# 第一章 绪论

## 1.1 研究背景

### 1.1.1 神经网络与深度学习

神经网络（Neural networks）是由具有适应性的简单单元组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对正式世界物体所作出的交互反应。神经网络中最基本的成分是神经元（neuron）模型，每个神经元与其他神经元相连，当它“兴奋”时，就会向相连的神经元发送化学物质，从而改变这些神经元内的电位。在神经网络模型中，神经元接收到来自n个其他神经元传递过来的输入信号，这些输入信号通过带权重的连接（connection）进行传递，神经元接收到的总输入值将与神经元的阈值（threshold）进行比较，然后通过激活函数（activation function）处理以产生神经元的输出。

随着云计算、大数据时代的到来，以深度学习（Deep Learning）为代表的复杂模型开始受到人们的关注。深度学习模型就是深层神经网络，通过增加隐藏层的层数和神经元的数目来获得学习能力更强大的模型。对于神经网络而言，参数越多则模型的复杂度越高，所占用的存储空间即容量（capacity）就越大，意味着其能完成更复杂的学习任务。目前已有多种深度学习算法模型，如卷积神经网络（Convolutional Neural Network，CNN）、深度置信网络（Depp Belief Network，DBN）、循环神经网络（Recursive Neural Network）。

CNN主要应用于计算机视觉领域，它由一个或多个卷积层（Convolutional Layer）组成，通常其后会跟着一个或多个全连接层（Fully Connected Layer），最后由sigmoid函数或者SoftMax单元输出分类。相比于拥有相同数量神经元的网络而言，CNN采取了权值共享（weight sharing）的策略，即让一组神经元使用相同的连接权，从而大幅减少了需要训练的参数数量。

### 1.1.2 深度学习与边缘计算

全球智能手机的快速发展，推动了移动终端和边缘计算（Edge Computing）的发展。而万物互联、万物感知的智能社会，则与物联网相伴而生，边缘计算系统也应声而出。边缘计算是指在靠近物或数据源头的一侧，采用网络、计算、存储、应用核心能力为一体的开放平台，就近提供最近端服务。其应用程序在边缘侧发起，产生更快的网络服务响应，满足行业在实时业务、应用智能、安全与隐私保护等方面的基本需求。边缘计算处于物理实体和工业连接之间，或处于物理实体的顶端。而云端计算，仍然可以访问边缘计算的历史数据。

深度神经网络已经演化成为了应用于计算机视觉任务中的先进技术。虽然这些神经网络具有非常强大的功能，但是其大量的参数消耗了相当大的存储空间和传输带宽。例如， AlexNet Caffe模型占用了200MB的空间，VGG-16 Caffe模型超过了500MB。因此，将这些深度神经网络部署在移动设备上是非常困难的。

对于主要开发移动设备软件的公司（如百度和Facebook），通过不同的APP商店下载的各种APP对于可执行二进制文件的大小非常敏感。例如，APP Store限制“apps above 100MB will not download until you connect to Wi-Fi”。尽管将深度神经网络运行在移动设备端拥有很多优势，如提供更好的隐私保护和实时处理、占据更少的网络带宽，然而，在巨大的存储开销面前，深度神经网络难以被部署到移动应用中。

### 1.1.3 深度学习与模型加速

运行一个大型神经网络需要将众多权值参数载入内存，还需要进行大量的矩阵乘积运算，因此需要大量的存储器带宽和强大的计算能力作为支持，同时也会消耗相当大的能源。这使得携带有深度神经网络的大功率应用难以部署。大型网络不适合片上存储，因此需要更昂贵的DRAM访问。例如以20fps运行1个10亿连接的神经网络将需要（20Hz）（1G）（640pJ）=12.8W仅用于DRAM访问，这远远超出了典型移动设备的功率包络。

在移动应用中，我们通常假定在服务器端执行训练（training），然后在移动设备端执行推断（inference）。但是，相对于服务器而言，移动设备在计算能力、内存容量和电池功耗方面有着极其严格的约束。因此，使深度神经网络能够适应移动设备的有限资源（如存储空间、电池功耗等）是很有必要的。

在探索各种各样的CNN网络结构中，ImageNet challenge已经获得了重大的进步。近几年的大致趋势是网络变得越来越深，从而导致参数的数量和卷积操作的次数大幅上涨。这种容量极大的网络在某种意义上有着相当大的推理成本，尤其是在嵌入式的传感器元件和移动设备上（Hao Li et al.(2017)）。对于这些资源有限的设备而言，除了模型的精确度之外，计算的高效性和网络的小体积可能是更关键的因素。另外，对于提供图像搜索和图像分类的Web服务而言，API通常每秒要提供成百上千张图像的服务，因此降低推理时间是非常重要的。

## 1.2 研究现状

### 1.2.1 深度学习的研究现状

目前，深度学习架构已应用于包括计算机视觉、语音识别、自然语言处理、音频识别、机器翻译等领域。通过使用深度学习架构，某些领域的效果已经达到了与人类工作相媲美或更加优越的效果。

深度学习本质上是一种复杂的，具有多层级的神经网络。其基本理论在上世纪八九十年代便已被Yuan Lecun等人提出。但在当时，由于大规模神经网络的训练极其耗时，相关硬件水平远远不能满足训练深度神经网络的巨大计算量，所以人工神经网络的发展非常缓慢，仅仅处在科研阶段。

1989年，Yann LeCun等人应用了标准的反向传播算法。后来他又提出了卷积神经网络多层结构学习算法。它利用空间相对关系的特性减少参数数目并提高了分类效果。

2006年，深度学习的概念由Hinton等人提出，他们的研究显示了如何有效地训练一层或多层前馈神经网络，依次将每层处理为一个无监督的限制玻尔兹曼器，然后使用监督反向传播对其参数进行微调，另外他们也提到了深度信仰网络的学习。

Yann LeCun等人在2015年指出：深度学习能够由多个处理层组成的计算模型来学习具有多个抽象级别的数据的特征。深度学习通过使用反向传播算法改变算法的各个参数来提取大型数据集中的特征。深度卷积神经网络在处理图像、视频和音频方面取得了突破，而且循环网络对连续性数据如文本和语音的处理来说打开了新的大门。

近年来，深度学习等人工智能算法进入了人们的视野。例如阿尔法狗(AlphaGo)是第一个击败人类职业围棋选手、第一个战胜围棋世界冠军的基于深度学习的人工智能系统。

### 1.2.2 模型压缩的研究现状

现有的很多神经网络模型具有大量的参数，而这些参数中往往存在大量的冗余，这也导致了存储空间和计算时间的浪费。参数压缩方法旨在消除这些冗余的部分，包括信息上的冗余以及空间上的冗余。

早期的方法是网络剪枝，包括有偏权重衰减，OBD和OBS。参数剪枝主要通过设计判断参数重要与否的准则，移除冗余的参数。有偏权重衰减是基于幅度的剪枝，OBD和OBS都是基于损失函数的Hessian矩阵来进行的。OBS优于基于幅度的方法和OBD，主要体现在：1）基于幅度的方法删除具有最小幅度的权重，OBD中假设Hessian矩阵是对角的，这都会导致错误地删除一些权重。而OBS规避了这两种错误。2）OBS允许剪掉更多的权重（训练集的误差相同的情况下），从而有更好的泛化能力。OBD和OBS都比基于幅度的剪枝精确，但是二阶导数需要额外的计算。Song Han等人在2015年成功地修剪了几个最先进的大规模网络，并表明在不损失精度的情况下，参数的数量可以减少一个数量级。

除了剪枝技术，许多研究人员在网络量化上进行了探索以此压缩网络。Hwang和Sung通过采用基于反向传播的再训练来优化定点设计。与浮点网络相比较，此文设计的具有三元权重（+1,0,-1）和3-bit信号的定点网络具有可忽略的性能损失。Anwar等提出了一种优化深度卷积网络的方法。首先使用L2误差最小化直接量化预训练网络的参数，逐一量化每个层，而其他层保持高精度的计算，然后用量化的权重对网络进行再训练，这种方法减少了网络参数量并提高了泛化能力。Chen等提出了一个新的网络架构——HashedNets，这显著减少了神经网络的存储需求，同时大部分保留了泛化性能，但是其只考虑了全连接层。Gong等人使用矢量量化压缩深度卷积神经网络。这种方法将模型压缩到员阿里的1/20但是有1%的精度损失。

目前，深度神经网络压缩的方法主要集中在剪枝、量化等。而且压缩的对象是针对深度卷积神经网络。但是基本都是主要针对全连接层进行压缩，压缩率不高而且精度有一定的损失，这些问题都亟待解决。

### 1.2.3 模型加速的研究现状

低秩分解技术也吸引了很多学者的注意。Denil等人利用这样的一个事实来进行研究：学习网络中的权重倾向于结构化。技术的核心在于：用两个较小矩阵的低秩乘积表示权重矩阵（字典学习）。在MLP和CNN还有RICA上进行实验，学习一小部分权重预测其余权重，最好的情况可以预测网络中超过95%的权重，而且并没有精度损失。由Denton等进行的后续工作提出可以加速CNN中的卷积运算的技术，其考虑基于奇异值分解（SVD）的几个基本张量分解。研究结果表明，提出的方法将强两层中权重的内存占用减少到原来的1/3~1/2。Lin等则尝试通过用全局平均池化代替全连接层来减少神经网络的参数数量。

还有一些其它尝试来加速网络，包括神经网络硬件加速器和迁移学习等。Chen等人提出了一种使用了新的数据搬移模式RS的CNN硬件加速器，以替代GPU等SIMD/SIMT架构的加速器。Shafiee等人提出了一种卷积神经网络加速器，这种基于忆阻器交叉开关的加速器能够有效降低模糊转换的开销。Hinton等提出了基于教室-学生网络的迁移学习方法，用教室网络来指导学生网络进行学习。Sergey等人借鉴迁移学习的思想，使用复杂网络中能够提供视觉相关位置信息的图像来监督小网络的学习。Google团队提出了一种轻量级的生成神经网络，取名为MobileNets，核心思想就是卷积核的巧妙分解，其可以有效的减少网络参数和加速网络。Xie等提出了一种新的结构-ResNeXt，ResNeXt对ResNet结构进行了改进。Peng等人定义了一个零的平均百分比的量来衡量每一个滤波器中激活为0的值的数量，来作为评价一个滤波器是否重要的标准。Yang等人提出了一种估计硬件能耗的工具，能够对模型的每一层计算它们的能量消耗，最大限度地降低模型的能耗。

## 1.3 研究内容及方案

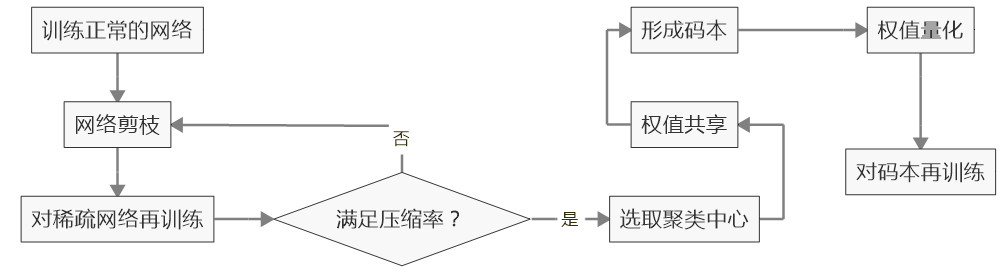
### 1.3.1 研究内容

如何在保证现有神经网络模型性能不变的情况下，有效减小神经网络模型的计算量和存储空间，成为了一个亟待解决的问题。神经网络压缩技术通过移除不重要的分支和权重共享等技术，可以降低大型神经网络其存储和计算消耗，使得其可以在移动设备上得以运行。**一方面，拥有百万级以上的深度神经网络模型内部存在大量冗余信息，因此并不是所有的参数和结构都对产生深度神经网络高判别性起作用**；另一方面，用**浅层或简单的深度神经网络无法在性能上逼近百万级的深度神经网络**。因此，通过压缩和加速原始深度网络模型，使之直接应用于移动嵌入式设备端，将成为一种有效的解决方案。

目前，深度神经网络压缩的方法主要集中在剪枝、量化和地址分解以及迁移学习等。而且压缩的对象是针对深度卷积神经网络。一般情况下，压缩和加速深度神经网络是两个不同的任务，两者之间存在区别，但又紧密联系。例如CNN分2种类型的计算层，即卷积层和全连接层。1）卷积层，是计算耗时最大的层，也是卷积神经网络能够获得高层语义信息重要层。在卷积层内，可以通过权值共享，减少对权值的大量存储。2）全连接层，不同于卷积层的局部感知，在全连接层中，每一个输出单元都与所有输入单元相关，通过密集的权值紧性连接，因此需要大量的参数。因为卷积层与全连接层内在的本职区别，通常把卷积层的计算加速和全连接层的内存压缩认为是2种不同的任务。这2类计算出之间又是紧密联系的，卷积层为全连接层提供分层的高层特征，全连接层通过分类指导卷积层的高判别力特征提取。本毕业设计研究基于剪枝的神经网络压缩技术，并实现对若干典型的神经网络模型进行压缩和加速，并通过实验研究系统性能。

本文研究了2大部分的工作，第一部分为神经网络的压缩，第二部分为神经网络的加速。在压缩工作中，主要采用了2种压缩方法，即稀疏化方法和卷积核剪枝，并实验对比该2种方法的压缩效率和精度损失。在加速工作中，主要采用了2种加速方法，即CP分解（PARAFAC分解）和Tucker分解（高阶SVD分解），并实验对比了该2种方法的加速效果和精度损失。

### 1.3.2 研究方案



图

本文的研究方案如图所示，包含网络剪枝、网络加速。

首先，基于剪枝的方法对网络去除网络中的冗余参数，以达到压缩的目的。在稀疏化网络的方法中，我们先通过训练得到一个准确率不错的网络模型，然后根据对网络准确率影响大小设定一个阈值，将低于该阈值的权值置零，最后再重训练剪枝后的网络，评估其性能。在卷积核剪枝的方法中，我们先通过训练得到一个准确率不错的模型，然后根据每一个卷积核的L1范数的大小去除卷积操作中的一些卷积核及其他们生成的下一层特征映射（feature map），最后再重训练剪枝后的网络，评估其性能。

其次，基于已经压缩的模型对卷积操作进行低秩分解，以达到加速的目的。在CP分解方法中，我们将一个卷积操作依次分解为4个卷积操作，每一个卷积操作对应的是不同维度的特征映射，最后重训练分解后的网络，并评估其性能。在Tucker分解中，我们将一个卷积操作依次分解为3个卷积操作，每一个卷积操作依次分解为3个卷积操作，每一个卷积操作对应的是不同维度的特征映射，最后重训练分解后的网络，并评估其性能。

本文基于深度学习框架PyTorch，利用深度学习的经典网络训练经典的大型数据集来测试不同压缩和加速方法的效果和精度损失，分别用LeNet-5网络训练MNIST-10数据集、AlexNet网络训练ImageNet数据集、VGG-16网络训练ImageNet数据集。

### 1.3.3 实验环境

#### 1.3.3.1 硬件

主板：TUF B360-PRO Gaming WIFI

CPU：12个逻辑CPU，型号：Intel(R) Core(TM) i7-8700 主频：3.20GHz

内存：32G

硬盘：SSD 500G

显卡：NVIDIA Geforce RTX 2070

本文主要依赖于NVIDIA 显卡，型号为GeForce RTX 2070。GeForce RTX 2070是仅次于GeForce RTX 2080的NVIDIA显卡，是NVIDIA目前GeForce Turing系列中计算速度第三快的台式机GPU。RTX 2070拥有2,304个着色单元（RTX 2080拥有2,944个着色单元）、采用GDDR6标准的8GB影像随机接达记忆器和256位内存带宽，运行频率为14GHz，最高可达448GB/s。NVIDIA在12毫米的鳍式场效应晶体管工艺上制造出TU106芯片，并包含深度学习超级采样（DLSS）和实时光线跟踪（RTRT）等功能。RTX 2070的性能与[GeForce GTX 1080](https://www.notebookcheck.net/Nvidia-GeForce-GTX-1080-Desktop-Review-Pascal-has-arrived.165500.0.html" \t "_self)相当，但在支持DLSS和RTRT的游戏中它将超过后者。因此，RTX 2070是一款非常快速的消费者台式机GPU。

#### 1.3.3.2 软件

Ubuntu 16.04 64位操作系统。

Nvidia 驱动

Cuda 9.0

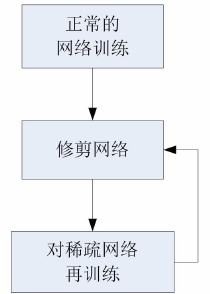
Cudnn 7.0

Pycharm professional 2019.1.1

# 第二章 网络压缩

## 2.1 基于权重剪枝的网络压缩

传统的卷积神经网络（CNN）训练完之后，模型十分庞大，全连接层的权值矩阵有十万、几百万个参数值，网络的全连接层参数最多，占据了模型的大部分。观察其参数可以发现，网络中很多参数的绝对值都很小，这些参数对CNN的训练或者测试结果作用很小，因此我们就可以尝试通过修剪网络将这些小值参数去掉，既可以减小模型的规模，有可以减少计算量。



基于稀疏化网络的模型压缩方法可以分为3步骤，算法的整体框架如图所示。首先，我们通过正常的网络训练得到准确率较高的网络，该网络中的可学习参数代表着神经元之间的连接（connection）。然后，我们设置一个阈值，将权重低于该阈值的连接从网络中移除，从而将连接十分密集的原始网络变为稀疏网络，实现初步的压缩，如图所示。最后，通过重训练剪枝后的稀疏网络来学习到最终的权重。重训练是十分重要的一步，因为如果网络经过剪枝后而不进行重训练，那么稀疏后的网络将会大大降低精度。

### 2.1.1 正则化的选择

选择正确的范数对剪枝和重训练的表现有很大的影响。一般来说，监督学习可以看做是最小化目标函数的问题，代价函数表示为



其中，第一项是损失函数，用于衡量我们的模型（分类或者回归）对第个样本的预测值和真实的标签值之间的误差。为了将训练的模型拟合给定的训练样本，因此我们要求该项项最小。但是为了防止过拟合（overfitting）问题，我们不仅要保证模型的训练误差较小，也希望模型的测试误差小，由此，我们需要对参数加上一个“惩罚因子”以约束该模型，即在代价函数上增加正则化函数。

正则化采用范数。矩阵的范数等于矩阵中各个元素绝对值之和，也被称作（稀疏规则算子），定义为



范数的引入就是为了完成特征自动选择及其可解释性。范数通过惩罚非零参数将其规范化，从而使之接近于0，这种方法会导致更多的参数接近于0。虽然L1范数能够在剪枝后获得较好的精度，但是重训练的效果不佳。

正则化采用范数。矩阵的范数等于矩阵中各元素的平方和之平方根，定义为：



是最常用的范数，我们常用的欧式距离公式就是一种范数。从学习理论的角度来说，范数可以防止过拟合，提升模型的泛化能力，从而被广泛地应用在解决机器学习里面的过拟合问题。在本文中采用范数，因为它能够获得更好的剪枝效果。因此，添加正则化的代价函数可以表示为：



### 2.1.2 随机失活

随机失活（dropout）被广泛地应用于防止过拟合，也被应用于重新训练。然而，在重训练过程中，随机失活概率必须被调整以适应模型大小的变化。随机失活是指在训练模式下以一定的概率使某些参数失活，然后在推断过程中恢复失活参数的作用。在网络剪枝中，被剪枝的参数将会永久失活，并无法在推断和训练过程中恢复。随着网络的参数变得稀疏，分类器将会选择能够提供有效信息的预测器，从而得到更小的预测方差，避免了过拟合。因为剪枝已经减小了模型的容量，重训练的随机失活概率必须更小。

令为第层的连接数，为原始网络的连接数，为重训练后的连接数，为第层的神经元个数。既然随机失活作用于神经元，且是关于的二次函数，根据等式



因此，在剪枝后的随机失活概率必须遵循等式



其中，代表了原始随机失活概率，代表重训练后的随机失活概率。

### 2.1.3 剪枝和参数适应

在重训练过程中，我们最好能够在剪枝后存活下来的权重上继续训练，而不是重新初始化被剪枝的层。CNN包含了敏感的相互适应特征：在初始化训练网络时梯度下降能够找到一个很好的解，但是在重新初始化一些层和重训练他们的时候并不能找到一个很好的解。因此，当我们重训练被剪枝的层时，我们应该保留存活下来的参数，而不是重新初始化他们。

从存活下来的权重开始重训练剪枝网络将会需要更少的计算量，因为我们不用对整个网络进行反向传播。同时，随着网络变得越深，神经网络发生梯度爆炸的可能性就会越高，这导致剪枝误差难以恢复。为了防止这种情况的发生，在剪枝全连接层后我们仅仅重训练全连接层而固定卷积层，在剪枝卷积层后我们仅仅重训练卷积层而固定全连接层。

### 2.1.4 迭代剪枝

从神经网络中学习到正确的连接是一个迭代的过程。重训练后再剪枝是一次迭代，在多次这样的迭代后，我们将会得到最少的连接数。相比于单步剪枝，该方法能够达到5倍至9倍的压缩效果且没有精度损失。每一次迭代都是寻找最好连接的贪心搜索。

### 2.1.5 剔除神经元

在剪去连接后，带有0输入和0输出的神经元将会被减去。从一个神经元剪去所有输入和输出的连接后，重训练阶段将会自动地使死去的神经元拥有0输入和0输出。这会发生是由于梯度下降和正则化。一个拥有0输入或者0输出的神经元将会对没有贡献，从而导致其梯度为0。因此，重训练过程中，死去的神经元将会自动地 被剔除掉。

## 2.2 基于卷积核剪枝的网络压缩

### 2.2.1 定义和符号

*  表示输入第个卷积层的输入通道数， 表示输入第个卷积层的输出通道数，也是第个卷积层的输入通道数。
* 为第个卷积层的feature map的高。
* 为第个卷积层的feature map的宽。
*  为第个卷积层的输入feature map，为第个卷积层的输出feature map，也是第个卷积层的输入feature map。
*  为作用于第个卷积层的个输入通道的卷积操作，它由个尺寸为的3维卷积核构成。

在对该层进行卷积核剪枝前，作用于第层的卷积操作乘积计算次数为

。

当作用于第层的第个卷积核被剪枝后，所生成下一层的第个feature map也会随之被剔除（即减少了1个输出通道），从而减少的乘积计算次数为

 。

同时，应用于下一层第个feature map的卷积核也会被剔除，因此作用于第层的卷积操作（即减少了1个输入通道），从而减少的乘积计算次数为

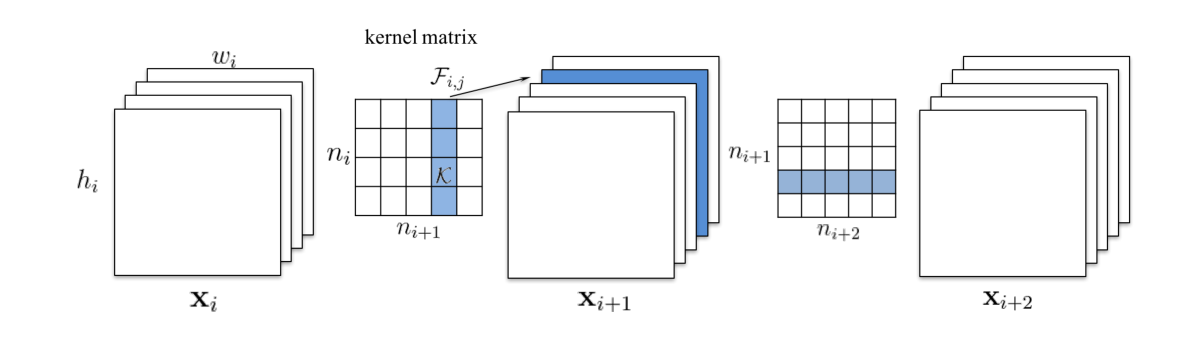
。

由此我们可得，对作用于第层的卷积操作剪去个卷积核将会使第层和第层都减少了原本计算量的。

### 2.2.2 单卷积层的剪枝策略

本文所采用的剪枝策略是从一个已经训练好的模型中剪去比较“无用”的卷积核，从而达到计算量的减少和模型的压缩，并且保证精度没有巨大的损失。由于作用于第层的卷积操作有个卷积核，每一个卷积核的尺寸都是 ，因此该卷积核权值的平均重要程度可以由范数来度量。代表了第层输出特征映射（feature map）的重要程度。相对于其他卷积核生成的特征映射而言，具有较小的的卷积核会生成激活值较小的特征映射，可以认为其生成的特征映射是较“无用”的。所以我们衡量一个卷积核重要性的方法，是计算每一个卷积核的范数，即。下面给出了对第个卷积层剪去个卷积核的算法：

1. 对于第层的每一个卷积核，计算其权重的绝对值之和 ，其中；
2. 对从小到大排序。
3. 修剪掉最小的个卷积核，以及这些卷积核所生成的特征映射。同时，作用于的所有卷积核都减少对应的个输入通道。
4. 第个卷积层由变成，而第个卷积层层由变成，剩余的权重被拷贝到新的卷积层中。



### 2.2.3 与权重剪枝的联系

卷积核剪枝与权重剪枝两种方法有相似性，两者都是基于权重的绝对值大小剪去重要性较低的权重。在权重剪枝中，当某一个卷积核的所有权重都低于阈值时，该卷积核也可能被完全地剔除。为了防止整个卷积核被剪去，权重剪枝必须很小心地调整阈值，而且无法预测有多少个卷积核会被剪去。

### 2.2.4 对最后一个卷积层的剪枝

如2.2.2中所叙述，对于卷积层的剪枝将会影响到下一层的特征映射通道数（帧数），进而影响到下一个卷积层的输入通道数。如果卷积层的下一层不是卷积层，而是全连接层，则需要另做分析。

记最后一个卷积层为，第一个全连接层为，剪枝后的为，对应的成为。用和表示最后一个卷积层的输入通道数和输出通道数。用和表示第一个全连接层的输入通道数和输出通道数，因此为的2维矩阵。

若对卷积层剪去个卷积核，则将会失去个输出通道数，因此。当某一个卷积核被剔除后，该卷积核所生成的特征映射也会被剔除。因为一个特征映射是一个大小为 的2维矩阵，因此作用在该特征映射上的全连接层将会减少的神经元个数为

，

其在数值上等价于

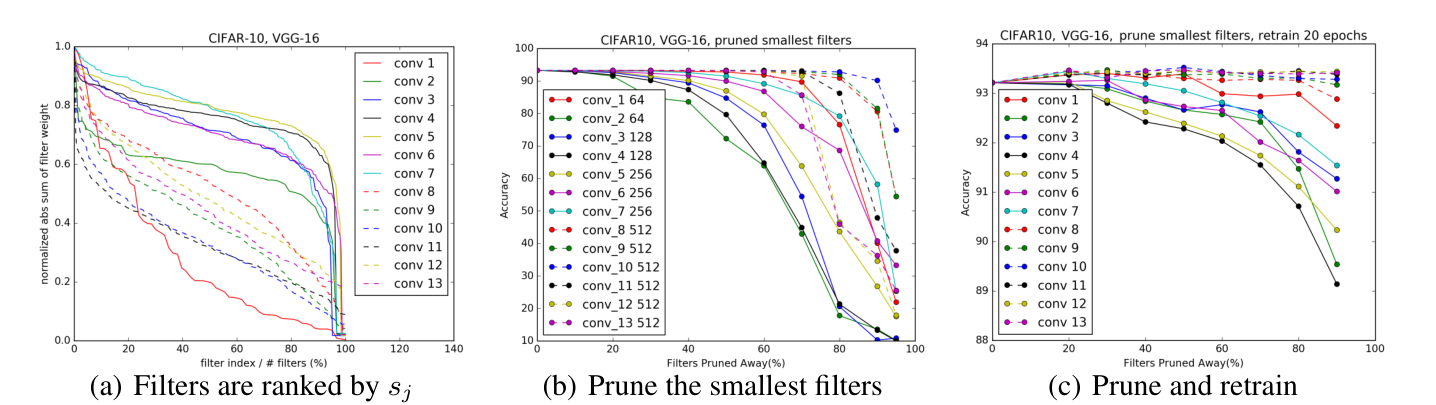
。

因此，我们可以得到剪枝后的第一个全连接层的输入通道数和输出通道数为

。

### 2.2.5 各卷积层对于剪枝的敏感程度分析

为了探究各卷积层对于剪枝的敏感程度，我们首先进行对每一个卷积层单独剪枝，然后分析剪枝后的神经网络在测试集上的正确率。图 展示了对每一个卷积层剪枝进行不同剪枝率后，该网络首次测试的精确度。从该图中可以看到，坡度越平缓的曲线所对应的卷积层，其对于剪枝的敏感程度越低。



### 2.2.6 全局剪枝策略

对于整个网络而言，多个卷积层的剪枝既可以逐层修剪，然后在重训练过程中迭代多次以弥补精度的损失；也可以直接对多个卷积层直接修剪，然后在重训练过程中多次迭代弥补精度的损失。从时间效率上来说，逐层修剪网络相对而言是比较耗时的，尤其是当网络的规模较大、卷积层数较多的情况下，逐层修剪网络将会耗费大量的时间。从网络的鲁棒性来说，直接对多个卷积层剪枝将会提高剪枝后网络的鲁棒性，因为剪枝和重训练都是在整体网络上进行的。因此，对于复杂的神经网络模型而言，一个全局的方法是很有必要的。

对于进行全局剪枝，我们考虑以下两种剪枝方案：

1. **各层独立剪枝算法**：在对某一个卷积层剪卷积核的时候不考虑其他层对该层的影响，因此在计算被剪去的卷积核的L1范数时，会考虑该卷积核中原有的所有参数。
2. **贪心算法**：若在前一个卷积层中卷积核被剪去，则考虑到生成的特征映射也会被剔除，因此的每一个卷积核将会减少1个输入通道，即的尺寸由减少为，所以在计算的L1范数的时候不考虑被减少的输入通道，即该卷积核有个权重不参与L1范数的计算。

图阐释了上述两种方案的区别。两种算法都能够使卷积层由原始尺寸变为。尽管贪心算法难以达到全局最优，但是它能够在整体地剪枝网络的前提下，达到较高的精确度。因此，本文采用贪心算法对全局神经网络进行剪枝。

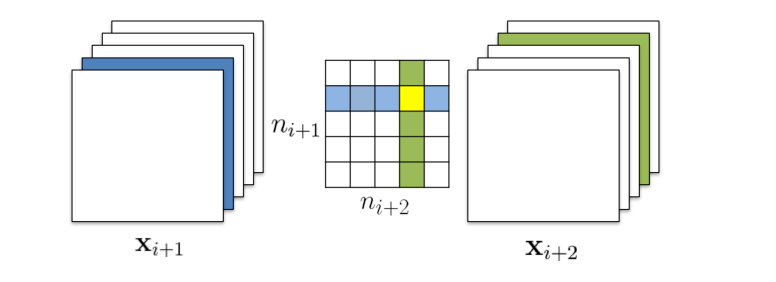


图 连续两个卷积层的剪枝示意图。各层独立剪枝算法是计算卷积核的所有参数（标记为绿色的部分，即上述的）而不考虑在前一层中被减去的特征映射（标记为蓝色的部分，即上述的），因此被标记为黄色的权重（个权重）也参与到L1范数的计算中。贪心算法并不计算黄色部分的权重。

### 2.2.7 重训练

在对网络剪枝后，需要重训练来恢复模型的精度。有两种重训练方法：

1. **剪枝1次后重训练**：对全局网络剪枝一次后就重训练，直到接近原来的精度。
2. **迭代剪枝和重训练**：每剪枝完一层先不剪枝下一次层，而是立即重训练以适应剪枝对于网络的改变。然后剪枝下一层，如此迭代。

剪枝1次后重训练的方法会将网络中的重要部分剪去，但是经过短时间的重训练后，精度能够恢复到原来的精度。但是，当一些对剪枝敏感的部分比剪去后就会难以恢复到原来的精度。因此，迭代剪枝和重训练方法会有更好的效果，但是要求更多的迭代次数。

# 第三章 低秩分解

在物理上，矩阵的秩度量的是矩阵的行列之间的相关性。秩的定义是：一个矩阵中的最大不相关的向量个数。一个矩阵是满秩的，当且仅当该矩阵的各行或各列是线性无关的。在TensorFlow和PyTorch等深度学习框架中，我们使用张量（Tensor）数据结构来表征所有的数据。与矩阵的秩不一样，张量的秩（rank）表征了张量的维度。秩为1的张量等价于向量，秩为2的张量等价于矩阵。

在网络剪枝后进行低秩分解能够有效地加速神经网络。应用低秩分解的方法可以将一个层分解为几个规模更小的层。虽然低秩分解后神经网络的层数增加了，但是FLOP（浮点操作/秒）次数的减少和权重的减小会加快神经网络的推理速度。与网络剪枝一样，低秩分解假定深层神经网络具有参数冗余的特性，而且一个层的权重可以被多个低秩张量表示，因此我们可以将低秩分解方法应用到层的权重上，把一个张量分解为多个张量乘积的近似。

## 3.1 奇异值分解（SVD）

在本文中，对全连接层的低秩分解采用奇异值分解。奇异值分解（Singular Value Decomposition，SVD）是在机器学习领域广泛应用的算法，通常应用于降维算法中的特征分解（Eigen Decomposition）、推荐系统（Recommendation System）以及自然语言处理（NLP）等领域。与特征分解一样，也是将一个矩阵分解为多个矩阵的乘积。但是SVD并不要求分解的矩阵为方阵。

### 3.1.1 SVD的定义

假设矩阵是一个的矩阵，定义矩阵A的SVD分解为：



其中，是一个的矩阵，是一个的对角阵，是一个的矩阵。的主对角线上每个元素都称为奇异值。和都是酉矩阵，即满足：

。

如果我们取出最大的奇异值，那么我们可以得到矩阵的一个近似：



是一个秩为的矩阵，在范数上非常接近于矩阵，因此是的一个良好近似。

### 3.1.2 SVD的计算

### 3.1.3 SVD应用于全连接层

在神经网络中，全连接层的本质是一个2维矩阵乘法，对于输入x和偏置单元b，一个全连接层可以表示为



我们可以对A进行奇异值分解，并且只保留最大的几个奇异值:



由该式我们可知，一个全连接层可以被两个全连接层替代：

* 第一个全连接层是一个的矩阵，且没有偏置单元，其权重来自。
* 第二个全连接层是一个的矩阵，具有偏置单元，其权重来自。

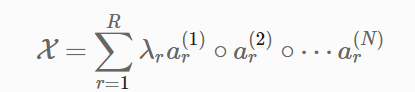
综上，一个全连接层的乘积计算次数从降低为。

## 3.2 CP分解

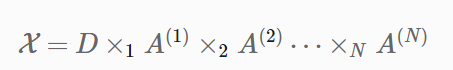
本小结将CP分解应用于神经网络的卷积层中，将一个卷积层分解为4个规模较小的卷积层，然后重训练整个网络以微调分解后的网络权重，从而减少卷积层的计算量，并且达到加速神经网络的目的。

### 3.2.1 CP分解的定义

1927年Hitchcock提出了张量的CP分解（Canonical Polyadic Decomposition，也叫做PARAFAC 或者CANDECOMP）。CP分解将一个N阶的张量分解为R个秩为1的张量和的形式，即：

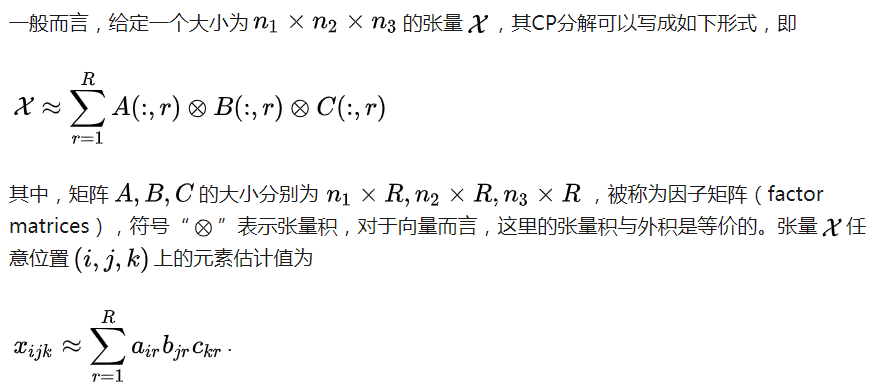


通常情况下是一个单位向量。定义那么上面的公式可以写为:

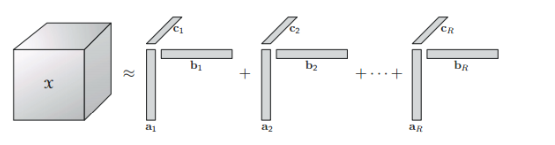


矩阵的表达形式即为：





例如，一个3阶张量的分解形式如下图所示：



对于一个n\*m且秩为R的矩阵A，为矩阵中第i行第j列的元素，其低秩分解表示为：

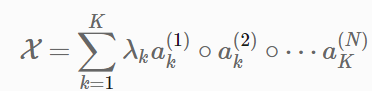


对于一个d维数组 ，其CP分解具有如下形式：

其中R被称为矩阵A的秩。值得注意的是，在张量中秩的定义不是唯一的，张量秩的个数求解是一个NP问题。在二维张量（矩阵）中，低秩分解可以采用类似于SVD的固定形式计算分解后的矩阵。但是，当张量的维度增加（即d>2时），目前还没有一种算法能够找到一个确切的张量的秩。因此，最常用的方法是采用不同的R分解张量，直到足够逼近原来的张量。

取SVD分解得到的前k个因子作为矩阵A的近似可以得到矩阵A的低秩近似。类似于矩阵中的定义我们取张量的前k个因子作为张量X的低秩近似即:



### 3.2.2 卷积核张量的近似

在卷积神经网络中，一个卷积核被描述为3维张量，其中两个维度表示在空间维度上的高和宽，第三个维度表示不同的输入通道或特征映射。CNN中最重要的一个操作是广义的卷积操作，即将输入张量变为输出张量，其中，卷积操作表示为：



其中，是一个4维张量，表示卷积操作，S是输入通道数，T是输出通道数，k为卷积核在空间上的高和宽。。

将4维张量进行秩为R的CP分解具有如下形式：



其中，是4个2维张量，它们是构成卷积操作的4个元素。将式4带入式3并进行分组排列，可得到式3的近似模式，如下表达式：



基于上式，我们可以得到从输入张量U计算输出张量V的4个带有较小规模卷积核的卷积核序列：

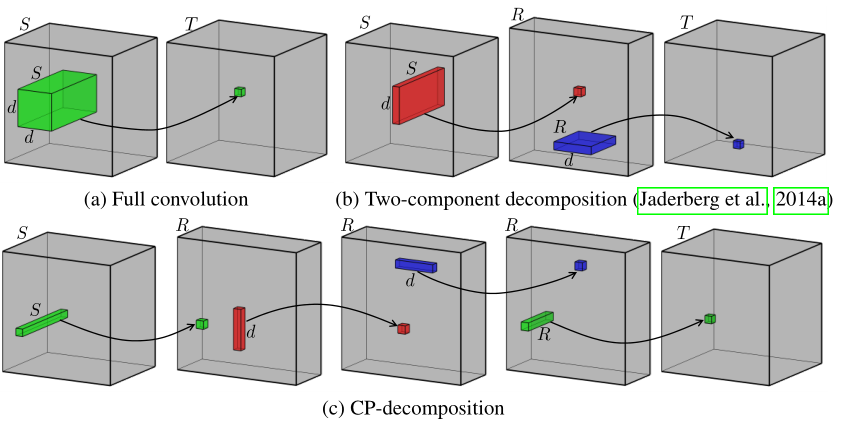








其中，，是中间张量。



### 3.2.3 实现和微调

上述4个卷积序列依次代表了4个卷积层。其中式中的和代表了 的卷积核，而和分别对应空间中水平和垂直轴上的长条形卷积核。此时，原本的一个卷积层被4个卷积层代替，这4个卷积层的维度信息如表所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 卷积层 | 高 | 宽 | 输入通道数 | 输出通道数 |
|  | 1 | 1 | S | R |
|  | d | 1 | R | R |
|  | 1 | d | R | R |
|  | 1 | 1 | R | T |

由于经过CP分解后，卷积层的数量增加了（有1个增加为4个），因此在反向传播过程中容易出现梯度爆炸的问题。为了避免梯度爆炸，我们将学习率设置的很小。

### 3.2.4 复杂度分析

CP分解之前，卷积操作中含有个可学习参数，因此对于原图中的每一个像素都需要个乘法运算。在CP分解之后，卷积操作含有个可学习参数，因此对于原图中的每一个像素都需要个乘法运算。通常来说d和T满足。当R满足 时，复杂度上将会相比于原始网络有倍的改进。

## 3.2 Tucker分解

# 第四章 实验