密　　级：
分 类 号：
学校代码：10075
学　　号：20161269

硕士学位论文

基于剪枝的卷积神经网络压缩方法研究

学位申请人：　靳丽蕾
指 导 教 师：　杨文柱　教授
学 位 类 别：　工学硕士
学 科 专 业：　计算机科学与技术
院 系 名 称：　网络空间安全与计算机学院
答 辩 日 期：　二〇一九年六月

Classified Index：　　CODE:10075

U.D.C：　　NO.20161269

Dissertation for the Degree of Master

Pruning-based Compression Method for Convolutional Neural Network

Candidate: Jin Lilei

Supervisor: Prof. Yang Wenzhu

Academic degree category: Master of Engineering

Specialty: Computer Science and Technology

College: School of Cyber Security and Computer

Defend date: June, 2019

河北大学

学位论文独创性声明

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

作者签名：靳丽茜 日期：2019 年 6 月 3 日

学位论文使用授权声明

本人完全了解河北大学有关保留、使用学位论文的规定，即：学校有权保留并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。学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

本学位论文属于

1、限制公开 □ ，在_____年_____月_____日限制期满后适用本授权声明。

2、不限制公开 ☑。

（ 请在以上相应方格内打“ √ ” ）

作者签名：靳丽茜 日期：2019 年 6 月 3 日

导师签名：杨之柱 日期：2019 年 6 月 3 日

保护知识产权声明

本人为申请河北大学学位所提交的题目为（基于剪枝的卷积神经网络压缩方法研究）的学位论文，是我个人在导师（杨文柱）指导并与导师合作下取得的研究成果，研究工作及取得的研究成果是在河北大学所提供的研究经费及导师的研究经费资助下完成的。本人完全了解并严格遵守中华人民共和国为保护知识产权所制定的各项法律、行政法规以及河北大学的相关规定。

本人声明如下：本论文的成果归河北大学所有，未经征得指导教师和河北大学的书面同意和授权，本人保证不以任何形式公开和传播科研成果和科研工作内容。如果违反本声明，本人愿意承担相应法律责任。

声明人：靳丽蕾　　日期：2019年6月3日

摘 要

卷积神经网络（Convolutional Neural Network, CNN）作为典型的深度学习模型，在图像分类、目标检测和语义分割等诸多领域取得了一系列的研究成果。随着问题规模和复杂度的增加，CNN 的参数与计算量都在成倍增长，使其对训练与运行环境特别是设备性能的要求相应提高。由于手机等移动设备以及嵌入式设备存在计算能力、存储空间等诸多方面的约束，使得现有 CNN 无法在这些资源受限设备上进行很好的部署使用。针对以上问题，模型压缩是一种有效的解决方法，因此研究 CNN 模型压缩方法有利于促进 CNN 更广泛的应用，具有很强的理论与现实意义。网络剪枝是一种降低网络复杂性和过拟合的有效方法，已经被广泛应用于压缩 CNN 模型。论文的主要工作如下：

1. 提出了一种基于改进权重剪枝的动态剪枝方法

深度卷积神经网络中存在巨大的参数冗余，权重剪枝能有效减少网络中的冗余参数。该方法对小于设定阈值的权重进行剪枝，但是这种方法对于误剪的权重无法恢复，这会对最后的模型识别精度产生一定的影响。针对该问题，提出了一种基于改进权重剪枝的动态剪枝方法。首先对原始网络中权重绝对值小于阈值的权重进行剪枝，然后动态更新权重的重要性系数，对误剪的权重进行动态恢复。实验结果表明所提方法在 LeNet-5 和 VGG-16 网络上的识别精度损失值与权重剪枝相比分别减少了 0.11%和 0.25%。

2. 提出了一种结合动态剪枝和卷积核剪枝的混合剪枝方法

在模型压缩中，单独使用动态剪枝或卷积核剪枝对卷积神经网络进行压缩，压缩后的模型中仍然存在较多冗余参数。针对这一问题，提出了一种结合动态剪枝和卷积核剪枝的混合剪枝方法。首先，修剪对卷积神经网络整体精度贡献较小的卷积核；其次，对剪枝过的模型再进行动态剪枝实现进一步的模型压缩。在 MNIST 和 CIFAR-10 数据集上的实验表明，与只进行动态剪枝或卷积核剪枝相比，混合修剪方法可以达到更高的模型压缩比。所提出的混合剪枝方法将 LeNet-5 网络压缩了 12.90 倍，识别精度仅损失了 0.99%；将 VGG-16 网络压缩了 19.13 倍，识别精度仅损失了 1.32%。

关键词 卷积神经网络 模型压缩 权重剪枝 卷积核剪枝 混合剪枝

本研究得到河北省自然科学基金（编号：F2015201033, F2017201069）资助

Abstract

As a classical model in deep learning, convolutional neural network (CNN) has achieved a series of research results in various applications such as image classification, object detection and semantic segmentation. However, as the scale and complexity of problem increasing, CNN's parameters and calculation complexity increase exponentially. It demands better training environment and equipment. The promising performance is accompanied by significant computation cost and memory cost. It is difficult to be applied to resource-constrained devices such as mobile phones and embedded devices. Model compression is an effective method to resolve this problem. Therefore, performing research on CNN model compression is conducive to the application of CNN, and it has theoretical and practical significance. Pruning has been widely used to CNN model compression. The main works of this dissertation are as follows.

1. A dynamic pruning method based on improved weight pruning

There is huge parameter redundancy in the deep convolutional neural network, and weight pruning is an effective method to reduce the network complexity. Weight pruning is to prune the weight less than a threshold, but this method is irreversible for the mistakenly pruned weights and has an impact on the final network performance. To address this problem, a dynamic pruning method based on weight pruning is proposed. It first prunes the weights below a threshold, then dynamically updates the importance coefficients of the weights, and dynamically restores the mistakenly pruned weights. Compared with the original weight pruning, the experimental results show that the accuracy loss of the proposed dynamic method is reduced by 0.11% and 0.25% on LeNet-5 和 VGG-16 respectively.

2. A mixed pruning method combining weight pruning with filter pruning

The compressed convolutional neural network just using weight or filter pruning alone still exists redundant parameters. To address this problem, a mixed pruning method combining weight pruning with filter pruning is proposed. Firstly, filters having little contribution to the overall performance of the convolutional neural network can be pruned. Secondly, the pruned model is further compressed by dynamic pruning to achieve further model compression.

Experiments on MNIST and CIFAR-10 datasets demonstrate that the proposed approach is effective and feasible. Compared with weight pruning or filter pruning, the mixed pruning can achieve higher pruning ratio of the model parameters. For LeNet-5, the proposed method can achieve a pruning rate of 12.90×, with just 0.99% drop in accuracy. For VGG-16, it can achieve a compression rate of 19.13×, incurring 1.32% accuracy loss only.

Keywords Convolutional neural network Model compression Weight pruning Filter pruning Mixed pruning

目 录

第一章 绪 论

1.1 研究背景及意义

近几年，卷积神经网络（Convolutional Neural Network, CNN）作为深度学习中的经典模型，在图像分类、目标检测和语义分割等诸多领域取得了一系列的研究成果。自 2012 年深度学习算法在识别精度方面表现出巨大优势之后，各种深度学习模型便相继涌现。但这些模型在不断刷新计算机视觉任务精度的同时，其模型深度和参数也在成倍增长。由表 1-1[1-4]可以看出，随着模型复杂度的增加，参数越来越多，模型规模也越来越大，这对 CNN 的训练环境和设备性能有很高的要求。CNN 通常要求设备具备足够大的存储空间和快速的推理速度，这阻碍了它们应用于资源受限设备上。移动设备（比如手机）和嵌入式设备存在存储空间、计算能力等很多方面的限制，使得 CNN 很难在这些设备上高效运行。模型压缩的关键在于如何在保持现有网络模型性能基本不变的情况下，通过减小网络的计算量和网络模型存储，使 CNN 在资源受限的设备上可以高效运行。因此，模型压缩与加速近几年来引起了学术界以及工业界的极大关注。

表 1-1 几种典型的卷积神经网络模型

模型	层数（层）	模型规模（MB）	浮点运算（B）	参数（M）	ImageNet Top-5 错误率 (%)
AlexNet	8	>200	1.5	60	16.4
VGG	19	>500	19.6	138	7.32
GoogleNet	22	~50	1.566	6.8	6.67
ResNet	152	230.34	11.3	19.4	3.57

自 AlexNet[1]在 2012 ILSVRC 上取得突破性成果后，越来越多的研究人员开始研究 CNN 模型。最具代表性的 CNN 模型，如 VGG[2]，GoogLeNet[3]，ResNet[4]和 DenseNet[5]，大大提高了模型识别精度。其中，VGG 网络的前 10 层卷积层（CONV1-1 到 CONV4-3）占据超过 90% 的计算量，全连接层（FC）占大约 5-10% 的计算量和 95% 的参数量，这为研究卷积神经网络的压缩提供了统计依据。研究表明，网络用其原始参数的一小部

分便能进行有效重建，这说明 CNN 的结构存在较大冗余。模型参数过多不仅会浪费内存、需要大量计算，而且还会导致严重的过拟合问题。因此，针对如何减少 CNN 中的冗余参数问题，该领域的许多研究人员进行了大量相关研究，这对于推动 CNN 的更广泛应用有着重要意义。

1.2 国内外研究现状

1.2.1 卷积神经网络研究现状

在图像识别领域，传统的图像分类方法（比如 SVM 分类器）一般只适用于小型的分类问题，对于 ImageNet 等大型图像分类的任务则存在一定的局限性。卷积神经网络是一种经典的深度学习架构，在图像识别中取得了很大的成功，其灵感来自于生物视觉感知机制。1990 年，LeCun 等人[6]发表了建立 CNN 框架的开创性论文，后来对其改进设计了一种具有多层的 LeNet-5 神经网络模型，可用于手写数字的识别。该模型可以使用反向传播算法进行训练[7]，在几乎没有预处理的情况下就可以获得原始图像的有效表示。由于当时缺乏大量的训练数据以及受到计算能力的限制，神经网络的研究很长一段时间停滞不前。随着计算机技术的发展，神经网络逐渐吸引了众多研究学者的关注，提出了各种 CNN 模型用于解决大规模图像分类和视频分类等更复杂的问题。

Krizhevask 于 2012 年提出的 AlexNet 模型与 LeNet-5 类似，但却包含更多的层数。在此之后，相继出现了 VGG，GoogLeNet，ResNet 和 DenseNet 等代表性的深度网络模型。从这些网络结构的演变来看，其变化趋势是网络越来越深。随着 CNN 越来越深，网络优化难、梯度消失以及过拟合等问题凸显出来。针对这些问题，研究学者提出了许多应对措施，主要包括权值初始化、随机梯度下降、批量标准化、跳跃连接等技术。

1.2.2 卷积神经网络压缩研究现状

目前有很多卷积神经网络压缩方法，大致可以分为五类：新型网络模块设计、知识蒸馏、低秩分解、网络量化、网络剪枝。

1. 新型网络模块设计研究进展

基于技巧和经验可以设计出高效的和体积小的新型网络结构，比如 SqueezeNet [8]、

MobileNet[9]、ShuffleNet [10]和 FractalNets[11]等。在模型设计中，使用 1x1 和 3x3 等这样较小的卷积核，网络尺度信息涵盖的也会更多，另外还可以改变网络层的组合。

AlexNet 是一个 8 层的卷积神经网络，模型大小为 200M。但该网络依然有三个 FC 层，90%以上的参数都集中于这三层。NIN[12]是一个 4 层的网络结构，模型大小却只有 AlexNet 网络的 1/10，主要因为 NIN 模型除了使用全局平均池化（GAP）替换 FC 层外，还提出了 1×1 卷积核。通过 1×1 卷积核，实现了通道的降维、升维，另外实现了跨通道信息的交互以及信息整合。GoogLeNet 及其改进版本和 ResNet 中使用 1×1 卷积核实现了特征降维，从而减少模型参数、去除冗余数据。

GoogLeNet 网络达到 22 层的关键是 Inception 模块的提出和 1×1 卷积核的使用。Inception 模块将前一层特征通过不同大小的卷积核（主要是 1×1、3×3 与 5×5）得到的结果进行级联，实现了多尺度特征的提取，从而有效提高了训练速度、减少了参数量。GoogLeNet 之后，ResNet 中残差模块的设计克服了梯度弥散的问题，使更深的网络依然拥有强大的表达能力。与 VGG-16 网络相比，ResNet-101 的识别错误率和网络模型都要小很多。而在 ResNet 之后，网络结构转向了设计模块化的结构，比如 SqueezeNet[8]网络采用模块化思想设计了 Fire module，该模块减少了参数量的同时还实现了高性能；MobileNets[9]采用深度可分离卷积的思想，即：将一个标准卷积分解为一个深度卷积和一个逐点卷积，该模型将计算量和参数量减少了数十倍。

新型网络模块的设计与直接在训练好的网络上进行压缩不同，其设计思想主要是对传统的卷积层进行改变，比如 Fire module [8] 和 ShuffleNet bottleneck unit [10]模块。新模块有效减少了模型参数量和计算量，但设计新模块需要有一定的经验技巧。

2. 知识蒸馏研究进展

知识迁移是一种只需很少的训练集就可以快速学习新的复杂概念的主要机制。Caruana 等人[13]在 1997 年提出了利用知识迁移进行模型压缩，他们训练了带有伪数据标记的强分类器的压缩模型，可以实现原始网络的输出结果。与训练得到的模型相比，原始网络为较大网络。但是，该方法仅限于浅层模型，不适用于规模更大的网络。

知识蒸馏（Knowledge Distillation, KD）将复杂网络提取的有用信息转移到简单网络中，使学习的小型网络具有接近复杂网络的性能。Hinton 等人[14]引入了知识蒸馏框架，

由教师模型指导学生模型的学习，以获得比教师模型精简但性能相近的网络。Sau 等人[15]扩展了网络压缩的师生框架，用老师模型去训练学生模型，对老师模型的输出结果添加噪声，以改进学生网络的性能。Ba 等人 [16]认为网络不需要很深，设计浅的学生模型也可以保持与老师模型近似的性能。与 Ba 等人结论不同，Romero 等人[17]提出了 FitNets，使用 Hints 方式训练该网络的前半部分，然后采用传统的知识蒸馏来训练整个 FitNets，实验表明网络深度对于提高网络性能十分关键。

沿着知识蒸馏的方向有几个延伸，比如 Korattikara 等人[18]通过训练一个参数化的学生模型来逼近 Monte Carlo 老师模型。先前的研究用软标签概率表示知识，而 Luo 等人[19]使用更高隐藏层中的神经元来表示知识，该神经元保存尽可能多的信息作为标签概率，这种方式更加紧凑。Chen 等人[20]通过将知识从之前的网络瞬时转移到每个新的更深或更宽的网络来加速实验过程。Yim 等人[21]提出了 FSP 方法，它实际上将原始网络的特征图之间的相关度作为知识迁移到学生网络中，同时使用了 L2 损失函数。Zagoruyko 等人[22]提出了注意力转移（Attention Transfer，AT），通过使用这种类型的信息，学生网络模仿教师网络的注意力以提高自身的性能。徐喆等人[23]提出了带比例因子的 KD 算法以评估样本的类间相近关系，利用比例因子误差作为损失函数的一部分来调整网络参数。

基于知识蒸馏的方法使用训练好的复杂网络指导简单网络的训练，从而得到与复杂网络近似的能力。然而，KD 只支持从头开始训练且只能应用于损失函数为 Softmax 的任务，这阻碍了它的广泛使用。另外，模型假设有时太过严格，使模型性能不能和其他类型的方法竞争。

3. 低秩分解方法研究进展

CNN 计算量的绝大部分在于卷积运算，因此减少卷积层有利于网络的压缩与加速。低秩分解适用于卷积层和全连接层，CNN 的卷积核可被视为 4D 张量，低秩分解可以消除 4D 张量中存在的大量冗余。而全连接层可以视为 2D 张量，低秩分解同样适用。奇异值分解（SVD）[24]将复杂矩阵分解为几个小矩阵减少参数。Denton 等人[25]利用卷积核中存在的线性结构来导出近似值，大大减少了计算量，实现了较好的模型压缩比，模型压缩比为 2～3 左右，加速比也是 2～3 左右，同时保持识别精度损失值在 1%以内。

基于 Go 分解（Go Decomposition, Go Dec）：郭锴凌提出了一种鲁棒性更好的低秩矩阵分解方法——Go Dec+[26]，该算法利用半二次优化和贪婪双边范式进行求解。基于非负矩阵分解的思想，徐梦珂等人[27]对低秩矩阵恢复模型中的低秩矩阵进行了非负因子分解。

低秩卷积核可用于卷积的加速。比如，高维 DCT（离散余弦变换）变换由 1D DCT 变换构成。Rigamonti 等人[28]提出了学习可分离的 1D 滤波器，出于字典学习的想法。Jaderberg 等人[29]利用跨通道或卷积核冗余来构建空间域中秩为 1 的卷积核的低秩卷积核来加速卷积层，实现 CPU 上 2.5 倍加速而不损失精度，GPU 上 4.5 倍加速而精度仅下降不到 1%。Lebedev 等人[30]提出了简单二步法加速卷积层，首先使用非线性最小二乘法来计算低秩 CP 分解，将 4D 卷积核张量分解成少量秩为 1 的张量和的形式；第二步，使用 CP 分解将具有小卷积核的四个卷积层替换原始的卷积层。Tai 等人[31]提出了一种计算低秩张量分解的新算法，用于去除卷积核中的冗余。该算法可找到分解的准确的全局优化器，而且比迭代方法更有效。

低秩分解可用于模型压缩和加速。然而，由于该方法包含分解操作，有昂贵的计算代价，因此实现起来并不简单。另外，因为不同层具有的信息不同，为此全局压缩是很重要的。但当前的方法都是逐层进行低秩近似，不能进行全局的参数压缩。最后，与原始模型相比，低秩分解需要模型多次的再训练才能实现收敛。

4. 网络量化

网络量化的主要思想是通过减少表示每个权重所需的比特数来压缩 CNN 模型。网络量化研究的一个方向是权值共享。权值共享的主要思想是多个权重连接共享同一个权重，HashedNets 模型[32]将权重连接随机的分组到散列桶中通过低代价的散列函数，同一个哈希桶中的权重连接共享一个参数值。Han 等人[33]提出通过权重量化共享一些权重，然后使用 Huffman 编码进一步压缩。

网络量化研究的另一个方向是低比特量化。Vanhoucke 等人[34]研究表明，参数的 8 比特量化可以使网络显著加速且识别精度损失较小。Gupta 等人[35]的工作在基于随机四舍五入的 CNN 训练中使用 16 比特定点表示，显著降低了内存使用和浮点运算。定点表示的数值范围有限，因此采用动态定点来表示网络的权重和激活函数。蔡瑞初等人[36]首先修剪对识别精度影响小的权重，然后使用动态定点方法对权重与激活函数进行处理。

由于量化函数不可导，因此大多数低比特量化方法通过直通估计来近似权重的梯度。所以 Leng 等人[37]把低比特量化方法转为 ADMM 可优化的目标函数。

这个方向的最新发展趋势是网络的二值化。二值化[38]是一种特殊的量化方式，对网络二值化只有两种取值，将权值和激活值二值化为 1 或者- 1 ，显著减少了存储空间和计算量。例如，BinaryConnect[39]将权重量化到 1 比特，BinaryNet [40]对前者进行扩展，将激活值也量化到 1 比特，XNORNet[41]提出了两个网络 Binary-Weight-Networks(BWN)和 XNOR-Net，BWN 只将权重量化为二值，而 XNOR 将权重和激活值都量化为二值。当处理 ResNet、GoogleNet 等大型 CNN 时，这种二进制网络的性能显著下降。

由于移动设备和嵌入式设备存在存储空间和计算能力的限制，CNN 难以应用在这种环境下。量化方法对于降低 CNN 的参数量和计算量十分有效，但在大型识别任务上模型识别精度损失严重。

5. 网络剪枝

网络剪枝是一种能降低网络复杂性和解决过拟合问题的有效方法，其目的是通过修剪不重要的权重或卷积核，以压缩 CNN 模型。

网络剪枝一般是在预先训练好的 CNN 模型中修剪冗余的参数。例如，Han 等人[33]提出深度压缩，修剪对准确率影响不大的权重连接。无数据参数剪枝[42]不需要任何训练或验证数据，仅修剪参数确保网络的整体结构保持不变，从而实现快速微调等操作。基于软权重共享[43]的方法在重新训练过程中实现了量化和剪枝。通过软权重共享和分解 Dirac 后验，重新训练预训练的 CNN。黄聪等人[44]利用权重的相似性对网络的全连接层进行剪枝，并提出了超参数 α 来控制剪枝比例，实现了 90%以上全连接层单元的剪枝。韩云飞等人[45]提出了一种结合网络剪枝与参数共享的压缩方法，为 CNN 模型在资源受限设备上的部署提供了可行方案。

现有的卷积核剪枝方法通常首先根据一些标准修剪卷积核，然后通过最小化损失函数来重新训练剪枝后的模型。根据使用损失函数的不同，可分为两类：第一类考虑整个模型的损失，因此直接修剪所有卷积核。例如，Weight Sum[46]通过计算权重绝对值的和来衡量卷积核的相对重要性，与该层中其他卷积核相比，权重绝对值的和越小，往往产

生的特征越弱。Hu 等人[47]根据神经元激活值的数据来修剪冗余的神经元，可以迭代的修剪激活值较低的神经元，这些神经元对最后的结果贡献很小，不会影响模型的性能。有些研究通过把正则化添加到损失函数中，将剪枝和再训练结合到一个框架中，比如 Liu 等人[48]提出的方法对批量归一化（BN）层中的缩放因子添加 L1 正则化，通过 L1 正则化将 BN 缩放因子的值推向零，以判断不重要的通道，因为每个缩放因子对应一个特定的卷积通道。第二类考虑逐层损失，因此逐层修剪卷积核。比如 He 等人[49]根据 LASSO 回归选择重要的通道，修剪多余的通道。ThiNet[50]使用贪婪选择算法选出不重要的卷积核，将卷积核剪枝问题定义成优化问题，根据 i+1 层的统计数据对 i 层进行剪枝。

卷积核剪枝的优点在于网络模型的加速以及存储空间的减少，但目前大多数方法往往以分层固定的方式进行卷积核的剪枝，且只能以局部方式逐层进行。针对这个问题，Lin 等人[51]提出了一种新颖的全局动态剪枝（GDP）方案，首先通过基于每个卷积核的先验知识的全局判别函数来全局修剪每层中不重要的卷积核，然后动态更新剪枝后的稀疏网络中卷积核的显著性以恢复误剪的卷积核，最后重新训练以恢复模型的准确度。Yu 等人[52] 提出了神经元重要性分数传播算法（NISP）来从网络整体考虑神经元的重要性。

网络剪枝方法的灵感来自 Oracle 剪枝方法，其主要思想是在模型中选择不重要的参数并将其修剪而不影响模型的效果。剪枝的关键是找到一种评估参数重要性的有效方法。目前，网络剪枝是最简单有效的卷积神经网络压缩方法。

综上所述，在保证网络压缩效果的同时尽量不损失识别精度，是目前卷积神经网络压缩研究中的努力方向。

1.3 研究内容和目标

1. 一种基于改进权重剪枝的动态剪枝方法

深度卷积神经网络中存在参数冗余的问题，权重剪枝是减少网络中冗余参数的有效方法；但是这种方法对于误剪的权重无法恢复，因此对最后的模型识别精度会产生一定的影响。针对这个问题，提出了一种动态剪枝方法。首先对原始网络中权重绝对值小于指定阈值的权重进行剪枝。然后动态更新权重的重要性系数，对误剪的权重进行动态恢复，从而降低因误剪而造成的精度损失。最后通过重新训练保留下的权重连接，以恢复模型识别精度。

2. 一种结合动态剪枝和卷积核剪枝的混合剪枝方法

在模型压缩中，单独使用动态剪枝或卷积核剪枝对卷积神经网络进行压缩，压缩后的模型中仍然存在较多冗余参数。针对这一问题，提出了一种结合动态剪枝和卷积核剪枝的混合剪枝方法。首先，修剪对卷积神经网络整体精度贡献较小的卷积核；其次，对剪枝过的模型再进行动态剪枝实现进一步的模型压缩。剪枝会对模型性能造成影响，通过重新训练来恢复模型的精度。

1.4 文章组织结构

根据上述工作内容和目标，各章节内容如下:

第一章介绍了模型压缩技术的研究背景与意义，阐述了卷积神经网络与模型压缩技术的研究现状。模型压缩方法分为五类：新型网络模块设计、知识蒸馏、低秩分解、网络量化、网络剪枝，并对各类方法的发展、特点和近期成果做了总结。针对卷积神经网络中参数存在冗余的问题，对网络剪枝中存在的问题及现阶段的研究进展进行了分析与总结。提出了本文的研究内容和目标。

第二章主要介绍了卷积神经网络（CNN）压缩的相关知识。首先介绍了模型压缩的必要性与可能性；然后介绍了 CNN 的基本组成，常见的卷积神经网络结构，简要说明了 CNN 的训练过程和正则化方法；最后对目前卷积神经网络压缩方法（新型网络模块设计、知识蒸馏、低秩分解、网络量化、网络剪枝）的基本原理做了简要介绍。

第三章提出了一种权重动态剪枝方法。首先对权重剪枝方法的基本原理进行了简单介绍，并指出了存在的问题。然后，介绍了针对权重剪枝问题提出的动态剪枝。最后是实验结果与分析，在 LeNet-5 和 VGG-16 网络上验证其改进效果。

第四章提出了一种混合剪枝方法。首先介绍了混合剪枝方法的框架，然后介绍了卷积核剪枝的原理，最后实验在 LeNet-5 和 VGG-16 网络上进行测试，验证混合剪枝方法的有效性。

第五章为总结与展望，对提出的动态剪枝方法与混合剪枝方法进行了总结，并对后续研究工作进行了展望。

最后是本文的致谢和在研究生期间取得的研究成果。

第二章　卷积神经网络压缩

2.1　卷积神经网络压缩的必要性与可行性

尽管卷积神经网络（CNN）在计算机视觉、自然语言处理、语音识别等诸多领域都取得了显著的成就，但在实际应用中往往受限于其巨大的参数量。以经典 VGG-16 网络为例，其参数数量达到了约 1.4 亿个，占用逾 500MB 存储空间，一张图像的识别任务需要进行 309 亿次浮点数运算（Floating-point operation，Flops）。如此巨大的存储代价以及计算开销，严重制约了深度神经网络在手机、嵌入式等小型设备上的应用。因此，如何在资源受限的设备上运行 CNN，是当前学术界与工业界研究的热点问题之一。

许多研究表明，卷积神经网络的参数存在冗余的问题。比如 Denil 等人[53]的研究发现，只给定原始参数的一个子集，便能有效地重构出剩余的参数。王征韬[54]对特征图进行可视化，表明了网络中参数存在冗余。曹文龙等人[55]对 CNN 存在的参数冗余以及计算复杂度进行了分析。

综上所述，卷积神经网络压缩不仅是必要的，也是可行性的。

2.2　卷积神经网络

2.2.1　卷积神经网络的基本组成

卷积层（Convolutional layer, CONV）、池化层（Pooling layer）和全连接层（Fully connected layers, FC）是卷积神经网络的主要组成部分。

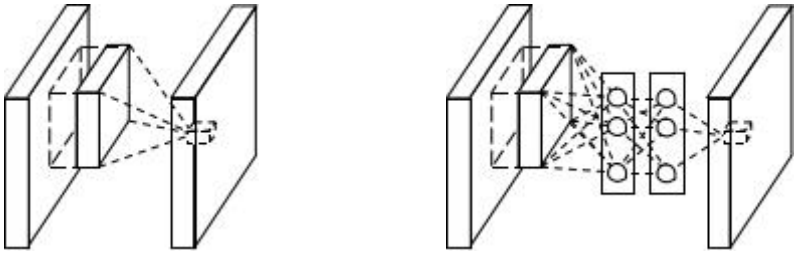
CNN 中的第一层是卷积层，由多个卷积核（filter）组成。卷积核可被看作一个滑动窗口，当输入某张图像时，卷积核在图像上从左至右，从上至下进行滑动，每次滑动对应的区域称为局部感受野，最后得到特征图。滑动操作是卷积核对图像进行卷积运算的过程，可以学习到输入图像的特征表示。不同的卷积核学习不同的特征，使用多个卷积核则可以学到更加丰富的特征。

卷积层之后一般是池化操作。通过池化操作来聚合输入特征通道的每个小区域内的信息，从而得到池化层，然后对结果进行下采样。池化的目的是保持平移、旋转、尺度的不变性，并保留主要特征且实现了降维。

在卷积层和池化层之后是全连接（FC）层。卷积运算与池化操作学习到的特征（局部信息）通过 FC 层整合为一个特征向量（即生成全局的语义信息），以实现最后的分类。

2.2.2 常见卷积神经网络结构

1. Network In Network 网络



(a) 线性卷积层　　(b) MLP 卷积层

图 2-1　线性卷积层和 MLP 卷积层的比较

CNN 的高级特征由某些低级特征组合而来，Lin 等人[11]受此启发提出 Network In Network（NIN）。其主要思想是“micro network”结构可以代替广义线性模型（GLM），并且在提高本地模型的抽象能力方面发挥重要作用。多层感知器（MLP）是对卷积层改进的一种算法，与线性卷积层相比在感受野中执行更复杂的计算。全局平均池化层与全连接层的不同之处在于前者更易于解释和有意义。线性卷积层和 MLP 卷积层的比较如图 2-1 所示。图 2-1 (a)显示了传统的线性卷积层，模型的抽象层次很低；图 2-1 (b)为代替传统卷积层的 MLP 卷积层，因为 MLP 是通用的近似函数，可以在 CNN 中使用反向传播进行训练。使用 ReLU 激活函数的 MLP 卷积层可以定义为：

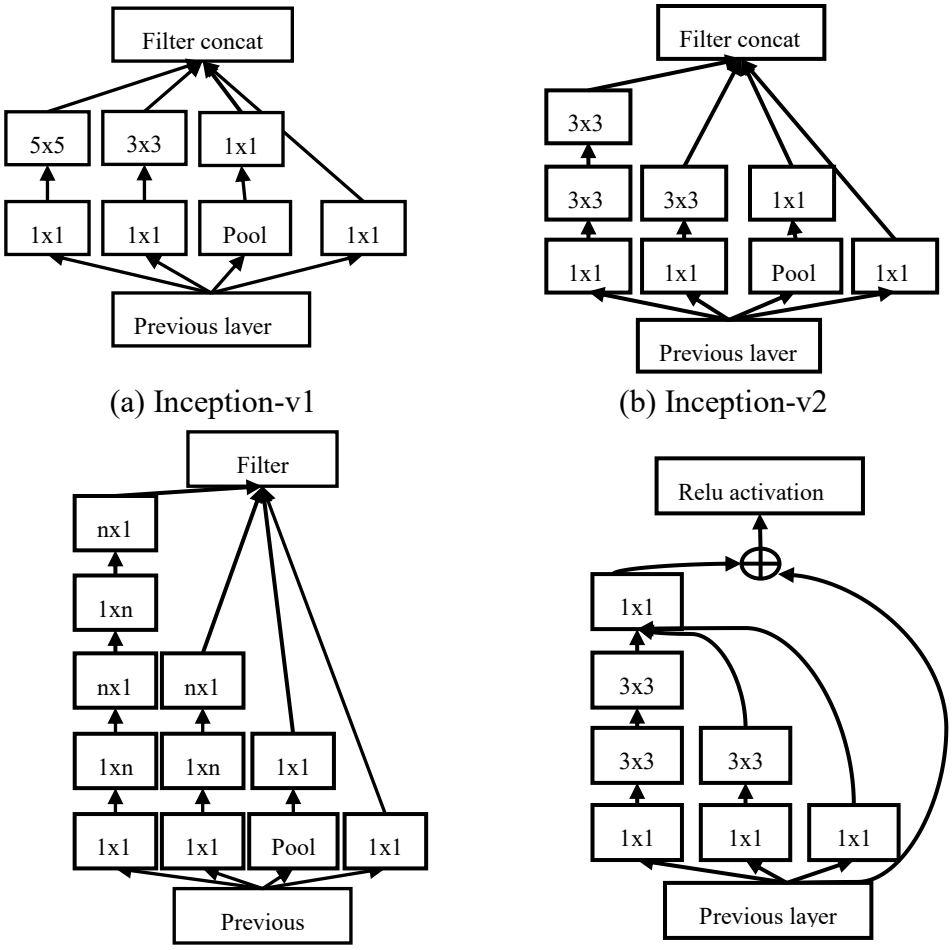
$$f_{i,j,k_n}^{1} = max(w_{k_n}^{T} f_{i,j}^{n-1} + b_{k_n}, 0) \tag{2.1}$$

其中，$n \in [1, N]$，N 代表 MLP 卷积层的层数，(i, j) 是特征图中的像素索引，$f_{i,j}^{0}$ 等价于 $x_{i,j}$，$x_{i,j}$ 代表以位置 (i, j) 为中心的输入块，k 用于进行特征图通道的索引。

2. Inception 模块

Inception 模块由 Szegedy 等人[3]引入，通过使用不同大小的卷积核来获得不同的特征，以近似最佳的稀疏结构。输入图像通过不同的卷积运算与池化操作获得不同的信息，并行处理这些运算并结合所有结果以获得更好的图像表征。如图 2-2(a)所示，Inception 模块包括一个池化操作和三种类型的卷积操作，在 3×3 与 5×5 卷积之前添加 1×1 卷积

以降维。在 Inception-v1 之后，提出了提高识别精度和降低计算复杂度的改进方法，图 2-2(b)(c)(d)为 Inception 模块的改进版本。如图 2-2(b)所示，Inception-v2[56]将 5×5 卷积替换为两个连续的 3×3 卷积以提升计算速度。如图 2-2(c)所示，Inception-v3[57]将一个二维卷积(n×n)分解为两个一维卷积(1×n 和 n×1)。如图 2-2(d)所示，受 ResNet[4]的启发，Inception-v4[58]将 Inception 模块与跳跃连接结合。



(a) Inception-v1　(b) Inception-v2

(c) Inception-v3　(d) Inception-v4

图 2-2　Inception 模块

3. 残差网络

随着网络深度的增加，准确率变得饱和，然后迅速下降，也就是说在深度适当的模型中添加更多层会导致更高的训练错误率。为此，He 等人[4]引入深度残差学习框架（ResNet）来解决网络加深带来的准确率下降问题（即退化问题）。如图 2-3 显示了 ResNet[4]的残差模块，这种残差学习结构通过前向神经网络与跳跃连接（shortcut connections）实现。跳跃连接只是执行恒等映射（identity mapping），它们的输出被添加

到堆叠层的输出中。Shortcut connections 不增加额外参数，网络的计算复杂度也没有增加，网络仍然能通过反向传播由 SGD 端到端的进行训练。ResNet 网络学习的是一个残差函数 F(x) = H(x) - x，只要 F(x) = 0，就构成了一个恒等映射 H(x) = x。

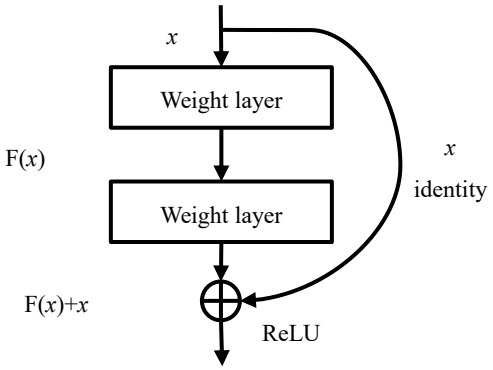


图 2-3　ResNet 的残差学习模块

4. DiracNets 网络

具有跳跃连接的 ResNet 应用在图像识别、目标检测等诸多领域，并表现出优异的性能，证明了 ResNet 残差学习框架的有效性。但 ResNet 也存在一些问题：首先是特征重用问题，上层激活后可能无法学习到有用的表示； 另外，把 ResNet 变深比变宽更有效，但网络到达一定的深度后，无法再提升识别精度；最后，网络实际的深度不明确，可能由最短路径决定。因此，使用 Dirac 权重参数化[59]的方法代替跳跃连接，它可以在没有跳跃连接的情况下训练非常深的普通网络，并实现几乎相同的性能。

使用 Dirac 对任何输入进行卷积会产生相同的输出，让 I 成为离散卷积算子的代数中的恒等式，即将 I 与输入 x 卷积得到相同的输出 x：

$$I * x = x \tag{2.2}$$

其中∗代表卷积操作。在二维的情况下，卷积可以表示为矩阵乘法，因此 I 是一个单位矩阵或一个克罗内克 δ 函数。将公式(2.2)推广到卷积层的情况，即公式(2.3)：

$$y = \widehat{W} * x \tag{2.3}$$

其中输入$x \in R^{M, N_1, N_2, \ldots, N_L}$由空间维度（$N_1$, N_2, …, N_L）的 M 个通道组成，权重$\widehat{W} \in R^{M,M,K_1, K_2, \ldots, K_L}$（组合 M 个卷积核），输入 x 与权重 W 卷积以产生 M 个通道的输出 y，即公式(2.3)。对公式(2.3)的权重$\widehat{W}$进行参数化得到公式(2.4)：

$$\widehat{W} = diag(a)I + W \tag{2.4}$$

其中$a \in R^M$是在训练期间可学习的缩放向量，W是权重向量。I是一个单位矩阵或一个克罗内克 δ 函数，也叫 Dirac delta 变换。a 的第 i 个元素对应于 W 的第 i 个卷积核的缩放。当 a 的所有元素都接近零时，公式(2.4)会变为简单的线性层$W * x$；当 a 的所有元素高于 1 且 W 很小时，Dirac 占主导地位，输出就与输入相同。

2.2.3　卷积神经网络的训练

CNN 的训练涉及两个过程，即前向传播（Forward propagation）与反向传播（Back propagation，BP）。前向传播是从输入层开始，将数据由前向后传播的过程，即经过 CONV 层、激活函数、Pooling 层以及 FC 层的处理得到最后的输出，通过损失函数（一般采用 cross entropy loss）得到预测值与目标值之间的误差。当前向传播得出的结果与预期不相符时进行反向传播；反向传播是通过损失函数训练网络、调节参数值的过程。

1. 前向传播

CNN 前向传播算法如表 2-1 所示。对于给定输入图像 x，经过 CONV 层的卷积运算、Pooling 层的池化操作，学习到高级抽象的特征，然后将学习到的特征传入全连接层进行信息整合，最后利用 Softmax 得到图像分类结果。

CONV 层的前向传播可定义为：

$$a^l = ReLU(\mathrm{z}^l) = ReLU(a^{l-1} * W^l + b^l) \tag{2.5}$$

其中 l 是网络层数，a^l代表输入张量，*代表卷积运算，W 是卷积核，b 代表偏置向量，激活函数使用 ReLU。将卷积核W^l和输入图像a^{l-1}进行卷积操作，加上偏置量 b，再通过激活函数 ReLU 得到输出。

Pooling 层的前向传播可定义为：

$$a^l = Pool(a^{l-1}) \tag{2.6}$$

pool 操作是指按照池化区域大小（Pooling size）k 和池化方式（一般使用最大池化或者平均池化）将输入张量变小的过程。将上一层 CONV 层提取的特征作为输入传到池化层，通过池化操作（Pooling）进行降维，目的是为了避免过拟合。

FC 层的前向传播可定义为：

$$a^l = \sigma(z^l) = \sigma(W^l * a^{L-1} + b^l) \tag{2.7}$$

通过 CONV 层与 pooling 层提取出的特征传到 FC 层中进行整合，然后分类。FC 层后，最后是 Softmax 输出层，表达式可表示为：

$$a^L = Softmax(z^L) = Softmax(W^L * a^{L-1} + b^L) \tag{2.8}$$

其中，判别函数使用 Softmax，得到最后分类结果。

表 2-1　CNN 前向传播算法

算法 1　CNN 前向传播算法
输入：输入图像, 层数 L, filter size K，padding P, stride S, pool size k, bias b 输出：CNN 的输出 a^L 1. get the input tensor a^l according to the P; 2. Initialize：W, b; 3. for l=2 to L-1 4. if layer l is CONV layer: 5. output $a^l = ReLU(z^l) = ReLU(a^{l-1} * W^l + b^l)$; 6. if layer l is Pooling layer: 7. output $a^l = Pool(a^{l-1})$; 8. if layer l is FC layer: 9. output $a^l = \sigma(z^l) = \sigma(W^l * a^{L-1} + b^l)$; 10. if layer l is output layer: 11. output $a^L = Softmax(z^L) = Softmax(W^L * a^{L-1} + b^L)$; 12. end for

2. 反向传播

卷积神经网络以及其它的深度学习模型都依赖损失最小化来学习模型参数，但神经网络模型不仅是非凸函数而且比较复杂，这会带来优化求解的困难。在这种情况下，深度学习模型采用随机梯度下降法[60, 61]和误差反向传播进行模型参数更新。针对 ILSVRC 分类或检测等大规模应用问题，一般采用批量随机梯度下降法[62]。

当前向传播预测的结果与真实结果不一致时，通过反向传播来训练网络。在求得预测值与目标值的总损失值（loss）后，随机梯度下降法优化参数（W, b）的值以尽可能最小化损失函数的值，求得最终 W 和 b 的值。

CNN 的训练过程为（如图 2-4 所示）：

步骤 1. 网络进行权重的初始化；

步骤 2. 前向传播：输入 x 经过 CONV 层、Pooling 层、FC 层和 Softmax 得到预测值；

步骤 3. 求出网络的预测值和目标值之间的损失值（loss）或误差；

步骤 4. 反向传播：当损失值超出容许范围时，将损失值传回网络中。求网络层中

神经元的损失值，依次求得 FC 层，Pooling 层，CONV 层的误差，各层的损失值相当于对网络总损失产生的影响。当损失值在容许范围内时，网络训练结束，否则转步骤 5。

步骤 5. 根据求得的损失值进行权重的更新。然后再进入到步骤 2，不停迭代，直到预测值和目标值之间的损失值在容许范围内，则完成训练。

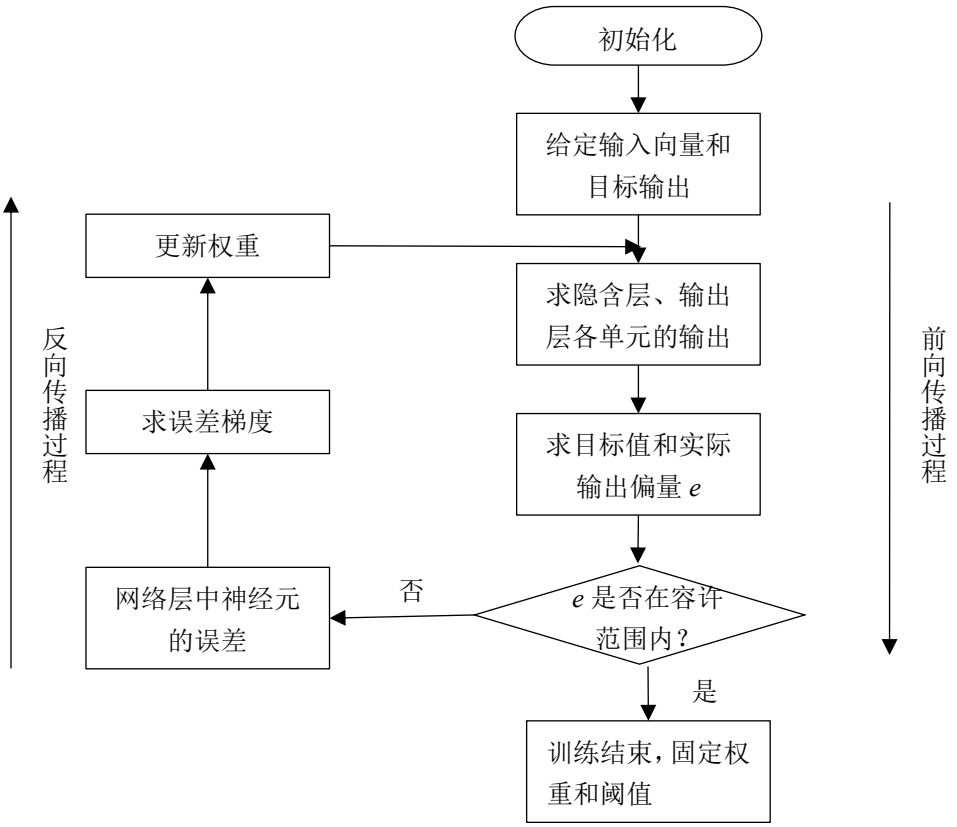
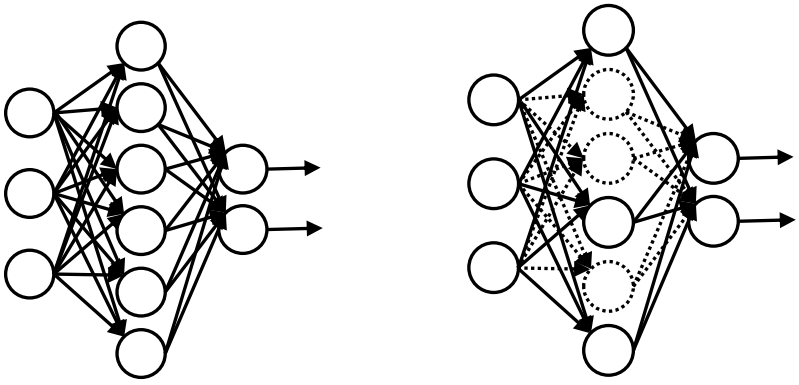


图 2-4　卷积神经网络的训练过程

2.2.4　正则化

1. Dropout

在网络训练过程中，Dropout[63]技术按照一定概率将网络中的神经元暂时停止工作。当网络参数太多，而训练样本又太少时，训练出来的模型容易过拟合，此时使用 Dropout 能够有效缓解过拟合的问题，一定程度上可达到正则化的效果。Dropout 只发生在卷积神经网络的训练阶段，CNN 的预测和测试阶段则不使用 Dropout。如图 2-5 所示为标准的神经网络与使用 Dropout 的神经网络模型。



（a）标准网络　　（b）使用 Dropout 的网络

图 2-5　标准网络与使用 Dropout 的网络

2. 数据增强（Data Augmentation）

在图像分类任务中，增加样本的方法之一是图像数据增强。如果初始数据集较小，就不能准确地进行网络训练，会对模型最后的识别精度产生一定的影响。图像数据增强就是对原始图像进行某种处理，让有限的图像数据产生更多的等价数据，在一定程度上提升模型的性能。常用的图像数据增强方法有图像翻转、平移、旋转、缩放等。

3. L1 和 L2 正则化

常用的正则化项包括 L1 和 L2 正则化项，都有防止过拟合的作用。

对模型参数 w 的 L2 正则项为:

$$\Omega(\theta) = \alpha||w||_2^2 \tag{2.9}$$

其中，$||w||_2^2$为权重参数的平方和，α的值一般为 1/2。L2 正则也经常被称作权重衰减。L2 正则化使得参数变小，所以它的抗扰动能力比较强。带 L2 正则化项的损失函数：

$$C = C_0 + \alpha\|w\|_2^2 \tag{2.10}$$

使用 L2 正则项的解不具有稀疏性（sparsity）。在求解过程中，L2 正则化对最后性能贡献不大的权重有抑制作用，使其变小，因此网络的复杂度变低，实现了正则化，可避免过拟合。

权重 w 的 L1 正则项为:

$$\Omega(\theta) = \|w\|_1 = \sum_i |w_i| \tag{2.11}$$

带 L1 正则化的损失函数：

$$C = C_0 + \alpha\|w\|_1 \tag{2.12}$$

其中，$\|w\|_1$为权重参数的绝对值。使用 L1 正则化，最后得到的解有很大概率为 0，因此具有稀疏性。L1 正则化使得网络中的权重 w 尽可能为 0，适用于选择出重要的特征。

2.3　卷积神经网络压缩方法

对卷积神经网络压缩的方法研究后，简要介绍五类方法的基本思想：新型网络模块设计、知识蒸馏、低秩分解、网络量化和网络剪枝。

2.3.1　新型网络模块设计

新型网络模块设计的主要思想是设计更高效的计算方式（主要是在卷积方式上进行改变），从而有效减小网络的参数且拥有良好的性能。目前有关设计网络模块的研究越来越多，如 SqueezeNet、MobileNet[9]、ShuffleNet [10]和 FractalNets[11]等。

以 SqueezeNet 网络为例进行介绍。SqueezeNet [8]中引进了 Fire Module(如图 2-6)，该模块包含两部分，squeeze 层与 expand 层，即将传统的一层卷积层设计为 squeeze 和 expand 两层。其中，与 GoogLeNet 思想类似，squeeze 层中仅使用 1×1 卷积核，降低 expand 层的输入通道数，达到减少参数数量的目的。expand 层中使用 1×1 和 3×3 卷积核的组合来提取特征，将 1×1 和 3×3 卷积得到的特征图在通道维度进行拼接。VGG 网络是堆叠的使用卷积操作，与 VGG 类似，SqueezeNet 网络的设计思想是使用堆叠的 Fire Module。

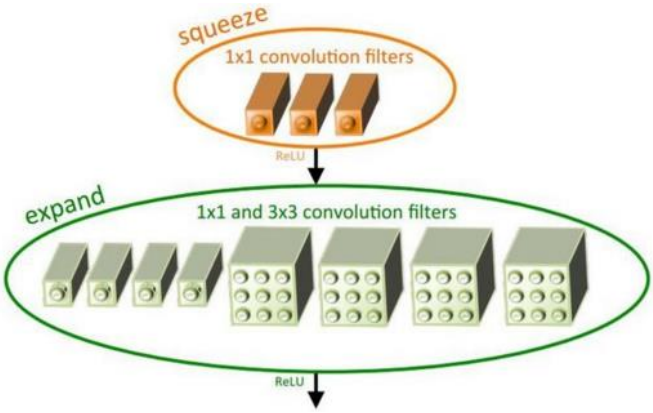


图 2-6　Fire Module [8]的基本组成

假设 Fire module 输入的特征图为 H×W×M。首先，特征图（H×W×M）经过 squeeze 层，然后得到 S1 个特征图（S1 < M ）。其次，expand 层使用 1×1 和 3×3 的卷积核，分别与 squeeze 层的特征图（H×W×S1）进行卷积运算，然后将运算结果进行拼接，最后得到 fire module 的输出特征图（H×M×(e1+e3)）。s1，e1，e3 是 fire module 的三个可

调的超参数。其中，s1 代表 squeeze 层中 1×1 卷积核的个数，e1 代表 expand 层中 1×1 卷积核的个数，e3 代表 expand 层中 3×3 卷积核的个数，

2.3.2 知识蒸馏

知识蒸馏[14]的主要思想是：通过采用预先训练好的复杂模型（Teacher model）的输出作为监督信号去训练另外一个简单模型（ Student model），从而在减小模型大小和计算量的同时，使简单网络能够具备与复杂网络近似的性能。

如图 2-7 所示，软目标（soft target）：来自复杂模型的预测输出，对应的带有温度参数 T 的目标，即在 Softmax 层使用合适的温度参数 T，最后训练得到的概率称为“软目标”。硬目标（hard target）：正常网络训练最后得到的概率称为“硬目标”。总损失（total loss）：软目标与硬目标的结合，作为小网络训练的损失函数。交叉熵损失（cross entropy loss）函数，用来衡量模型预测值和真实值的误差（loss）。

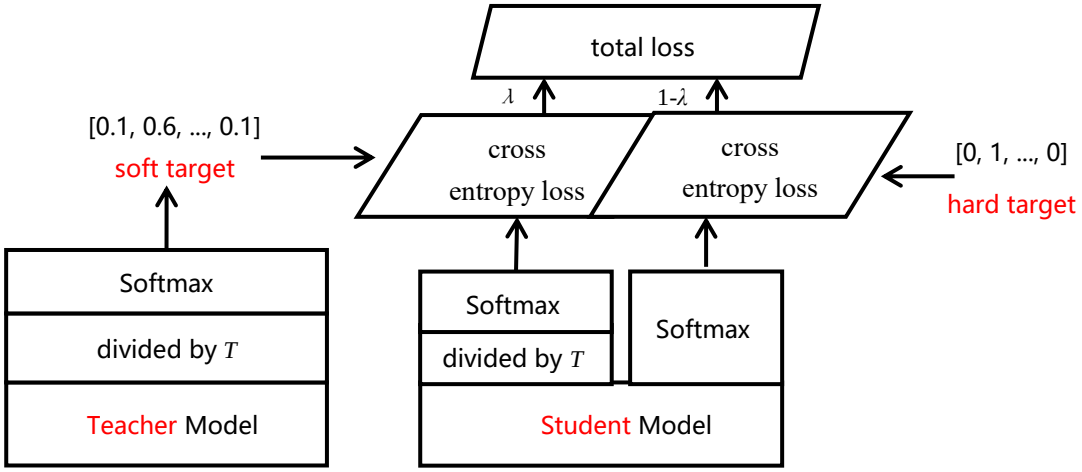


图 2-7 知识蒸馏示意图

知识蒸馏过程：

步骤 1. 训练复杂模型：先用硬标签训练复杂模型。

步骤 2. 计算软目标：利用训练好的复杂模型来计算软目标，在复杂模型中，对所有 Softmax 层的输入 Logits 值除以一个温度参数 T 再经过 Softmax 输出，获得软化的概率分布（0~1），即软目标作为监督信息，取值分布较为缓和，可定义为公式(2.13)。

步骤 3. 得到软目标之后，与硬目标加权结合来计算简单模型训练时的损失（total loss），通过加权系数 λ 来调节两个损失函数的比重。若软目标交叉熵的 λ 越大，表明

越依赖复杂模型的贡献，但训练后期需要适当减小软目标的比重，让硬目标帮助鉴别困难样本。

$$q_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}} \tag{2.13}$$

其中，z_i 表示 Softmax 第 i 类的输入 Logits，q_i 表示 Softmax 第 i 类的输出软目标。公式(2.13)是在 Softmax 基础上增加了一个温度参数 T，用来控制预测概率的平滑程度。T 的值越大，分布越缓和；T 的值减小，易放大错误分类的概率，引入不必要的噪声。

2.3.3　低秩分解

低秩分解把一个卷积神经网络的参数矩阵通过张量分解，用它的低秩特性做逼近。目前低秩分解方法有：奇异值分解(SVD) 、CP 分解、Tucker 分解、Tensor Train 分解和 Block Term 分解。这些分解方法都是将初始权重矩阵通过分解用多个低秩矩阵来近似，极大减少了矩阵计算量。例如可以通过 SVD[24]得到原矩阵的低秩近似，有效减少矩阵运算的计算量。

对于 SVD，从任意为 $m\times n$ 的矩阵 A 开始。存在正交矩阵 U 和 V 以及对角矩阵 $\sum$，使得$A = U\sum V^T$。在这种情况下，U 是 $m\times m$ 且 V 是 $n\times n$，因此 $\sum$ 是与 A 具有相同维度的矩阵。$\sum$ 的对角线值为 $\sum_{ii} = \sigma_i$，可以被安排为非负值并且按顺序递减。非负值被称为 A 的奇异值。对于矩阵 A，U 和 V 的列被称为左和右奇异向量。通过 SVD 选择正交基，以便转换由最简单形式的矩阵表示，即对角线。

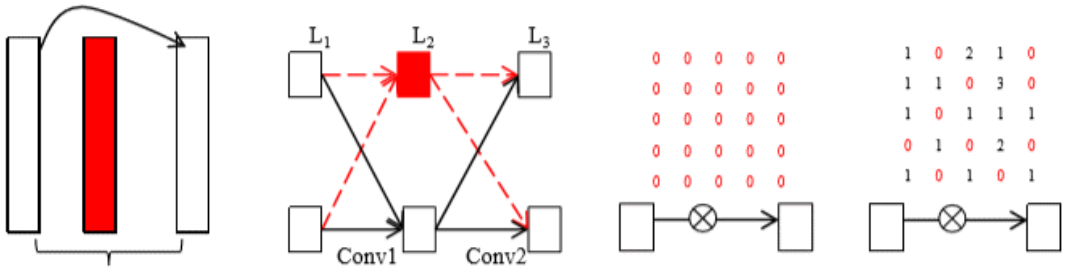
2.3.4　网络量化

网络量化的主要思想是：用更少的比特数近似表示权重所用的比特数，以实现卷积神经网络的压缩。量化模型有 Deep Compression，BinaryNet，TenaryNet 等。网络模型的参数一般使用 32 bit 浮点数来表示，但实际无需这样高精度的表示。因此，使用量化减少比特数，使用 8 bit 整数近似地表示 32 bit 浮点数进行存储和计算，从而降低模型的存储空间。

Deep Compression[33]主要分为三个部分：剪枝，量化，哈夫曼编码。其中，量化将表示每个连接的位数从原来使用的 32 bit 压缩到 5 bit 等更小的值。若输入神经元与输出神经元各有 4 个，权重与梯度则为 4×4。权重被量化为 4 个区间，即只需存储 4 个码字和 16 个 2 bit 的索引。在码字更新期间，相同区间内的梯度进行相加，乘以学习速率并

从上次迭代中的共享质心中减去。

2.3.5 网络剪枝



(a) 层间剪枝　　(b) 通道剪枝　　(c) 卷积核剪枝　　(d) 权重剪枝

图 2-8　按剪枝粒度分为四类剪枝[64]

网络剪枝的主要思想是通过一定的评判标准，将比较重要的权重、通道或卷积核保留下来，并将不重要的部分剪掉，得到压缩的网络。然后，微调剪枝后的网络以恢复模型的准确率。目前剪枝方法主要有 Deep Compression、APoZ、ThiNet 等。如图 2-7 所示，按照剪枝粒度[64]（从左到右，从粗到细）分类，网络剪枝主要有以下四类：层间剪枝、通道剪枝、卷积核剪枝、权重剪枝。

1. 层间剪枝

层间剪枝是修剪完整的隐藏层（图 2-8(a)的中间层部分），层间剪枝会影响网络的深度，使深层网络转换为浅层网络。增加深度可提高网络性能，因此层间剪枝需要智能技术来缓解性能下降的问题。

2. 卷积核剪枝与通道剪枝

特征图剪枝（即通道剪枝）会删除大量卷积核，并且可能会影响网络的性能。如图 2-8(b)所示的架构，修剪一个特征图，会将与其连接的卷积核也一起修剪。特征图剪枝会影响网络层的宽度，从而直接获得更薄的网络而不需要稀疏表示。如图 2-8(c)所示，卷积核剪枝是修剪对最后贡献不大的卷积核，将其整个移除。

如果将某个特征图进行剪枝，与它相连的卷积核也会一同被移除；同样，对某个卷积核剪枝后，对应的通道也被修剪。比如 ThiNet[50]剪枝的主要过程是：首先，卷积核选择依据是：通过通道剪枝来引导卷积核的剪枝，即如果可以用第 i+1 层的输入通道的一个子集作为第 i+1 层的输入且近似得到第 i+1 层的输出，那么这个子集以外的通道就可

以去掉了；其次，将第一步第 i+1 层中选出的不重要的通道子集和对应的 i 层的卷积核去掉，最后进行微调恢复由于剪枝损坏的泛化能力，提高模型的准确率。除了上述衡量策略，衡量卷积核重要性的标准还有许多。根据卷积核中所有权重的绝对值之和作为该卷积核的评价指标[46]，修剪绝对值低的卷积核。大型卷积神经网络中的大多数神经元的激活趋于零，而这些激活为零的神经元是冗余的，使用零的平均百分比[47]（APoZ）衡量每一个卷积核中激活为零值的数量，以此作为评价卷积核的标准。

3. 权重剪枝

如图 2-8(d)所示，卷积核中部分值为零。权重剪枝通过修剪不重要的权重，即将对最后结构贡献不大的小权重置为零，以得到稀疏化的网络。在 Deep Compression[33]中进行剪枝的主要过程是：首先正常训练网络，得到训练好的权重；然后设定一个阈值，将小于阈值的权重置为零，此时网络变成了稀疏连接的网络；最后重新训练这个稀疏网络，得到最终结果。

2.4　本章小结

本章重点介绍了卷积神经网络压缩的相关知识，分别从以下几个方面对其进行说明：首先介绍了卷积神经网络压缩的必要性与可行性。然后介绍了卷积神经网络的基本组成，常见的卷积神经网络结构，简要说明了卷积神经的训练过程（前向传播与反向传播），几种正则化方法。最后对目前卷积神经网络压缩方法（新型网络模块设计、知识蒸馏、低秩分解、网络量化、网络剪枝）的基本原理做了简要介绍。

第三章　一种权重动态剪枝方法

3.1　权重剪枝

3.1.1　权重剪枝原理

权重剪枝通过修剪卷积神经网络中冗余的参数，以达到压缩 CNN 的效果。该方法过程如下：

步骤 1. 评估权重的重要性。正常训练 CNN，得到训练好的权重。

步骤 2. 权重剪枝。设定阈值，修剪低于阈值的权重，即将不重要的权重置为零。

步骤 3. 重新训练。保留下来的权重通过重新训练进行微调，以恢复模型精度。

剪枝时，不重要的权重置为 0，此时修剪的权重正常保存时所占空间大小不变，因此采用 Compressed Sparse Row (CSR)或 Compressed Sparse Column (CSC)格式存储稀疏矩阵。如图 3-1 所示 CSR 的存储原理图，CSR 通过：数值（values），非零元素的列下标（column indices），行偏移量（row offsets） 来存储稀疏矩阵。CSC 与 CSR 原理一致。通过使用 CSR 或 CSC，能将一个 m × n 矩阵的存储数量减少至： 2a+m+1(CSR)或 2a+n+1(CSC)，其中 a 为非零数值。

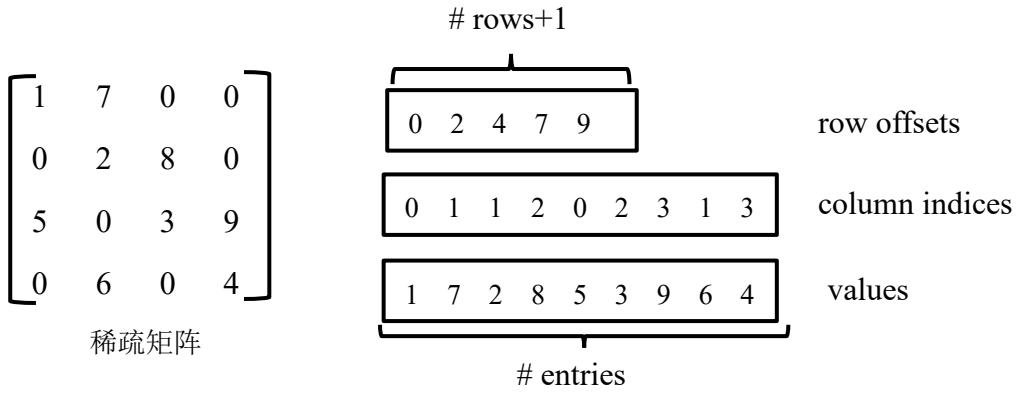


图 3-1　Compressed Sparse Row (CSR)存储原理图

因数据量有限、训练参数多并且训练过度，卷积神经网络面临过拟合的问题，目前正则化和 Dropout 是防止过拟合常用的两种方法。正常训练 CNN 后，得到训练好的权重。在权重剪枝时，卷积层和全连接层会具有不同的敏感度。

3.1.2 正则化

增加网络的深度可以获得更好的特征表示，但是网络的复杂性也会随之增加，使得网络难以优化并且更易过拟合。防止过拟合的最直接方法是增加所使用数据集的数量和减小使用的网络结构。但是，增加数据集的数量并不简单。减小网络结构固然可以有效减少参数数量，但会影响最后的识别效果，因为越深的网络一般表达能力更强。而正则化和 Dropout 的使用旨在解决过拟合的问题。

L2 正则化是最常用的正则化方法，表示为公式(3.1)：

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2 \tag{3.1}$$

其中 C 是正则化后的损失函数，C_0 是原始的损失函数，λ 表示正则化因子。所以常用的交叉熵损失函数可以表示为：

$$C = -\frac{1}{n}\sum_x [y\, ln\, a + (1-y)\, ln(1-a)] + \frac{\lambda}{2n}\sum_w w^2 \tag{3.2}$$

其中 x 代表样本，n 代表样本总数，y 是实际值，a 是输出值。由公式(3.2)，正则化是通过在损失函数中加入权重惩罚因子来增加损失值以减小权重。使用正则化因子调整正则化：λ 越大，越倾向于减小权重；λ 越小，越倾向于减小原始的损失函数。在剪枝过程的第一次训练中，确定权重的重要程度。正则化程度将影响权重的大小，从而影响网络中的哪些连接需要进行剪枝。

与 L2 正则化不同，Dropout 通过改变网络结构解决过拟合的问题。Dropout 在训练期间以一定概率将某些权重置为零，通过设置剪枝的概率，增加网络稀疏性，可按公式(3.3)和(3.4)计算：

$$C_i = N_i N_{i-1} \tag{3.3}$$

$$D_r = D_o\sqrt{\frac{C_{ir}}{C_{io}}} \tag{3.4}$$

其中 C_i 表示第 i 层的连接数，N_i 表示第 i 层的神经元数目，C_{io} 表示原来的连接数，C_{ir} 表示再训练时的连接数，D_o 表示原始的 Dropout。

3.1.3 敏感度

卷积层（CONV）和全连接层（FC）都可以进行剪枝，但网络层的类型不同具有的敏感度也不同，因此它们的剪枝阈值对精确度也会有不同的影响。表 3-1 为敏感度

分析算法，对网络逐层进行敏感度分析。

表 3-1　敏感度分析算法

算法 2 敏感度分析
输入：训练好的网络 输出：精度损失 Accuracy loss(*Al*) 1. $L \leftarrow \{L_1, L_2, \ldots, L_i, \}$;　// L 为网络层总数 2. for each item $i \in L$ do 3. set a threshold *t*; 4. weight pruning according to *t*; 5. compute *Al* = *O_acc* - *P_acc*　// *Al* 是精度损失值 // *O_acc* 是原始精度, *P_acc* 是剪枝后的精度 6. rank *Al*; 7. if *Al* of the CONV layers are greater than *Al* of the FC layers then 8. the CONV layers are more sensitive to pruning than the FC layers; 9. otherwise 10. the FC layers are more sensitive to pruning than the CONV layers; 11. end if 12. end for

3.2　基于改进权重剪枝的动态剪枝方法

深度神经网络中存在巨大的参数冗余，通过剪枝方法能有效减少网络中不重要的权重和卷积核的数量。权重剪枝方法是修剪小于指定阈值的权重，得到稀疏连接的网络，再微调保留下来的权重。这种方法对于误剪的权重无法恢复，这会对最后的模型识别精度产生一定的影响。针对这个问题，提出了一种基于改进权重剪枝的动态剪枝方法。

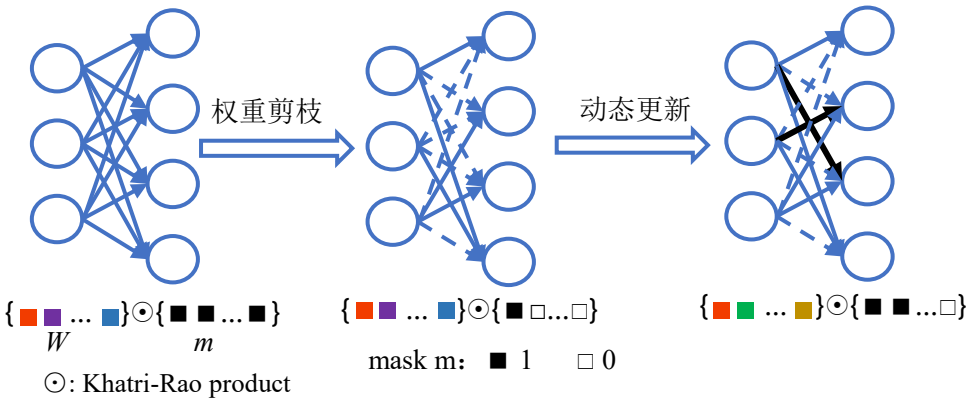


图 3-2　动态剪枝示意图

如图 3-2 所示为动态剪枝过程的示意图。动态剪枝方法中，对于每层的权重矩阵，加入一个与其形状相同的掩码矩阵，即得到一个带有二进制掩码的预训练模型。首先对

网络中权重绝对值小于指定阈值的权重进行剪枝（虚线代表要剪枝的权重连接，实线代表保留下来的权重连接）。其次，在剪枝的整个稀疏网络上动态更新权重的重要性系数，然后对误剪的权重进行动态恢复（粗实线即原先权重剪枝方法中误剪的部分）。最后重新训练网络，微调保留下来的权重系数，以提高剪枝网络的识别精度。

权重剪枝的目标是修剪不重要的权重，然而网络中的权重若是被误剪，则会造成模型精度损失。为了解决这一问题，动态剪枝方法引进一个二进制掩码（mask，m）变量，以暂时掩盖不重要的权重。于是问题可转换成以下优化问题：

$$\min_{W_l, m_l} L(W_l \odot m_l) \ \ s.t. \quad m_l^{(i,j)} = h_l\left(W_l^{(i,j)}\right) \tag{3.6}$$

$$|m_l| \le \beta, l = 1, 2, \ldots, L$$

以第 l 层为例，其中 L (·)是网络的损失函数，W_l 是原始矩阵，m_l 是这一层的掩码矩阵，两者相乘即可得到剪枝后的参数矩阵。h_l(·)是参数 W 在当前层的判别函数，其用来决定当前层的权重是否重要。函数 h_l(·)的输出值是二值的，即如果 m=1，则对应的权重是重要的；如果 m=0，则对应的权重不重要。$\beta \in (0，1]$是剪枝阈值。

W_l 的更新方案如下：

$$W_l^{(i,j)} = W_l^{(i,j)} - \eta \frac{\partial}{\partial\left(W_l^{(i,j)} m_l^{(i,j)}\right)} L(W_l \odot m_l) \tag{3.7}$$

其中 η 是学习率，⊙代表 Khatri-Rao 积。这种方法不仅更新了重要的参数，而且还更新了与 m_l 中 0 相对应的权重，这样就可以恢复误剪的权重。用于表示权重重要性的 h_l (·)函数定义如下：

$$h_l\left(W_l^{(i,j)}\right) = \begin{cases} 1, & \left|W_l^{(i,j)}\right| > \beta \\ 0, & otherwise \end{cases} \tag{3.8}$$

表 3-2 为动态剪枝的算法，具体过程为：首先，给定一个带有二进制掩码的预训练模型，将其对应的二进制掩码矩阵初始值全设为 1（mask=1）。其次，确定要剪枝的权重。基于权重的的平均绝对值和方差设置阈值，对原始网络中权重绝对值低于阈值的权重进行剪枝。最后，通过随机梯度下降（SGD）方法更新 W_l 和 m。根据判别函数 h_l (·)和当前层的 W_l 更新掩码的值 m，根据公式(3.8)和当前损失函数的梯度∇L 更新权重的值 W_l，迭代地动态更新滤波器和全局掩码，以提高修剪网络的准确率。

表 3-2　动态剪枝算法

算法 3 动态剪枝算法

输入：训练数据 D=$\{(x,y)\}$，阈值 β，学习率 η，参考模型$\widehat{W}_l$: $0 \le l \le C$

输出：更新的权重矩阵及其二进制掩码$\{W_l, m_l: 0 \le l \le C\}$

Initialize：$W_l \leftarrow \widehat{W}_l$, $m_l \leftarrow 1$, $0 \le l \le C$, $\beta \leftarrow 1$, iter $\leftarrow 0$;
repeat
  Forward Pass:
    Choose a minibatch of network input from D;
    loss computation with $(W_0 \odot m_0), \ldots, (W_C \odot m_C)$;
  Backward Pass:
    for l = 0, …, C do
      Update m_l by function $h_l(\cdot)$ and the current W_l, with a probability of σ(iter);
      Update W_l by Eq. (3.7) and the current loss function gradient∇L;
    end for
  Update:
    iter $\leftarrow$ iter + 1, $\beta \leftarrow f(\eta, \text{iter})$;
until iter reaches its desired maximum

3.3　实验结果与分析

本实验使用 LeNet-5 和 VGG-16 两种经典的 CNN 架构作为实验对象。采用 MNIST [65]手写数字识别和用于物体识别的 CIFAR-10 [66]两种数据集验证动态剪枝方法的有效性。MNIST 数据集包含 10 类，即数字 0 到 9，训练集 60,000 个，测试集 10,000 个。CIFAR-10 包含 60,000 张图像，由 10 类 32×32 彩色图像组成，每个类别有 6000 张图像，训练集有 50,000 张，测试集有 10,000 张。深度学习框架采用基于 Python 的 PyTorch [67]框架。

3.3.1　LeNet-5 剪枝结果与分析

如图 3-3 所示，是 LeNet-5 网络在 MNIST 数据集上训练得到的卷积层第一层（CONV1）中的参数分布。CONV1 层中大部分权重集中在 0 处附近，对网络的贡献较小，权重剪枝方法将 0 值附近的较小的权重置 0，使这些权重不被激活，最终在尽量不损失网络识别精度的前提下达到压缩网络的目的。该方法虽然在一定程度上压缩了网络模型，但是这种方法对于误剪权重无法恢复，这会对最后的识别精度产生一定的影响。针对这个问题，提出的动态剪枝方法不仅能有效压缩模型，还可以对误剪的权重进行恢复，使最后的识别精度更好。

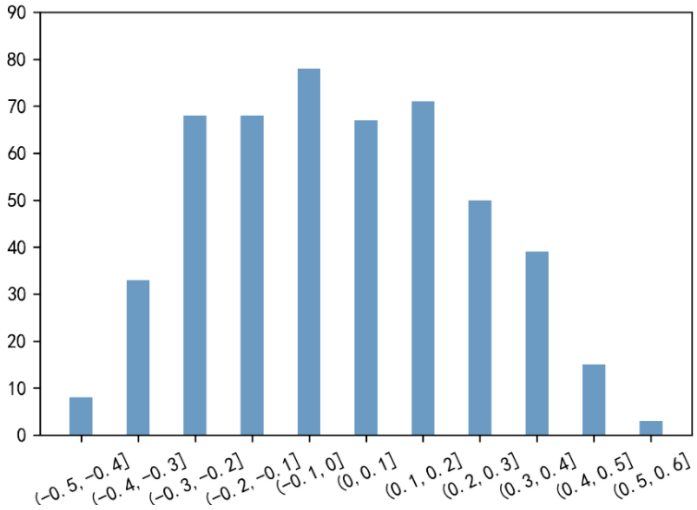


图 3-3　LeNet-5 中 CONV1 的参数分布

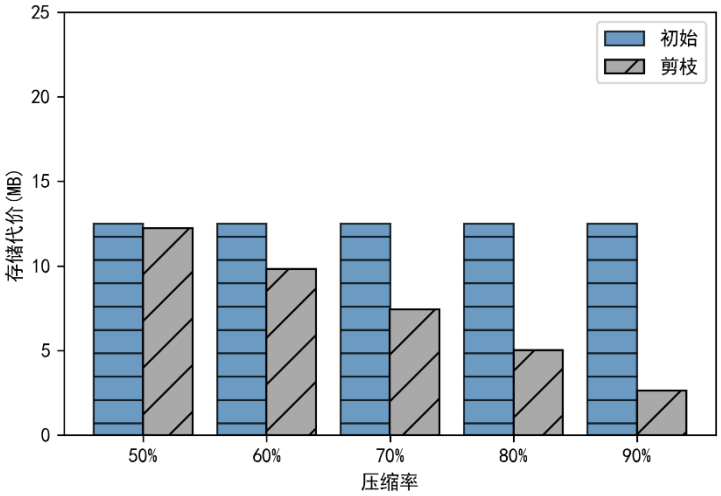


图 3-4　不同压缩率下存储代价的比较

实验中使用 LeNet-5 网络，它有两个卷积层，两个下采样层，两个全连接层和一个分类层，在 MNIST 数据集上的错误率为 0.8%。在 MNIST 数据集上进行剪枝和训练 CNN。动态剪枝与二个基准比较：（1）原始模型：正常训练一个小的 CNN 不进行剪枝。（2）权重剪枝：对网络中低于阈值的所有权重进行剪枝以获得压缩模型。

使用 LeNet-5 网络在 MNIST 数据集上评估权重剪枝方法的性能。给定一个带有二进制掩模（mask，m）的预先训练好的网络，形状大小与每层的权重张量形状完全相同。掩码的值用来决定权重是否重要：若权重重要，则 m=1；若权重不重要，则 m=0。对需

要剪枝层的权重取绝对值，低于阈值的权重进行剪枝，即将权重对应的掩码值设置为 0。如图 3-4 所示，不同的压缩率下，LeNet-5 网络上存储代价的比较。使用不同压缩率对 LeNet-5 网络进行权重剪枝，最后得到剪枝后的模型。当压缩率为 90%时，LeNet-5 网络的存储代价最小。

LeNet-5 权重剪枝实验的过程如下：

步骤 1. MINST 数据集预处理。使用 PyTorch 中的 torchvision.transforms.ToTensor() 函数对数据集进行处理，将 PIL.Image 或 numpy.ndarray 转换为形状为 C × H × W 的 32 位浮点张量（torch.FloatTensor），并在[0.0,1.0]范围内进行标准化。数据集目录结构如图 3-5 所示，raw 目录下为原始的数据集格式，processed 为处理后的.pt 文件。

```
Mnist
|—processed
      |—train.pt
      |—test.pt
|—raw
      |—t10k-images-idx3-ubyte
      |—t10k-labels-idx1-ubyte
      |—train-images-idx3-ubyte
      |—train-labels -idx1-ubyte
```

图 3-5　MNIST 数据集目录结构

步骤 2. 加载模型（Model）。加载 LeNet-5 模型，获取已经训练好的二进制文件 checkpoint，其中存储了权重，偏置，梯度等值。加载 MNIST 数据集，MNIST 类别 classes = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)。

步骤 3. 权重剪枝。使用 Batch Gradient Descent（SGD）进行网络优化，损失函数使用交叉熵损失函数（cross entropy loss），超参数值的设定（learning rate(lr)=0.01, momentum=0.9, weight_decay=1e-6）。由于卷积层与全连接层的敏感度不同，因此要剪枝的数量也会不一样，根据α 来控制每层的剪枝速率。其中阈值选择的标准为：α × weight.std（权重的标准差）。LeNet-5 网络中的权重取绝对值，迭代的修剪低于阈值的权重，设置其对应的掩码值（mask=0）。针对掩码值为零的权重，将修剪后的权重位置保存在地址簿和掩码簿中（addressbook=[]，maskbook=[]），保存在其中的权重在训练中可以保持为零。

步骤 4. 微调（finetune）：权重剪枝会造成准确率的下降，重新训练网络，微调保留下来的权重连接，迭代直到返回识别准确率最佳的值（best_accuracy）。可定义 finetune() 函数，括号内为定义的四个参数：finetune (model_weights, best_accuracy, epochs, lr)。通过调整保留的权重，恢复模型的准确率，最后保存模型。

使用 LeNet-5 网络在 MNIST 数据集上评估动态剪枝方法，验证该方法的可行性以及与权重剪枝相比所具有的优势。识别精度作为评估指标，对两种方法进行评估。如表 3-3 所示，LeNet-5 网络动态剪枝的算法，具体过程为：首先，对 MNIST 数据进行预处理，加载预训练的 LeNet-5 网络。然后根据稀疏度的不同得到最佳识别精度的模型，掩盖不重要的权重，根据判别函数 $h_l(\cdot)$和当前的权重矩阵更新掩码矩阵 m 的值，恢复误剪权重，直到达到当前训练步骤所需的稀疏度水平。修剪后的权重位置保存在 addressbook[]和 maskbook[]中，最后得到剪枝后的 LeNet-5 模型。

如表 3-4 所示，提出的动态剪枝方法与原始的权重剪枝相比，压缩效果近似的情况下，精度损失值与权重剪枝相比减少了 0.11%，这表明了动态剪枝方法通过恢复误剪的权重可以提升网络模型的性能。

表 3-3　LeNet-5 网络动态剪枝算法

算法 4　LeNet-5 网络动态剪枝算法
输入：MNIST=$\{(x,y)\}$，阈值 β，学习率 η，参考模型$\widehat{W_l}: 0 \le l \le C$ 稀疏度sparsity：$s \leftarrow \{50.0, 60.0, 70.0, 80.0, 90.0\}$，
输出：$\{W_l, m_l: 0 \le l \le C\}$:更新的权重矩阵及其二进制掩码
1. Initialize：$W_l \leftarrow \widehat{W_l}$, $m_l \leftarrow 1$, $0 \le l \le C$, $\beta \leftarrow 1$, iter $\leftarrow 0$;
2. Data preprocessing
3. transforms.RandomCrop();
4. transforms.ToTensor();
5. transforms.Normalize();
6. transforms.RandomHorizontalFlip();
7. torchvision.datasets. MNIST ();
8. Load net = LeNet(' LeNet-5');
9. while each item $i \in s$ do
10. mask pruned weights;
11. repeat
12. Forward Pass:
13. choose a minibatch of network input from CIFAR10;
14. loss computation with $(W_0 \odot m_0), \ldots, (W_C \odot m_C)$;
15. Backward Pass:

```
16.          for l = 0, …, C do
17.             update m_l by function h_l(·) and the current W_l
18.             with a probability of σ(iter);
19.             update W_l by Eq. (3.7) and the current loss function gradient∇L;
20.          end for
21.       Update:
22.          iter ← iter + 1, β ← f(η, iter);
23.    until iter reaches its desired maximum
24.    if s reaches its desired sparsity then
25.       save index of pruned weights in addressbook[] and maskbook[];
26.    end if
27. end while
```

表 3-3　剪枝前后 LeNet-5 模型的性能变化对比

方法	压缩	准确率
初始模型	1×	99.17%
权重剪枝[33]	11.72×	98.80%
动态剪枝	11.64×	98.91%

3.3.2　VGG-16 剪枝结果与分析

VGG-16 是一个 16 层的 CNN，有 13 个卷积层和 3 个 FC 层，在 CIFAR-10 数据集上进行剪枝和训练该 CNN。与 3.4.1 节类似，提出的动态剪枝方法与两种基准在 CIFAR-10 数据集上进行对比实验。

使用 VGG-16 网络在 CIFAR-10 数据集上评估权重剪枝的性能。创建和 VGG-16 相同的架构，但将掩码和阈值变量添加到需要进行剪枝的层。变量掩码与网络层的权重张量具有相同的形状，确定哪些权重参与图的正向执行。然后，将操作添加到训练图中，该图监视层中权重大小的分布并确定层阈值，掩盖低于该阈值的所有权重，达到当前训练步骤所需的稀疏度水平。

VGG-16 网络权重剪枝实验过程：

步骤 1. 数据预处理。CIFAR10 类别 classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')。对于 PyTorch，表 3-5 为数据预处理使用的主要函数。

步骤 2. 加载模型（Model）。加载 VGG16 模型 net = VGG('VGG16')，获取已经训练好的二进制文件 checkpoint，其中存储了权重，偏置，梯度等值。

表 3-5　数据预处理主要函数

transforms.RandomCrop()	数据随机裁剪
transforms.ToTensor()	数据转为张量（Tensor）
transforms.Normalize()	数据归一化
transforms.RandomHorizontalFlip()	数据随机水平翻转
torchvision.datasets.CIFAR10()	加载 CIFAR10 的训练集/测试集

步骤 3. 权重剪枝（Prune）。使用 SGD 进行网络优化，损失函数使用交叉熵损失函数，网络参数值的设定（learning rate=0.01, momentum=0.9, weight_decay=5e-4）。根据稀疏度的不同得到最佳识别精度的模型，掩盖不重要的权重，直到达到当前训练步骤所需的稀疏度水平。修剪后的权重位置保存在 addressbook[]和 maskbook[]中，保存在其中的权重在训练中可以保持为零。

步骤 4. 微调（Fine-tuning）。权重剪枝会造成准确率的下降，重新训练网络，微调保留下来的权重连接，恢复模型的准确率。

使用 VGG-16 网络在 CIFAR-10 上评估提出的动态剪枝方法。除了数据集和加载网络不同之外，VGG-16 网络动态剪枝过程与 LeNet-5 相同，具体过程为：首先，对 CIFAR10 数据进行预处理，加载预训练的 VGG-16 网络。然后根据稀疏度的不同得到最佳识别精度的模型，掩盖不重要的权重，根据 $h_l(\cdot)$函数和当前的权重矩阵更新掩码矩阵 m 的值，恢复误剪权重，直到达到当前训练步骤所需的稀疏度水平。修剪后的权重位置保存在 addressbook[]和 maskbook[]中，最后得到剪枝后的 VGG-16 模型。

表 3-6　剪枝前后 VGG-16 模型的性能变化对比

方法	压缩	准确率
初始模型	1×	88.68%
权重剪枝[33]	13.33×	87.32%
动态剪枝	13.28×	87.57%

表 3-6 是提出的动态剪枝方法与权重剪枝方法压缩效果与识别准确率的比较。动态剪枝方法是权重剪枝方法的改进，解决了误剪权重无法恢复的问题。与权重剪枝相比，动态剪枝使 VGG-16 网络在 CIFAR-10 数据集上压缩效果近似的情况下，识别精度损失

与权重剪枝相比减少了 0.25%。

3.4　本章小结

本章主要对权重剪枝方法进行了改进，提出了一种动态剪枝方法。首先，对权重剪枝方法的基本原理进行了研究。其次，针对该方法中对于误剪权重无法恢复且对最后的识别精度会产生一定影响的问题，提出了动态剪枝方法，以恢复误剪的权重。最后，使用 LeNet-5 和 VGG-16 两种经典的网络架构分别在 MNIST 和 CIFAR-10 两种数据集上进行实验，验证动态剪枝方法的有效性。实验结果表明所提方法在 LeNet-5 和 VGG-16 上的识别精度损失与权重剪枝相比分别减少了 0.11%和 0.25%。

第四章　一种混合剪枝方法

在模型压缩中，单独使用动态剪枝或卷积核剪枝对卷积神经网络进行压缩时，压缩后的模型中仍然存在较多冗余参数。针对此问题，提出了一种结合动态剪枝和卷积核剪枝的混合剪枝方法。首先，修剪对卷积神经网络整体性能贡献较小的卷积核；其次，进行动态剪枝，修剪网络中低于阈值的权重，以实现进一步的模型压缩。剪枝会对模型性能造成影响，可通过微调来恢复模型的识别精度。

4.1　CNN 混合剪枝框架

混合剪枝的框架如图 4-1 所示。给定一个原始 CNN 模型，首先，进行卷积核剪枝，修剪相对不重要的卷积核。其次，进行动态剪枝，以实现进一步的模型压缩。使用混合剪枝方法得到的剪枝模型与原始 CNN 模型相比网络中的参数更少，实现了网络的极大压缩。

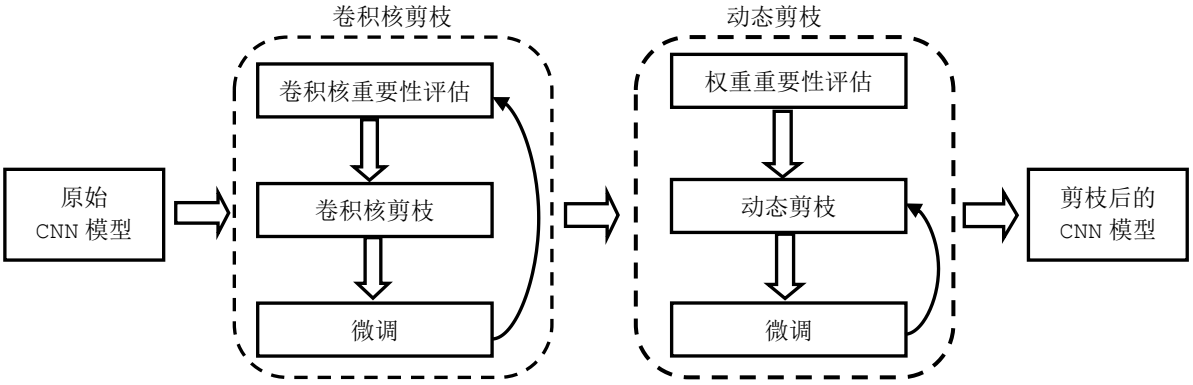


图 4-1　CNN 混合剪枝框架

4.2　卷积核剪枝

4.2.1　卷积核剪枝原理

卷积核剪枝过程描述如下：

步骤 1. 收集训练样本。从训练集的每个类别中随机选择 10 个图像（用于通道选择）。对于每个输入图像，使用不同的通道和不同的空间位置随机采样 10 个实例。

步骤 2. 剪除不重要的通道及其相对应的卷积核。通过 i+1 层的统计数据来进行 i 层的剪枝，若 i+1 层输入的通道子集可以近似得到 i+1 层的输出，则 i+1 层输入的其余通道可安全进行剪枝。i+1 层输入中的通道由 i 层中的卷积核产生，因此可把对应 i 层的卷

积核安全地移除。使用表 4-2 中的贪心算法得到要移除通道的子集，相对应的卷积核也可被修剪。

步骤 3. 剪枝后会损坏模型的泛化能力，将整个网络微调一次或两次恢复其性能。

步骤 4. 最后，判断剪枝是否结束。如果剪枝停止，对网络再进行多次微调获得性能更好的模型；否则，重复步骤 1-3 继续对下一层进行剪枝。

表 4-1　卷积核剪枝算法

算法 5　卷积核剪枝算法
输入：训练数据 D={(x,y)}, 预训练 CNN 模型 输出：移除通道子集 T 1. initialize：$T \leftarrow \emptyset, U \leftarrow \{1,2,\ldots,C\}$; 2. data preprocessing; 3. load net = LeNet/VGG('LeNet5/VGG16'); 4. collecting training examples $\{(\hat{x}_i,\hat{y}_i)\}$: 5. forward pass: 6. choose an input image *i*, 7. conduct the forward run to find the input and output of layer *i* + 1; 8. $\hat{x} = \{\hat{x}_1,\hat{x}_2,\ldots,\hat{x}_C\}$ and $\hat{y}$ can be obtained via Eq. (4.1) to Eq. (4.3); 9. a greedy algorithm (illustrated in Table 4-2): 10. output: a subset of channels *T*; 11. minimize the reconstruction error via Eq. (4.7); 12. fine-tuning the whole network

卷积核剪枝算法如表 4-1 所示。针对网络模型的第 *i*+1 层进行压缩，使用第 *i*+1 层的输入，即第 *i* 层的输出来模拟其输出的线性过程。如果可以使用 *i*+1 层输入通道的子集来近似 *i*+1 层的输出，那么 *i*+1 层的其它通道便可安全移除。剪枝后的 CNN 模型具有更少的通道和卷积核，而且几乎不损失精度。通过优化方法找到不重要的通道及其对应的卷积核，即线性组合的值接近于零的部分，具体过程主要包括收集训练样本、用于通道选择的贪心算法和最小化重构误差。

4.2.2　收集训练集

收集训练集以评估通道的重要性，确定通道是否可以安全地移除。由 y 表示的元素从 *i*+2 层中随机采样。$i+1$ 层中的相应卷积核和滑动窗口也可以根据其位置来确定。卷积运算如下式：

$$y = \sum_{c=1}^{C}\sum_{k1=1}^{K}\sum_{k2=1}^{K}\widehat{W}_{c,k1,k2} \times x_{c,k1,k2} + b \tag{4.1}$$

进一步定义如公式(4.2)：

$$\hat{x}_c = \sum_{k_1=1}^{K}\sum_{k_2=1}^{K} \widehat{W}_{c,k_1,k_2} \times x_{c,k_1,k_2} + b \tag{4.2}$$

然后，公式(4.1)卷积操作可以简化为公式(4.3)：

$$\hat{y} = \sum_{c=1}^{C} \hat{x}_c \tag{4.3}$$

假如我们能找到一个子集$S \subset \{1，2，\ldots, C\}$，使公式(4.3)总是成立，那对于$c \notin S$的任何的$\hat{x}_c$可以安全移除而不改变 CNN 模型的结果。公式(4.3)对于变量$\hat{x}$和$\hat{y}$的所有实例，不会总是成立，可以通过手动提取它们的实例以找到子集 *S*，使得公式(4.4)近似正确。

$$\hat{y} = \sum_{c\in S}^{C} \hat{x}_c \tag{4.4}$$

4.2.3　通道选择

卷积核剪枝的关键是衡量卷积核的重要性，常用的方法包括：结构化稀疏性、基于神经元的输出、基于下一层的输入通道选择等。训练之后，移除一些不重要的卷积核，然后每一层剪枝后微调一遍或两遍，以便恢复模型的识别精度。

表 4-2　用于通道选择的贪心算法

算法 6　用于通道选择的贪心算法

输入：训练集$\{(\hat{x}_i, \hat{y}_i)\}$和压缩率 *r*
输出：移除通道的子集 *T*

```
1.   T ← ∅;   U ← {1,2,…,C};
2.   while |T| < C × (1 − r) do
3.     min_value← +∞;
4.     for each item i∈U do
5.       tmpT ← T ∪ {i};
6.       compute value from Eq. (4.6) using tmpT;
7.       if value < min_value then
8.         min_value←value; min_i←i;
9.       end if
10.    end for
11.    move min_i from U into T;
12.  end while
```

给定一组 *m* 个训练样本$\{(\hat{x}_i, \hat{y}_i)\}$，*m* 是图像数量和位置数量的乘积，原来的通道选择问题就变成了下面的优化问题。

$$\underset{S}{argmin} \sum_{i=1}^{m}\left(\hat{y}_i - \sum_{j\in S} \hat{x}_{i,j}\right)^2, s.t \quad |S| = C \times r, \quad S \subset \{1,2,\ldots,C\}. \tag{4.5}$$

其中，|S|是子集 *S* 中元素的数量，*r* 是预定义的压缩率。等价的，*T* 表示被移除通道的子集，即$S \cup T = \{1，2，\ldots, C\}$，$S \cap T = \phi$，就可以最小化下面的公式(4.6)来替代公式(4.5)：

$$\underset{T}{argmin} \sum_{i=1}^{m}\left(\sum_{j\in T} \hat{x}_{i,j}\right)^2, \ \text{s.t}\ |T| = C \times (1-r), T \subset \{1,2,\ldots,C\}. \tag{4.6}$$

公式(4.6)等价于公式(4.5)，但|T|的个数通常是小于|S|的，因此公式(4.6)比公式(4.5)更快更简单。表 4-2 为用于通道选择的贪心算法（greedy algorithm），可以用来解决公式(4.6)的优化问题。假设 T 表示已移除通道的子集，其初始值为空集。U 是所有通道的集合，对于每个 $i \in U$，通过公式(4.6)计算值，每次选择添加一个通道使得通过当前样本得到的误差最小，最后获得移除通道的子集。这种方法是局部最优的，可通过微调来弥补这种方法造成的影响。

4.2.4 最小化重构误差

在确定要保留哪些卷积核之后，通过通道的加权来进一步减少重构误差：

$$\hat{w} = \underset{w}{argmin} \sum_{i=1}^{m} (\hat{y}_i - w^T \hat{x}_i^*)^2 \tag{4.7}$$

其中$\hat{x}_i^*$表示通道选择后的训练样本。公式(4.7)是典型的线性回归问题，具有使用普通最小二乘法的独特闭合形式解。w 中的每个元素可以被视为对应卷积核通道的缩放因子，这种缩放操作为微调提供了更好的初始化，网络可达到更高的准确性。

4.3 实验结果及分析

实验使用经典的 LeNet-5 和 VGG-16 两种网络架构，各自在 MNIST 数据集和 CIFAR-10 数据集上进行验证。与原始网络、动态剪枝和卷积核剪枝结果分别进行对比，验证混合剪枝方法的有效性。最后，将混合剪枝结果与现有方法（如 APoZ，Weight Sum 和 Taylor）进行比较。本实验在 PyTorch 环境下进行。

4.3.1 LeNet-5 剪枝结果与分析

实验使用 LeNet-5 网络在 MNIST 数据集上进行训练和剪枝。混合网络剪枝与三个基准比较：（1）原始模型：正常训练一个小的 CNN 不进行剪枝。（2）动态剪枝：修剪不重要的权重后，对误剪的权重进行恢复。（3）卷积核剪枝：移除重要性低的卷积核。

在 3.4.1 节中介绍了使用 LeNet-5 网络在 MNIST 数据集上评估动态剪枝方法的性能。同样条件下评估卷积核剪枝方法的性能。从收集用于通道选择的训练集开始，从训练集中的每个类别中随机选择 10 个图像构成评估集。并且对于每个输入图像，用不同的通道和不同的空间位置随机采样 10 个实例。因此，总共有 1000 个训练样本用于通过贪心算法找到最佳通道子集。实验证明了这种选择的有效性（每个类 10 个图像，每个图像 10 个位置），足以进行神经元重要性评估。每层剪枝后微调一遍（学习率=0.001）。

当所有层都被修剪完，整体的网络微调 10 遍以获得更高的准确度（学习率 0.001~0.00001）。

使用 LeNet-5 网络在 MNIST 数据集上评估混合剪枝方法的性能。卷积核剪枝的主要优点是该方法能直接影响网络结构的大小，删除对最后结果贡献不大的卷积核得到的网络可实现一定程度的压缩。对剪枝后的模型，再进行动态剪枝实现进一步压缩。如图 4-2 所示，LeNet-5 在 MNIST 上使用混合剪枝方法得到的剪枝参数与剩余参数的对比。FC 层可用全局平均池化层（GAP）替换，但 MNIST 数据集和 LeNet-5 网络比较简单，因此本实验中保留了全连接层。针对 LeNet-5 网络剪枝，对 CONV 层与 FC 层中的权重和卷积核进行剪枝操作。

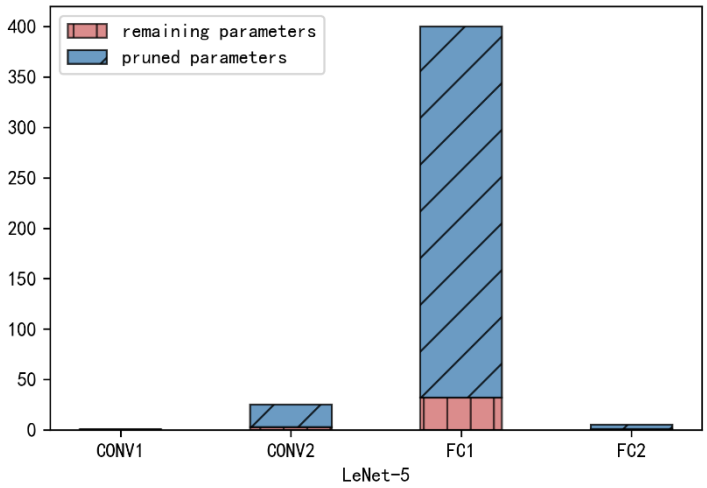


图 4-2　LeNet-5 在 MNIST 上每层参数减少统计

表 4-3　剪枝前后 LeNet-5 模型的性能变化

剪枝方法	准确率	加速	压缩
初始模型	99.17%	1.00×	1.00×
动态剪枝	98.91%	1.00×	11.64×
卷积核剪枝[47]	98.32%	3.23×	3.78×
混合剪枝	98.18%	3.23×	12.90×

如表 4-3 所示，剪枝前后的 LeNet-5 模型的性能变化。对于 LeNet-5 模型，使用权重剪枝虽然可以实现较好的压缩效果，但是没有明显的加速效果。而卷积核剪枝方法，可以同时实现网络的压缩与加速，但是压缩效果与权重剪枝相比效果略差。而使用所提

出的混合剪枝方法可以实现 12.90 倍压缩和 3.23 倍加速。

4.3.2 VGG-16 剪枝结果与分析

VGG-16 是一个 16 层的 CNN，有 13 个卷积层和 3 个全连接层，在 CIFAR-10 数据集上进行剪枝和训练 CNN。与 4.1 节类似，在 CIFAR-10 数据集上，将提出的混合剪枝方法与三种基准进行对比实验。

使用 VGG-16 网络在 CIFAR-10 数据集上评估动态剪枝的性能。创建和 VGG-16 相同的架构，但将掩码和阈值变量添加到需要进行剪枝的层。变量掩码矩阵与网络层的权重张量具有相同的形状，确定哪些权重参与图的正向执行。然后，将操作添加到训练图中，该图监视层中权重大小的分布并确定层的阈值，暂时掩盖低于该阈值的所有权重，达到当前训练步骤所需的稀疏度水平。动态更新掩码的值，恢复误剪的权重以减少识别精度的损失。

使用 VGG-16 网络在 CIFAR-10 数据集上评估卷积核剪枝方法的性能。CIFAR-10 数据集包含从 10 个类别中抽取的 50,000 个训练图像。首先收集训练集用于通道选择，对于每个输入图像，使用不同的通道和不同的空间位置随机采样 10 个实例，然后从训练集中的每个类别中随机选择 10 个图像以构成我们的训练集（包括评估集）。因此，总共有 1000 个训练集（图像数量与对应位置数量的乘积）用于通过贪心算法找到最优通道子集。微调期间，网络每修剪一层后微调一遍（学习率=0.0001）。当所有层都被剪枝后，微调 15 遍以恢复准确率（学习率 0.001~0.00001，小的学习率可降低损失值）。

使用 VGG-16 网络在 CIFAR-10 数据集上评估混合剪枝方法的性能。首先移除不重要的卷积核，再对剪枝后的模型进行动态剪枝实现进一步的压缩。如图 4-3 所示，VGG-16 在 CIFAR-10 数据集上使用混合剪枝方法得到的剪枝参数与剩余参数的对比。结果表明，所提出的混合剪枝方法显著减少了参数的数量。VGG-16 网络的卷积层占有约 90%的浮点运算，而 FC 层有 89.36%的参数。出于模型加速的考虑，修剪 VGG-16 的卷积层。对于 FC 层，使用 GAP 层替换 FC 层这一策略会更简单、更有效。

剪枝前后的 VGG-16 模型的性能变化如表 4-4 所示。与其它基准相比，混合剪枝方法可以实现更好的压缩与加速。对于 VGG-16 模型，使用所提出的混合剪枝方法可以实现 19.13 倍压缩和 3.31 倍加速。

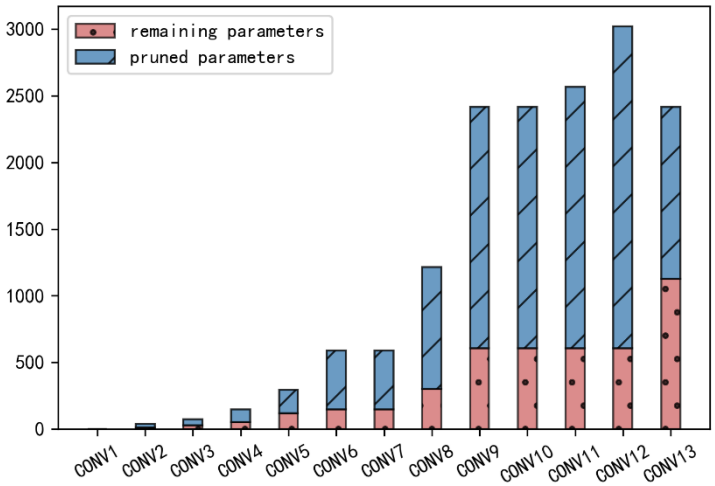


图 4-3　VGG-16 在 CIFAR-10 上每层参数减少统计

表 4-4　剪枝前后 VGG-16 模型的性能变化

剪枝方法	准确率(%)	加速	压缩
初始模型	88.68	1.00×	1.00×
动态剪枝	87.57	1.00×	13.28×
卷积核剪枝[50]	87.68	3.31×	12.64×
混合剪枝	87.36	3.31×	19.13×

4.3.3　对比现有方法

针对不同的卷积核选择方法，在 Dogs vs. Cats Kaggle 数据集上对其性能进行实验评估，该数据集包含 25000 张狗和猫的图像。训练新的全连接层，VGG-16 中的所有 FC 层都将被删除并在新数据集上（Dogs vs. Cats Kaggle）进行微调。然后，获得该微调模型之后，以不同的压缩率逐层对网络剪枝。每一层剪枝之后微调一遍，最后一层剪枝完微调多遍以提高准确性，使用不同的卷积核选择策略重复该过程若干次。实验表明微调 15 次时，准确率即趋于稳定。除了卷积核选择标准外，其它所有设置均保持不变。

实验过程如下：

步骤 1. 猫狗数据集制作：PyTorch 的数据加载方法是 datasets.ImageFolder()，使用这个函数对数据集进行处理，默认的自动给图片标签命名为 001，002，003…，有助于损失函数的计算。训练（train）数据集中包含 2000 张狗的图像和 2000 张猫的图像，测试（test）数据集中包含 800 张狗的图像和 800 张猫的图像用于测试网络性能。得到的

猫狗数据集目录结构如图 4-4 所示。

```
Root
|—train                    |—test
|—|—dog                    |—|—dog
|      |—001.png           |      |—001.png
|      |—002.png           |      |—002.png
|      |—…                 |      |—…
|—|—cat                    |—|—cat
|      |—001.png           |      |—001.png
|      |—002.png           |      |—002.png
|      |—…                 |      |—…
```

图 4-4　猫狗数据集目录结构

步骤 2. 训练：使用猫狗数据集在预训练的 VGG-16 网络上进行微调。去掉 VGG 网络的全连接层，根据自己的数据类别数（num_class）来训练新的全连接层，总共迭代 20 次以得到微调后的模型。

步骤 3. 剪枝：把重要性相对较低的卷积核和权重移除。

步骤 4. 微调：恢复由于剪枝损坏的泛化能力的必要步骤。

如图 4-5 所示，在 Dogs vs. Cats Kaggle 数据集上针对不同卷积核选择标准的剪枝结果。对不同卷积核选择标准的性能进行了对比，APoZ 旨在减少参数数量，但其性能有限。相反，泰勒（Taylor）[68]标准旨在模型加速，并且只对卷积层进行剪枝。但计算过程可能非常耗时，因此他们使用泰勒展开来近似剪枝的优化问题。与其他选择方法相比，数据驱动（Data-driven）的通道选择标准性能最优。

将提出的 CNN 混合剪枝方法与几种现有剪枝方法进行比较，结果如表 4-5 所示。与数据驱动（Data-driven）的通道选择标准相比，泰勒（Taylor）标准性能稍差。随机（Random）选择标准表现出相当好的性能。但是，这个标准鲁棒性不好，并且在压缩所有层之后识别精度非常低，在实践中并不适用。权重和（Weight Sum）标准由于它只考虑权重的大小，具有比较差的识别准确率。实际上，如果去除大量小的卷积核也会对识别精度产生很大影响。提出的混合剪枝方法是在数据驱动的通道选择方法基础上进行动态剪枝，实现了 17.40×的压缩与 3.31×的加速，结果优于其他方法。

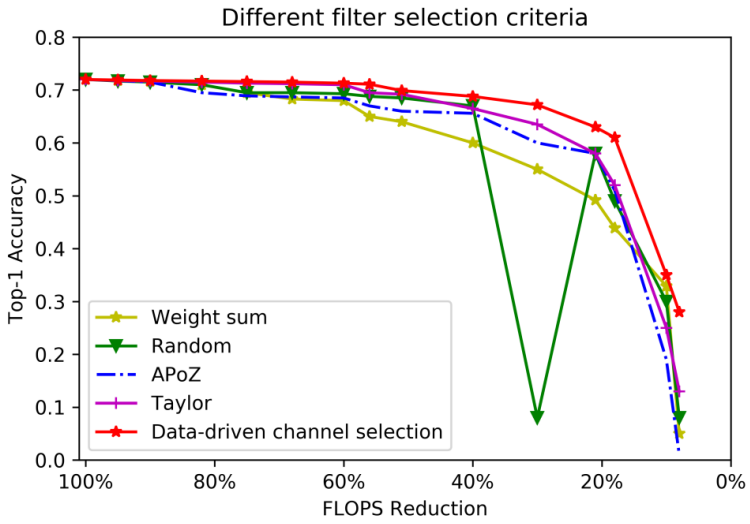


图 4-5　不同卷积核选择标准的性能比较

表 4-5　在 VGG-16 上与现有方法的比较

剪枝方法	准确率	加速	压缩
初始模型	98.70%	1.00×	1.00×
Weight Sum[46]	97.59%	1.80×	2.80×
APoZ[47]	96.69%	1.00×	2.04×
Taylor[68]	97.50%	2.82×	3.83×
Data-driven[50]	97.68%	3.31×	12.64×
混合剪枝	97.14%	3.31×	17.40×

4.4　本章小结

在模型压缩中，单独使用动态剪枝或卷积核剪枝，压缩后的卷积神经网络仍然存在较多参数冗余问题。针对这个问题，提出了一种结合卷积核剪枝和动态剪枝的混合剪枝方法。首先删除不太重要的卷积核，达到压缩网络的初步目的；然后对剪枝后的模型再进行动态剪枝，以实现进一步的压缩；剪枝会对模型性能造成影响，通过重新训练来恢复模型的精度。实验结果表明，提出的方法有效减少了 CNN 中存在的参数冗余，实现了网络加速与压缩。在识别精度分别损失 0.99%和 1.32%的情况下，将 LeNet-5 网络压缩了 12.90 倍，将 VGG-16 网络压缩了 19.13 倍。

第五章　总结与展望

5.1　工作总结

论文的主要工作包括以下两个部分：

1. 在权重剪枝部分，针对误剪权重无法恢复的问题，提出了一种动态剪枝方法。根据掩码判别函数和权重更新函数，动态剪枝方法更新剪枝网络中权重的显著性以及对应的掩码值，然后恢复被误剪的权重以降低误剪率。实验结果表明所提出的动态方法在 LeNet-5 和 VGG-16 上识别精度损失值与权重剪枝相比分别减少了 0.11%和 0.25%，

2. 在模型压缩中，单独使用动态剪枝或卷积核剪枝对卷积神经网络进行压缩，压缩后的模型中仍然存在较多冗余参数。针对这一问题，提出了一种结合动态剪枝和卷积核剪枝的混合剪枝方法。该方法有效减少了 CNN 中存在的参数冗余，实现了网络的进一步压缩。所提出的混合剪枝方法将 LeNet-5 压缩了 12.90 倍，识别精度仅损失 0.99%；将 VGG-16 网络压缩了 19.13 倍，识别精度仅损失 1.32%。

5.2　工作展望

近几年，模型压缩受到了许多研究者的关注和探索。但是，模型压缩技术目前尚处于发展阶段，还存在很多挑战和问题。本文提出的算法，主要针对现有模型进行剪枝操作的研究，在尽量不影响模型性能的情况下获得了一定效果，但是还有较大的提升空间。

接下来的工作可以在以下两个方面继续进行探索：

1. 在 CNN 压缩的研究中，剪枝是一项有效的压缩技术，其关键之处在于如何衡量权重或者卷积核的重要程度。而对于目前的权重或者卷积核选择策略，理论上几乎无法保证哪个选择策略是最优的。因此，研究出一个更优的选择策略是需要进一步研究的问题。

2. 在 CNN 剪枝的研究中，本文提出了一种混合剪枝方法。与卷积核剪枝与权重剪枝相比，混合剪枝方法能够更加有效的压缩 CNN。该方法仅在 LeNet-5 与 VGG-16 网络上进行了实验并得到了验证，接下来的工作将在规模更大的神经网络上进行实验以观察模型压缩的综合性能。

参考文献

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton. Imagenet classification with deep convolutional neural networks[C].Proceedings of Advances in Neural Information Processing Systems (NIPS), 2012,60(2):1097-1105.

[2] K. Simonyan, A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition[C].Proceedings of the In International Conference on Learning Representation(ICLR), 2015, arXiv:1409.15556v6.

[3] C. Szegedy, W. Liu, Y. Jia, et al. Going deeper with convolutions[C].Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), 2015:1-9.

[4] K. He, X. Zhang, S. Ren, et al. Deep Residual Learning for Image Recognition[C].Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), 2016:770-778.

[5] G. Huang, Z. Liu, VDM Laurens, et al. Densely Connected Convolutional Networks[C].Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), 2017: 2261-2269.

[6] B. B. LeCun, J. S. Denker, D. Henderson, et al. Handwritten digit recognition with a backpropagation network[C]. Proceedings of Advances in Neural Information Processing Systems (NIPS), 1990:396-404.

[7] R. Hechtnielsen . Theory of the backpropagation neural network[M].Neural networks for perception. 1992(2):65-93.

[8] F. N. Iandola, S. Han, J. D.William, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv:1602.07360, 2016.

[9] A. G. Howard, M. Zhu, B. Chen, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861, 2017.

[10] X. Zhang, X. Zhou, M. Lin, et al. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. arXiv:1707.01083, 2017.

[11] G. Larsson, M. Maire, G. Shakhnarovich. FractalNet: Ultra-Deep Neural Networks without Residuals. arXiv:1605.07648v4, 2017.

[12] M. Lin, Q. Chen, S.Yan. Network in Network. arXiv:1312.4400, 2013.

[13] C. Bucila, R. Caruana, A. Niculescu-Mizil. Model compression[C].ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD), 2006:535-541.

[14] G. Hinton, O. Vinyals, J. Dean. Distilling the Knowledge in a Neural Network[C].Proceedings of Advances in Neural Information Processing Systems (NIPS), 2014:2644−2652.

[15] B. B. Sau, V. N. Balasubramanian. Deep model compression: distilling knowledge from noisy teachers. arXiv:1610.09650, 2016.

[16] J. B. Lei, R. Caruana. Do Deep Nets Really Need to be Deep? [C].Proceedings of Advances in Neural Information Processing Systems (NIPS), 2013:2654-2662.

[17] A. Romero, N. Ballas, S. E. Kahou, et al. Fitnets: Hints for Thin Deep Nets[C].Proceedings of the In International Conference on Learning Representation(ICLR), 2015:1-13.

[18] A. Korattikara Balan, V. Rathod, et al. Bayesian dark knowledge[C].Proceedings of Advances in Neural Information Processing Systems (NIPS), 2015:3420–3428.

[19] P. Luo, Z. Zhu, Z. Liu, et al. Face model compression by distilling knowledge from neurons[C]. Proceedings of the AAAI Conference on Artificial Intelligence(AAAI), 2016:3560–3566.

[20] T. Chen, I. Goodfellow, J. Shlens. Net2net: Accelerating learning via knowledge transfer[C]. Proceedings of the In International Conference on Learning Representation(ICLR), 2016:1-12.

[21] J. Yim, D. Joo, J. Bae, et al. A Gift From Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning[C].Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), 2017:4133-4141.

[22] S. Zagoruyko, N. Komodakis. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer[C].Proceedings of the In International Conference on Learning Representation(ICLR), 2017:.

[23] 徐喆,宋泽奇. 带比例因子的卷积神经网络压缩方法[J].计算机工程与应用, 2018, 54, No.907(12):110-114+156.

[24] X. Zhang, J. Zou, X. Ming, et al. Efficient and accurate approximations of nonlinear convolutional networks[C].Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), 2014:1-9.

[25] E. L. Denton, W. Zaremba, J. Bruna, et al. Exploiting linear structure within convolutional networks for efficient evaluation[C].Proceedings of Advances in Neural Information Processing Systems (NIPS), 2014:1-11.

[26] 郭锴凌. 低秩分解及其在计算机视觉中的应用[D]. 广州: 华南理工大学, 2017.

[27] 徐梦珂,许道云,魏明俊. 基于非负矩阵分解的低秩矩阵恢复模型[J]. 计算机与数字工程, 2017(6):1019-1024.

[28] R. Rigamonti, A. Sironi, V. Lepetit, et al. Learning separable filters[C].Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), 2013:2754–2761.

[29] M. Jaderberg, A. Vedaldi, A. Zisserman. Speeding up Convolutional Neural Networks with Low Rank Expansions. arXiv:1405.3866v1, 2014.

[30] V. Lebedev, Y. Ganin, M. Rakhuba, et al. Speeding-up convolutional neural networks using fine-tuned cp-decomposition[C].Proceedings of the In International Conference on Learning Representation (ICLR), 2015:1-11.

[31] C. Tai, T. Xiao, X. Wang, et al. Convolutional neural networks with low-rank regularization [C].Proceedings of the In International Conference on Learning Representation(ICLR), 2016:1-11.

[32] W. Chen, J. T. Wilson, S. Tyree, et al. Compressing neural networks with the hashing trick [C].Proceedings of the International Conference on Machine Learning(ICML), 2015.

[33] S. Han, H. Mao, and W. J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding[C].Proceedings of the In International Conference on Learning Representation(ICLR), 2016, 56(4):3-7.

[34] V. Vanhoucke, A. Senior, M. Z. Mao. Improving the speed of neural networks on cpus[C].Deep Learning and Unsupervised Feature Learning Workshop, NIPS, 2011.

[35] S. Gupta, A. Agrawal, K. Gopalakrishnan, et al. Deep learning with limited numerical precision[C]. Proceedings of International Conference on International Conference on Machine Learning (ICML), 2015, 37:1737-1746.

[36] 蔡瑞初,钟椿荣,余洋等. 面向“边缘”应用的卷积神经网络量化与压缩方法[J/OL]. 计算机应用, http://kns.cnki.net/kcms/detail/51.1307.TP.20180423.0846.002.html

[37] C. Leng, Z. Dou, H. Li, et al. Extremely Low Bit Neural Network: Squeeze the Last Bit Out with ADMM[J]. National conference on artificial intelligence, 2018: 3466-3473.

[38] 王磊,赵英海,杨国顺, 等. 面向嵌入式应用的深度神经网络模型压缩技术综述[J]. 北京交通大学学报, 2017, 41(6):34-41.

[39] M. Courbariaux, Y. Bengio, and J. David, “Binaryconnect: Training deep neural networks with binary weights during propagations[C]. Proceedings of Advances in Neural Information Processing Systems (NIPS), 2015:3123-3131.

[40] M. Courbariaux and Y. Bengio, “Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. arXiv:1602.02830, 2016.

[41] M. Rastegari, V. Ordonez, J. Redmon, et al. Xnor-Net: ImageNet Classification using Binary Convolutional Neural Networks[C]. Proceedings of the European Conference on Computer Vision (ECCV), 2016.

[42] S. Srinivas, R. V. Babu. Data-free parameter pruning for deep neural networks[C].Proceedings of the British Machine Vision Conference, 2015, 31:1-12.

[43] K. Ullrich, E. Meeds, M. Welling. Soft weight-sharing for neural network compression[C].Proceedings of the In International Conference on Learning Representation(ICLR), 2017.

[44] 黄聪,常滔,谭虎,等. 基于权值相似性的神经网络剪枝[J]. 计算机科学与探索, 2018, 119(08):92-99.

[45] 韩云飞,蒋同海,马玉鹏,等. 深度神经网络的压缩研究[J/OL]. 2018, 35(10). [2017-09-27]. http://www.arocmag.com/article/02-2018-10-001.html.

[46] H. Li, A. Kadav, I. Durdanovic, et al. Pruning Filters for Efficient ConvNets[C].Proceedings of the In International Conference on Learning Representation(ICLR), 2016.

[47] H. Hu, R. Peng, Y. W. Tai, et. al. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv:1607.03250, 2016.

[48] Z. Liu, J. G. Li, Z. Q. Shen, et al. Learning efficient convolutional networks through network slimming[C].Proceedings of The IEEE International Conference on Computer Vision(ICCV), 2017:2755–2763.

[49] Y. H. He, X. Y. Zhang, J. Sun. Channel pruning for accelerating very deep neural networks[C]. Proceedings of the 16th IEEE International Conference on Computer Vision (ICCV), 2017:1398-1406.

[50] J. H. Luo, J. Wu, W. Lin. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression[C].Proceedings of The IEEE International Conference on Computer Vision(ICCV), 2017:5068-5076.

[51] L. Shaohui, J. Rongrong, L. Yuchao, et al. Accelerating Convolutional Networks via Global & Dynamic Filter Pruning[C].Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2018.

[52] Y. Ruichi, L. Ang, C. Chufu, et al. NISP: Pruning Networks using Neuron Importance Score Propagation [C].Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[53] M. Denil, B. Shakibi, L. Dinh, et al. Predicting parameters in deep learning[C].Proceedings of Advances in Neural Information Processing Systems (NIPS), 2013.2:2148–2156.

[54] 王征韬. 深度神经网络压缩与优化研究[D]. 四川: 电子科技大学, 2017.

[55] 曹文龙,芮建武,李敏.神经网络模型压缩方法综述[J/OL]. 2019, 36(3). [2018-04-17]. http://www.arocmag.com/article/02-2019-03-062.html.

[56] S. Ioffe, C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[C]. Proceedings of the International Conference on on Machine Learning (ICML), 2015:448-456.

[57] C. Szegedy, V. Vanhoucke, S. Ioffe, et al. Rethinking the Inception Architecture for Computer Vision[C].Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2016:2818-2826.

[58] C. Szegedy, S. Ioffe, V. Vanhoucke, et al. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning[C].Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[59] S. Zagoruyko, N. Komodakis. DiracNets: Training Very Deep Neural Networks Without Skip-Connections[C].Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[60] L. Bottou. Large-scale machine learning with stochastic gradient descent[C]. Proceedings of COMPSTAT, 2010.

[61] R. G. J. Wijnhoven, P. H. N. de. Fast training of object detection using stochastic gradient descent[C]. Proceedings of the International Conference on Pattern Recognition (ICPR), 2010.

[62] G. Danner, M. Jelasity. Fully Distributed Privacy Preserving Mini-batch Gradient Descent Learning[M]. Distributed Applications and Interoperable Systems. 2015.

[63] N. Srivastava, G. E. Hinton, A. Krizhevsky, et al. Dropout: a simple way to prevent neural networks from overfitting [J]. Journal of Machine Learning Research, 2014, 15(1):1929-1958.

[64] S. Anwar, W. Y. Sung. Coarse pruning of convolutional neural networks with random masks[C]. Proceedings of the International Conference on Learning Representation(ICLR), 2017:134−145.

[65] Y. Lecun, L. Bottou, P. Haffner. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.

[66] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[67] A. Paszke, S. Gross, S. Chintala, et al. Automatic differentiation in PyTorch[C]. Proceedings of Advances in Neural Information Processing Systems (NIPS), 2017.

[68] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient transfer learning[C]. Proceedings of the International Conference on Learning Representation(ICLR), 2017.2(7):1-17.

致 谢

不知不觉，三年的研究生生活到了尾声。回首这段时间，不禁有些激动与感慨，这篇论文的完成不仅仅是我个人的力量，还有整个实验室老师们和同窗们的帮助，所以我要由衷的感谢帮助过我的老师们和同窗们。

首先，衷心感谢我的导师杨文柱教授，他知识渊博，为人和善。在我研究生三年里，杨老师严谨的治学风格，办事认真以及一丝不苟的工作作风，都深深感染了我，对我的学习以及生活都产生了一定的影响。进入计算机视觉领域，我所取得的成果离不开杨老师的耐心指导。完成论文的过程中，杨老师总能在我迷茫和犹豫不决的时候引导我不偏离轨道，走向正确的道路。

另外，感谢实验室的老师们和同学们，感谢和你们一起度过的那些年。感谢实验室的王思乐老师、陈向阳老师、陈丽萍老师、崔振超老师和卢素魁老师，从他们身上我学到了很多。感谢实验室的师姐们，尤其是刘晴师姐，在我学习上、找工作上给予了我巨大的帮助。感谢与我同级且同一实验室王铭羽与吴少策，共同度过了互相帮助互相鼓励的 2 年时光。感谢实验室的师弟师妹们，你们学习上的认真，生活上的活跃铭记于心。

最后，衷心感谢我的爸爸妈妈，感谢他们对我的养育之恩，永远在背后无条件默默的支持着我。感谢我的三个舍友，在学习、找工作等很多方面给了我很大鼓励与帮助，因为彼此的鼓励支持，才使得我能够更坚强的面对学习和生活中的困难。

感谢河北省自然科学基金（项目编号：F2015201033，F2017201069）提供资金支持！

藉此论文最后完成之际，再次衷心感谢所有给予我帮助的老师们和同学们。

攻读学位期间取得的科研成果

[1] 靳丽蕾，杨文柱，王思乐等. 一种用于卷积神经网络压缩的混合剪枝方法[J]. 小型微型计算机系统，2018, 39(12):2596-2601.

[2] W. Z. Yang, L. L. Jin, S. L. Wang, et al. Thinning of Convolutional Neural Network with Mixed Pruning[J]. IET Image Processing, 2019. DOI:10.1049/iet-ipr.2018.6191.