分支上进行多尺度的卷积运算，在多个感受野上进行特征提取，为了减少计算量，使用两个  $3 \times 3$  卷积代替  $5 \times 5$ ，使用  $1 \times 7$  和  $7 \times 1$  卷积代替  $7 \times 7$  卷积。

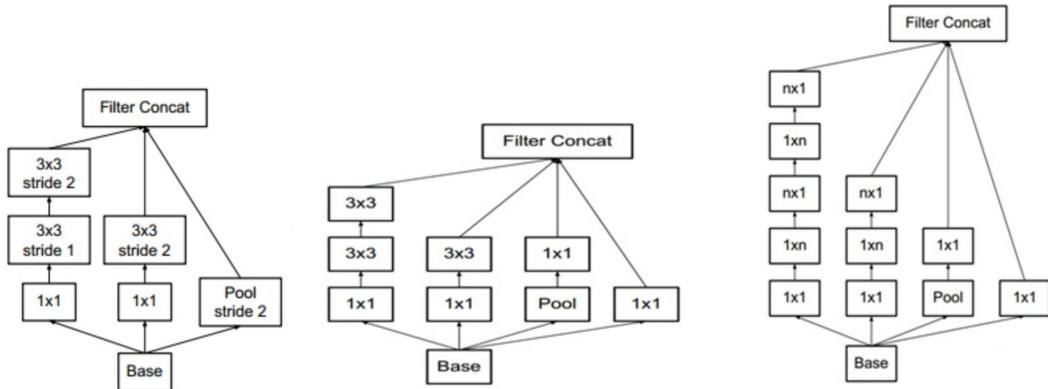


图 4.5 3 种的残差结构

#### 4.3.4 线性网络与多分支网络对比

多分支网络结构如，残差模块通过 shortcut 快捷连接，直接将训练损失回传，避免了深层神经网络的退化问题，可以训练更深的网络；inception 模块利用多分支卷积，通过合并多个感受野下的特征使得网络训练性能更好。它们都能提升网络参数的表达能力，对与网络训练是有意义的。但是，相比于单路结构，多分支结构对内存不友好，同时不利于并行化，并不能完全利用计算和内存资源。如图 4.6 所示，具有分支的结构会保存之前计算的特征图，在 add 或 cat 之前都需要常驻内存，对内存不友好。他们的优缺点如表 4.1 所示，可以看到，多分支网络训练友好部署不友好，线性网络则恰好相反。

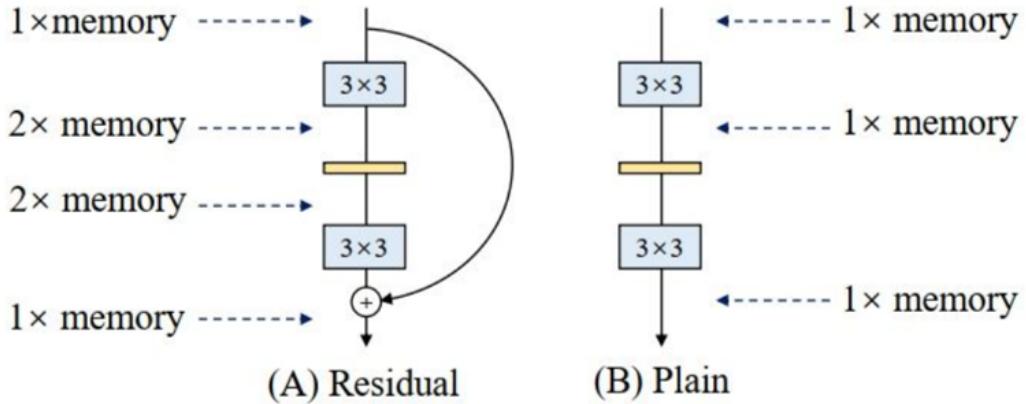


图 4.6 残差结构与线性网络内存使用对比

表 4.1 线性网络与多分支网络优缺点

网络类型	优点	缺点
线性网络	并行度高 内存效率高 推理速度更快	不利于反向传播 训练不友好
多分支网络	利于反向传播 多尺度感受野 参数效率高	不利于并行 内存效率低

#### 4.4 基于重参数化的残差模块融合

相比于各种多分支架构（如 ResNet, Inception, DenseNet, 各种 NAS 架构等），近年来 VGG 式的线性网络模型鲜有关注，主要自然是因为性能差。例如，有研究认为，ResNet 性能好的一种解释是 ResNet 的分支结构（Shortcut）产生了一个大量子模型的隐式集成（因为每遇到一次分支，总的路径就变成两倍），单路架构显然不具备这种特点。既然多分支架构是对训练有益的，而我们想要部署的模型是单路架构，我们针对多分支网络中的典型网络模型 ResNet 提出一种解耦训练时和推理时的方法，将网络模型的训练和部署区分，流程如下：

- 1) 训练 ResNet 网络模型
- 2) 将残差模块进行分支融合，去除 Shortcut 连接，等价转换为单个卷积核
- 3) 部署融合后模型

这样就可以同时利用残差结构训练时的优势（性能高）和线性模型推理时的好处（速度快、省内存）。这里的关键显然在如何将 ResNet 的残差模块进行处理和融合。

为了能够顺利的融合残差模块，我们需要对残差模块进行一定修改。如

图4.7和图4.8所示，在训练时，确保能够在训练之后通过代数运算合并卷积核，我们将残差连接中的非线性层去掉，这是我们对ResNet的残差结构唯一的修改，基本不影响ResNet模型精度。但是去掉非线性层后，相互连接的卷积层就能够进行纵向融合成为一个卷积核，如图4.7所示，对于包含两个 $3 \times 3$ 卷积核的BasicBlock残差模块，去掉非线性层之后能够将整个残差模块融合为一个 $5 \times 5$ 的卷积核，虽然一个 $5 \times 5$ 的卷积核参数量比两个 $3 \times 3$ 卷积核稍大，但是在实际运行时其速度更快，且去除了分支结构，内存消耗更少。对于Bottleneck残差模块的融合如图4.8所示则更为划算，融合之后的卷积核仍为 $3 \times 3$ 的卷积核。

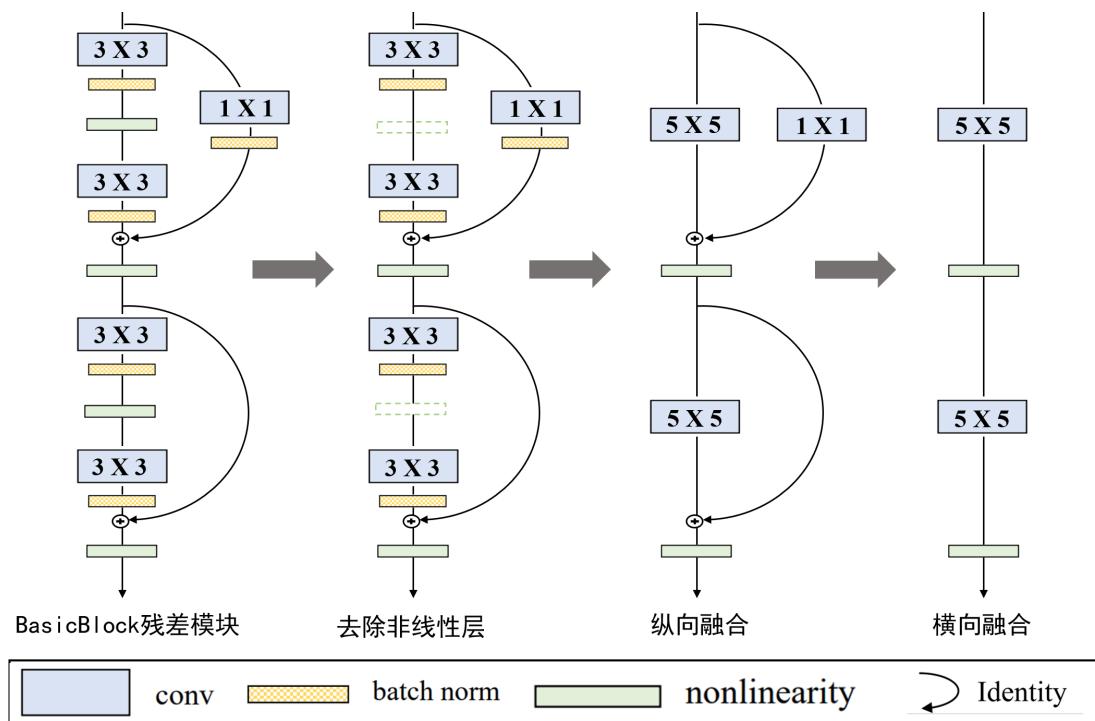


图4.7 BasicBlock 残差结构融合示意图

我们知道，不包含非线性层的卷积网络模块可以等价为一个线性函数，因此可以一步一步将其进行等价转换和等价合并。下面将具体描述残差网络的融合原理和融合方法，包括：

- 1) 卷积层和批归一化层融合
- 2) 卷积层纵向融合
- 3) 卷积层横向融合

#### 4.4.1 卷积层和批归一化层合并

卷积层(Conv层)和批归一化层(BN层)很容易进行合并为一个卷积层，这是在部署时进行计算图优化的一个常用手段，通过合并Conv + BN层，可以减少网络层数，提升网络性能。其融合原理如下：

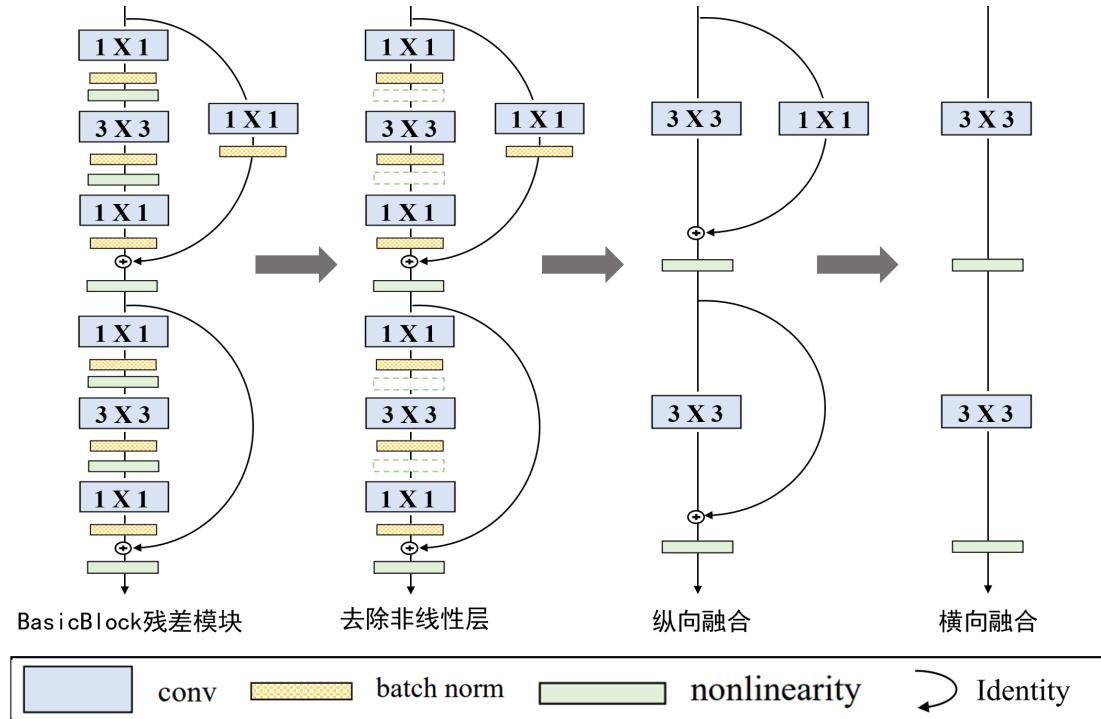


图 4.8 Bottleneck 残差结构融合示意图

卷积层公式为:

$$Conv(X) = WX + b \quad (4.1)$$

BN 层公式为:

$$BN(X) = \gamma * \frac{-mean}{\sqrt{var}} + \beta \quad (4.2)$$

带入有:

$$\begin{aligned} BN(Conv(X)) &= \gamma * \frac{WX + b - mean}{\sqrt{var}} + \beta \\ &= \frac{\gamma * WX}{\sqrt{var}} + \frac{\gamma * (b - mean)}{\sqrt{var}} + \beta \end{aligned} \quad (4.3)$$

令:

$$\begin{aligned} Conv_{fused} &= BN(Conv(X)) \\ W_{fused} &= \frac{\gamma * WX}{\sqrt{var}} \\ B_{fused} &= \frac{\gamma * (b - mean)}{\sqrt{var}} + \beta \end{aligned} \quad (4.4)$$

则:

$$\begin{aligned} Conv_{fused}(X) &= BN(Conv(X)) \\ &= W_{fused}X + B_{fused} \end{aligned} \quad (4.5)$$

融合之后, 新的卷积层的权重 (weight) 为  $W_{fused}$ , 其偏置 (bias) 为  $B_{fused}$ 。

Conv 层与 BN 层融合代码如下:

Conv层与BN层融合

---

```

1  def merge_bn(conv: nn.Conv2d, bn: nn.Conv2d):
2      w, b = conv.weight.data, conv.bias.data
3      gamma = bn.weight.data
4      running_mean = bn.running_mean
5      running_var = bn.running_var
6      eps = bn.eps
7      std = (running_var + eps).sqrt()
8
9      w = conv[0].weight.data * ((gamma / std).reshape(-1, 1, 1, 1))
10     b = conv[1].bias.data - running_mean * gamma / std
11
12     return w, b

```

---

#### 4.4.2 卷积层纵向融合

直接的卷积层由于中间没有非线性层，其公式可表示为：

$$\begin{aligned}
 Conv_2(Conv_1(X)) &= W_2(W_1X + b_1) + b_2 \\
 &= W_2W_1X + W_2b_1 + b_2 \\
 &= (W_2W_1)X + (W_2b_1 + b_2)
 \end{aligned} \tag{4.6}$$

令：

$$\begin{aligned}
 Conv_{fused}(X) &= Conv_2(Conv_1(X)) \\
 W_{fused} &= (W_2W_1) \\
 b_{fused} &= (W_2b_1 + b_2)
 \end{aligned} \tag{4.7}$$

则：

$$\begin{aligned}
 Conv_{fused} &= Conv_2(Conv_1(X)) \\
 &= (W_2W_1)X + (W_2b_1 + b_2) \\
 &= W_{fused} + b_{fused}
 \end{aligned} \tag{4.8}$$

融合之后，新的卷积层的权重 (weight) 为  $W_{fused}$ ，其偏置 (bias) 为  $B_{fused}$ 。为了详细说明，假设输入特征图特征图尺寸为  $[1, 2, 3, 3]$ ，连续经过一个  $3 \times 3$  卷积层和  $1 \times 1$  卷积层，输出特征图为  $[1, 2, 1, 1]$ ，如图 4.9 所示，为连续  $3 \times 3$  卷积层和  $1 \times 1$  卷积层的计算过程，由于  $1 \times 1$  卷积比较特殊（非  $1 \times 1$  卷积核融合计算稍显复杂，融合后的卷积核会增大，示例使用比较简单的包含  $1 \times 1$  卷积的融合），此时可以看做由  $1 \times 1$  卷积核先对  $3 \times 3$  卷积核参数做卷积运算，将所得的参数当做新卷积层参数，再对输入进行卷积。Conv $3 \times 3$  与 Conv $1 \times 1$  纵向融合代码如下：

Conv $3 \times 3$  与 Conv $1 \times 1$  纵向融合

```

1  # 先Conv3x3再Conv1x1
2  def merge_3_1(conv3: nn.Conv2d, conv1: nn.Conv2d):
3      k1, k3 = conv1.weight.data, conv3.weight.data
4      b1, b3 = conv1.bias.data, conv3.bias.data
5
6      print(k1.shape, k3.shape)
7      k = F.conv2d(k3.permute(1, 0, 2, 3), k1).permute(1, 0, 2, 3)
8      b = (k1 * b3.reshape(1, -1, 1, 1)).sum((1, 2, 3)) + b1
9      print(b.shape)
10     return k, b
11
12  # 先Conv1x1再Conv3x3
13  def merge_1_3(conv1: nn.Conv2d, conv3: nn.Conv2d):
14      k1, k3 = conv1.weight.data, conv3.weight.data
15      b1, b3 = conv1.bias.data, conv3.bias.data
16
17      k = F.conv2d(k3, k1.permute(1, 0, 2, 3))
18      b = (k3 * b1.reshape(1, -1, 1, 1)).sum((1, 2, 3)) + b3
19      print(b.shape)
20      return k, b

```

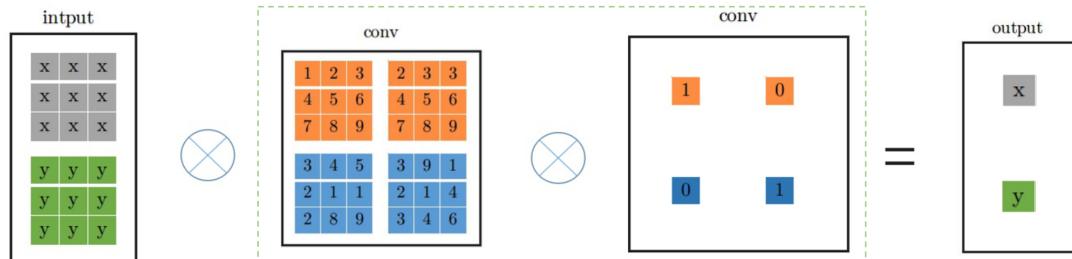


图 4.9 Conv3\*3 与 Conv1\*1 纵向融合

#### 4.4.3 卷积层横向融合

在残差结构中，使用逐元素相加的方式合并不同的分支，对于特征图，要求不同分支的特征图维度完全一样，对于卷积参数融合，要求不同分支的卷积核参数维度完全一样，因此，对于 1x1 卷积和 Identity 连接需要特殊处理。

##### 1) 1x1 卷积等价扩充：

对于卷积操作，小卷积核可以看做是大卷积核的一种特例，小卷积核可以用大卷积核来进行描述，如图 4.10 所示，1x1 卷积核可以进行补 0，将其扩充为 3x3 卷积核，由 3x3 卷积核表示。

##### 2) Identity 层等价卷积核：

identity 层就是输入等于输出，即 input 中每个通道每个元素直接输出到

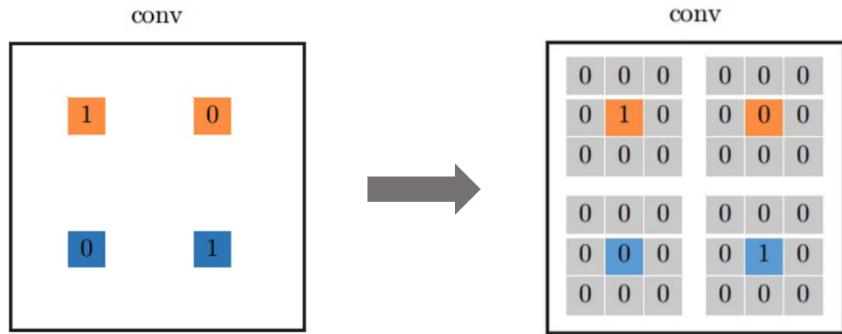


图 4.10 Conv1\*1 卷积核等价扩充表示

`output` 中对应的通道，即相当于值为 1 的深度可分离卷积，使用普通卷积表示为单位矩阵，因此，`identity` 可以等效成如下图 4.11 的 Conv1\*1 的卷积形式。我们进一步可以将 `identity -> Conv1*1 -> Conv3*3` 的形式，如下图 4.12 所示。

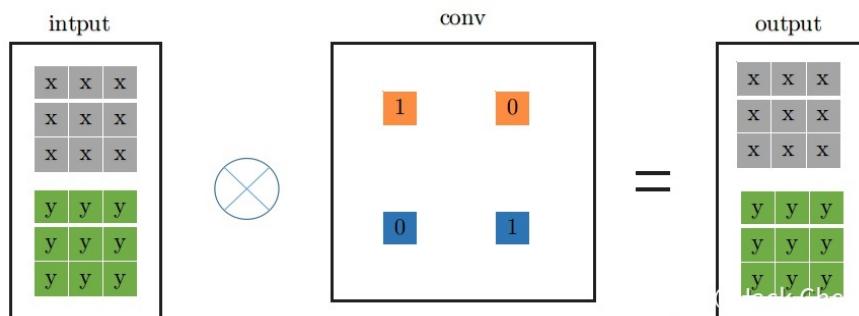


图 4.11 identity 层的 Conv1\*1 表示

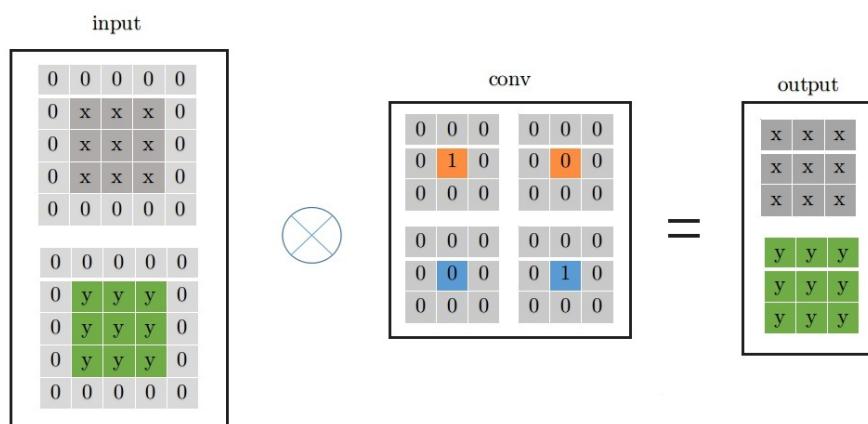


图 4.12 identity 层的 Conv3\*3 表示

为了详细说明，假设输入特征图尺寸为  $(1, 2, 3, 3)$ ，输出特征图尺寸与输入特征图尺寸相同，且  $\text{stride}=1$ ，如图 4.13，展示是 Conv3\*3 的卷及过程：

Conv3\*3 卷积由上图所示，首先将特征图进行  $pad = kernel\_size/2$ ，然后从上

图左上角红色为止做卷积运算，最终得到右边 output 输出。如图 4.14 是 Conv1\*1 卷积过程：

同理，Conv1\*1 跟 Conv3\*3 卷积过程一样，从上图中左边 input 中红色位置开始进行卷积，得到右边的输出，观察 Conv1\*1 和 Conv3\*3 的卷积过程，从红色起点位置开始，走过相同的计算路径，因此，将 Conv1\*1 跟 Conv3\*3 进行融合，只需要将 Conv1\*1 卷积核 padding 成 Conv3\*3 的形式，然后于 Conv3\*3 相加，再与特征图做卷积（这里依据卷积的可加性原理）即可，也就是 Conv1\*1 的卷积过程变成如下图 4.15 形式：

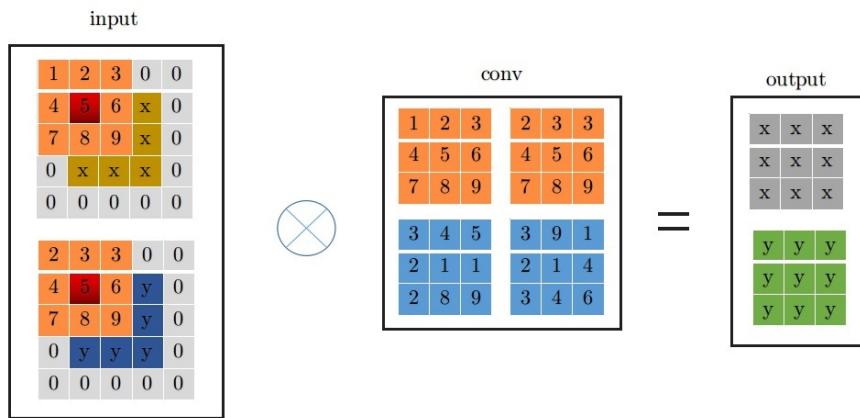


图 4.13 Conv3\*3 卷积运算

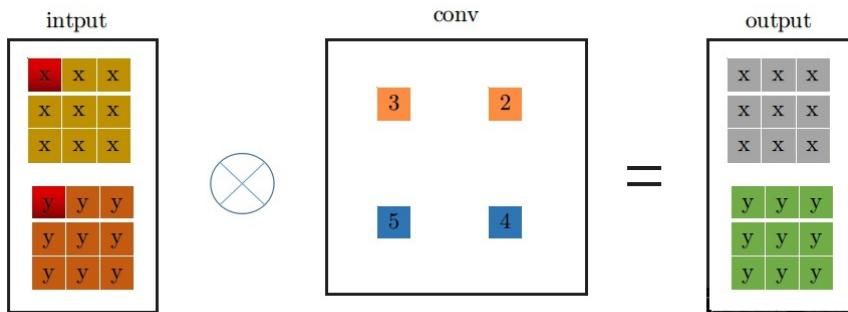


图 4.14 Conv1\*1 卷积运算

在将分支转化为同等的卷积结构后，如图 4.16 将 Conv1\*1 与 identity 都表示为 Conv3\*3 形式，吸收 BN 层，然后线性可加。

## 4.5 残差模块融合收益分析与通道调整

通过移除残差模块中的非线性层，可以很容易的将其进行合并，但是直接合并而不经任何修改可能会导致参数量的剧增，并不能带来明显的收益。下面将对 ResNet 网络中的不同残差模块的融合收益进行分析，并根据结果调整残差模块

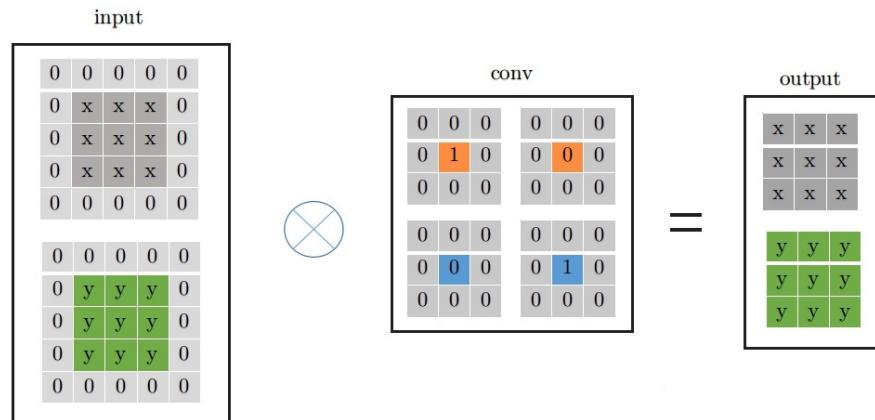
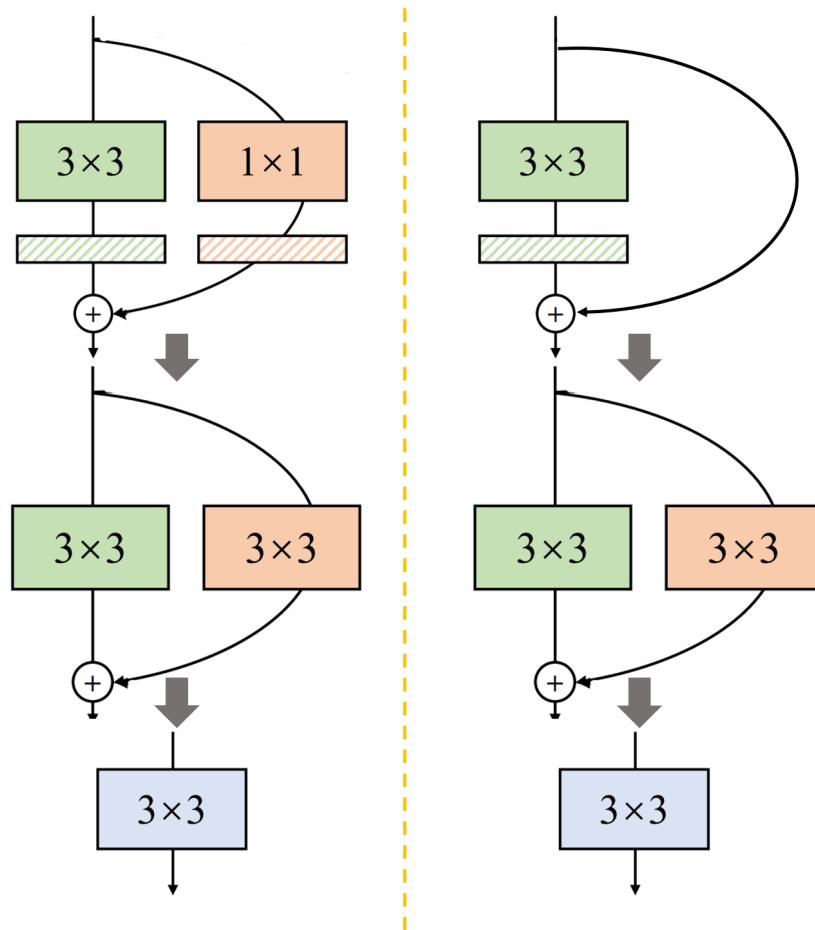
图 4.15 Conv $3 \times 3$  和 Conv $1 \times 1$  合并步骤 3

图 4.16 残差模块横向融合

的通道结构。

#### 4.5.1 BasicBlock 残差模块

BasicBlock 残差模块由一个 Shortcut 残差连接与两个连续的  $3 \times 3$  卷积核组成，但是如果遇到需要下采样的情况，

## 1. 无下采样的 BasicBlock 残差模块

### 4.5.2 Bottleneck 残差模块

## 4.6 实验结果与分析

在本节中，我们在算力较高搭载 V100 GPU 的服务器上和算力较低的嵌入式设备 TX2 上进行了部署推理实验，比较了融合分支后的网络模型对比原始 ResNet 的推理性能。同时也在 Cifar100 和 ImageNet 数据集上进行了训练，对比融合分支后的网络模型与原始 ResNet 的精度差异。

### 4.6.1 实验设置

实验使用简单数据增强后的 Cifar100、ImageNet 数据集，训练 120 个周期，推理批量尺寸 (batchsize) 为 256，速度单位为示例/秒，服务器显卡为 NVIDIA V100，嵌入式设备为 NVIDIA TX2，使用 TensorRT 作为测试的软件环境。在实验对比中，我们将所提出针对残差结构的分支融合方法应用于 ResNet 上，并与原始的 ResNet 在运行速度，模型精度，内存消耗量上进行了比较。

#### 1. 硬件平台说明

我们实验的训练服务器使用 Intel Xeon E5 服务器，配有 2 张 NVIDIA V100 显卡，其具体配置如表 reftbb:server 所示。

表 4.2 训练服务器配置表

OS	Ubuntu 16.04 Xenial
CPU	2 * Intel Xeon E5-2620 v4 @ 32x 3GHz
GPU	2 * Nvidia Tesla V100
RAM	256GB DDR4

在部署时还在嵌入式平台上进行测试，使用 Nvidia TX2 作为部署环境，其搭载四核 ARM®Cortex®-A57 MPCore，8GB 256 位 LPDDR4 内存，操作系统为 Ubuntu 18.04。其具体配置如表 4.3 所示。

表 4.3 Nvidia TX2 配置表

OS	Ubuntu 18.04 bionic
CPU	HMP Dual Denver + Quad ARM A57
GPU	Nvidia Pascal 256 CUDA cores
RAM	8GB 128-bit LPDDR4
DISK	32GB eMMC

## 4.6.2 实验结果

表4.4为分别在Cifar100和ImageNet上的训练结果，本次测试将分支融合部署的ResNet18、ResNet34、ResNet50、ResNet101与其原始模型进行对比，实验表明，我们的分支融合方法在误差范围内对模型精度并不影响。

表4.5 Cifar100上训练结果对比

模型	Top1 准确率
ResNet18	78.61
<b>ResNet18*</b>	<b>78.83</b>
ResNet34	78.84
<b>ResNet34*</b>	<b>78.63</b>
ResNet50	77.88
<b>ResNet50*</b>	<b>79.26</b>
ResNet101	80.16
<b>ResNet101*</b>	<b>80.87</b>
ResNet152	80.99
<b>ResNet152*</b>	<b>80.34</b>

表4.6 ImageNet上训练结果对比

模型	Top1 准确率
ResNet18	71.16
<b>ResNet18*</b>	<b>71.21</b>
ResNet34	74.17
<b>ResNet34*</b>	<b>74.36</b>
ResNet50	76.31
<b>ResNet50*</b>	<b>76.48</b>
ResNet101	77.21
<b>ResNet101*</b>	<b>77.28</b>
ResNet101	77.78
<b>ResNet101*</b>	<b>77.56</b>

表4.7为在服务器端与嵌入式端实际部署时的推理速度对比。本次测试将分支融合部署的ResNet18、ResNet34、ResNet50、ResNet101、ResNet152与其原始模型进行对比，实验表明，在公平的训练设定下，同精度的分支融合部署的模型在速度和内存占用方面显著优于原有模型，效果非常不错。

表4.7 ImageNet上训练结果对比

模型	TX2 速度	V100 速度
ResNet18	71.16	2442
<b>ResNet18*</b>	<b>72.41</b>	<b>3256</b>
ResNet34	74.17	1419
<b>ResNet34*</b>	<b>74.46</b>	<b>2339</b>
ResNet50	76.31	719
<b>ResNet50*</b>	<b>77.78</b>	<b>792</b>
ResNet101	77.21	430
<b>ResNet101*</b>	<b>78.78</b>	<b>460</b>

## 4.7 本章小结

考虑到模型训练时和部署时的注重点不同，借助重参数化的思想，本章提出了针对残差结构的分支融合方法，优化部署时残差网络模型推理效率和内存效率。通过去除残差结构中的非线性层，在部署前融合多分支结构，在不损失模型精度的前提下，去除模型分支结构同时减少模型层数，提高部署时内存效率和运行效率。首先，讨论了到线性网络结构和多分支网络结构各自的优点和局限性，其次通过微调 ResNet 网络结构，解耦网络的训练和部署，在训练时使用多分支残差网络结构，在部署时将其转化为线性网络结构，同时利用了线性网络和多分支网络的优点而规避它们的缺点，最终获得在速度和精度两方面的性能提升。

## 第5章 总结与展望

近年来，深度神经网络在许多领域，尤其是视觉识别任务中取得了巨大的成功。然而，现有的深度神经网络模型计算成本高，内存密集，阻碍了它们在内存资源较低的设备或在有严格延迟要求的应用中部署。随着深度神经网络的研究与发展，目前已成为许多智能系统的基本工具。与此同时，这些网络的计算复杂度和资源消耗也在不断增加，现有的深度神经网络模型需要消耗大量的计算资源且占用大量内存，从而组队网络的部署造成了重大挑战，特别是在实时应用或资源受限设备上的部署。因此，网络的优化和加速已经成为目前深度学习研究的一个热门话题。本文以深度卷积神经网络为研究对象，从不同的角度探索研究深度卷积神经网络在嵌入式平台下部署的优化与加速算法，在有限的计算资源的前提下尽可能的发挥出深度学习的性能优势，摆脱硬件和功耗的束缚，扩展深度学习的应用场景。

### 5.1 全文总结

由于在嵌入式设备和边缘设备计算和存储资源有限，现有多数卷积神经网络在该场景下的部署和应用局限性比较大。本文主要针对嵌入式环境下计算和存储资源受限的问题，对卷积神经网络在嵌入式设备上的部署加速和优化进行研究。

首先，广泛调研目前的神经网络压缩和优化研究现状，对模型压缩和优化加速的向相关算法进行了广泛的研究，总结了该领域现有的几类主流研究方向：模型修剪、矩阵/张量分解、模型参数量化、高效网络设计、知识蒸馏、分支融合与重参数化，对各类算法和研究进行了大量的文献阅读和探索性的实验，进行一定的总结和综述。

其次，探讨了如何在低算力嵌入式设备上适配神经网络或者紧凑的网络设计，即考虑通过合理选择神经网络架构和规模压缩方案，来有效地实现约束型硬件资源上的模型部署和推理。针对此问题，借助动态网络的思想和运行时内存交换技术，本章提出了任务感知的子网络切换算法，解决了在内存约束的嵌入式设备上部署 DNN 模型时内存不足的问题。通过动态网络的思想为不同类的数据训练一个专属子网，并在运行时根据输入数据动态切换子网进行推理，可以有效消除这种冗余，加快推理速度，并有效减少运行时内存需求。在运行时，我们的方法检测任务类型，并将关键子网的模型参数从外部存储交换到内存。与传统的网络剪枝方法相比，该方法在保持较高的推理精度的同时，进一步减少了内存需求。

和推理速度。

然后，分析了多分支网络和线性网络的结构差异性与优缺点，针对具有残差结构的多分支网络，提出了针对残差结构的分支融合方法，将内存低效和低并行度的残差结构通过用重参数化技术对其分支进行融合，将多分支的 ResNet 合并成为一个类 VGG 的线性网络，提高网络部署运行时的内存效率和并行度，节省网络资源消耗，加快网络推理速度。

## 5.2 后续工作展望

近年来，卷积神经网络在预测精度方面发展非常快，但随着准确度提高，模型推理速度也会随着模型计算复杂度增加而变慢。因此如何在精度和速度之间取得更好的平衡一直是模型设计过程中一个非常大的挑战。此外，目标任务或者硬件平台的差异性、理论复杂度和实际速度差异性和平台或业务的额外约束都极大地加大了模型设计难度。我们之前的工作，主要基于动态模型和重参数化两个方面，他们有较大的不同，但是其共同点是将模型训练阶段和部署阶段分离，训练时和部署时模型结构并不相同，结合训练阶段和部署阶段各自的特性进行网络设计。未来的工作将主要围绕着模型训练和部署之间的特性，考虑嵌入式设备的资源局限性，着重于在部署时的效率优化，进行进一步的研究。

## 参 考 文 献

- [1] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
- [2] LUONG M T, PHAM H, MANNING C D. Effective approaches to attention-based neural machine translation [J]. arXiv preprint arXiv:1508.04025, 2015.
- [3] AMODEI D, ANUBHAI R, BATTENBERG E, et al. End to end speech recognition in english and mandarin [J]. 2016.
- [4] BAHDANAU D, CHO K, BENGIO Y. Neural machine translation by jointly learning to align and translate [J]. arXiv preprint arXiv:1409.0473, 2014.
- [5] ESTEVA A, KUPREL B, NOVOA R A, et al. Dermatologist-level classification of skin cancer with deep neural networks [J]. nature, 2017, 542(7639): 115-118.
- [6] 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.
- [7] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks [C]//Advances in neural information processing systems. 2012: 1097-1105.
- [8] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition [J]. arXiv preprint arXiv:1409.1556, 2014.
- [9] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.
- [10] JOUPPI N P, YOUNG C, PATIL N, et al. In-datacenter performance analysis of a tensor processing unit [C]//Proceedings of the 44th annual international symposium on computer architecture. 2017: 1-12.
- [11] CAMBRICON. Mlu [EB/OL]. <https://www.cambricon.com/>.
- [12] CLEMENT F, MARTINI B, CORDA B, et al. Neuflow: A runtime reconfigurable dataflow processor for vision [C]//Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on. 2011: 109-116.
- [13] FARABET C, POULET C, HAN J Y, et al. Cnp: An fpga-based processor for convolutional networks [C]//2009 International Conference on Field Programmable Logic and Applications. IEEE, 2009: 32-37.
- [14] QADEER W, HAMEED R, SHACHAM O, et al. Convolution engine: balancing efficiency & flexibility in specialized computing [C]//Proceedings of the 40th Annual International Symposium on Computer Architecture. 2013: 24-35.

- 
- [15] CHEN T, DU Z, SUN N, et al. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning [J]. ACM SIGARCH Computer Architecture News, 2014, 42(1): 269-284.
  - [16] CHEN Y, LUO T, LIU S, et al. Dadiannao: A machine-learning supercomputer [C]//2014 47th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE, 2014: 609-622.
  - [17] DU Z, FASTHUBER R, CHEN T, et al. Shidiannao: Shifting vision processing closer to the sensor [C]//Proceedings of the 42nd Annual International Symposium on Computer Architecture. 2015: 92-104.
  - [18] CHEN Y H, KRISHNA T, EMER J S, et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks [J]. IEEE journal of solid-state circuits, 2016, 52(1): 127-138.
  - [19] QIU J, WANG J, YAO S, et al. Going deeper with embedded fpga platform for convolutional neural network [C]//Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2016: 26-35.
  - [20] CHAKRADHAR S, SANKARADAS M, JAKKULA V, et al. A dynamically configurable co-processor for convolutional neural networks [C]//Proceedings of the 37th annual international symposium on Computer architecture. 2010: 247-257.
  - [21] DENIL M, SHAKIBI B, DINH L, et al. Predicting parameters in deep learning [J]. arXiv preprint arXiv:1306.0543, 2013.
  - [22] LECUN Y, DENKER J S, SULLA S A, et al. Optimal brain damage. [C]//NIPS: volume 2. Citeseer, 1989: 598-605.
  - [23] DENTON E, ZAREMBA W, BRUNA J, et al. Exploiting linear structure within convolutional networks for efficient evaluation [J]. arXiv preprint arXiv:1404.0736, 2014.
  - [24] NOVIKOV A, PODOPRIKHIN D, OSOKIN A, et al. Tensorizing neural networks [J]. arXiv preprint arXiv:1509.06569, 2015.
  - [25] LIN M, CHEN Q, YAN S. Network in network [J]. arXiv preprint arXiv:1312.4400, 2013.
  - [26] VANHOUCKE V, SENIOR A, MAO M Z. Improving the speed of neural networks on cpus [J]. 2011.
  - [27] ANWAR S, HWANG K, SUNG W. Fixed point optimization of deep convolutional neural networks for object recognition [C]//2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2015: 1131-1135.
  - [28] HWANG K, SUNG W. Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1 [C]//2014 IEEE Workshop on Signal Processing Systems (SiPS). IEEE, 2014: 1-6.
  - [29] GONG Y, LIU L, YANG M, et al. Compressing deep convolutional networks using vector quantization [J]. arXiv preprint arXiv:1412.6115, 2014.

- [30] CHEN W, WILSON J, TYREE S, et al. Compressing neural networks with the hashing trick [C]//International conference on machine learning. PMLR, 2015: 2285-2294.
- [31] MOLCHANOV P, TYREE S, KARRAS T, et al. Pruning convolutional neural networks for resource efficient inference [J]. arXiv preprint arXiv:1611.06440, 2016.
- [32] ANWAR S, SUNG W. Compact deep convolutional neural networks with coarse pruning [J]. arXiv preprint arXiv:1610.09639, 2016.
- [33] YANG T J, CHEN Y H, SZE V. Designing energy-efficient convolutional neural networks using energy-aware pruning [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 5687-5695.
- [34] HE Y, ZHANG X, SUN J. Channel pruning for accelerating very deep neural networks [C]// Proceedings of the IEEE International Conference on Computer Vision. 2017: 1389-1397.
- [35] NARANG S, ELSEN E, DIAMOS G, et al. Exploring sparsity in recurrent neural networks [J]. arXiv preprint arXiv:1704.05119, 2017.
- [36] GUO Y, YAO A, CHEN Y. Dynamic network surgery for efficient dnns [J]. arXiv preprint arXiv:1608.04493, 2016.
- [37] WEN W, WU C, WANG Y, et al. Learning structured sparsity in deep neural networks [J]. arXiv preprint arXiv:1608.03665, 2016.
- [38] LI H, KADAV A, DURDANOVIC I, et al. Pruning filters for efficient convnets [J]. arXiv preprint arXiv:1608.08710, 2016.
- [39] LIU Z, LI J, SHEN Z, et al. Learning efficient convolutional networks through network slimming [C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 2736-2744.
- [40] LUO J H, WU J, LIN W. Thinet: A filter level pruning method for deep neural network compression [C]//Proceedings of the IEEE international conference on computer vision. 2017: 5058-5066.
- [41] ANWAR S, HWANG K, SUNG W. Structured pruning of deep convolutional neural networks [J]. ACM Journal on Emerging Technologies in Computing Systems (JETC), 2017, 13(3): 1-18.
- [42] MAO H, HAN S, POOL J, et al. Exploring the regularity of sparse structure in convolutional neural networks [J]. arXiv preprint arXiv:1705.08922, 2017.
- [43] VENKATESH G, NURVITADHI E, MARR D. Accelerating deep convolutional networks using low-precision and sparsity [C]//2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2017: 2861-2865.
- [44] ZHU C, HAN S, MAO H, et al. Trained ternary quantization [J]. arXiv preprint arXiv:1612.01064, 2016.

- [45] RASTEGARI M, ORDONEZ V, REDMON J, et al. Xnor-net: Imagenet classification using binary convolutional neural networks [C]//European conference on computer vision. Springer, 2016: 525-542.
- [46] ZHOU S, WU Y, NI Z, et al. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients [J]. arXiv preprint arXiv:1606.06160, 2016.
- [47] MIYASHITA D, LEE E H, MURMANN B. Convolutional neural networks using logarithmic data representation [J]. arXiv preprint arXiv:1603.01025, 2016.
- [48] DING X, GUO Y, DING G, et al. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks [C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 1911-1920.
- [49] DING X, ZHANG X, MA N, et al. Repvgg: Making vgg-style convnets great again [J]. arXiv preprint arXiv:2101.03697, 2021.
- [50] DING X, ZHANG X, HAN J, et al. Diverse branch block: Building a convolution as an inception-like unit [J]. arXiv preprint arXiv:2103.13425, 2021.
- [51] KRIZHEVSKY A, HINTON G, et al. Learning multiple layers of features from tiny images [J]. 2009.
- [52] FACEBOOK. Pytorch [EB/OL]. <https://pytorch.org/>.
- [53] TENCENT. Ncnn [EB/OL]. <https://github.com/Tencent/ncnn>.
- [54] NVIDIA. Tensorrt [EB/OL]. <https://developer.nvidia.com/zh-cn/tensorrt>.
- [55] SRINIVAS S, BABU R V. Data-free parameter pruning for deep neural networks [J]. arXiv preprint arXiv:1507.06149, 2015.
- [56] HAN S, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural networks [J]. arXiv preprint arXiv:1506.02626, 2015.
- [57] HAN S, MAO H, DALLY W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding [J]. arXiv preprint arXiv:1510.00149, 2015.
- [58] ULLRICH K, MEEDS E, WELLING M. Soft weight-sharing for neural network compression [J]. arXiv preprint arXiv:1702.04008, 2017.
- [59] LEBEDEV V, LEMPITSKY V. Fast convnets using group-wise brain damage [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 2554-2564.
- [60] ZHOU H, ALVAREZ J M, PORIKLI F. Less is more: Towards compact cnns [C]//European Conference on Computer Vision. Springer, 2016: 662-677.
- [61] RIGAMONTI R, SIRONI A, LEPETIT V, et al. Learning separable filters [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2013: 2754-2761.
- [62] JADERBERG M, VEDALDI A, ZISSERMAN A. Speeding up convolutional neural networks

- with low rank expansions [J]. arXiv preprint arXiv:1405.3866, 2014.
- [63] LEBEDEV V, GANIN Y, RAKHUBA M, et al. Speeding-up convolutional neural networks using fine-tuned cp-decomposition [J]. arXiv preprint arXiv:1412.6553, 2014.
- [64] TAI C, XIAO T, ZHANG Y, et al. Convolutional neural networks with low-rank regularization [J]. arXiv preprint arXiv:1511.06067, 2015.
- [65] WU J, LENG C, WANG Y, et al. Quantized convolutional neural networks for mobile devices [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 4820-4828.
- [66] GUPTA S, AGRAWAL A, GOPALAKRISHNAN K, et al. Deep learning with limited numerical precision [C]//International conference on machine learning. PMLR, 2015: 1737-1746.
- [67] CHOI Y, EL-KHAMY M, LEE J. Towards the limit of network quantization [J]. arXiv preprint arXiv:1612.01543, 2016.
- [68] COURBARIAUX M, BENGIO Y, DAVID J P. Binaryconnect: Training deep neural networks with binary weights during propagations [J]. arXiv preprint arXiv:1511.00363, 2015.
- [69] COURBARIAUX M, HUBARA I, SOUDRY D, et al. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1 [J]. arXiv preprint arXiv:1602.02830, 2016.
- [70] MEROLLA P, APPUSWAMY R, ARTHUR J, et al. Deep neural networks are robust to weight binarization and other non-linear distortions [J]. arXiv preprint arXiv:1606.01981, 2016.
- [71] HOU L, YAO Q, KWOK J T. Loss-aware binarization of deep networks [J]. arXiv preprint arXiv:1611.01600, 2016.
- [72] LIN Z, COURBARIAUX M, MEMISEVIC R, et al. Neural networks with few multiplications [J]. arXiv preprint arXiv:1510.03009, 2015.
- [73] IANDOLA F N, HAN S, MOSKEWICZ M W, et al. SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size [J]. arXiv preprint arXiv:1602.07360, 2016.
- [74] XIE S, GIRSHICK R, DOLLÁR P, et al. Aggregated residual transformations for deep neural networks [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1492-1500.
- [75] HOWARD A G, ZHU M, CHEN B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications [J]. arXiv preprint arXiv:1704.04861, 2017.
- [76] ZHANG X, ZHOU X, LIN M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6848-6856.
- [77] ZOPH B, LE Q V. Neural architecture search with reinforcement learning [J]. arXiv preprint arXiv:1611.01578, 2016.

- [78] BUCILUă C, CARUANA R, NICULESCU-MIZIL A. Model compression [C]//Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. 2006: 535-541.
- [79] BA L J, CARUANA R. Do deep nets really need to be deep? [J]. arXiv preprint arXiv:1312.6184, 2013.
- [80] HINTON G, VINYALS O, DEAN J. Distilling the knowledge in a neural network [J]. arXiv preprint arXiv:1503.02531, 2015.
- [81] ROMERO A, BALLAS N, KAHOU S E, et al. Fitnets: Hints for thin deep nets [J]. arXiv preprint arXiv:1412.6550, 2014.
- [82] KORATTIKARA A, RATHOD V, MURPHY K, et al. Bayesian dark knowledge [J]. arXiv preprint arXiv:1506.04416, 2015.
- [83] LUO P, ZHU Z, LIU Z, et al. Face model compression by distilling knowledge from neurons [C]//Proceedings of the AAAI Conference on Artificial Intelligence: volume 30. 2016.
- [84] CHEN T, GOODFELLOW I, SHLENS J. Net2net: Accelerating learning via knowledge transfer [J]. arXiv preprint arXiv:1511.05641, 2015.
- [85] ZAGORUYKO S, KOMODAKIS N. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer [J]. arXiv preprint arXiv:1612.03928, 2016.
- [86] GOOGLE. Tensorflow [EB/OL]. <https://www.tensorflow.org/>.
- [87] APACHE. Mxnet [EB/OL]. <https://mxnet.apache.org/>.
- [88] JIA Z, THOMAS J, WARSZAWSKI T, et al. Optimizing dnn computation with relaxed graph substitutions [J]. SysML 2019, 2019.
- [89] DING Y, ZHUL, JIA Z, et al. Ios: Inter-operator scheduler for cnn acceleration [J]. Proceedings of Machine Learning and Systems, 2021, 3.
- [90] HAO M, LIU Y, ZHANG X, et al. Labelenc: A new intermediate supervision method for object detection [C]//European Conference on Computer Vision. Springer, 2020: 529-545.

## 致 谢

时光如水，岁月如梭，转眼间，多姿多彩而又忙碌充实的硕士生涯即将落下帷幕。回首这个历程，在这三年的光阴里，科大不仅给了我学习知识的平台，也给我各方面的锻炼提供了机会。在这过程中，我收获了很多，不仅是知识上的丰富，也有能力和阅历的锻炼增长，马上将要走出校门，走上社会继续贡献自己。在这短短的时光，很多人给予了我无私帮助和辛勤的指导，无论是生活上还是学习上，即将毕业，我要把最真挚的感谢和祝福送给他们。

首先要感谢帮助过我的老师们，您们兢兢业业，无私奉献，只为教育我们成长，丰富我们学识。在您们的带领下，才有我的今天。还有我的导师王超老师和朱宗卫老师，在三年的硕士生涯中，老师给予我无数的教导、指引、关爱和帮助。特别是朱宗卫老师，全程指导本论文的研究和撰写过程。从论文的开题到中期检查到最终上交，无论是在实验研究方法还是在论文写作上，每一步都有朱老师的细心指导，给了我许多宝贵的意见，令我获益良多。同时在这个过程中，老师锲而不舍的精神，认真负责的态度和创新性的思维都给了我极大的鼓励与启发，激励着我追求卓越，奋力拼搏。所以，非常荣幸能在朱老师的指导下，顺利完成了硕士论文撰写，为自己的硕士生涯画上一个圆满的句号。在此向您们表达我衷心的感谢！

其次，还要感谢实验室各位同学和师弟们对我生活上和学习上的关心和帮助，感谢他们在我学习中对我的大力支持以及在论文写作中给予我的中肯建议，没有他们的无私帮助，我的论文实验不可能如此顺利的完成。再次真诚的感谢所有帮助过我的老师和同学和师弟们，在以后的生活中，我也一定会继续努力，不断前进。

最后，由衷的感谢我的家人和焦璐屹同学对我的理解和支持，他们的鼓励、关怀和期待，是我为之奋斗，不断前行的动力。

通过撰写本次硕士论文，我不仅对专业知识有了更深一步的理解，也提高了自己独立思考问题，独立解决问题的能力，培养了我对于科学实验的严谨态度。在实验中，我明白做任何事情都不会一帆风顺的，所以要细心，更要有耐心。当然，由于自己的知识所限，在论文中肯定有考虑不到的地方，希望各位评审老师能够耐心指出，让我能够更进一步。所以，衷心的感谢评审本论文和参加论文答辩的专家、教授，谢谢你们无私奉献，在此向你们致以最诚挚的敬意！

## 在读期间发表的学术论文与取得的研究成果

### .1 硕士期间发表的论文

- 1) FAN W, **HUANGHE L**, ZONGWEI Z, et al. Overcoming memory constraint for improved target classification performance on embedded deep learning systems [C]//IEEE International Conference on High Performance Computing and Communications. 2020. (HPCC)(CCF-C), 2020 年 9 月
- 2) **HUANGHE L**, ZONGWEI Z, CHENG J, et al. Subnetworkenabled taskaware swapping for efficient dnn on dramconstrained embedded devices [C]//Design Automation Conference.2021: WIP119s1. (DAC)(CCF-A), 2021 年 3 月

### .2 硕士期间参与的项目

- 3) 2018-2019 年, 参与搭建高性能计算实验室中心的“具有海量数据智能处理能力的高能效智能云计算平台”, 该平台最终成功申请成为中国科学技术大学超级计算中心苏州分中心。
- 4) 2019 年 12 月参加国家级比赛“海量光学遥感数据智能处理算法大赛”, 成功获得一等奖 30 万科研基金奖励, 带队老师朱宗卫。