装订线

上海大学

毕业论文（设计）

题目：深度学习模型压缩算法研究

学　　院：机电工程与自动化学院

专　　业：自动化

学　　号：18122537

学生姓名：霍英杰

指导教师：修贤超

起讫日期：2022.2.21-2022.5.27

SHANGHAI UNIVERSITY

毕业论文（设计）

UNDERGRADUATE THESIS(PROJECT)

装订线

题 目：深度学习模型压缩算法研究

学 院	机电工程与自动化学院
专 业	自动化
学 号	18122537
学生姓名	霍英杰
指导教师	修贤超
起讫日期	2022.2.21-2022.5.27

目　　录

装订线

摘要

近年来，深度神经网络的性能和算力不断提高，在诸多领域都取得了显著的成就。然而深度神经网络模型规模和复杂度的增长会产生巨大的计算成本和内存消耗，这给深度神经网络模型在移动端上的部署带来了很大的困难。因此，模型压缩算法受到了广泛的关注，并成为了当前深度学习领域的研究热点。经过一段时间的发展，出现了大量优秀的模型压缩算法，剪枝、量化以及知识蒸馏就是其中的佼佼者。本文对这三种模型压缩算法进行了系统性的分析和研究，并分别在神经网络模型上进行了实验，以求在基本不损失模型精度的情况下实现对模型的压缩和加速。本文的研究工作可归纳为以下三部分：

（1）针对神经网络模型存在大量的低贡献度冗余参数的问题，采用了两种剪枝算法，分别是非结构化剪枝和结构化剪枝。非结构化剪枝以参数的绝对值作为剪枝评判标准，在多层感知器上实现了基于权重粒度的剪枝。结构化剪枝以参数的 L2 范数作为剪枝评判标准，在卷积神经网络上实现了基于卷积核粒度的剪枝。两种算法均以微小的精度损失实现了良好的压缩效果。

（2）针对神经网络模型参数的存储方式存在较大冗余量的问题，采用了两种量化算法，分别是聚类量化和线性量化。两种算法在卷积神经网络的卷积层和全连接层上均实现了对参数的低比特量化，有效降低了模型参数的位宽，并且精度损失极低。融合了剪枝和量化两种算法，设计了完整的基于剪枝-量化的模型压缩策略，并在卷积神经网络上验证了融合算法的可行性和有效性。

（3）针对小型神经网络模型在学习和预测上存在很大局限性的问题，采用了知识蒸馏算法。通过大规模的高性能教师网络训练小规模的学生网络，使其获得与大规模网络近似的性能和精度。算法给教师网络的输出层加上了温度参数，以增强其预测结果的信息熵，并重新定义了学生网络的损失函数，以教师网络的预测结果和真实标签共同指导学生网络的学习训练过程，显著提高了其性能、精度以及泛化能力，实现了对原网络模型的压缩和加速。

关键词：剪枝；量化；知识蒸馏；模型压缩；神经网络

ABSTRACT

In recent years, the performance and computing power of deep neural networks have been continuously improved, and remarkable achievements have been made in many fields. However, the growth of the scale and complexity of deep neural network model will produce huge computing cost and memory consumption, which brings great difficulties to the deployment of deep neural network model on mobile terminal. Therefore, model compression algorithm has been widely concerned and has become a research hotspot in the field of deep learning. After a period of development, a large number of excellent algorithms have emerged in the field of model compression, among which pruning, quantization and knowledge distillation are outstanding ones. In this paper, the three model compression algorithms are systematically analyzed and studied, and experiments are carried out on the neural network model respectively, in order to realize the compression and acceleration of the model without losing the model accuracy. The research work of this paper can be summarized into the following three parts:

（1）To solve the problem that the neural network model has a large number of redundant parameters with low contribution, two pruning algorithms are adopted, namely unstructured pruning and structured pruning. Unstructured pruning based on weight granularity is implemented on Multilayer Perceptrons with the absolute value of parameters as pruning evaluation criterion. Structured pruning based on convolution kernel particle size was implemented on convolutional neural network with L2 norm of parameters as pruning criterion. Both algorithms achieve good compression results with little precision loss.

（2）To solve the problem of large redundancy in the storage mode of neural network model parameters, two quantization algorithms are adopted, namely clustering quantization and linear quantization. Both algorithms realize the low bit quantization of parameters at the convolution layer and the full connection layer of the

convolutional neural network, and effectively reduce the bit width of the model parameters, and the precision loss is very low. In addition, the pruning-quantization algorithm and pruning-quantization algorithm are combined to design a complete model compression strategy based on pruning-quantization, and the feasibility and effectiveness of the fusion algorithm are verified on the convolutional neural network.

（3）Knowledge distillation algorithm is adopted to solve the problem that small neural network model has great limitations in learning and prediction. The performance and precision of the large-scale teacher network are similar to that of the large-scale network by training the small-scale student network. Temperature parameters are added to the output layer of teacher network to enhance the information entropy of prediction results. The loss function of the student network is redefined, and the prediction results of the teacher network and real labels are used to guide the learning and training process of the student network. The performance, accuracy and generalization ability of the teacher network are significantly improved, and the original network model is compressed and accelerated.

Keywords: Pruning; Quantization; Knowledge distillation; Model compression; Neural networks

装订线

第一章 绪论

1.1 研究背景及意义

2016 年 3 月，由 Google 旗下 DeepMind 公司开发的阿尔法围棋（AlphaGo）横空出世，接连战胜了当时代表了人类职业围棋顶尖水平的柯洁、李世石以及数十位来自中日韩的围棋高手，这代表着人工智能机器人 AlphGo 在棋力的比拼上完全战胜了人类。随着 AlphGo 人工智能机器人在围棋领域的大获成功，人工智能成为了这几年来相当热门的话题，在诸多领域都得到了广泛的关注和应用，并不断推动着相关领域的发展。深度学习作为 AlphGo 的主要工作原理，有力推动了人工智能相关技术的变革和进步，展现了非常高的研究价值和巨大的发展潜力。在世界各国大力支持信息化和数字化推进的大环境下，发展信息技术成为了国家引领创新和驱动转型不可或缺的先导力量。这使得近年来计算机科学在深度与广度上都取得了显著的进步，运算和存储能力有了极大的提升。深度学习技术由此获得了充裕的发展空间，并依托于强劲的计算能力和庞大的存储空间在工业界和学术界都贡献了大量的成果。深度学习是机器学习领域的重点研究方向，其技术在计算机视觉[1-3]、语音识别[4-6]、自然语言处理[7][8]、自动驾驶[9][10]等领域均得到了广泛的应用，普及到了交通运输、网上购物、影视娱乐、医疗健康、军事安防等各种各样的现实场景，不仅便捷了人类社会日常的生活和生产，还协调促进了各学科领域的发展。

图 1-1 中国棋手柯洁与 Alphgo 对弈

深度神经网络[11]（Deep Neural Network，DNN）是深度学习领域的核心技术，拥有类似人脑神经网络的结构，具有从海量数据中提取知识和特征的能力，通过设计深度学习算法以及构建和训练神经网络模型，能够有效解决许多复杂的实际问题。首个神经网络的数学模型出现在 1943 年，即数理逻辑学家 Pitts 和心理学家 McCulloch 共同建立的 McColloch-Pitts（MP）模型[12]，由于模型结构过于简单并且当时的软件和硬件水平受限，使其难以应用在实际问题上，因此对神经网络的理论研究停滞不前。直到上世纪八十年代，在 Hinton 与 Lecun 相继提出反向传播算法[13]以及识别手写字体的卷积神经网络[14]（Convolutional Neural Networks, CNN）后，神经网络才逐渐突破了发展瓶颈。在 2014 年，AlexNet[15]开创性的使用 GPU 提升运算速度，显著加快了训练模型的过程，并部署了梯度消失问题的解决方案，从而极大的促进了深度学习的研究。在深度神经网络不断衍生出新研究方向的同时，得益于软件和硬件的快速发展，深度神经网络能够在高性能平台的支持下有效解决更复加杂的问题，例如卷积神经网络，以其独有的卷积运算结构，在图像处理方面表现优异，被普遍应用在计算机视觉领域。在神经网络性能飞速提升的同时，模型的规模和复杂程度呈几何式增长，包含数百万级甚至上亿参数的大规模的深度网络模型不仅对计算机的中央处理器和图像处理器有极高的要求，还会产生高昂的运算成本和大量的数据冗余。卷积神经网络从最早的 3 层的 LeNet[16]发展到 8 层的 AlexNet[15]（见图 1-2）再到 19 层的 VggNet[17]，还有拥有更深层数的 ResNet[18]和 DenseNet[19]，网络结构一步步变得更深更宽更复杂，虽然精确度和算力显著上升，但是庞大的模型和计算量成为了实际应用的瓶颈。

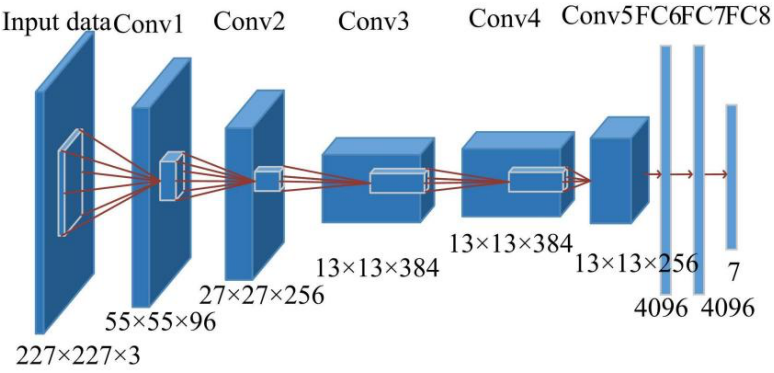

图 1-2 AlexNet 网络结构（来自[15]）

在智能移动设备飞快普及的今天，便携性变得愈发重要，然而大规模的神经网络难以直接部署在资源相当有限的移动端设备，这使得深度学习领域许多优秀的科研成果无法实际应用在诸如智能手机、平板电脑等计算、能耗和存储资源都十分受限的移动设备上。在移动设备的算力和性能难以支持深度学习部署的困境下，学界和工业界推出了一些针对移动端深度学习部署的解决方案。例如将模型直接部署在云端上[20][21]，移动设备通过网络与云端服务器进行信息的传输交互，由此模型能够在云端充裕的资源条件下进行推理运算，同时也保障了模型精度和数据处理准确度。但是云端部署的的解决方案还是存在一些弊端。第一，云端十分依赖网络进行信息传输交互，网络信号波动会导致数据传输有延迟或中断以至于推理运算有误甚至失败，而且智能手机的面容识别解锁方式和智能语音助手都需要在离线的环境下依然可以正常运行，但是云端却无法提供离线服务。第二，云端服务器会承担大量用户的深度学习计算需求，运营的成本大大增加。第三，用户数据必须上传云端服务器，服务的实时性和用户数据的隐私性得不到保障。基于以上三点原因，可见在云端上部署深度学习模型并不是良好的解决方案。

在深度学习相关研究不断深入的同时，研究学者发现大规模的深度神经网络存在着大量的冗余信息[22][23]，这些参数以及结构对于神经网络的预测结果影响甚微。对此类冗余进行系统性和规模性的精简不仅可以在一定程度上降低过拟合现象，更是能大大压缩神经网络模型体积以及运算成本，并且模型的精度也有所保障。这种解决方案能够实现大模型的压缩和优化，我们就可以将压缩后也保持着高准确的小模型应用在资源有限的移动设备上，深度学习领域大量的科研成果也得以转移到小型设备上进行有效应用。学术界和工业界也对深度学习模型压缩和加速技术投入了广泛的关注，各种各样的神经网络模型压缩和加速的部署策略层出不穷，已经成为了深度学习领域新的研究热点。模型压缩本质上就是要在保持原模型功能和精度的条件下，对模型进行轻量化处理，可以不改变原有的模型结构进行参数和维度的缩减，也可以重新设置更为简单的网络结构，达到显著减少模型存储量和计算量的效果，而如何在模型压缩后保持高精度和近似的性能是一个非常有挑战性的问题。大型神经网络模型有着细致严谨的结构，在经过重复的学习和训练后具有优秀的精度和稳定性，这是所有网络参数和结构共同作用的结

果，对其中的每一部分进行修改和删减都会对模型精度造成影响。因此非常需要科学实用的模型压缩算法和深度神经网络压缩和加速策略，以此平衡模型精度和规模之间的关系并减小精简参数结构对模型推理运算结果的影响，达到最大程度上化解复杂模型与小型设备矛盾的目的，最终实现其在资源受限的移动设备上的有效部署。本课题便是以研究深度神经网络模型压缩和加速的算法以及策略为出发点，以求在基本不损失模型精度的条件下，节约参数并降低其计算时间。总之，这是一个算法研究与实际应用相结合的前沿性课题，具有重要的科学意义和实用价值。

1.2 国内外研究现状

随着近年来便携设备的快速发展，人们对于深度学习模型的部署需求大大增加，许多设备都亟需深度神经网络模型强大的算力来提供更智能便捷的服务。为适应当前社会对大数据处理的迫切需求，国内外近年来对深度学习模型压缩算法的研究工作层出不穷[24-27]。但由于深度学习的理论分析是一个复杂的问题，神经网络的压缩算法也具有相当的挑战性，涉及计算机、 控制工程、优化等多个领域，所以现有的成果还是较少，仍亟需相关理论、算法与应用的突破。综合这几年模型压缩技术的研究方向和热点来看，根据其技术方法的特征，可主要分为三类：参数剪枝（parameter pruning）、参数量化（parameter quantization）、知识蒸馏（knowledge distillation），下面分别对这三类的研究现状进行初步的介绍。

（1）参数剪枝：剪枝算法从本质上来说是对模型的整体参数、滤波、卷积层、通道的重要性进行权衡，并去除结果影响较低的部分，故称之为剪枝算法。对于冗余量非常大的网络有很好的压缩和加速作用，并且能在一定程度上解决网络的过拟合问题。Han 教授发表的 2016ICLR 最佳论文[28]指出了神经网络中存有大量冗余性，将 AlexNet 这种深层模型的体积压缩了 35 倍并保持了其精度的下降在很小的范围内，通过剪除权值低于预设值的连接，实现对网络的稀疏化，减少网络的冗余和小信息量的权重，最后对剪枝完毕的稀疏网络结构进行再训练，以提高其性能和精度。剪枝的主要对象可大致分为如下四类：权重（Weights）、神经元（Activation）、滤波器（Filters）、通道（Channels），除去以上四类近年

还有对 Group-wise 剪枝和对 Stripe 剪枝。基于剪枝粒度的差异，可以将剪枝算法分为非结构化剪枝以及结构化剪枝。非结构化剪枝的粒度最细，即针对单个参数（权重）进行剪枝，将模型的参数张量中被判断重要性低于阀值的参数置零，以此稀疏参数张量，实现对模型的压缩和加速。Guo 等人[29]在 2016 年提出的的动态连接剪枝就是典型的非结构化剪枝，Aghasi 等人[30]在 2017 年提出的通过求解凸优化规划移除 DNN 每层连接也是这种方法，其结构如图 1-3 所示，二者均取得了不错的成果。但这种方法也存在很大的问题，即虽然修剪率高，但会构成非

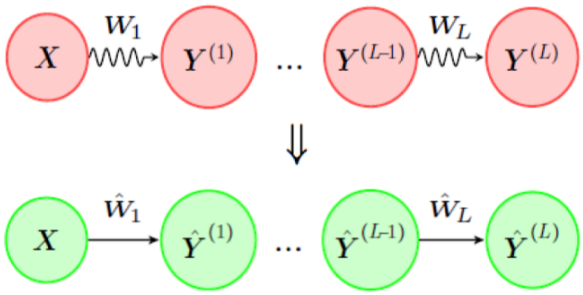

图 1-3 基于求解凸优化规划对 DNN 层剪枝（来自[30]）

结构化稀疏滤波器（稀疏矩阵），由于通用硬件不支持加速稀疏矩阵计算，所以需要专用硬件才能达到压缩和加速效果。结构化剪枝针对滤波和通道甚至层进行剪枝，且不改变原始的卷积结构，稀疏矩阵进行加速的问题得到了解决，无需专用硬件也能实现压缩加速，结构化剪枝通过直接对参数张量进行修剪以达到产生结构化稀疏的目的。近年关于结构化剪枝的研究成果颇为丰硕，He 等人[31]提出了用 LASSO 回归来选择通道以及最小二乘重构进行通道的修剪，Liu 等人[32]通过对 BN 层缩放因子 γ 进行优化，由于 γ 的大小能直接影响特征图的分布，所以可将其作为衡量通道重要性的指标，将贡献度低的通道按预设剪枝率剪除，如图 1-4 所示。还有 Li 等[33]人采用了基于滤波器的 L1 范数进行剪枝，删除 L1 范数值小于阀值的滤波，而后重新训练恢复精度。近年来在 Filter 和 Channel 进行剪枝的算法占多数，但是在剪枝率无明显变化的情况下精度的提升相当有限，当然还存在一些端到端的模型剪枝方法，其论文发表较少就不再赘述了。

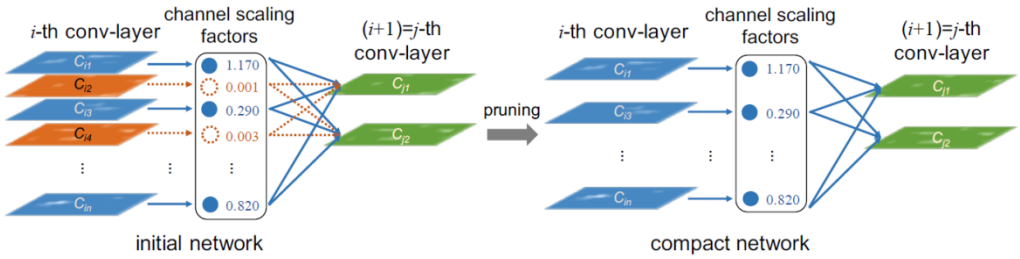

图 1-4 基于通道粒度的剪枝（来自[32]）

（2）参数量化：参数量化方法是指在不改变原有网络结构的情况下用更少的比特来表示每个参数，网络中的权重、激活值、梯度和误差等参数都可以通过改变其位宽来显著减少存储和计算资源的耗费。模型量化的优点很多，首先，可应用在各种模型上，且量化执行后无需重新训练。其次，非常节省存储空间以及内存资源，如前文所述，量化后占用空间的下降是显而易见的，这为在移动端部署模型提供了良好的条件。最后，参数量化带来了更快的推理速度以及更少的能耗，大部分处理器一般都支持 8 比特数据处理的加速，而且量化后的低比特数据在内存中移动更快，对能源的消耗更小。但是参数量化还是存在一定的局限性，例如信息量丢失导致的精度下降，还有特殊位宽量化不适用于很多已有训练方法和硬件架构，相对其他方式不够灵活。根据量化程度的不同可以将参数量化分为二值量化和浮点量化，浮点量化就是以更低精度的参数如 8 比特参数取代原来标准的 32 比特参数，而将参数 8 比特量化可以只付出少量的精度下降的代价便能将模型的体积缩小约四倍。例如 Vanhoucke 等人[34]就将权重参数量化到了 8 位，大幅提高了运算速度，且精度损失非常小。二值量化就是将神经网络浮点参数均置为同精度并离散的＋1 和－1，仅仅只占一比特空间，相较于 32 比特参数的网络尺寸骤缩 31 倍，但这种方法难以保持网络的精度和表达能力。Rastegari 等人提出的 BWN 和 XNOR-Net[35]以及 Courbariaux 等人提出的 Binaryconnect[36]和 BinaryNet[37]都是表现十分优秀的二值网络，其中 BWN 进行的是权重参数的二值化，而 XNOR-Net 对输入数据也进行了二值化处理。参数量化根据映射函数是否为线性还可以分成两类，即线性量化和非线性量化。不同于非线性量化如聚类量化，线性量化的聚类中心呈规则的均匀分布，通过较为简单线性变换就可以让原始数据和量化后数据互相转换。由于卷积和全连接都是比较基础的线性计算，所

以可直接使用量化后的数据进行推理运算，在压缩模型体积的同时推理计算速度明显上升。多数浮点量化的方法都是线性量化，例如 Jacob B 等人[38]提出的在前向推理时将权重值和激活值量化为 8 比特，将浮点数运算转换为整数运算，其流程如图 1-5 所示。聚类量化就是典型的非线性量化，同样也是效果非常优异的参数量化方法，是由 Gong 等人[39]开创性地提出的，他们将 K-Means 聚类应用在量

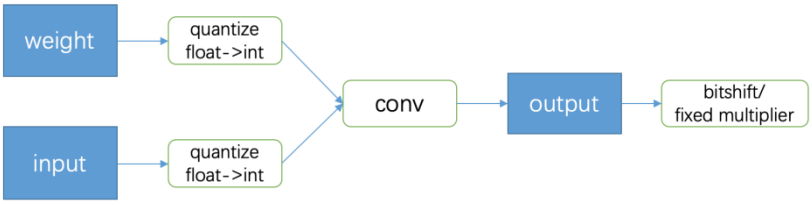

图 1-5 卷积在 8bit 量化推理时的流程（来自[38]）

化全连接层参数上，很好的兼顾了模型体积和精度，Wu 等人[40]在后来将聚类量化方法扩展应用到了卷积层参数上，K-Means 聚类算法可以将样本分类并算出聚类中心，我们只需要将存储聚类中心作为权重索引值，以样本中心加索引的方式替代原始权重信息，大大降低了内存消耗。Han 等人[28]在 ICLR2016 的最佳论文里就用到了聚类量化的方法，如图 1-6 所示，他们用 K-Means 算法对每一层权重矩阵聚类获得几个不同的集群，用每个集群的中心值作为同一集群的共享权重，而后只需要储存集群权重的索引即可，最后再调整训练模型对权重作一补偿，在不改变梯度信息的情况下恢复精度损失。还有一些论文发表较少的方法也取得了不错的效果，例如三值量化和混合位宽，但与上文中的方法相比优势不大。

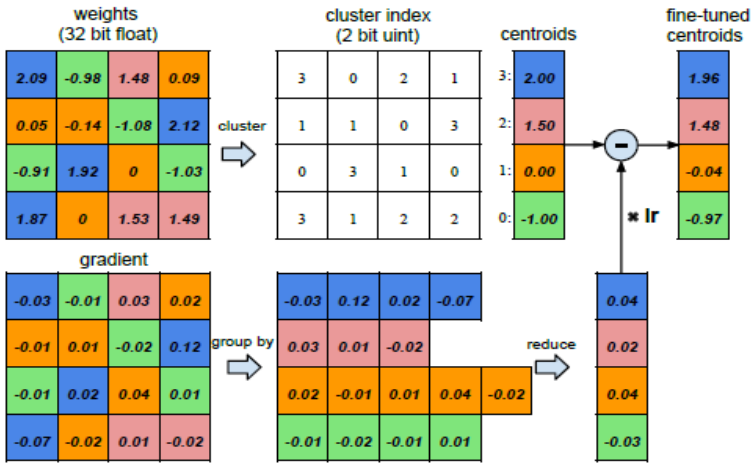

图 1-6 Han 等人进行聚类量化时的基本方法（来自[28]）

（3）知识蒸馏：知识蒸馏是一种基于教师-学生架构的模型压缩方法，由性能优秀且精度高的大规模深度神经网络作为教师模型，而结构简单的轻量化小型网络作为学生模型，复杂度高的大型教师网络可以通过 SoftMax 函数输出一个类别分布概率，而小型的学生网络可以学习教师网络的类别分布输出，以此完成教师网络对学生网络的知识迁移，使得学生网络能够获得更接近大型复杂教师网络的优异性能和表现力。Bucilua 等人[41]最早提出基于知识迁移（Knowledge Transfer）的深度学习模型压缩算法，这种方法可以训练内置的含有伪数据标记的强分类器的压缩模型，在输入标记后的样本后，经过训练可以复现原始分类器也就是原大型网络的输出。而完整的神经网络知识蒸馏思想是由 Hinton 团队在 2015 年提出的[42]，其思想框架如图 1-7 所示，他们认为 one-hot 形式标注的训练样本，其标签信息熵过低，而采用 SoftMax 分类器输出的类别分布概率拥有更好的表现力，能很好的表达出同一输入经模型推理运算后关于各个类别的置信度，而这能更好的体现数据样本的相似结构。这些信息蕴含着网络的泛化和表现能力，具有非常高的学习价值，于是他们使用教师-学生模型，以 one-hot 形式的标注作为硬知识，又将教师模型的 SoftMax 函数输出的概率信息蒸馏出来作为软知识，相辅相成共同作为学生网络学习的知识，即实现了模型压缩的效果，又提升了学生网络的性能。Remero 等人[43]还针对学生网络结构提出了 FitNets 网络，在那个时期性能优秀的网络深度一般更大，因此文章选择的学生网络比教师网络更窄更深，FitNets 网络将 Hints 设置为教师模型特征提取器的中间层输出，结合 Softmax 函数的输出对学生网络进行知识蒸馏，使得学生网络获得更充沛的先验知识。而 Ba 等人[44]观点与 Remero 等人的观点不同，他们以深度网络作为教师模型，将学生模型设计为更浅更宽的浅层网络，经过训练后得到新的浅层网络模型（只有一层的神经网络模型）可以和深度模型达成一样的效果，也就证明了在参数数量相同的情况下，存在更好的算法可使浅层网络拥有跟深度网络一致的性能，网络深度不是模型性能绝对必要的条件。Zagoruyko 等人[45]还提出了基于注意力迁移的知识蒸馏方法，注意力图是指多通道输出特征图激活后（激活函数值）的加和，这种方法通过迁移注意力图松弛了 FitNets 方法所提的假设条件，将注意力图也定义为一种知识，然后通过知识蒸馏训练学生模型以得到更好的性能表现。关于知识

蒸馏领域的一些拓展研究成果也很优秀，不论是传统的知识蒸馏还是基于经典方法的扩展都取得了不错的效果，可见其实用性较高且效果不错，但也存在泛化性不好的局限，仍有较大进步空间。

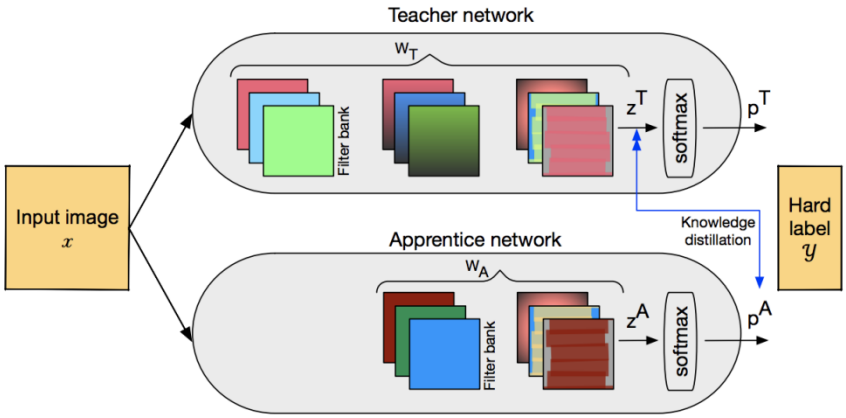

图 1-7 Hinton 等人提出的知识蒸馏架构（来自[42]）

装 订 线

除此之外，还有紧凑网络设计[46-48]、神经网络架构搜索[49-51]、低秩因子分解[52-54]等其他方法。其中低秩因子分解主要是对原始网络张量进行处理，将其作为满秩矩阵，然后使用低秩矩阵逼近来实现网络参数的简化，此方法是利用原始网络张量矩阵信息冗余和低秩的特征进行模型压缩的，将低秩矩阵分解成小型矩阵的乘积后还可以进一步压缩原始网络模型参数量。低秩因子分解方法多在较小规模网络和大卷积核上有不俗的表现，Jaderberg 等人[52]就对卷积核进行了分解，Wen 等人[53]还提出了将权重信息协调到低秩空间的正则化方法。但因其实现难度较大，因此在相关研究成熟后的这几年发展相当缓慢。低秩分解实现难度体现在两方面，首先，矩阵分解操作十分复杂，会耗费高额的计算成本。其次，由于是逐层分解，所以低秩分解不利于压缩全局参数。再次，使用低秩分解方法需要对分解压缩后的网络进行大量的重新训练才能达到收敛。最后，近几年出现的很多新网络采用的 1×1 小卷积核不利于使用低秩分解方法，比如 Lebedev 等人[54]提出的针对四维卷积核的 CP 分解，这使得模型压缩和加速难以实现。基于以上原因，低秩因子分解方法已经不再是主流的研究方向。

1.3 论文结构安排

本文主要研究深度学习模型压缩算法，概述了现有的模型压缩方法，并且针对剪枝、量化和知识蒸馏三种方法进行了深入研究。以保证模型精度基本不损失为前提条件，设计了多种神经网络压缩和加速算法，分别在神经网络模型上进行了实验，并根据实验结果数据分析了每种方法的优劣和实际效果。总体来看，本文的结构安排如下：

第一章是绪论部分，首先阐述了模型压缩的研究背景及意义，然后概括的介绍了目前模型压缩算法的研究现状，最后简述了文章的研究内容和主要工作。

第二章深入研究了剪枝算法，首先从结构化剪枝和非结构化剪枝两方面切入，设立了评估参数重要性的标准。然后分别在卷积神经网络和多层感知机上完成了卷积核粒度和权重粒度的剪枝实验。最后实现了对网络模型的加速和压缩，有效减少了网络模型的冗余量，并且降低了网络发生过拟合的概率。

第三章深入研究了量化算法以及剪枝-量化融合算法，首先对聚类量化和线性量化两种较为普遍的量化方法进行了初步的介绍。然后运用两种方法分别实现了对卷积神经网络模型的卷积层以及全连接层的参数压缩，将其量化为不同位宽，大大降低了参数的存储空间。最后分析了融合剪枝和量化算法的可行性，并设计了具体的策略和算法，将剪枝和量化两种方法融合在一起，实现了对 LeNet_5 网络模型的压缩和加速，取得了不错的效果。

第四章深入研究了知识蒸馏算法，首先对知识蒸馏的框架和思想做了详实的介绍和概述，并根据其框架和思想设计了算法。然后实现了对教师网络 SoftMax 分类器输出信息的软化，使其很好的表达出了教师网络内的暗知识。接着通过重新定义学生网络学习训练过程中的损失函数，使得从教师网络中提取出的知识能有效指导学生网络的学习训练。最后使小规模的学生网络获得了跟大规模的教师网络近似的性能和精度，并且进一步提升了学生网络的泛化能力。

第五章是对论文整体内容的总结，以及对未来研究工作的展望。

第二章 基于剪枝的压缩算法研究

近年来，剪枝算法领域出现了很多优秀的思路，在深度学习模型的复杂度逐年升高的环境下，能够有效对模型进行压缩和加速，并且去除模型的参数、运算以及结构上的大量冗余，减少过拟合问题出现的概率。而且大多数剪枝策略都非常灵活，适用于多种情况，因此剪枝算法的研究和应用价值都比较高。本章主要对剪枝算法的两个方向，即结构化剪枝与非结构化剪枝进行介绍，并针对实际问题进行理论分析，然后分别设计算法在 MLP 和 CNN 上进行实验，最终对所得实验结果进行评估和讨论，给出调参建议以及对此方法的评价。

2.1 针对权重的非结构化剪枝

多层感知器（MLP，Multilayer Perceptron)是结构非常基础的前馈人工神经网络模型，由大量的简单神经元构成，每一个神经元都接受上一层神经元的大量输入，这导致网络推理的运算非常缓慢，并且也存在数据冗余的情况。由于网络的整体性能受局部性能的影响有限，因此如果对每层的权重进行非结构化剪枝就可以在保持精度的前提下大大提升其推理运算速率。本节基于此针对 MLP 网络模型设计非结构化剪枝算法，并详细阐述算法原理和实现步骤，然后在对比实验中通过选取不同的剪枝率对每一层的 Weights 进行剪枝，最终得出损失精度最小的最佳剪枝率范围。

2.1.1 问题分析与方法介绍

MLP 网络模型结构的复杂度一般由隐藏层数、节点数量和激活的方式等决定，复杂的 MLP 网络在经历大量训练后很可能会产生过拟合（overfitting）现象，即训练时网络模型对抽样误差也进行了学习拟合。这导致训练后的模型只在训练集上表现良好，而在测试集上表现很差，其泛化能力非常弱。所以模型复杂度高且参数数量大不仅会占用更多储存空间资源且拖慢处理运算的速率，还会产生过拟合问题，导致模型泛化能力差，无法有效应用在其余数据集上。

为了解决这一问题，可以采用非结构化剪枝算法，以此降低网络模型的复杂

度。非结构化剪枝即是细粒度的剪枝，可以对权重连接级别的参数进行剪枝，对卷积神经网络来说，可以从其卷积核张量中挑选低重要性参数，用掩膜的方式来遮蔽从而使其不参与计算（等同于置 0）。这种不规则的剪枝方法灵活性非常高，但会构成稀疏矩阵，需要特殊硬件以及专用库的优化支持，导致在很多深度神经网络模型上难以实现，实用性低。而解决 MLP 网络冗余和过拟合问题需要对细粒度参数（Weights 和 Bias）进行剪枝处理，因此非结构化剪枝方法比较适合用于 MLP 网络模型的压缩和加速。机器学习的正则化方法可以减弱网络模型的过拟合以及增强网络模型的泛化能力，Drop Connect[55]就是通过对部分神经元之间的连接权重作置零处理来对模型进行正则化的，这与非结构化剪枝处理权重的思想是一致的，在一些任务上二者可以达到相似的效果。综上所述，对 MLP 网络进行非结构化剪枝不仅可以实现模型的压缩和加速，还可以解决其过拟合问题。

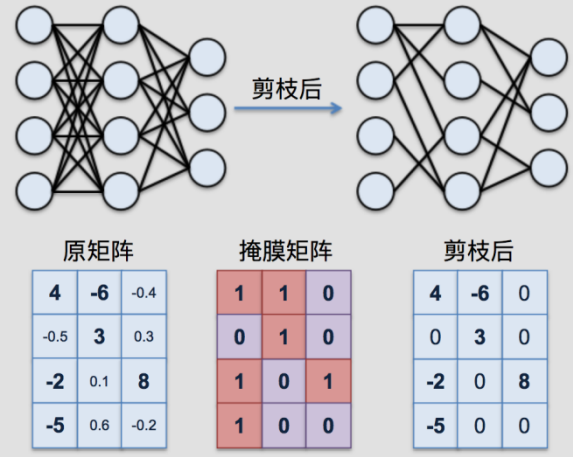

图 2-1 非结构化剪枝原理图

2.1.2 算法原理和实现过程

在 MLP 模型上可剪枝的参数分为三类，即神经元（Neuron）、权重（Weights）和偏置（Bias）。本算法就是对权重进行剪枝，属于非结构化剪枝的方法，而对权重以及神经元进行剪枝也是最早期且最简单基础的剪枝技术。这种算法的差异主要在于评判权重或神经元重要性的指标（预设阈值）以及对每层的剪枝比例，在设置了预设阈值和剪枝率后，就可以对网络模型进行剪枝。剪枝步骤可以具体分为四步：（1）训练一个 MLP 网络模型；（2）定义评估权重的重要性的方法，并用具体的数值来衡量重要性；（3）对每层权重参数进行评估，并根据剪枝率

设置一个预设阀值；（4）去除重要性指标低于阀值的权重参数得到剪枝后网络。

根据 MLP 网络模型的结构，线性模块的表达公式为：

$$\hat{y} = w[0] \cdot x[0] + w[1] \cdot x[1] + ... + w[p] \cdot x[p] + b \qquad (2\text{-}1)$$

其中$\hat{y}$为对输出 y 的估计值，x 和 w 分别为样本特征值及其权重，这是一个特征值加权求和的过程。MLP 网络的隐藏层就在进行这样的加权求和计算过程，计算结果传入神经元节点后经过激活函数非线性化，得到的结果作为下一层的输入继续运算。由此可知，输出特征值主要是由权重值的大小决定的，绝对值越小越接近零的权重值对输出的贡献度越小，因此针对 MLP 的权重剪枝可以根据权重值的绝对值大小进行排序，以此作为判断权重参数重要性的指标。然后要根据剪枝率设置阀值，就要确定这个多维数组的观测值从小到大排列的任意百分比分位数，可以用 numpy 里的 np.percentile 函数实现，将权重矩阵和剪枝率 p%输入后，会返回权重参数的绝对值从小到大排序处于 p%位置的第 p 百分位数。而后将返回值预设为阀值，将绝对值小于阀值的权重置零，即完成对 MLP 模型的权重剪枝。

在具体实现时，需要构建 Mask 矩阵，以 Mask 矩阵作为掩膜对全连接层的 Weights 矩阵进行处理即可得到剪枝后的稀疏矩阵。我们首先在全连接层每层设立 Mask 标志位然后增加一个储存 Mask 的变量，而后将权重参数拉伸为一维数组并用阀值进行筛选，生成一个 bool 矩阵，并将大于阀值的设为 True，小于阀值的设为 False，以此生成一个 bool 矩阵，将其转化为 0-1 矩阵，True 转化为 1，False 转化为 0，这个矩阵就是作为掩膜的 Mask 矩阵。最后将 Mask 矩阵与初始权重矩阵对应的参数相乘，以掩膜的方式将需要剪枝的权重置零，最终完成对 MLP 网络模型权重的非结构化剪枝。

2.1.3 实验结果分析

（1）数据集以及实验模型

本实验选用的是 MNIST 数据集，数据集包含了 250 个人的手写数字图片，其中一半来源于高中生，剩余的来源于人口普查局。7 万张图片中 6 万张作为训练集，剩下 1 万张作为测试集，均由 28×28 像素的 0-9 的手写数字灰度图片组成，标签为 one-hot 编码形式的十个数字 0-9。其中数字均经过尺寸标准化且位于图像

中心，每一图像都被平展并转换为 28×28 个特征的一维 numpy 数组，图 2-2 展示了数据集中的 64 张图片，图 2-3 对其中一张图片进行了像素化显示。

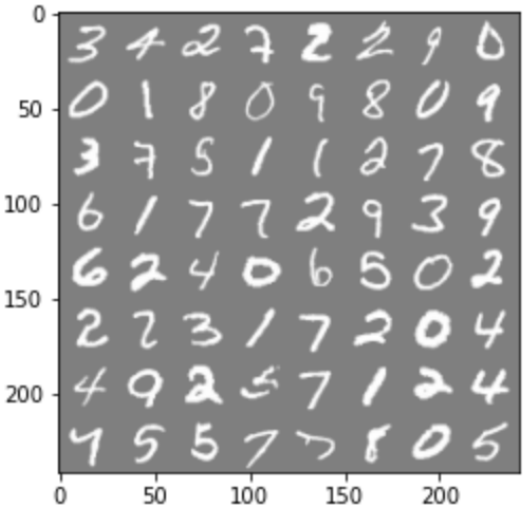

图 2-2 数据集中的 64 张图片

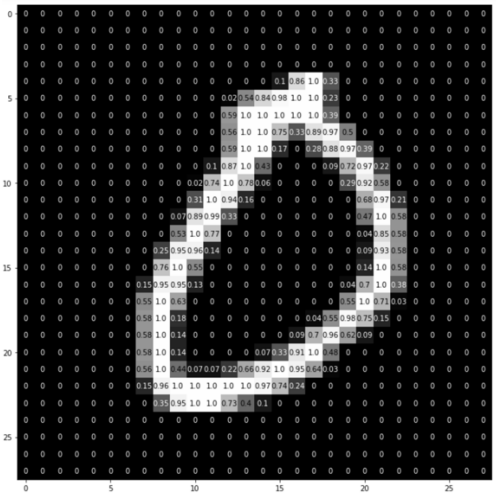

图 2-3 图片 0 的像素化显示

实验选用的网络模型为 MLP，由三层全连接层以及两层激活函数层组成，batch_size 设置为 64，训练论数设置为 2，训练完毕后在 MNIST 上的 average loss 为 0.1066，准确率为 9638/10000 (96%)。

（2）实验结果分析

设置不同的剪枝率对模型进行非结构化权重剪枝，其精度变化如下表：

表 2-1 MLP 在不同剪枝率下的精度变化

剪枝率	模型精度	Average Loss
10%	9674/10000（97%）	0.1070
20%	9679/10000（97%）	0.1038
30%	9667/10000（97%）	0.1067
40%	9651/10000（97%）	0.1095
50%	9633/10000（96%）	0.1124
60%	9638/10000（96%）	0.1208
70%	9583/10000（96%）	0.1503
80%	8884/10000（89%）	0.3926
90%	5166/10000（52%）	1.1256

由表中数据可得，在 MLP 模型经过剪枝后，再到剪枝率从 10%到 20%的过程中，模型的准确率一直是上升的，也就是说通过剪枝一部分低重要性的权重参数，可以一定程度上提升模型的泛化能力，解决过拟合问题，获得更高的精度。剪枝率从 20%到 90%的过程中，模型的精度持续下降，特别是当剪枝率为 80%和 90%时，模型的精度骤降，跌落到 90%以下，故可知，MLP 网络模型中大约有 20%到 30%的权重系数对结果的贡献度高，因此我们的剪枝率应当设置在 70%较为合理，既能保持一定准确率，又完成了对模型参数量的压缩，降低了计算成本。

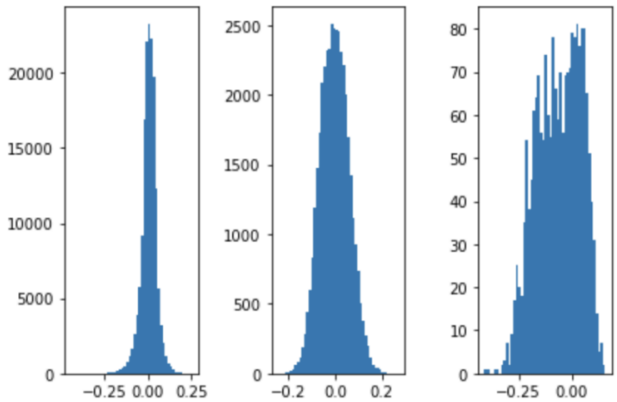

图 2-4 剪枝前的权重参数分布

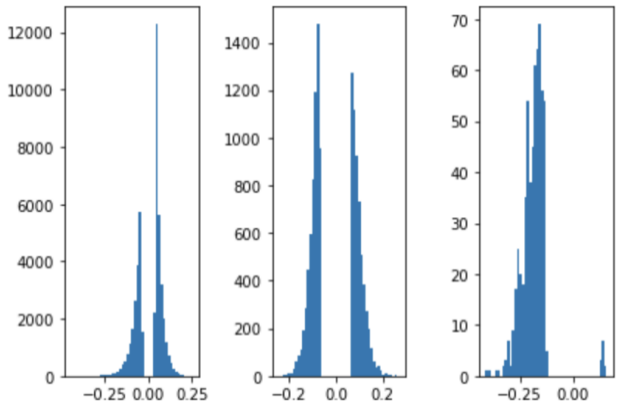

图 2-5 剪枝率为 70%的权重系数分布

由图 2-4 以及图 2-5 的对比，可以看出在经过剪枝后，接近 0 点的权值都被删除了(置 0），证明对 MLP 网络模型权重参数的非结构化剪枝实验是成功的。

2.2 针对滤波器的结构化剪枝

卷积神经网络在深度学习领域的应用相当普遍，因此研究针对于卷积神经网络模型的压缩和加速策略是非常有价值的。由于非结构化剪枝会造成卷积核权重成为不规则稀疏矩阵，而现有的深度学习框架和处理器底层库无法对稀疏矩阵进行加速及优化，因此不适合卷积神经网络的大多数应用场景。这时使用结构化剪枝的方法就能很好的解决这个问题，较好的实现对卷积神经网络模型的压缩和加速。本节就将介绍非结构化剪枝的算法及策略，在卷积神经网络的滤波器的层面上进行剪枝，并对比剪枝前后的模型精度变化，而后通过重训练提高其精度，以求在不损失精度的情况下实现模型压缩和加速。

2.2.1 问题分析与方法介绍

随着卷积神经网络模型被设计的越来越复杂，层宽和层深都显著增长，模型运用在较为小型的数据集上便会产生大量的数据冗余和额外的运算成本，为了降低模型体积以及减少参数数量，可以对模型进行剪枝。而由于当前的硬件仍然不支持对非结构化剪枝处理后的卷积神经网络进行加速优化，所以需要对其采用结构化剪枝的方式，规则化稀疏模型，使其可以和当前深度线性架构及硬件相结合，实现模型的进一步压缩和加速。

结构化剪枝的对象主要是通道、卷积核以及层，是粗粒度的剪枝方法，所以可以保留原始的卷积结构，无需专用硬件就能实现压缩和加速。结构化剪枝还可以分为三个维度即长宽维度、卷积核维度、卷积层维度。由于长宽维度是压缩卷积核的长度或宽度，虽然不影响单一卷积核结构的规则性，但在滤波器层面来看，卷积核组成的张量依然是稀疏不规则的，难以加速，因此应用价值较低。而卷积层维度作为剪枝粒度最粗的方法，直接以卷积层为基本单位进行剪枝操作，即使能实现模型压缩和加速，但对精度影响过大，不好控制。这两种维度的剪枝都较为少见，实际价值不如卷积核维度的剪枝，而且卷积核维度的剪枝不仅不会影响其四维张量的规则性，易于实现压缩和加速，对模型精度的影响也没那么大，因此采用选择卷积核维度的剪枝方法对模型进行压缩和加速是比较好的选择。Drop Out[56]是对神经元输出置零即从网络模型中删去神经元的一种正则化方法，对于卷积神经网络来说，一层的神经元可以对应为一层的特征图，将特征图看作神经元的话就可以将 Drop Out 也类比一种结构化的剪枝。而正则化可以增强模型的泛化能力且减弱模型的过拟合，因此卷积神经网络进行结构化剪枝也可以达成这种效果，可见进行结构化剪枝后还有可能增强网络模型的表现力。

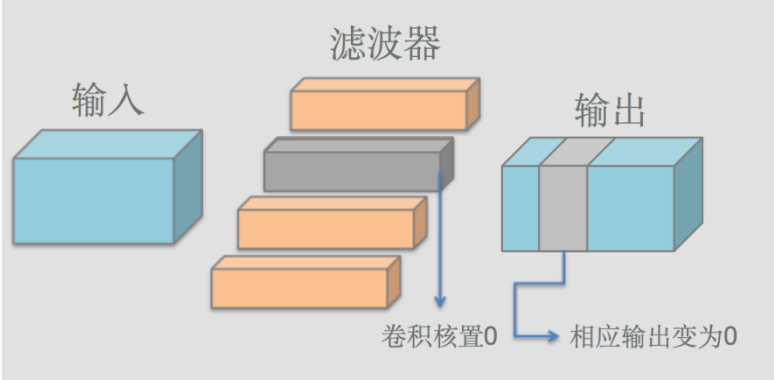

图 2-6 结构化剪枝原理图

2.2.2 算法原理和实现过程

在卷积神经网络上进行卷积核粒度的剪枝，重点在于定义评估卷积核重要性的方法，并用具体的数值对其重要性进行表示，步骤与上文在 MLP 上进行非结构化剪枝相似。首先，训练一个 CNN 模型；其次，制定评估卷积核的重要性的标准，且用易于表示的数值来衡量；再次，计算表示卷积核重要性的数值，然后

根据剪枝率预先设定一个评判的阀值；最后，删除重要性指标低于阀值的卷积核，得到剪枝后网络。完成以上步骤后，对模型进行重训练调整，提高其精度。

算法采用 L2 范数来评估卷积核的重要性，范数是指可以表示“距离”概念的函数，而矩阵范数可以衡量矩阵引起变化的大小，也就是说通过对卷积核参数矩阵的 L2 范数计算，可以根据其大小判断卷积核对模型最终输出结果的贡献度。L2 范数偏小的卷积核生成的特征图的数值一般偏小，反之，L2 范数偏大的卷积核生成的特征图的数值一般偏大。L2 范数的计算公式为：

$$\| x \|_2 = \sqrt{\sum_i^n x_i^2} = \sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2} \tag{2-2}$$

实质上就是对卷积核参数矩阵中的每一项求平方和然后开方，L2 范数的规则项越小，代表卷积核权重参数矩阵中的每个参数都很小，均接近于 0。而后通过计算滤波器各个卷积核 L2 范数的大小，就可以对其进行排序。在确定剪枝率后，使用 np.percentile 函数，得到卷积核参数矩阵的 L2 范数从小到大排序处于 p%位置的第 p 百分位数，将其设置为阀值超参数，将小于阀值的卷积核删除，完成对卷积神经网络的结构化剪枝。在剪枝后测试模型精度，而后通过重训练的方法微调模型，使得其精度接近甚至超过原始模型精度。

具体实现时，与在 MLP 实现非结构化剪枝算法相似，首先改造其二维卷积层，即设立判断是否与 Mask 矩阵相乘的标志位。然后将卷积核参数张量变成 numpy 的数据形式，在计算其 L2 范数后用阀值超参数进行筛选，Mask 矩阵可以通过 np.where 函数实现。最后基于 Mask 矩阵删除 L2 范数小于阀值的卷积核，在模型除去被评估为低贡献度的卷积核后，进行重训练微调，提升模型精度，最终完成对于卷积神经网络滤波器的结构化剪枝。

2.2.3 实验结果分析

（1）数据集以及实验模型

本实验选用 MNIST 数据集，实验模型为卷积神经网络模型，由三层卷积层、两层池化 层、三层激活函数层和一层全连接层组成。batch_size 设置为 64，经过两轮训练后，模型的 average loss 为 0.0245，准确率为 9923/10000 (99%)。

（2）实验结果分析

设置不同的剪枝率对模型进行卷积核粒度的结构化剪枝，测试剪枝后模型的准确率，而后重训练模型微调精度，所得实验数据如下：

表 2-2 CNN 在不同剪枝率下的精度变化以及微调后精度

剪枝率	模型精度	Loss	微调后精度	Loss
10%	9911/10000（99%）	0.0280	9916/10000（99%）	0.0282
20%	9909/10000（99%）	0.0357	9909/10000（99%）	0.0311
30%	9900/10000（99%）	0.0682	9916/10000（99%）	0.0285
40%	9837/10000（98%）	0.2153	9922/10000（99%）	0.0291
50%	9310/10000（93%）	1.1084	9886/10000（99%）	0.0353
60%	1243/10000（12%）	2.3248	9674/10000（97%）	0.1033

装订线

根据表中的实验数据，可以发现在剪枝率较小时，例如 10%、20%和 30%时，精度的下降幅度非常小，这说明了卷积神经网络模型中确实存在大量的冗余参数，在进行剪枝后，可以达到提升泛化能力，防止过拟合的效果。而且模型经过剪枝率为 10%和 30%的结构化剪枝后，通过微调能恢复到统一精度，甚至 30%剪枝率下的模型 average loss 更小，由此可知模型的结构在精简后也能拥有同样甚至更好的表现力，模型的结构也存在一定程度上的冗余。在剪枝率为 50%和 60%时，模型精度下降幅度较大，特别是在 60%的剪枝率下模型的精度仅为 12%，与上一节的非结构化剪枝相比，可见粒度更大的剪枝显然对于模型精度的影响更大。同时微调后的模型精度都保持的非常不错。例如在经过剪枝率为 50%和 60%的剪枝后，通过微调可以将模型精度恢复到 99%和 97%，效果显著，因此在剪枝后对模型进行重训练微调非常有必要性。综合来看，剪枝率设置在 50%比较合适，精度下降的幅度较小，且通过微调能恢复为近似原始模型的精度，较好的兼顾了模型的精度和压缩的效果。

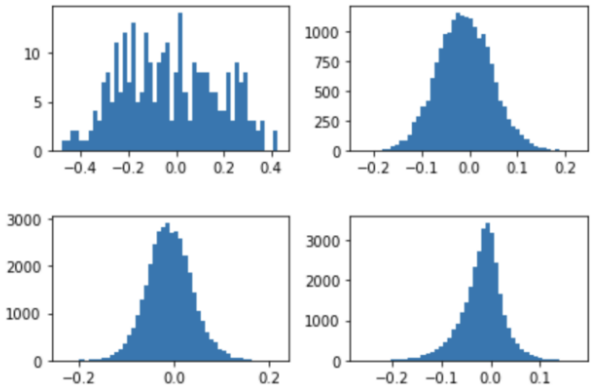

图 2-7 原始模型的权重参数分布

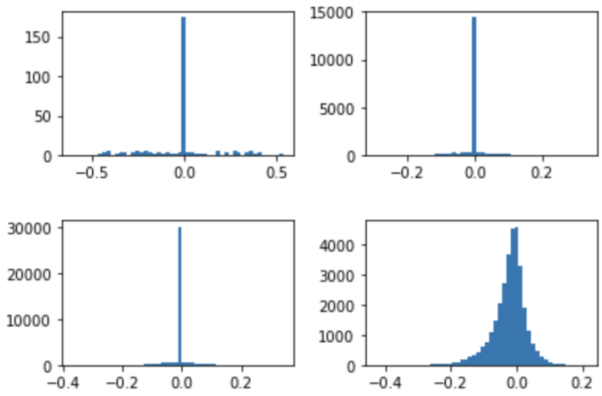

图 2-8 剪枝率为 50%的权重参数分布

由图 2-7 和图 2-8 的对比，可以发现结构化剪枝后，权重参数分布变化非常大，前三层卷积层权重系数大部分都被剪枝（置 0）了，而全连接层的权重系数分布未受影响。粒度较大的结构化剪枝更加彻底，对剪枝后模型进行微调后以更精简的结构也能保持不错的精度。

2.3 本章小结

本章主要从两个方向对剪枝算法进行了研究，分别是非结构化剪枝和结构化剪枝。首先，本章讨论分析了两种剪枝方法需要解决的问题，并对二者的概念、思路以及优劣做了初步的介绍。其次，本章基于剪枝的两个方向分别设计了算法，并详细阐述了算法原理，系统性的陈述了具体实现的步骤和过程。最后，本章通过分别在 MLP 和 CNN 上进行剪枝实验，达到理论和实践相互印证的目的，并且

在分析实验结果后客观评价了两种剪枝方法，给出了部分针对调参和实现过程的建议，对两种方法的实际效果得出了初步的结论。

第三章 基于量化的压缩算法研究

在众多深度学习模型压缩算法中，量化是非常有效且实用的方法之一，狭义上的量化就是将连续的信号离散化，如麦克风将连续的声音离散量化为数字信号。而在神经网络中，参数本来就不是连续的，量化就是改变参数的储存方式，通过参数共享或低比特量化，在保证网络模型精度的同时减少其存储规模。本章主要讨论量化算法的实现和效果，详细阐述聚类量化和线性量化的原理和应用场景，而后设计算法并在卷积神经网络上验证其实现效果。最终还会将剪枝和量化两种压缩方法做一结合，使用剪枝-融合算法对卷积神经网络模型进行压缩和加速。

装订线

3.1 聚类量化算法

聚类量化属于非线性的量化方法，可以将卷积神经网络各层的参数聚集成不同类别，而类别内所有参数就都能用一个参数进行表示，由此达成参数共享的目的。在实际储存中只需保留聚类中心和标签，极大的缓解了模型的存储压力。本节就将基于聚类量化算法对卷积神经网络模型实现压缩和加速，并分析不同量化粒度对精度的影响，评估较为适合的粒度对模型进行聚类量化，尽可能避免精度损失的问题。

3.1.1 问题分析与方法介绍

参数量化方法最核心的思想就是尽可能降低参数的位宽，位宽越小的参数占用更少的储存空间。例如存储 32 比特的浮点型参数所需空间是存储 8 比特的整型参数的四倍，可见位宽的下降带来的储存空间的提升是显著的，而且小位宽参数运算的速度也更快，能很好的降低设备的能耗。但量化方法不可避免会使网络参数丢失掉部分信息量，所以必然会对模型精度造成一些影响，而下降的精度就只能耗费时间成本通过重训练微调恢复。因此选择合适的量化方法和量化粒度，尽量模型平衡精度损失和模型压缩及加速效果，是设计量化算法的关键点。

聚类量化在参数量庞大的情况下，实现效果和稳定性较好。聚类方式是非线

性量化的一种主要手段，通过按科学的标准将数据分割成不同的类别（簇）进行处理，使同类别（簇）内的参数尽可能的接近，保持较高的相似性，并且对于不同类别（簇）的参数，尽量能够使其差异性最大化。在对数据样本进行分类后，将每类的聚类中心（簇质心）作为同一类内参数的共享权值，而聚类量化后需要储存的只有索引标签和聚类中心，相当于减少了参数的位宽，大大降低了储存和运算的成本。例如，将一组 32 位浮点型数据样本聚类成 16 簇，那么存储数据的位数就变成了 4 位，等同于压缩了 8 倍。而聚类量化方法还可以作用在网络正向传播和反向传播两个阶段，网络正向传播时，整体权值由各类共享的聚类中心替代，网络反向传播时，同样可以对参数的梯度值进行聚类量化，根据其学习率对聚类中心进行更新。由此可见聚类量化方法具有很不错的压缩和加速效果。

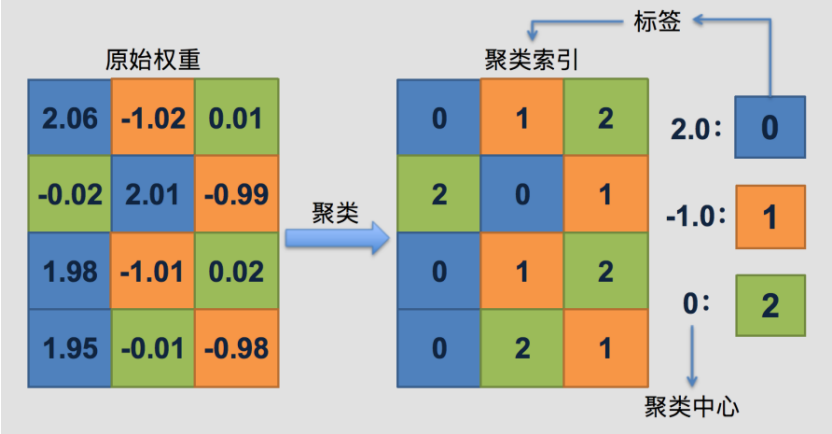

图 3-1 聚类量化原理图

3.1.2 算法原理和实现过程

聚类量化方法通过 K-Means 算法[57]来实现聚类，K-Means 就是一种非常实用的基于迭代求解的聚类分析算法，其迭代过程分为四步：（1）当我们想将全体参数分为 Q 个簇时，K-Means 算法就可以随机找出 Q 个对象（质点），使其成为最初的聚类中心（簇质心）；（2）计算数据样本中每个参数与聚类中心的欧氏距离，根据所得欧式距离按就近分配的准则将各个参数分配到与其间隔距离最短的聚类中心，共同形成一个聚类；（3）进行聚类中心的更新，针对类中所有的参数进行均值计算，并将其设为新的聚类中心；（4）判断是否满足终止条件，若没有则重复执行（2）和（3）。而终止条件可以为以下三种的任意一种：a. 没

有（或最小数目）参数被执行重新分配给其他不同聚类的操作，即参数聚类的类别不再发生改变；b. 聚类中心不再更新（改变）或有最小数目的聚类中心更新（改变）；c. 得到局部最小的误差平方和。

其中欧式距离的公式为：

$$d(x,y) := \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\cdots+(x_n-y_n)^2} = \sqrt{\sum_{i=0}^{n}(x_i-y_i)^2} \quad (3\text{-}1)$$

更新聚类中心的公式为：

$$Center_k = \frac{1}{\mid C_k \mid}\sum_{x_i \epsilon C_k} x_i \quad (3\text{-}2)$$

误差平方和的计算公式为：

$$S_E = \sum_{i=1}^{r}\sum_{j=1}^{n_i}(X_{ij}-\overline{X_i})^2 \quad (3\text{-}3)$$

K-Means 算法经过反复迭代后，会得到最优的聚类结果，最终可根据其返回的聚类中心（Centroids）和标签（Labels）具体实现聚类量化。在对卷积神经网络的权重参数具体实现聚类量化时，需要预先构造其卷积层和全连接层以添加重要变量，即聚类量化所需要的标签、判断是否被量化过的标志位以及期望的聚成类数，然后根据期望得到的压缩后位宽计算聚集类数，将需要执行聚类量化层的权重参数和类数输入 K-Means 函数处理。详细步骤如下：首先，将要进行聚类量化的权重参数矩阵转化为一维数组，在设定需要分成几类后，使用 K-Means 函数对其进行聚类，函数会返回样本参数的聚类中心和标签（均为一维）。其次，创建标签和位置的映射表，再将一维标签转换为标签矩阵。再次，以标签作为索引，以聚类中心作为权重参数索引值，根据标签枚举每一个中心点的值从而生成聚类量化后的权重参数矩阵。最后，在权重参数更新后，测试模型精度，若较低则可通过重新训练进行微调，从而完成基于聚类量化方法对卷积神经网络进行压缩和加速的目的。

3.1.3 实验结果分析

（1）数据集以及实验模型

本实验选用 MNIST 数据集，实验模型为卷积神经网络模型，由三层卷积层、三层激活函数层、两层池化层和一层全连接层组成。batch_size 设置为 64，经过两轮训练后，模型的 average loss 为 0.0245，准确率为 9923/10000 (99%)。

（2）实验结果分析

对卷积层和全连接层的权重参数进行不同粒度的聚类量化，并测试量化后模型的精度变化，所得实验数据如下：

表 3-1 对 CNN 卷积层和全连接层进行聚类量化后的精度变化

量化粒度	聚类数量	量化后模型精度	Average Loss
2 bit	4	9869/10000（99%）	0.0612
3 bit	8	9887/10000（99%）	0.0498
4 bit	16	9913/10000（99%）	0.0309

由上表数据可以看出，即使将聚类量化的粒度设置很小，其精度也能保持在较高的水准，且不需要微调。因此只想要实现对模型在存储空间上的压缩，聚类量化是非常实用且有效的方法，但由于在推理运算时聚类量化还需要通过索引读取数据，所以不适用于作为模型的加速策略。同时随着量化粒度的提升，可以观察到模型精度也一直在上升，可以得到聚类量化后模型的精度是随着聚类数量的上升而上升的，聚类数量越多，模型的拟合效果越好，性能越接近原始模型。在量化粒度为 2 比特时，模型的精度保持良好，也就是说对实验模型进行 2 比特量化可以最大化模型的压缩效果，同时还能保证模型的精度损失较小，因此应对本模型采用 2 比特聚类量化策略进行模型压缩。

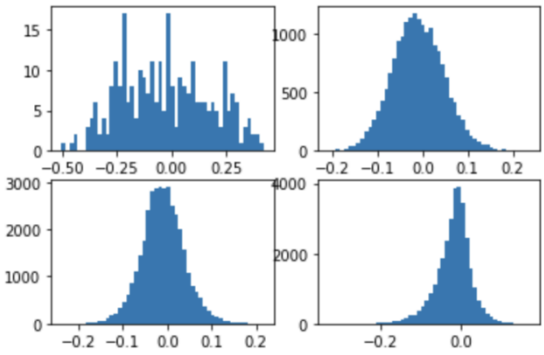

图 3-2 原始模型权重参数分布

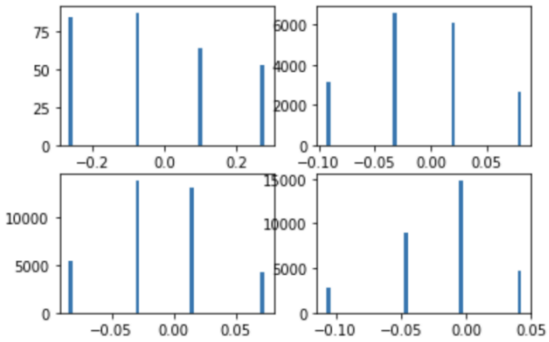

图 3-3 2 bit 聚类量化后的模型权重参数分布

由图 3-1 以及图 3-2 可以看到，在 2 比特聚类量化后，卷积层和全连接层的权重参数被分为了 4 类，均由各自的聚类中心表示，占用的储存空间更小，对模型的参数的简化程度较高。

装订线

3.2 线性量化算法

线性量化是最为普遍成熟的一种量化方法，其 8 比特量化策略[38]就在工业界有着非常成熟的应用，在用整型数据运算代替浮点数据运算后，能在极小的精度损失情况下压缩模型体积以及大幅降低计算成本。本节将对线性量化方法作系统的介绍，并设计算法验证其实际效果，分析量化到不同位宽对于精度的影响，评估线性量化适用的场景并确定最佳的量化比例，避免影响模型性能。

3.2.1 问题分析与方法介绍

由于非线性量化的聚类中心是不是规则均匀分布的，虽然可以在位宽相同的情况下有效降低误差且压缩能力更强，但是由于必须通过索引读取数据进行计算，所以对于模型推理运算的速度优化幅度较小，因此多用于需要压缩模型体积的场景，无法解决提升模型计算速度和降低计算成本的问题。而线性量化可以很好的解决这一问题，经过线性量化后的数据与初始数据之间可以通过基础的线性运算互相转化，并且卷积层和全连接层的数据参与的也是简单的线性运算，因此线性量化后的数据可以直接参与推理运算过程，这就让线性量化可以同时兼顾压缩模型体积与提升模型运算效率及降低计算成本两方面的工作，能够很好的适用于对计算速度和成本有较高要求的模型。

线性量化跟聚类量化的主要区别，就在于其聚类中心是均匀分布的，但本质上二者都是通过建立函数的映射使处理后的数据尽可能的利于存储和计算，量化的实现效果主要就取决于高精度的浮点型数据在映射到低精度的整型数据时信息量丢失的程度。而线性量化的关键之处就在于线性映射关系的建立，据此可以将其分为对称模式和非对称模式，以 8 比特量化为例，对称量化选取参数中的正负最大绝对值作为上下界限对区间[-128，127]进行映射，非对称量化则是根据参数的实际取值范围对区间[0，255]进行映射。可见在对称模式下，对量化范围的利用没有非对称模式精确充分，若是浮点型数据样本分布范围聚集在一侧，另一侧量化范围就被白白浪费了。如在浮点数据样本全为正数或负数的情况下，对称量化实际上等同于失去了一位的表示空间，但对称模式非常利于实现，相较非对称模式少了而外的逻辑成本，因此还要根据实际状况对两种线性量化模式进行选择。总之，线性量化方法由于其发展更为成熟稳定，能兼顾压缩和加速，具备较高的研究和实用价值。

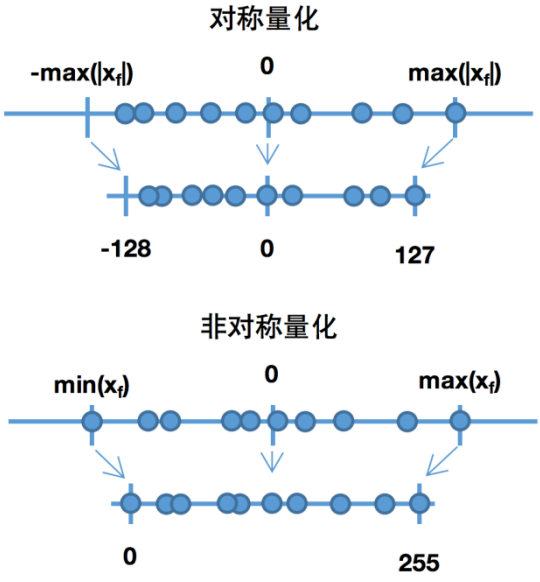

图 3-4 对称量化和非对称量化的示意图

3.2.2 算法原理和实现过程

线性量化算法原理较为简单，主要根据量化后的位宽设定线性变换关系，浮点数值向定点数值转化公式如下：

$$Q = Round(\frac{R}{S} + Z) \tag{3-4}$$

定点向浮点转化（反量化）公式为：

$$R = (Q - Z)S \tag{3-5}$$

式中 *R* 为浮点数据，*Q* 为量化后的整数值，*Z* 为浮点型的偏移量，或零点（最小值）相对应的量化数值，*S* 为缩放因子（缩放系数），Round 是具有四舍五入以及向上或向下近似取整能力的函数。

一般来说，非对称模式的 *S* 求解公式如下：

$$S = \frac{R_{max} - R_{min}}{Q_{max} - Q_{min}} \tag{3-6}$$

也可以表示为：

$$S = \frac{R_{max} - R_{min}}{2^n - 1} \tag{3-7}$$

对称模式的 S 求解公式为：

$$S = \frac{max(|\ R\ |)}{2^{n-1} - 1} \tag{3-8}$$

Z 的求解公式如下：

$$Z = Q_{max} - \frac{R_{max}}{S} \tag{3-9}$$

也可以表示为：

$$Z = Q_{min} - \frac{R_{min}}{S} \tag{3-10}$$

其中 *n* 表示要量化的比特（位宽），对称量化时 *Z* 可简单的取零，非对称量化时 *Z* 可以简单的取浮点数据样本最小值，以此简化计算。在确认要量化的比特后，就可以得到 *Q* 的范围，例如 8 比特量化，*Q* 的取值范围就是[0，255]（非对称量化）或[-128，127]（对称量化），均可表示 256 个定点整数值，而后由已知的 *R* 的范围对 *S* 和 *Z* 进行求解，从而根据参数 *S* 和 *Z* 对 *R* 进行量化以及对 *Q* 进行反量化，最终可以对模型参数实现非对称模式的线性量化。

具体实现时为了高效利用量化范围，算法采用非对称模式进行线性量化，逐层对权重参数进行 8 比特线性量化，每层的权重的缩放因子 *S* 以及偏移量 *Z* 是共用的。在模型训练完毕后进行量化，分为四步：（1）准备数据集，对模型进行训练和测试；（2）改造卷积层和全连接层，添加三个变量，即判断是否量化的

标志位以及缩放因子 S 和偏移量 Z；（3）根据浮点型（32 比特）权重参数的数值范围和期望的量化后位宽（8 比特）确定缩放因子 S 和偏移量 Z；（4）使用量化指标参数对卷积层和全连接层逐层线性量化。

本算法采取非对称模式的线性量化将 32 位的浮点数转换为了不同位宽的定点数，例如 8 比特量化，可将原浮点数据从[R_{min}，R_{max}]的范围区间线性映射到[0，255]的范围区间。最终量化后的模型数据可直接参与卷积层和全连接层的计算过程，并通过反量化可以转换为原始数据的近似值。成功达成了压缩模型体积并提高模型推理运算效率的目的。

3.2.3 实验结果分析

（1）数据集以及实验模型

本实验选用 MNIST 数据集，实验模型为卷积神经网络模型，由三层卷积层、三层激活函数层、两层池化层和一层全连接层组成。batch_size 设置为 64，经过两轮训练后，模型的 average loss 为 0.0245，准确率为 9923/10000 (99%)。

（2）实验结果分析

本实验对模型的卷积层和全连接层的权重参数进行非对称模式下的线性量化，将其量化到不同位并记录其精度变化，实验数据见下表：

表 3-2 对 CNN 卷积层和全连接层进行线性量化后的精度变化

量化粒度	量化后模型精度	Average Loss
2 bit	9731/10000（97%）	0.1004
4 bit	9921/10000（99%）	0.0239
8 bit	9922/10000（99%）	0.0245

由上表数据可以看出，在量化粒度为 4 比特和 8 比特时，模型精度非常高，本模型较为适合选取 4 比特粒度的线性量化进行压缩，其精度与原始模型相差无几。而通过反量化我们可以在推理运算的时候对模型进行加速，同时兼顾了压缩模型体积和降低模型计算成本，因此最好选用 4 比特粒度的线性量化策略对本模型进行压缩和加速。

```
原始数据：
tensor([[0.7200, 0.4660, 0.6461, 0.4650],
        [0.7090, 0.4304, 0.8222, 0.4107],
        [0.2993, 0.8848, 0.1022, 0.8660]])
进行4bit量化并反量化后数据：
tensor([[0.7304, 0.4696, 0.6261, 0.4696],
        [0.7304, 0.4174, 0.7826, 0.4174],
        [0.3130, 0.7826, 0.1043, 0.7826]])
```

图 3-5 反量化后数据与原始数据对比

由上图可以看到，原始数据经过 4 比特量化并接着进行反量化后，其数值较原始数据改变的幅度不大，在模型进行运算推理的过程中，以这部分数据的精度损失为代价，可以显著降低模型的计算成本。

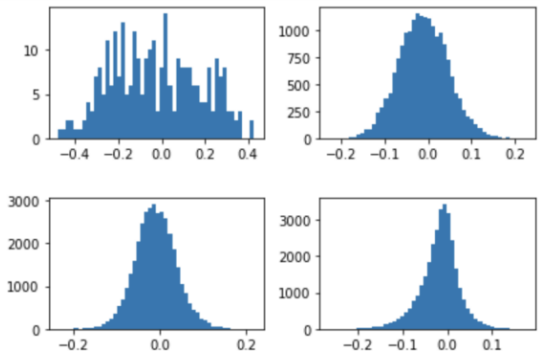

图 3-6 原始模型权重参数分布

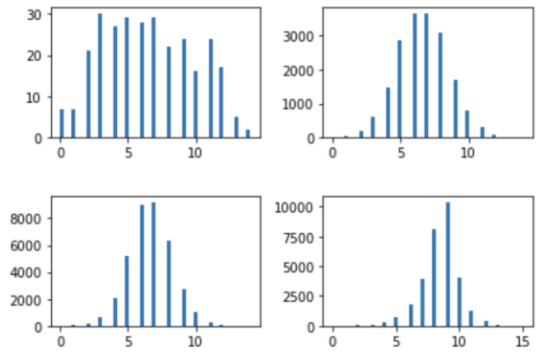

图 3-7 4bit 线性量化后模型的权重参数分布

由图 3-4 以及图 3-5，可以看出各层的权重参数都由 32 位的浮点数量化成为了 4 位的定点数，映射在[0，15]的区间上，非对称模式下的 4 比特线性量化成功完成。

3.3 剪枝-量化融合算法

随着神经网络模型的发展愈发趋向于大规模和复杂化，大量的冗余参数不仅拖慢了模型推理运算的速度还使模型体积便得更加臃肿，导致网络处理难以胜任实时性高的任务，不利于将其部署在资源受限的移动端设备上。而单一的压缩方法一般很难实现大规模模型的压缩和加速优化，所以将较为成熟且实用的模型压缩方法灵活的结合起来处理这一问题是非常有必要的。因此本节主要研究如何融合剪枝和量化两种方法，在保证低精度损失的条件下，达成对规模较大模型进行效果较好的压缩和加速的目的。本节首先是对融合算法可行性做讨论分析，并给出初步的方案，然后阐述算法原理和具体实现细节，最后通过分析实验结果得出结论。

装订线

3.3.1 融合算法可行性分析与方案设计

要将剪枝和量化算法较好的融合在一起，就要考虑两个方法的优势和劣势，尽量使二者在实现过程中扬长避短，从而达到 1 加 1 大于 2 的效果。剪枝算法主要通过对参数和结构的删减对网络模型进行精简，保留重要性高的参数和结构，并舍去贡献度低的参数和结构，起到“取其精华，去其糟粕”的作用。量化算法则是降低网络模型参数的储存位数，在确保精度损失较小的情况下，使参数的位宽尽可能的小，用更简单的参数实现相同的效果，是化繁为简的过程。基于剪枝和量化两种算法的原理和特点，可以发现二者在进行压缩的过程中均不会发生冲突，是从两个角度实现模型压缩的效果的，因此二者有很大的融合空间，在一定程度上可以起到互补的作用。那么决定两种算法作用的先后顺序就显得尤为重要，如果采用先量化后剪枝的方案，会导致剪枝算法无法对量化后的参数进行很好的重要性评估，存在重要性较高的参数或结构被删除的风险，不利于模型精度的保持。基于量化不改变网络模型结构的特点，应当采用先剪枝后量化的方法，剪枝后的网络模型更加精简，被舍去的参数和结构一般来说量化的价值也比较低，变相节约了量化算法的工作量。在完成剪枝后，需要对模型进行重训练微调以弥补其精度损失，然后对恢复精度后的模型进行量化处理。在两次压缩后，模型的精度一般会有较大的损失，这时继续进行重训练微调，若精度难以恢复，则

要选择更保守的剪枝和量化策略，付出降低部分模型压缩程度的代价以换取模型更高的精度，不断调整直到网络精度和压缩效果能实现较好的平衡。

3.3.2 具体实现过程及细节

本次融合算法基于卷积神经网络的经典模型 LeNet_5，具体实现的第一步，就是初始化模型后加载 MNIST 数据集进行训练，待训练完毕后对模型分别进行剪枝和量化。首先对其进行剪枝处理，由于之后还要对模型进行量化，所以应该从整体上把握模型需要被剪枝的参数。这里采用对权重参数的全局非结构化剪枝，这里的剪枝率设定为 90%，只保留 10%的重要权重参数，判断权重参数重要性的指标为 L1 范数，其公式如下：

$$|| x ||_1 = \sum_i | x_i | \tag{3-11}$$

旨在计算权重参数矩阵内各个参数的绝对值之和，以此删除整个网络模型中 L1 范数最低的 90%的权重连接，并且对得到的稀疏网络进行训练微调来恢复精度。然后进行量化处理，为了使网络模型的推理运算过程都是整数和移位运算，就要对权重和偏置以及激活值都进行 8 比特的线性量化，其中激活值的区间范围可以从大量的训练样本中统计得出，并且基于尽量减少精度损失的原则，应采用非对称量化的模式。接着对模型的每一层的参数逐层进行量化，将 32 位浮点数量化为 8 位的定点数，通过在模型计算过程中执行反量化，就可以复原成与其精度近似的值参与运算。量化的精度损失就是来源于上述两个过程，一个是量化过程丢失一部分信息量，另一个是计算过程由于反量化带来输出值的偏差。因此在量化后，需要注意模型精度是否损失严重，视情况对模型进行微调。以上步骤即融合算法的具体实现过程和细节。

3.3.3 实验结果分析

（1）数据集以及实验模型

本实验选用 MNIST 数据集，实验模型为 LeNet_5 网络，是由 LeCun 等人在 1998 年提出的卷积神经网络模型，主要用于手写字体识别任务，一共由 7 层组成，分别为卷积层、池化层和全连接层，是结构较为简单的卷积神经网络。将其网络

的初始学习率设为 0.1 并迭代训练 100 轮后，其 training_loss 为 0.8975，准确率为 7195/10000 (72%)。

（2）实验结果分析

首先，实验的剪枝率设置为 90%，针对其卷积层和全连接层的权重参数进行了非结构化的全局剪枝，这里的参数重要性指标为 L1 范数。全局剪枝后的各层的稀疏度见下表：

表 3-3 LeNet_5 网络全局剪枝 90%后各层的权重稀疏度

Layer	Sparsity
Conv1	21.33%
Conv2	61.62%
Conv3	98.67%
Fc1	60.48%
Fc2	42.02%

被剪枝的权重参数主要分布在 conv1 和 fc2，即这两层对模型最终结果的贡献度较低。对剪枝后的模型重新迭代训练 100 轮后，其 training_loss 为 0.7407，准确率为 7880/10000 (79%)，精度较剪枝前有明显提高，说明网络参数存在较大冗余。

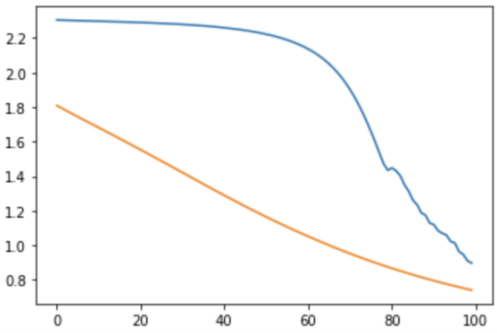

图 3-8 原始模型和剪枝后模型迭代训练过程的 training_loss 变化

上图蓝线为原始网络的 training_loss 变化，橙线为剪枝 90%后网络的 training_loss 变化。可见在剪枝后，网络的 training_loss 下降更加平滑，且 training_loss 比原始网络模型小，网络在参数量变少的情况下的整体学习效果和学习效率更好。

在重训练调整了网络精度后，对其进行了非对称模式下的 8 比特线性量化，经过剪枝-量化算法压缩加速后的模型准确率为 7887/10000 (79%)，对比原始网络的模型精度有所提升，融合算法较好的完成了对模型的压缩和加速。

3.4 本章小结

本章主要围绕量化算法展开，分别介绍了聚类量化和线性量化两种方向，详细阐述了二者的算法原理以及实现步骤和过程，并设计算法在卷积神经网络上进行了验证，通过对实验结果的分析客观评价了二者的实际效果。还将量化和剪枝两种方法结合在了一起，在分析了融合算法的可行性后规划设计了具体方案，陈述了具体实现的步骤和细节，最后在 LeNet_5 网络上初步验证了融合算法的效果。本章不仅对量化方法的两个研究方向进行了讨论和实践，还在此基础上融合第二章的剪枝方法对网络进行进一步的压缩和加速，实现了对量化算法较为综合全面的研究。

第四章 基于知识蒸馏的压缩算法

知识蒸馏是一种十分简单高效的模型压缩与加速方法，在学界和工业界存在普遍的研究应用，是非常有代表性和研究价值的模型压缩算法。知识蒸馏的核心思想主要是通过高性能的大规模网络模型去“教”结构较为简单的小型网络模型，使其突破自身架构的限制从而获得更加优秀的性能，实现更好的学习效果。本章主要对知识蒸馏方法的原理和框架进行详实的介绍，并设计算法在卷积神经网络上进行模型压缩实验，最后通过对比实验结果分析其优化效果以及实现时需要考虑的问题，提出个人的想法和建议。

4.1 问题分析与方法介绍

深度学习模型压缩现有的主流方法，例如剪枝和量化等算法，他们的共通点在于都是在训练好的现有网络模型上对其参数和结构进行调整和处理，这就导致了如果网络模型本身的架构存在一些问题和弊端，仅靠剪枝和量化等算法是无法解决整体架构层面上的不合理的。因此出现了从网络模型的设计角度解决问题的压缩方法，如紧凑网络设计[46-48]方法，通过设计紧凑而精简的网络变相解决模型压缩和加速的问题。但基于新的网络模型的设计难度系数较高，需要耗费较大时间和人力成本，还有可能面临新的挑战和阻碍，且要解决在现有硬件和深度学习架构环境下的部署问题，在很多情况下不是一个好的选择。知识蒸馏算法的出现较好的解决了这个问题，使我们可以付出较小的代价得到性能出色的小规模网络模型，找到了实现模型压缩和加速的一条捷径。

知识蒸馏算法的思想是来源于教师传授知识给学生的行为，所以也被称为知识迁移[23]，在教师-学生这一架构下，教师模型一般选用复杂程度高且性能强大的大规模网络模型，而学生模型则均为较精简的小型网络模型。知识蒸馏的核心任务就是提取教师模型里较为完备和准确的数据和信息作为知识，以供学生模型在训练的过程中参考和学习，最终获得更好的表达能力和更高的精度。因此如何有效从教师模型中提取学生模型需要的知识是成为了关键的问题，通常我们同时从两种渠道获取学生模型需要的知识，分别是将教师模型的 SoftMax 分类器的输出

蒸馏出来作为软知识，同时将样本的真实标签信息作为硬知识，将二者同时提供给学生模型进行学习，由此更好的发挥了对学生模型的指导作用。经过训练学习后的学生模型基本上具有了独当一面的能力，能在节约了大量储存空间和推理运算成本的情况下，弥补与大规模网络模型之间的性能差距，拥有了与教师模型较为近似的精度和性能表现。

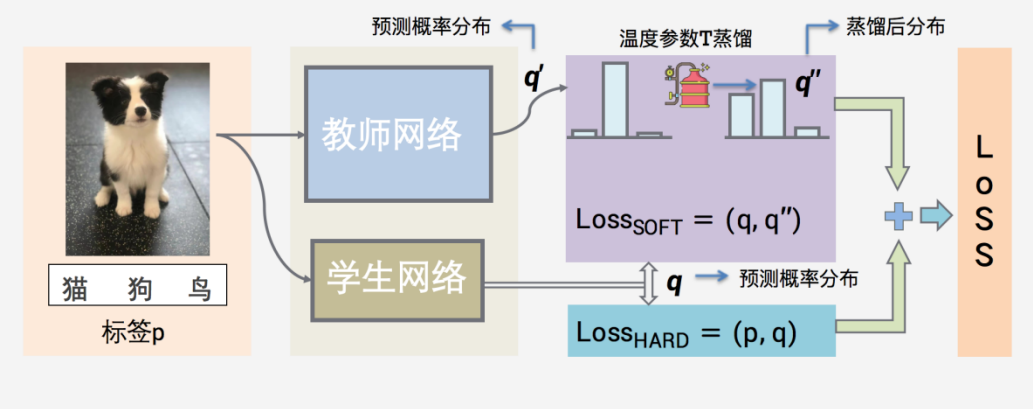

图 4-1 知识蒸馏算法原理图

4.2 算法原理和实现过程

知识蒸馏算法本质上是通过重新定义学生模型的损失函数（Loss function）来对其进行优化训练，使其能够依靠学习教师模型的知识提高自身精度和性能的。损失函数是衡量模型的预测值与实际真实值之间的差距程度的工具，本算法使用的是交叉熵损失函数（Cross Entropy Loss Function），其非常适合应用在神经网络处理分类问题时的场景，其公式可表示为：

$$L = \frac{1}{N}\sum_{i} L_i = -\frac{1}{N}\sum_{i}\sum_{c=1}^{M} y_{ic} log(p_{ic}) \tag{4-1}$$

其中 M 表示标签（类别）总数，y_{ic}为符号函数（取 0 和 1），若样本 i 的真实标签（类别）为 c 则取 1，反之取 0，p_{ic}表示预测训练样本 i 的标签（类别）为 c 的概率。在正常训练神经网络模型时，真实标签是由 one-hot 编码的，可以通过这个式子计算得出损失函数值的大小来判断模型预测数据和真实数据的差距有多大。但是在知识蒸馏算法中，损失函数主要有两部分组成，一个是用硬知识（hard target）算得的 student loss，由学生模型 SoftMax 分类器输出的预测值和 one-hot 编码的真实标签即实际值通过交叉熵损失函数公式计算得出，并用L_{hard}来表示。

另一个是用软知识（soft target）算得的 distill loss，这部分需要计算的是学生模型与教师模型各自预测输出的类别概率之间的交叉熵，用L_{soft}来表示。通常情况下，将向量z作为卷积神经网络模型最后一层即全连接层的输出，那么z_i就表示第 i 类输出的预测值，则模型预测输入的数据样本类别属于 i 类的概率p_i可以用 SoftMax 函数计算，其公式可表示为：

$$p_i = \frac{exp(z_i)}{\sum_j exp(z_j)} \tag{4-2}$$

其输出的是一个较为接近 one-hot 编码形式的概率分布，例如[0，1 ，0， 0]（one-hot）和[0.01，0.92，0.05，0.02]（SoftMax）。在神经网络模型处理普通的分类问题时，用模型的 SoftMax 分类器的输出和 one-hot 形式的真实标签输出（硬知识）算出的交叉熵损失就是硬知识，用L_{hard}表示。但是 one-hot 形式的标签信息量（信息熵）非常低，不利于整体知识的体现，难以表征各类信息之间的关系，无法提高学生模型的泛化能力。因此知识蒸馏算法引入了一个温度参数 T 来对 SoftMax 函数的输出进行“软化”，带有温度参数 T 的公式可表示为：

$$p_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{4-3}$$

这样就可以获得更加平滑的输出，与 one-hot 编码的标签（0-1 结构）不同，各个类别的概率差距没有那么大，T 的数值设置的越大，其分布就更均匀缓和。这种方法就可以更好的提取教师网络的暗知识。例如手写字体识别时，手写数字 5 被误判归类为 6 的概率非常低，但是这种概率仍然比将 5 错判为 1 的概率高得多，其中就蕴含了 1 和 6 的图像相比，6 的图像与 5 的图像更接近的信息，这种信息就可被称为暗知识，能很大程度上体现模型的泛化能力。将教师模型这部分软知识“蒸馏”出来迁移到学生模型中，能传递更大的信息量从而更好的指导其训练学习。因此需要用带温度参数 T 的 SoftMax 函数对教师模型和学生模型的输出进行软化，使这二者软化后的 SoftMax 分类器的输出趋向一致，即最小化两个输出的交叉熵损失。而后将得到的L_{soft}与L_{hard}这两部分结合起来作为学生模型的总体损失函数，并用α和β作为两部分的权重系数，学生模型的总体损失函数可

加权表示为：

$$L = \alpha L_{soft} + \beta L_{hard} \tag{4-4}$$

$$L_{soft} = -\sum_{j}^{N} p_j^T log(q_j^T) \tag{4-5}$$

$$L_{hard} = -\sum_{j}^{N} c_j log(q_j) \tag{4-6}$$

其中p_j^T和q_j^T分别是教师模型和学生模型的温度参数为*T*的SoftMax分类器输出在第*j*类的值，c_j为one-hot形式标注的类别信息，其中在真实类别取1，其他类别全部取0，表示在第*j*个类别的真值（Ground-truth），q_j为正常训练神经网络模型时不带温度参数T的SoftMax分类器输出在第j类的值，*N*为总类别数量。以上就是知识蒸馏算法的原理部分，整个算法的核心主要是对学生网络的损失函数的重新定义，加入了从教师网络蒸馏出来的知识，通过对温度参数*T*的调节控制学生网络训练学习的方向。*T*设置的较高（大于1）则学生网络学到更多的暗知识，获得的信息量更大，有效提升了其泛化能力，*T*设置的较小，则学生网络受噪声的干扰更小，虽然获得的信息量较少，但其拟合能力会变得更好。在具体实现的时候一般根据学生网络的体积（参数量）来决定温度参数*T*的大小，若是非常精简的学生模型，那么*T*的取值就不宜过大，因为参数量小的网络难以消化太大的信息量，可以适当的减少一些，效果更佳。

具体的实现过程分为四步：（1）使用正常的标签训练教师模型；（2）设定温度参数*T*，计算教师模型经过带温度参数*T*的SoftMax分类器输出的值，将其作为蒸馏出来的软知识；（3）在训练学生模型前，首先计算正常的标签（由one-hot形式编码）与不带温度参数*T*的SoftMax分类器的输出的交叉熵损失，作为由硬知识算得的student loss，用L_{soft}表示。然后计算学生模型在相同温度参数*T*下的SoftMax分类器的输出与教师模型蒸馏出来的软知识的交叉熵损失，作为由软知识算得的distill loss，用L_{hard}表示。并设置权重系数α和β的大小（二者的和为1），通过权重系数来确定总体损失函数两部分的比例；（4）基于总体损失函数训练学生模型，在这个过程中学生模型对教师模型的知识进行学习，不断提高其

网络性能和精度；（5）训练完毕后的学生模型此后均按正常卷积神经网络的常规方式处理任务，对训练后的学生模型进行测试，并与没有使用知识蒸馏算法进行优化，经过正常训练后的模型进行对比。

4.3 实验结果分析

（1）数据集以及实验模型

本实验选用 MNIST 数据集，教师模型是由两层卷积层、两层全连接层和两层 Dropout 层组成的卷积神经网络模型，batch_size 设置为 64，在迭代训练 10 轮后，教师模型的 average loss 为 0.03192，准确率为 9905/10000 (99%)。学生模型是由三个全连接层组成的神经网络模型，batch_size 设置为 64，在迭代训练 10 轮后，学生模型的 average loss 为 0.1133 准确率为 9798/10000 (98%)。

（2）实验结果分析

对于手写数字 9，教师网络使用正常 SoftMax 函数预测输出的概率分布与使用带温度参数 T（T=10）的 SoftMax 函数预测输出的概率分布之间的差别如下图：

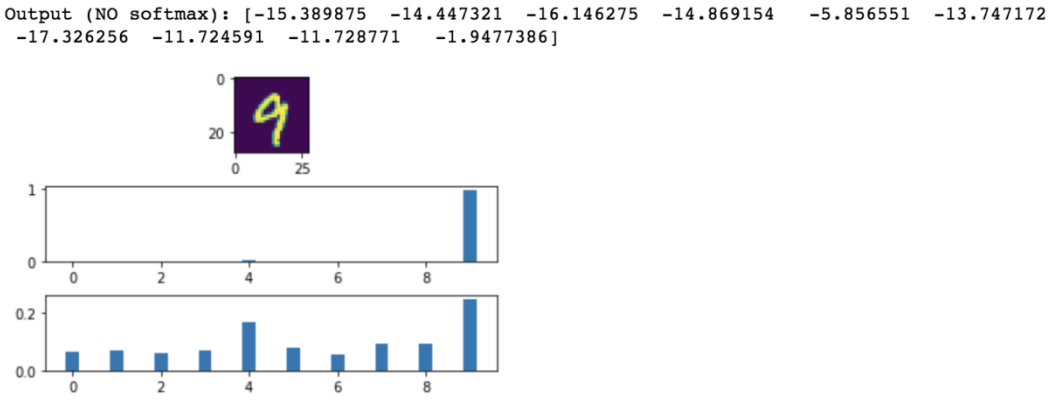

图 4-2 温度参数 T 对教师模型预测输出的影响

可以看到正常的 SoftMax 概率分布非常陡峭，预测数字为 9 的概率最大，另外有微小的概率为 4，其余概率均极接近 0。而带温度参数 T（T=10）的 SoftMax 概率分布更加均匀平滑，且携带的信息量更大，我们可以发现 9 的概率最大，但 4 和 7、8 的概率也不低，这说明 4、7、8 和 9 之间相似程度较高，各类之间的关系也得以体现。因此给 SoftMax 函数加上温度参数 T，对教师模型的知识进行蒸馏可以得到这些暗知识，更有助于学生模型在训练学习的过程中提高网络的泛化

能力，获得更优秀的性能。

用知识蒸馏算法对学生模型进行优化训练后，学生网络的 average loss 为 0.0505，准确率为 9844/10000 (98%)，相比未用知识蒸馏算法优化的学生模型精度更高，与教师模型相比精度损失也非常低。因此知识蒸馏算法能通过大规模模型有效的训练高性能的小规模模型，使其获得与大规模模型近似的精度和性能，变相实现了对模型的压缩和加速。整个学习过程中三个模型的 loss 和准确率变化如下图：

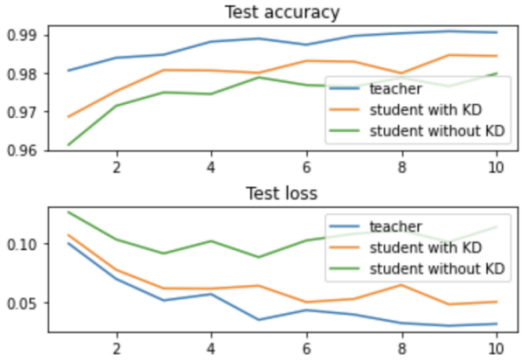

图 4-3 三种模型迭代过程的 loss 和准确率变化折线图

可以看出用知识蒸馏算法优化后学生模型在整个迭代训练的过程中，其精度和性能较正常训练的学生模型有明显优势，一直保持着接近教师模型的精度和性能，很好的证明了知识蒸馏算法的有效性和实用性。

4.4 本章小结

本章主要对知识蒸馏算法进行研究，首先，分析了部分模型压缩算法都在面临的问题，并系统性的介绍了知识蒸馏算法作为解决方案，对知识蒸馏算法的整体思想做了概述。其次，详细阐述了知识蒸馏算法的原理以及具体实现步骤和过程，对带温度参数 T 的 SoftMax 分类器和教师网络中的暗知识进行了较为细致的探讨。最后，以对比两种 SoftMax 分类器对教师网络暗知识的不同影响为铺垫，在卷积神经网络上验证了知识蒸馏算法，并通过对比实验，充分体现出知识蒸馏算法的优越性和实用性，有力证明了算法的实际效果。

第五章 总结与展望

5.1 论文总结

在深度学习模型体积愈发庞大，复杂度逐年升高的大环境下，将这些模型部署在资源相当受限的移动端便携设备上成为了亟需应对的挑战。因此研究深度学习模型压缩算法不仅具有重要的科学意义，还能与实际应用紧密结合，帮助深度学习模型部署在各类设备上，在工业界的实际应用中具有现实意义。本文对现有的深度学习模型压缩算法进行了综合研究，简述了深度学习模型压缩的研究背景及意义，对国内外研究现状进行了总结。并且对其中四个主流研究方向即剪枝、量化、知识蒸馏和低秩分解进行了系统性概括，灵活运用了剪枝、量化和知识蒸馏三种算法实现对深度神经网络模型的压缩和加速。论文所做工作可以归纳为以下几部分：

（1）深入研究了剪枝算法，对剪枝的两个方向即结构化剪枝与非结构化剪枝分别进行了介绍，并针对实际问题设计了具体剪枝策略，在 MLP 上进行了剪枝粒度为权重的非结构化剪枝实验，并且在卷积神经网络上进行了剪枝粒度为卷积核粒度的结构化剪枝实验。两次剪枝都有效的除去了网络模型中大量的冗余参数，并且均能保证网络的精度损失在极小的范围内。

（2）深入研究了量化算法，详实的论述了聚类量化和线性量化的原理和适用场景，并针对实际问题设计了不同的量化方案，对所用算法的原理进行了细致的分析，阐述了具体的实现过程，而后在卷积神经网络上将参数量化到了不同的位宽，并且兼顾了模型的精度和压缩的效果。还将剪枝和量化两种算法较好的融合了起来，设计了融合算法的压缩策略，并使用剪枝-融合算法对 LeNet_5 网络进行了压缩和加速，取得了非常不错的效果，从而实现了对剪枝和量化算法的结合应用。

（3）深入研究了知识蒸馏算法，完整叙述了知识蒸馏方法的思想和框架，对知识蒸馏算法中温度参数 T 与教师模型的暗知识之间的联系进行了深入分析。通过重新定义学生网络的损失函数，使其能结合软知识和硬知识进行学习训练，

还详细阐述了基于知识蒸馏思想所设计算法的原理和具体实现过程，并在卷积神经网络模型上进行了对比实验。使用知识蒸馏算法优化训练后的学生模型精度和性能有明显的提高，由此验证了知识蒸馏算法的实用性和优越性。

总而言之，本文概述了模型压缩领域的背景知识和研究现状，并对其中三个研究方向进行了深入的研究，基于不同的模型压缩思路均制定了模型压缩策略以及设计了可行的算法。而后在深度学习框架平台 PyTorch 下，搭建了深度神经网络模型且对每种策略和算法都进行了实践验证，通过对实验数据的分析，评估了每种算法的精度损失和压缩效果，证明了其可行性以及有效性。

5.2 未来工作展望

本文主要研究的三种方法即剪枝、量化和知识蒸馏，虽然都经过了实验并展现了一定的成效，但是仍然是比较基础简单的方法，存在很多的限制和弊端，需要在未来进一步的改进，包括以下方面：

（1）非结构化剪枝会生成权重参数构成的稀疏矩阵，目前的硬件和深度学习框架不能支持对其的优化和加速，所以应当将更多的目光聚焦在结构化剪枝上。本文提出的结构化剪枝策略主要是基于 L2 范数的，但在实际应用中，单一化的参数重要性评估标准不利于剪枝的进行，参数对模型结果的影响不止体现在其数值上，很可能还会跟其所处位置有关，应当设立更严谨和多维的剪枝标准，才能进一步深化剪枝的程度，提升对模型的压缩能力，并尽可能避免精度的损失。

（2）剪枝-量化融合算法也是应用了非结构化剪枝，局限性非常大，应当将量化和结构化剪枝结合在一起，使融合算法更具普适性。而且融合算法主要是对网络参数和结构进行简化和修整，会存在精度损失较大的可能性，可以考虑用知识蒸馏算法优化其训练学习的过程，使压缩后的网络模型在经过训练学习调整后，其精度和泛化能力能显著提高。

（3）知识蒸馏算法只考虑了如何有效提取教师网络的知识供学生网络学习，没有考虑到教师网络和学生网络的整体结构对算法实现的影响，应进一步深化研究，探究网络结构对知识蒸馏算法的影响，选取框架更适配的教师网络和学生网络，进一步提升学生网络的学习效果，更好的实现对模型的压缩和对精度的保障。

并且由于实验设备受限，无法在硬件上实际测试模型优化后推理运算速度的变化，只能通过观察模型结构和参数量的改变，理论推断出是否能提高模型的推理运算速度，这是本文实验上的不足之处。在以后的工作中应当设计更严谨的实验，多方位评估模型压缩算法的效果，以及通过对比不同算法压缩和加速模型的幅度，对这些算法进行择优和改进。

致　谢

时光荏苒，四年的大学本科生活即将迎来终点，四年前初次走进校园的景象好似就在昨天。在毕业论文即将完成之际，向所有支持和帮助我的家人、老师、好友、同学表示诚挚的感谢。

首先感谢的是我的指导老师修贤超老师，上海的疫情给老师们和同学们的工作都带来了许多的不便，因此在进行毕业设计的初期，内心还是有很多担忧和迷茫的，在此衷心感谢老师对我的耐心和负责，是您让我能够笃定且踏实的逐步完成毕业论文。

其次感谢我的老师和同学们，老师执教的认真和负责，使我这四年受益匪浅，在疫情期间同学们相互的互助和鼓励使我保持了良好的心态，在四年的相处中，我们都给彼此留下了非常美好的回忆。同时也感谢我的学校，提供给我广阔的平台，和优质的资源，为我的学业生活提供了良好的环境。

最后感谢我的家人，是他们的爱与鼓励支持着我，翻过一座座山，看见更广阔的风景，希望他们永远健康快乐。

装订线

参考文献

[1] Ji R, Wen L, Zhang L, et al. Attention Convolutional Binary Neural Tree for Fine-Grained Visual Categorization[C]// 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020.

[2] Oza P, Patel V M. One-Class Convolutional Neural Network[J]. IEEE Signal Processing Letters, 2019, 26(2):277-281.

[3] Li X, Guivant J E, Kwok N, et al. 3D Backbone Network for 3D Object Detection[J]. 2019.

[4] Hinton G, Deng L, Yu D, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition[J]. IEEE Signal Processing Magazine, 2012, 29(6):82-97.

[5] Graves A, Mohamed A R, Hinton G. Speech Recognition with Deep Recurrent Neural Networks[J]. Acoustics, Speech, and Signal Processing, 1988. ICASSP-88. 1988 International Conference on, 2013, 38.

[6] Xiong W, Droppo J, Huang X, et al. Achieving Human Parity in Conversational Speech Recognition[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2016, PP(99).

[7] Klein G, Kim Y, Deng Y, et al. OpenNMT: Open-Source Toolkit for Neural Machine Translation[J]. 2017.

[8] Lan Z, Chen M, Goodman S, et al. ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations[J]. 2019.

[9] Li P, Zhao H, Liu P, et al. RTM3D: Real-Time Monocular 3D Detection from Object Keypoints for Autonomous Driving[J]. European Conference on Computer Vision, 2020.

[10] Ancha S, Raaj Y, Hu P, et al. Active Perception Using Light Curtains for Autonomous Driving[M]. 2020.

[11] Lecun Y, Bengio Y, Hinton G. Deep Learning[J]. Nature, 2015, 521(7553):436-444.

[12] Mcculloch W S, Pitts W. A Logical Calculus of The Ideas Immanent in Nervous

Activity[J]. The Bulletin of Mathematical Biophysics, 1943, 5(4):115-133.

[13] DE Rumelhart, Hinton G E, Williams R J. Learning Representations by Back Propagating Errors[J]. Nature, 1986, 323(6088):533-536.

[14] Lecun Y. Generalization and Network Design Strategies[C]// Connectionism in Perspective. Elsevier, 1989.

[15] Krizhevsky A, Sutskever I, Hinton G. ImageNet Classification with Deep Convolutional Neural Networks[J]. Advances in Neural Information Processing Systems, 2012, 25(2).

[16] Lecun Y, Bottou L. Gradient-based Learning Applied to Document Recognition[J]. Proceedings of The IEEE, 1998, 86(11):2278-2324.

[17] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.

[18] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[J]. IEEE, 2016.

[19] Huang G, Liu Z, Laurens V, et al. Densely Connected Convolutional Networks[J]. IEEE Computer Society, 2016.

[20] Kamei K, Nishio S, Hagita N, et al. Cloud Networked Robotics[J]. IEEE Network, 2012, 26(3):28-34.

[21] Mohanarajah, Gajamohan, Hunziker, et al. Rapyuta: A Cloud Robotics Platform.[J]. IEEE Transactions on Automation Science & Engineering, 2015.

[22] Cheng Y, D Wang, Zhou P, et al. A Survey of Model Compression and Acceleration for Deep Neural Networks[J]. 2017.

[23] 耿丽丽，牛保宁. 深度神经网络模型压缩综述[J]. 计算机科学与探索，2020, 14(9):1441-1455.

[24] 曾焕强，胡浩麟，林向伟等. 深度神经网络压缩与加速综述[J]. 信号处理，2022, 38(01):183-194.

[25] 高晗，田育龙，许封元等. 深度学习模型压缩与加速综述[J]. 软件学报，2021, 32(01):68-92.

装订线

[26] 周逸伦. 深度学习模型量化及相关压缩技术研究[D]. 上海交通大学, 2020.

[27] Choudhary T, Mishra V, Goswami A, et al. A Comprehensive Survey on Model Compression and Acceleration[J]. Artificial Intelligence Review, 2020, 53(3).

[28] Song H, Mao H, Dally W J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding[C]// ICLR. 2016.

[29] Guo Y, Yao A, Chen Y. Dynamic Network Surgery for Efficient DNNs[J]. 2016.

[30] Aghasi A, Abdi A, Nguyen N, et al. Net-Trim: Convex Pruning of Deep Neural Networks with Performance Guarantee[J]. 2016.

[31] Zhang X, He Y, Jian S. Channel Pruning for Accelerating Very Deep Neural Networks[C]// IEEE International Conference on Computer Vision. IEEE Computer Society, 2017.

[32] Zhuang L, Li J , Shen Z, et al. Learning Efficient Convolutional Networks through Network Slimming[J]. IEEE, 2017.

[33] Li H, Kadav A, Durdanovic I, et al. Pruning Filters for Efficient ConvNets[J]. 2016.

[34] Vanhoucke V, Senior A, Mao M Z. Improving The Speed of Neural Networks on CPUs[C]// Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011. 2011.

[35] Rastegari M, Ordonez V, Redmon J, et al. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks[C]// European Conference on Computer Vision. Springer, Cham, 2016.

[36] Courbariaux M, Bengio Y, David J P. BinaryConnect: Training Deep Neural Networks with Binary Weights During Propagations[J]. MIT Press, 2015.

[37] Courbariaux M, Bengio Y. BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1[J]. 2016.

[38] Jacob B, Kligys S, Chen B, et al. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference[J]. 2017.

[39] Gong Y, Liu L, Ming Y, et al. Compressing Deep Convolutional Networks Using Vector Quantization[J]. Computer Science, 2014.

装订线

[40] Wu J, Cong L, Wang Y, et al. Quantized Convolutional Neural Networks for Mobile Devices[J]. IEEE, 2016.

[41] Bucila C, Caruana R, Niculescu-Mizil A. Model Compression[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'06). Computer Science Cornell University, 2006.

[42] Hinton G, Vinyals O, Dean J. Distilling The Knowledge in A Neural Network[J]. Computer Science, 2015, 14(7):38-39.

[43] Romero A, Ballas N, Kahou S E, et al. FitNets: Hints for Thin Deep Nets[J]. Computer ence, 2015.

[44] Ba L J, Caruana R. Do Deep Nets Really Need to be Deep?[J]. Advances in Neural Information Processing Systems, 2014, 3:2654-2662.

[45] Zagoruyko S, Komodakis N. Paying More Attention to Attention: Improving The Performance of Convolutional Neural Networks via Attention Transfer[J]. 2016.

[46] Iandola F N. Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <0.5MB Model Size[J]. 2016.

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

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

[49] Brock A, Lim T, Ritchie J M, et al. SMASH: One-Shot Model Architecture Search through HyperNetworks[J]. 2017.

[50] Wu B, Keutzer K, Dai X, et al. FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search[C]// 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2019.

[51] Liu C, Zoph B, Neumann M, et al. Progressive Neural Architecture Search[J]. 2017.

[52] Jaderberg M, Vedaldi A, Zisserman A. Speeding up Convolutional Neural Networks with Low Rank Expansions[J]. Computer ence, 2014, 4(4):XIII.

[53] Wen W, Xu C, Wu C, et al. Coordinating Filters for Faster Deep Neural

Networks[C]// 2017 IEEE International Conference on Computer Vision (ICCV). IEEE, 2017.

[54] Lebedev V, Ganin Y, Rakhuba M, et al. Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition[J]. Computer Science, 2014.

[55] Larsen J, Hansen L K, Svarer C. Regularization of Neural Networks Using DropConnect. 2001.

[56] Hinton G E, Srivastava N, Krizhevsky A, et al. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors[J]. Computer Science, 2012, 3(4): 212-223.

[57] Macqueen J. Some Methods for Classification and Analysis of MultiVariate Observations[C]// Proc of Berkeley Symposium on Mathematical Statistics & Probability. 1965.