学校代码： 10385 分类号：

研究生学号：1611401015 密 级：



**基于深度卷积神经网络的移动端花卉识别系统**

**Flower Recognition System of Mobile Based on the Deep Convolutional Neural Network**

作者姓名： **陈浩**

指导教师： **李国刚**

实际单位导师： **马国光**

专业学位类别： **工程硕士**

专业学位领域： **电子与通信工程**

研究方向： **深度学习**

所在学院： **信息科学与工程学院**

论文提交日期：**……年…月…日**

|  |
| --- |
| 学位论文独创性声明  本人声明兹呈交的学位论文是本人在导师指导下完成的研究成果。论文写作中不包含其他人已经发表或撰写过的研究内容，如参考他人或集体的科研成果，均在论文中以明确的方式说明。本人依法享有和承担由此论文所产生的权利和责任。  论文作者签名： 签名日期： |

|  |
| --- |
| 学位论文版权使用授权声明  本人同意授权华侨大学有权保留并向国家机关或机构送交学位论文的复印件和电子版，允许学位论文被查阅和借阅。本人授权华侨大学可以将本学位论文的全部内容或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。  论文作者签名： 指导教师签名：  签 名 日 期： 签 名 日 期： |

摘要

基于深度卷积神经网络（简称CNN）的花卉识别技术是近年来的一个热门研究方向。凭借强大的非线性特征表达能力和学习能力，使得基于CNN的计算机视觉模型不断突破各种图像识别任务的技术瓶颈，但与此同时模型的深度和大小也在成倍的增长。CNN有上百万个计算节点与参数，这意味着硬件设施要有较强的算力与较大的存储资源。然而在周边应用（比如移动端和嵌入式平台）中，设备的计算能力较低，内存也较小，因此部署起来有较大难度。论文围绕这一课题展开研究，探索了一系列优化策略来使模型尺寸更小、预测时间更短、识别精度更高。论文的主要工作包括：

(1) 回顾了CNN的发展历史，从人工神经网络、全连接神经网络、经典卷积神经网络到轻量级卷积神经网络，随后分析了CNN的时间复杂度和空间复杂度，最后引入了尺度、速度和精度都取得了较好权衡的轻量级卷积神经网络MobileNet-V2。

(2) 网络训练方面，提出了一种结合Momentum的RMSProp优化算法，和其它优化算法相比，该算法在相同的迭代轮数内，损失函数下降得更快，正确率上升得更高。另外本文采用同步模式的双GPU并行训练来缩短训练时间，并探索了迁移学习、L2正则化、数据增强等不同的训练策略对最终精度的影响。

(3) 模型量化与压缩方面，提出了一种高效8-bit整数运算神经网络量化方案，该方案能将CNN的浮点运算转化为高效8-bit整数运算，减小模型大小的同时缩短模型预测时间，而精度下降却非常低。

本文选用Oxford-102 Flower公共数据集进行验证，研究阶段进行了大量细致的对比实验，并与当前学术界的花卉识别方法进行了比较。本文设计的基于深度卷积神经网络的移动端花卉识别系统，APK安装包为7.95MB，小米6手机单帧预测时间为74ms，Top-1正确率为94.7%，在尺度、速度和精度等方面都具有较好的性能。

关键词：图像识别 轻量级卷积神经网络 模型训练 模型量化

**Abstract**

Flower recognition technology based on deep convolutional neural network (CNN) has been a popular research direction in recent years. With its strong non-linear feature expression ability and learning ability, the computer vision model based on CNN continuously breaks through the technical bottleneck of various image recognition tasks, but at the same time, the depth and size of the model are also growing exponentially. CNN has millions of computing nodes and parameters, which means that hardware facilities should have strong computing power and large storage resources. However, in the surrounding application (such as mobile terminals and embedded platforms), the device has low computing power and small memory, so it is difficult to deploy. This paper focuses on this subject and explores a series of optimization strategies to make the model smaller in size, shorter in prediction time and higher in recognition accuracy. The main work of this paper includes:

(1) This paper reviews the development history of CNN, from artificial neural network, fully connected neural network, classical convolutional neural network to lightweight convolutional neural network, then analyzes the time complexity and space complexity of CNN, and finally introduces the lightweight convolutional neural network MobileNet-V2 with a good balance of scale, speed and precision.

(2) In terms of network training, a RMSProp optimization algorithm combining Momentum is proposed. Compared with other optimization algorithms, the loss function decreases faster and the accuracy rises higher within the same iteration cycle. In addition, this paper adopts dual-gpu parallel training of synchronous mode to shorten the training time, and explores the influence of different training strategies such as transfer learning, L2 regularization and data augmentation to the final accuracy.

(3) In terms of model quantification and compression, an efficient 8-bit integer arithmetic neural network quantization scheme is proposed, which can convert CNN’s floating-point arithmetic into efficient 8-bit integer arithmetic, reduce the model size and shorten the model prediction time, with few decrease of accuracy.

In this paper, the Oxford-102 Flower public data set was selected for verification. A large number of detailed comparative experiments were carried out in the research stage, and the results were compared with current flower recognition methods in the academic world. Our flower recognition system of mobile based on the deep convolutional neural network , APK installation package is 7.95 MB, the single frame prediction time of xiaomi 6 mobile phone is 74ms, and the Top-1 accuracy is 94.7%, show that it has good performance in scale, speed and precision.

**Key Words:** image recognition lightweight convolutional neural network model training model quantization

目录

第1章 绪论 1

1.1 研究背景及意义 1

1.2 国内外研究现状 2

1.2.1 数据集的选择 2

1.2.2 传统机器学习方法 2

1.2.3 深度学习方法 3

1.3 论文的组织结构 5

第2章 深度神经网络的前向预测 7

2.1 人工神经网络 7

2.1.1 生物动机与单个神经元建模 7

2.1.2 全连接神经网络 9

2.2 卷积神经网络 10

2.2.1 卷积层 10

2.2.2 池化层 13

2.2.3 全连接层 13

2.2.4 Batch Normalization层 14

2.2.5 Softmax层 14

2.3 经典卷积神经网络 15

2.3.1 LeNet-5 15

2.3.2 AlexNet 16

2.3.3 VGG-Net 17

2.3.4 GoogLeNet 18

2.3.5 ResNet 20

2.4 轻量级卷积神经网络 21

2.4.1 MobileNet-V1 21

2.4.2 标准卷积与深度可分解卷积的复杂度分析 23

2.4.3 MobileNet-V2 26

2.5 本章小结 28

第3章 深度神经网络的反向训练 30

3.1 特征学习 30

3.1.1 卷积核的作用 30

3.1.2 多层特征学习 31

3.2 深度神经网络的训练流程 31

3.2.1 损失函数 31

3.2.2 梯度下降算法 32

3.2.3 反向传播算法 33

3.2.4 改进的优化算法 37

3.2.5 训练流程概述 41

3.3 防止过拟合 42

3.3.1 大数据算法与过拟合现象 42

3.3.2 数据增强 44

3.3.3 正则化 45

3.3.4 dropout 47

3.4 加快收敛速度 48

3.4.1 迁移学习 48

3.4.2 多GPU并行训练 50

3.4.3 指数学习率衰减 53

3.5 本章小结 54

第4章 高效8-bit整数运算神经网络 55

4.1 神经网络的量化与压缩技术概述 55

4.2 高效8-bit整数运算神经网络量化方案 56

4.2.1 神经网络权值分布特征 56

4.2.2 均匀仿射量化 58

4.2.3 卷积计算转化为矩阵乘法 59

4.2.4 量化的整数运算矩阵乘法 61

4.2.5 典型的融合层实现 62

4.2.6 模拟量化训练 64

4.3 本章小结 66

第5章 实验结果与分析 67

5.1 数据集及实验环境 67

5.1.1 数据集 67

5.1.2 实验环境 69

5.2 模型训练相关实验 72

5.2.1 结合Momentum的RMSProp优化算法与其他优化算法对比 72

5.2.2 双GPU并行训练与单GPU训练对比 74

5.2.3 不同训练策略对最终精度的影响 75

5.3 8-bit模型量化相关实验 76

5.3.1 未模拟量化训练 76

5.3.2 模拟量化训练 77

5.4 与当今学术界的对比 77

5.5 移动端效果图 78

5.6 本章小结 80

总结与展望 81

参考文献 83

致谢 86

附录A Oxford-102 Flower数据集花卉类别 87

附录B 相关实验开源代码 90

在学期间的研究成果及发表的学术论文 91

第1章 绪论

1.1 研究背景及意义

鲜花是世界上最繁盛的物种之一，全世界目前发现的花卉的种类已经达几十万种。我国幅源辽阔，气候地跨三带，花卉种类繁多，资源丰富，是世界公认的花卉宝库。

花卉识别系统的研究在植物学领域是一个很重要的课题。花卉识别最原始的方法是观察花卉的生活习性、形态结构等特征，然后与已记录的花卉类别标本进行比较，最终确定花卉的种类。这种分类方法完全是人工的，工作量很大，并且需要在有丰富植物学知识和分类经验的科研人员的指导下才能顺利进行。

随着计算机技术和数字图像处理技术的发展，人们开始探索利用计算机来自动进行花卉识别的方法。花卉分类属于图像分类中较难的细粒度（Fine-Grained）图像分类问题。细粒度图像分类与粗粒度图像分类的差别在于，粗粒度图像分类的类间差异性比较大，比如像分类狗、蝴蝶和显示器（Caltech-256）等多种不相关类别。而细粒度图像分类的类间相似度很高，通常属于同一大类中的不同小类，例如花卉中不同种类花卉的区分，鸟类中不同类别鸟的区分等。此外，花卉所处环境的复杂性、花卉图像的特殊性（光照变化、视角变化、形态变化和颜色变化等）也给花卉图像的分类研究造成了很大的困难。

花卉识别的难点在于人们无法通过文字来准确描述花朵的信息，因此使用机器学习和人工智能的方法，直接通过图像来获取花朵的类别信息就成了一个主要的研究方向。目前主要分为两种方法，一种是传统机器学习方法，另一种是深度学习方法。传统机器学习方法主要是提取花卉的颜色、形状和纹理等特征来计算出花卉图像之间的相似性，进而确定花卉的种类。而深度学习方法则以CNN为代表，通过超大规模样本训练让模型做到对特征的自动提取和抽象。CNN可以自动找出复杂且有效的高阶特征，免去了人工特征提取的繁琐。目前基于CNN的深度学习方法由于能够取得更高的正确率，已逐步取代传统的机器学习方法，成为图像分类的主流算法。

此外，随着智能手机的大规模普及，人们通过手机可以很容易地拍摄到花卉的图像，将该花卉识别系统移植到手机平台后，用户可以快速便捷地了解花卉的种类，对非专业人士识别和鉴赏花朵起到一定的帮助，具有较好的实用价值。

1.2 国内外研究现状

1.2.1 数据集的选择

在图像分类中，数据集的获取是至关重要的。数据集的获取通常有两种方式，第一种是自采集，第二种是使用公共数据集。自采集的好处是可以根据自己的需要来进行数据的挑选，和实际问题贴合比较紧密，坏处是需要耗费大量的人力物力，而且有些高质量的数据集的构建还需要相关的专业知识（比如医疗图像）。采用自采集的方法通常很难与现有方法进行公正的比较，在自建数据集上的效果好坏往往很难证明该算法的优劣；而使用公共数据集的好处是可以将自己的算法与学术界的前沿算法进行最直接的比较，另外也节省了数据采集的人力物力成本，坏处就是对数据的掌控性没有自采集那么强。目前学术界主流的花卉识别算法研究通常采用第二种方法。

目前学术界对花卉识别算法进行算法验证所采用的的数据集，一般都是英国牛津大学Visual Geometry Group（VGG）组所创建的Oxford-102 Flower数据集。该数据集包含102类花卉图像，每类图像有40~250张，总共8189张图像。该数据集兼顾了花卉图像识别中所有的难点，包括光照变化、视角变化、形态变化和颜色变化等。此外不同类别的花卉之间具有很大的相似度，同一类别的花卉又有很大的差异性。该数据集被固定分为三个集合，分别为训练集（6149张）、验证集（1020张）和测试集（1020张）。

1.2.2 传统机器学习方法

传统机器学习方法主要是基于特征提取和浅层学习结构进行图像分类，特点是需要人工提取图像的特征。基于传统机器学习的花卉识别方法通常分为三步：图像预处理（分割）、特征抽取和分类器训练。目前学术界采用传统机器学习方法来进行花卉识别的主要研究成果如下：

(1) Nilsback和Zisserman[1]最早在花卉图像分类上取得较为显著的进展。他们首先提出了花卉图像分割算法用于分割花卉图像的前景和背景。将花卉的前景分割出来后，提取花瓣特征、纹理形状特征、边缘形状特征和颜色特征四种特征，然后将提取的特征送入SVM分类器训练，最终Top-1正确率72.8%。

(2) Chai等[2]在此基础上提出了一种新型的可扩展的联合过滤分割算法BiCoS，该算法与传统花卉图像分割算法相比更加简单，且分割效果更好。而用于分类的特征提取方法和Nilsback等人相比并没有太大的不同，最终Top-1正确率79.4%。

(3) Angelova等[3]改进了之前的算法。他们首先检测可能属于物体的low-level区域，然后使用propagation算法对原图像进行一个全分割，提取原图像的HOG特征1和全分割后的图像的HOG特征2，然后将两种特征连接起来送入SVM分类器训练，最终Top-1正确率80.7%。

采用传统机器学习的花卉识别方法虽然都在各自的年代都取得了较好的检测效果，但同时也存在较大的局限性。首先，花卉图像分割算法步骤复杂且繁琐，而且即便是目前最好的花卉图像分割算法依旧存在着分割失败的问题。此外，采用手工特征提取的方法进行图像识别，分类的正确率很大程度上要依赖于特征的选择也就是研究人员的经验问题，而特征的提取往往是非常耗时耗力的，研究人员必须在该领域有非常深入的理解。除此之外，提取的特征往往不具有扩展性，可能在某一个数据集上面表现不错但是在其它数据集上面就只有很低的准确率。

1.2.3 深度学习方法

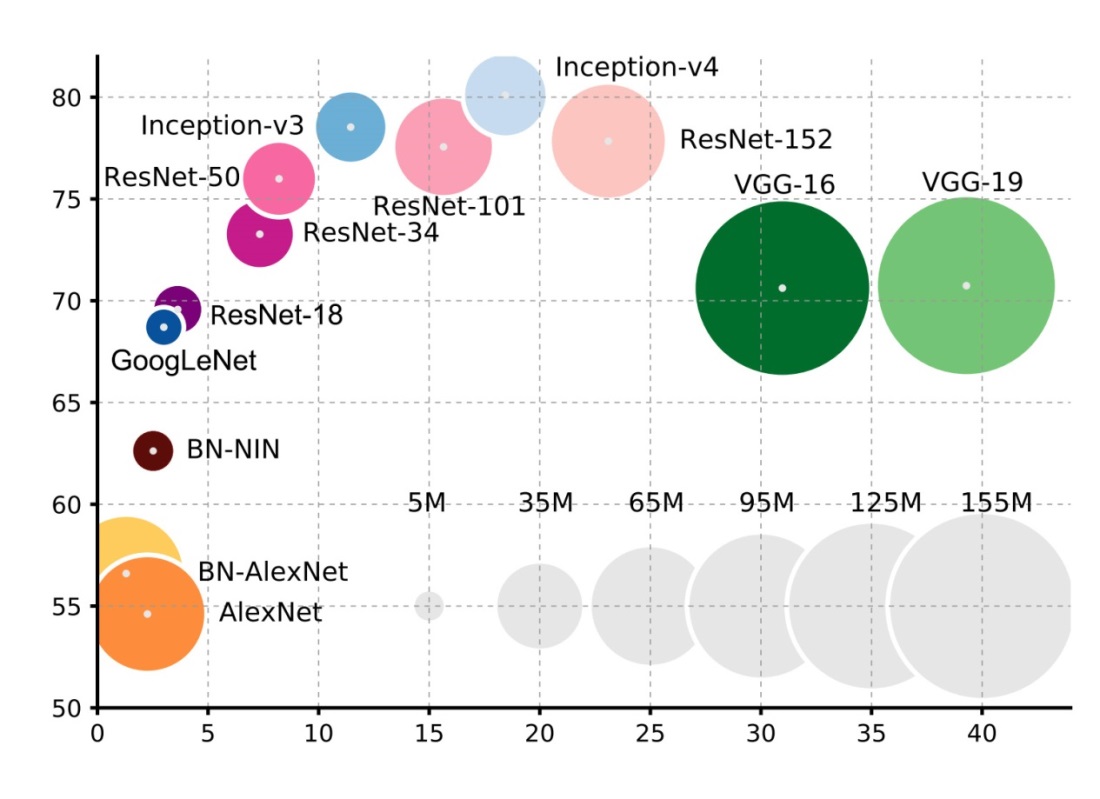
深度学习主要以CNN为代表，通过超大规模样本训练让模型做到对特征的自动提取和抽象。以CNN为代表的自动特征提取方法可以让模型自动找出复杂且有效的高阶特征，免去了人工特征提取的繁琐。CNN无需进行复杂且重复的图像预处理（分割）工作，可以直接使用图像的原始像素作为输入。此外，CNN还对一定程度内的平移、旋转、缩放、视觉改变、亮度调整等畸变具有不变性，有着很强的泛化性。国内外许多研究人员开始采用CNN对花卉进行分类，并且取得了不错的分类效果，主要研究成果如下：

(1) 程[4]采用AlexNet（Top-5错误率16.4%），用ILSVRC数据集训练好的网络参数初始化网络层（最后一层全连接层除外），以较大的学习率训练最后一层全连接层，以较小的学习率微调瓶颈层。程针对Oxford-102 Flower这一细粒度花朵分类数据集，分别训练了粗分类模型和细节分类模型，然后将2种模型的分类结果融合，最终Top-1正确率为82%。

(2) Hu等[5]采用GoogLeNet（Top-5错误率6.67%），先在ILSVRC数据集上训练网络，之后替换掉最后一层全连接层，再在Oxford-102 Flower数据集上微调网络。Hu等认为细粒度花朵分类应关注特定部分（如叶、茎和花瓣）而不是整张图像，因此采用selective search算法裁剪出整张图像的region proposal，再对花朵图像的region proposal进行训练，最终Top-1正确率为88.4%。

(3) Xia等[6]采用Google Inception Net系列的Inception-v3（Top-5错误率3.5%），先在ILSVRC数据集上训练网络，之后替换掉最后一层全连接层。Xia等对一些数量较少的花朵类别额外增加样本量，之后单独训练最后一层全连接层，最终Top-1正确率为94%。

采用CNN虽然可以获得很高的正确率，但也存在缺点，即运算量过大。CNN的前向传播过程需要进行大量浮点数的乘加运算。Canziani等[7]人认为CNN预测一张图片的时间主要花费在浮点数的乘加运算，因此可以采用浮点数的乘加运算总次数（floating point operations，简称FLOPs）来有效评估运行时间。图1.1展示了最近几年ILSVRC挑战赛各网络架构的Top-1正确率，浮点数乘加运算总次数和参数的数量之间的关系，其中气泡大小与网络参数数量成正比。



浮点数乘加运算总次数（G- FLOPs）

Top-1正确率（%）

图1.1 Top-1正确率vs浮点数乘加运算总次数，模型大小∝参数数量

从上图可以看出，通常正确率越高的CNN，运算量也越大。目前基于CNN的花卉识别研究的总趋势是使用更深层更复杂的网络来实现更高的正确率，然而更精准的网络也同时伴随着更大的运算量。CNN对硬件要求较高，目前大多数基于CNN的花卉识别模型由于计算复杂度过高，通常很难在计算能力有限的移动端做到实时识别。此外，移动平台的内存通常较小，而大多数CNN对内存也有一定的要求。因此将CNN部署到移动端或嵌入式设备这类资源有限的平台时，我们应权衡好精度、速度和模型大小这三者的关系。此外，将模型量化和压缩，使其模型尺寸更小、预测更快、耗电更低，使其更适合部署到移动端也是非常有必要的。

综上所述，基于CNN的自动特征提取方法由于能够取得更好的分类效果，已逐步取代传统的手工特征提取方法，成为图像分类的主流算法。本文仍然采用CNN来进行花卉识别研究。此外，针对移动端或嵌入式设备这类计算能力有限且资源较少的平台，本文采用了一种小型、低延迟且准确率较高的轻量级深度CNN MobileNet-V2[8]。另外，本文采用了一种高效8-bit整数运算神经网络量化方案使模型尺寸更小、预测更快，更适合部署到移动端。

1.3 论文的组织结构

论文组织结构安排如下：

第1章首先介绍了本论文的研究背景及意义，并阐述了国内外的研宄现状：数据集的选择、基于花卉识别的传统机器学习方法和深度学习方法。另外介绍了本论文的研究目标与研究内容，列举了论文的组织结构。

第2章介绍了深度神经网络的前向预测过程。首先介绍了人工神经网络，交代了它的由来和其中最普遍的全连接神经网络，接着过渡到卷积神经网络，以及最早的卷积神经网络LeNet-5和历年ILSVRC挑战赛经典的卷积神经网络架构，最后引入了轻量级卷积神经网络MobileNet-V2。

第3章介绍了深度神经网络的反向训练过程。首先介绍了特征学习方面的一些理论知识，然后介绍了深度神经网络的具体训练流程，紧接着介绍了训练过程中的过拟合现象和如何解决过拟合，以及加快训练速度的一些常用方法。

第4章首先简要概述了神经网络的量化与压缩技术，然后详细介绍了高效8-bit整数运算神经网络的一整套量化实现方案。

第5章对本论文中提出的算法进行实验展示以及分析，包括模型训练相关实验、8-bit模型量化相关实验以及与当今学术界的对比实验。通过严谨的实验验证以及详细的分析，验证了本文算法的有效性。

最后，对本论文作了总结，分析了本文中的一些不足之处和可以改进的地方，并对未来工作进行了展望。

第2章 深度神经网络的前向预测

本章主要介绍深度神经网络的预测（inference）。本章首先介绍了人工神经网络，交代了它的由来和其中最普遍的全连接神经网络。卷积神经网络是在全连接神经网络的基础上专门为解决图像问题而提出来的，它主要采用了卷积加池化的组合结构来有效提取图像的特征。一个优秀的卷积神经网络是有一定的结构规律的，本章随后介绍了最早的卷积神经网络LeNet-5,历年ILSVRC挑战赛经典的卷积神经网络架构，以及轻量级卷积神经网络MobileNet系列，它们对于卷积神经网络的设计具有良好的借鉴意义。

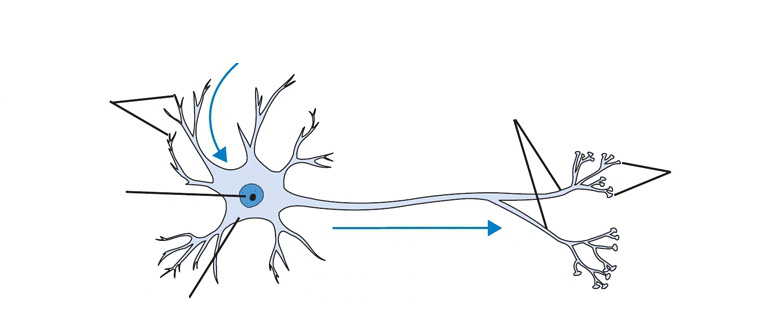
2.1 人工神经网络

神经网络算法最初是从生物神经系统建模得到了启发，但随后独自发展而形成一个工程应用问题，并在机器学习领域获得良好效果。它模仿了生物神经网络的结构和功能，形成一个与之类似的数学模型。

2.1.1 生物动机与单个神经元建模

大脑是由神经元这一基本单元构成，人类的神经系统中大约有860亿个神经元，它们被大约1014~1015个突触连接起来。图2.1展示了一个生物学的神经元。每个神经细胞都从它的树突传入输入信号，然后沿着它唯一的轴突传出输出信号。轴突在末端会开始产生分枝，通过轴突末梢和其他神经元的树突连接。

图2.1 生物神经元



神经冲动传入细胞体

树突

细胞核

细胞体

轴突

神经冲动从细胞体传出

轴突分支

轴突末梢

图2.2展示了一个生物神经元的数学模型。在神经元的计算模型中，沿着轴突传播的信号（*x*0）将基于突触的突触强度（*w*0），与其他神经元的树突进行乘法交互（*w*0*x*0 + *w*1*x*1 + *w*2*x*2）。突触的强度（即权重*w*），是可训练的且能够控制一个神经细胞对于另一个神经细胞的影响强度，使其兴奋（正权重*w*）或使其抑制（负权重*w*）。在神经元的计算模型中，树突将输入信号传入细胞体，来自树突的所有输入信号在细胞体中相加，如果最终之和大于某个阈值，那么这个神经细胞将会激活，并向其轴突输出一个激活信号。



**其他神经元的轴突**

**突触**

**树突**

**细胞体**

**激活函数**

**轴突输出**

图2.2 单个神经元的数学模型

如图2.2所示，单个神经元的输出表达式为：

|  |  |  |
| --- | --- | --- |
|  |  | （2.1） |

权重 *w* 和偏置 *b* 为可训练并改变的变量， *f* 为激活函数，常用的激活函数有Sigmoid函数、Tanh函数和ReLu函数等。表2.1列举了常用激活函数的表达式：

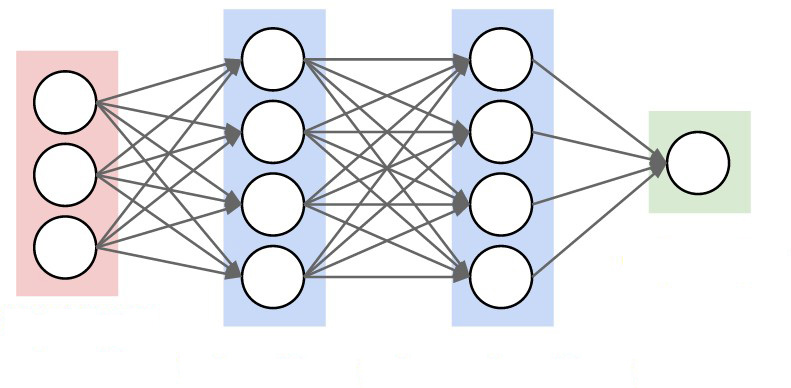
表2.1 常用激活函数的表达式

|  |  |
| --- | --- |
| 名称 | 表达式 |
| Sigmoid |  |
| Tanh |  |
| ArcTan |  |
| ReLu |  |
| PReLu |  |
| ELU |  |
| SoftPlus |  |

2.1.2 全连接神经网络

所谓的神经网络就是将许多个单一的神经元联结在一起，这样，一些神经元的输出就可以是另一些神经元的输入。神经网络模型是由神经元分层构成的，而不是类似生物神经元一样聚合成团状。神经网络模型中最普遍的层是全连接层，全连接层中的神经元与其前后两层的神经元是全部连接的，但是同一层内的神经元之间没有连接。图2.3为全连接神经网络的结构图例：

图2.3 全连接神经网络



**输入层**

**隐含层1**

**隐含层2**

**输出层**

神经网络中除了输入层和输出层外，中间的网络层统一称为隐含层，因为不能直接观测到它们的值。图2.3是一个3层的全连接神经网络（一般输入层不计算在内），有两层含4个神经元的隐含层。隐含层的神经元个数与层数没有固定值，通常是根据实际问题具体设计。通常来说隐含层的神经元个数越多，隐含层的层数越深，越能拟合出复杂的函数，但与此同时也伴随着参数增多导致难以训练，以及极易陷入过拟合等问题。因此设计一个优秀的全连接神经网络模型需要根据实际情况综合考虑并权衡利弊。

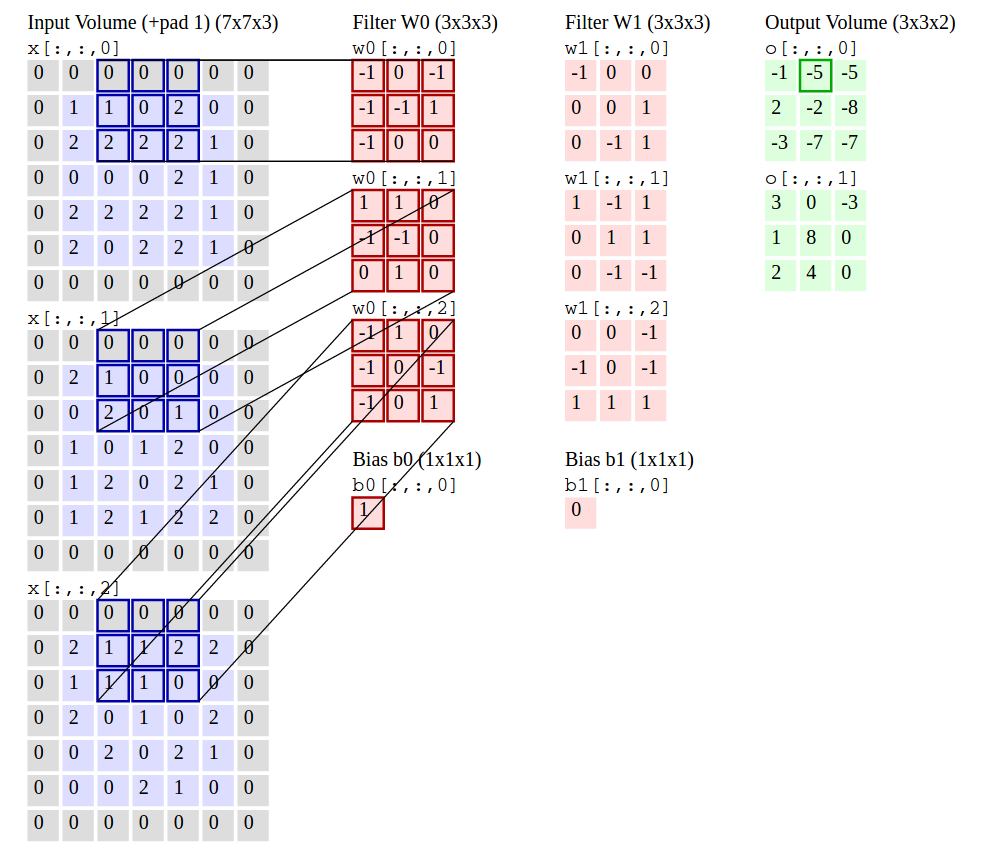
2.2 卷积神经网络

卷积神经网络和全连接神经网络结构类似，也是通过一层一层的神经元连接起来的。两者最大的不同在于，全连接神经网络中的神经元与其前后两层的神经元是全部连接的；而卷积神经网络中的前后两层神经元只有部分连接，它将每一层卷积层的神经元组织成一个宽度方向、高度方向和深度方向的三维矩阵。卷积神经网络主要由三种类型的层构成：卷积层、池化层和全连接层。通过将这些不同类型的层叠加起来，就可以组成一个完整的卷积神经网络。

2.2.1 卷积层

卷积神经网络的核心层是卷积层，网络中大部分的计算量都是卷积运算产生的。卷积层主要由一些参数可训练的滤波器构成的。每个滤波器在宽度和高度上都比较小，但是在深度上和输入数据的深度相同。在前向传播的时候，每个滤波器沿着输入数据的宽方向和高方向滑动（卷积），然后计算滤波器的参数值和相对应的输入数据值的内积。而步长则定义了滤波器每次移动的像素值。当滤波器在输入数据的宽方向和高方向滑过后，会生成一个二维的特征图（feature map），特征图给出了在每个空间位置处滤波器的反应。在前向传播的时候，当滤波器看到某些类型的视觉特征时，特征图中相应空间位置处的节点就会激活。具体的视觉特征可能是第一层中某些颜色的斑点，或者某些方向上的线条、边和角，甚至可以是更高网络层中的复杂形状和图案等高阶特征。在每个卷积层中，网络都会有一整个集合的滤波器，每个滤波器都会生成一个不同的二维特征图，将这些二维特征图在深度方向上重叠起来就组成了输出特征图谱。

图2.4为卷积操作示意图。其中输入数据是蓝色，权重数据是红色，输出数据是绿色。输入数据的宽度为5，高度为5，深度为3，输入数据在图像边缘处进行了0填充。有2个滤波器w0和w1，其中每个滤波器的宽度为3，高度为3，深度为3。每次卷积操作滤波器以2的步长沿着宽和高方向滑动，蓝色的输入数据和红色的滤波器逐元素相乘，然后求其总和，最后加上偏差，便得到对应位置的绿色的输出数据。卷积运算的本质其实就是在滤波器和输入数据的局部区域间做点积运算。

图2.4 卷积操作示意图

卷积神经网络中的卷积层采用了局部连接(local connection)和权值共享(weight sharing)的结构，相比全连接神经网络能够大大降低参数量和训练复杂度，并有效减轻过拟合问题。所谓的局部连接，即卷积层的每个神经元只与输入数据的一个局部区域进行连接，该局部区域通常叫做神经元的感受野(receptive field)。感受野的尺寸其实就是滤波器的空间尺寸。一个卷积层可以有多个不同的滤波器，而每一个滤波器都会生成一个不同的二维特征图。同一个二维特征图中的每一个像素都来自完全相同的滤波器，这就是所谓的权值共享。

采用卷积神经网络中的卷积操作后，不管输入图像大小如何变化，需要训练的参数数量只和滤波器大小、滤波器个数有关。使用多层卷积操作后，网络可以抽象组合出更高阶的特征。相比全连接神经网络使用多个隐含层逐层提取特征，卷积神经网络的多层卷积抽象表达能力更强，效率更高。

2.2.2 池化层

在连续的卷积层之间往往会插入一个池化层。池化层的作用是降低数据体的空间尺寸，从而减少最后全连接层中的参数个数。使用池化层既可以加快计算速度，也可以有效减轻过拟合问题。

和卷积层类似，池化层的前向传播过程也是通过移动一个类似过滤器的结构完成的。卷积层是在过滤器和输入数据的局部区域间做点积运算，而池化层是计算输入数据的局部区域内的最大值或者平均值。计算最大值的池化层称为最大池化层，计算平均值的池化层称为平均池化层。实践中通常采用最大池化层。还有一点不同的是，卷积层的过滤器是横跨整个深度的，而池化层的过滤器只在一个深度方向上进行操作。因此池化层的过滤器除了要沿着输入数据宽度方向和高度方向滑动外，还要沿着深度方向滑动。

图2.5为池化操作示意图，池化过滤器的宽度和高度为2，滑动步长为2。左边的数据体经过池化操作后尺寸从224×224×64缩小到了112×112×64。池化操作只会改变数据体的宽度和高度，不会改变数据体的深度。右边为最大池化操作示例，每次池化操作过滤器会沿着数据体的宽度、高度和深度方向滑动，并从2×2的矩形方块区域中选取最大像素值作为输出。

图2.5 池化操作示意图



2.2.3 全连接层

在经过多轮卷积层和池化层的处理后，卷积神经网络的最后几层一般是采用全连接层来处理最后的分类结果。全连接层和全连接神经网络一样，每一层的神经元与其前后两层的神经元是全部连接的，但是同一层内的神经元之间没有连接。全连接层的参数通常是网络中最多的，但产生的计算量通常是网络中最少的。网络中大部分的计算量通常是卷积运算产生的。经过多层卷积层和池化层的处理后，可以认为图像中的信息已经被抽象成表达能力很强的特征向量，此时用1~2个全连接层处理后便可以得到很好的分类结果。

2.2.4 Batch Normalization层

我们知道神经网络的训练本质就是为了学习数据分布。而神经网络的训练是一个复杂的过程，网络一旦训练起来参数就要发生更新，因此除了输入层的数据外，网络每一层的输入数据分布是一直在发生变化的。一旦网络某一层的输入数据的分布发生改变，那么这一层网络就需要去适应学习这个新的数据分布，因此只要网络的前面几层发生微小的改变，那么后面几层就会被累积放大下去，导致训练数据的分布一直在发生变化，严重影响网络的训练速度。我们把网络中间层在训练过程中，数据分布的改变称之为Internal Covariate Shift[9]。而Batch Normalization[9]的提出就是为了解决中间层数据分布发生改变的情况。

Batch Normalization其实是将输出数据做归一化处理后，再做一次变换重构。重构过程中引入了两个可学习的参数*γ*、*β*，这两个参数可以恢复出原始网络所要学习的特征分布。

Batch Normalization层的前向传播过程公式如下：

|  |  |  |
| --- | --- | --- |
|  |  | （2.2） |

在训练阶段中， 和 为一个batch的均值和标准差，而在预测阶段则为整个训练集的均值和标准差。

2.2.5 Softmax层

采用卷积神经网络来处理图像分类问题时，最后一层全连接层的输出为每一种类别的置信度，但通常会增加一层Softmax层将输出归一化为0~1区间的概率分布值。假设网络的最后一层全连接层的输出为*y*1，*y*2，…*y*n，那么经过Softmax回归处理之后的输出为：

|  |  |  |
| --- | --- | --- |
|  |  | （2.3） |

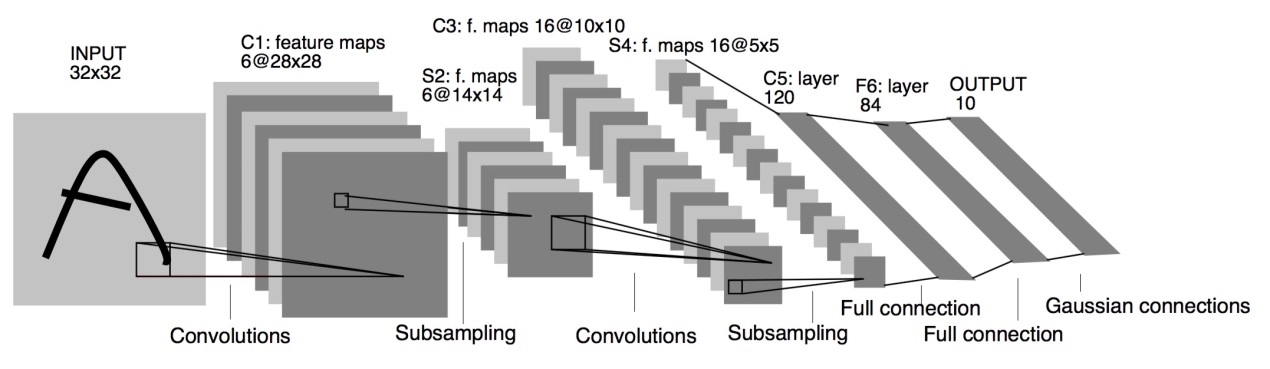
经过Softmax回归处理后，每一个输出值可以看成神经网络认为图像为该类别的概率有多大，网络所有类别的输出值的总和为1。

2.3 经典卷积神经网络

采用卷积层、池化层和全连接层任意组合得到的神经网络有无限多种，但一个优秀的卷积神经网络通常是有一定的结构和组合规律的。本节首先介绍最早的卷积神经网络模型LeNet-5，然后介绍历年ILSVRC挑战赛经典的卷积神经网络架构。ILSVRC[10]全称ImageNet Large Scale Visual Recognition Challenge，是计算机视觉领域最权威、最具影响力的竞赛，每年都会吸引全球众多工业巨头和知名高校参与。比赛项目有图像分类、目标位置检测、视频检测、图像分割等，其中图像分类主要是对ImageNet子数据库中的1000种类别进行分类，比赛采用Top-5分类错误率和Top-1分类错误率作为模型性能的评测指标。

2.3.1 LeNet-5

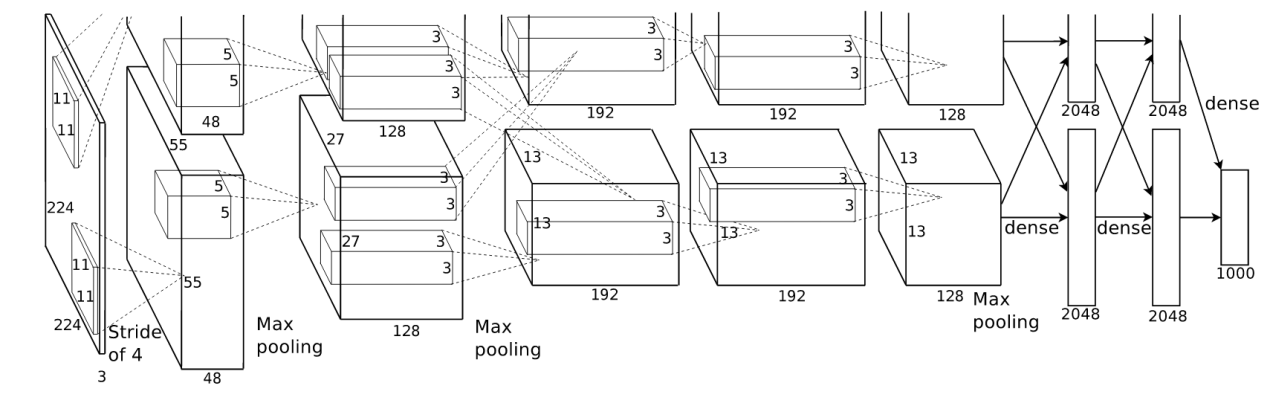
卷积神经网络最早可以追溯到Yann LeCun在1998年提出的LeNet-5[11]，它是第一个成功应用于手写数字识别的卷积神经网络。LeNet-5一共有7层，即3层卷积层、2层平均池化层、1层全连接层和1层输出层，模型结构如图2.6所示。网络的输入为32×32的黑白图像，后面有3个卷积层，其中前2个卷积层的后面连着平均池化层。卷积层的过滤器尺寸为5×5，步长为1，第一个卷积层有6个过滤器，第二个卷积层有16个过滤器，第三个卷积层有120个过滤器。平均池化层的过滤器尺寸为2×2，步长为2。在多层卷积层和池化层之后为全连接层，有84个隐含节点。模型的最后一层为10个输出节点的欧式径向基函数，它处理最后的分类结果。LeNet-5在MNIST数据集可以达到99.2%的准确率，可以说是领先时代的重大突破。LeNet-5是卷积神经网络的开山之作，它奠定了现代卷积神经网络的基石。

图2.6 LeNet-5网络结构

2.3.2 AlexNet

卷积神经网络的开山之作LeNet-5虽然可以达到很高的准确率，但随后几年因为计算能力的制约，卷积神经网络的研究一直没有太大突破，神经网络也一度被支持向量机（Support Vector Machine，简称SVM）等超越而陷入低谷，在21世纪初的很多年里几乎被大家遗忘，相关研究一直都不受重视。这一段是神经网络的一次冰期。

在2012年的ILSVRC中，Alex等提出了8层的深度卷积神经网络模型AlexNet[12]，以巨大优势摘得了当年比赛的第一名（Top-5错误率16.4%），领先第二名（Top-5错误率26.2%）超过10个百分点的准确率，震惊了整个学术界。AlexNet可以算是LeNe-5的一种更深更宽的版本，它继承了LeNet-5的思想，用多层卷积加池化的组合来提取特征，最后用全连接层对特征进行组合分类。AlexNet首次在卷积神经网络中应用了ReLU激活函数、Dropout和LRN等新技术，同时使用了2块GTX 580 GPU进行并行加速运算。AlexNet的网络结构如图2.7所示。

图2.7 AlexNet网络结构

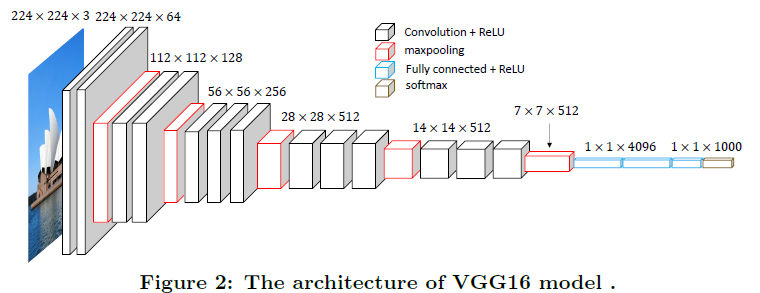
AlexNet总共有8层（不包括池化层和LRN层，因为它们没有可训练的参数），有5个卷积层、3个池化层、2个LRN层和3个全连接层。AlexNet的前5层为卷积层，后3层为全连接层，LRN层出现在第一层和第二层卷积层后，最大池化层出现在2个LRN层和最后一个卷积层后。8层中的每一层在最后都使用了ReLU激活函数。

AlexNet的输入为224×224的RGB三色通道图像，第一个卷积层的过滤器尺寸为11×11，步长为4，但随后的卷积层过滤器尺寸都比较小，都是5×5或者3×3，步长都为1。而最大池化层的尺寸都为3×3，步长都为2。AlexNet在最后的几层全连接层中使用了Dropout技术，通过在训练过程中随机忽略一部分神经元来避免模型陷入过拟合。AlexNet的输出层采用了Softmax层来对1000种类别进行分类。

AlexNet是神经网络在低谷期后的第一次发声，在当年以压倒性优势夺冠后便瞬间点燃了卷积神经网络研究的热潮，同时也推动了深度学习在图像识别、目标检测、自然语言处理、语音识别和强化学习等领域的扩展。在AlexNet之前，每一年的ILSVRC参赛作品都是做传统机器学习的特征提取工程，然后使用SVM进行分类。而AlexNet夺冠后，每一年ILSVRC的冠军都是基于深度学习和卷积神经网络。可以说AlexNet的提出确立了深度卷积神经网络在计算机视觉的统治地位。

2.3.3 VGG-Net

VGG-Net[13]是2014年ILSVRC挑战赛的亚军（Top-5错误率7.3%），它由牛津大学Visual Geometry Group和谷歌DeepMind的研究人员一起研发。VGG-Net的网络结构如图2.8所示。

图2.8 VGG16网络结构

VGG-Net结构非常简洁，整个网络都采用了同样大小的卷积过滤器（尺寸为3×3，步长为1）和最大池化过滤器（尺寸为2×2，步长为2），通过不断加深网络的深度来获得更高的性能。VGG-Net有五段大卷积层，第一段和第二段有2个串联的小卷积层，第三段、第四段和第五段有3个串联的小卷积层，每一段大卷积层尾部都会连接一个最大池化层来压缩数据体。每一段大卷积层的卷积过滤器个数一样，越往后卷积过滤器个数越多。从第一段大卷积层到第五段大卷积层卷积过滤器个数分别为：64、128、256、512和512。

VGG-Net中多个完全一样的小卷积层串联是非常有用的设计。两个串联的3×3卷积层相当于1个5×5的卷积层，而三个串联的3×3卷积层相当于1个7×7的卷积层。三个串联的3×3卷积层的参数个数只有1个7×7的卷积层的

(3×3×3) / (7×7) = 55%，此外3个串联的3×3卷积层可以做3次非线性变换（使用3次ReLU激活函数），而1个7×7卷积层只能做1次非线性变换。多次非线性变换可以增强卷积神经网络的特征学习能力。

VGG-Net的创新点在于通过反复堆叠3×3的小卷积层来减少参数个数以及获得更好的非线性变换，同时VGG-Net也证明了加深网络的深度可以获得更好的性能。虽然VGG-Net参数非常多并且计算量也非常大，但它的拓展性很强并且泛化性很好，因此VGG-Net仍旧被广泛应用于许多计算机视觉场合。

2.3.4 GoogLeNet

GoogLeNet[14]是2014年ILSVRC挑战赛的冠军（Top-5错误率6.7%），它最大的特点是保持计算量和参数量都较少的同时，又有很好的分类性能，可以说是非常优秀并且非常实用的模型。GoogLeNet有22层，比AlexNet的8层或者VGG16的19层还要深，但GoogLeNet的计算量（浮点数乘加总次数）只有1.5G，比VGG16的15G要少很多。GoogLeNet的参数个数只有6.8M，比AlexNet的60M和VGG16的138M都要少。

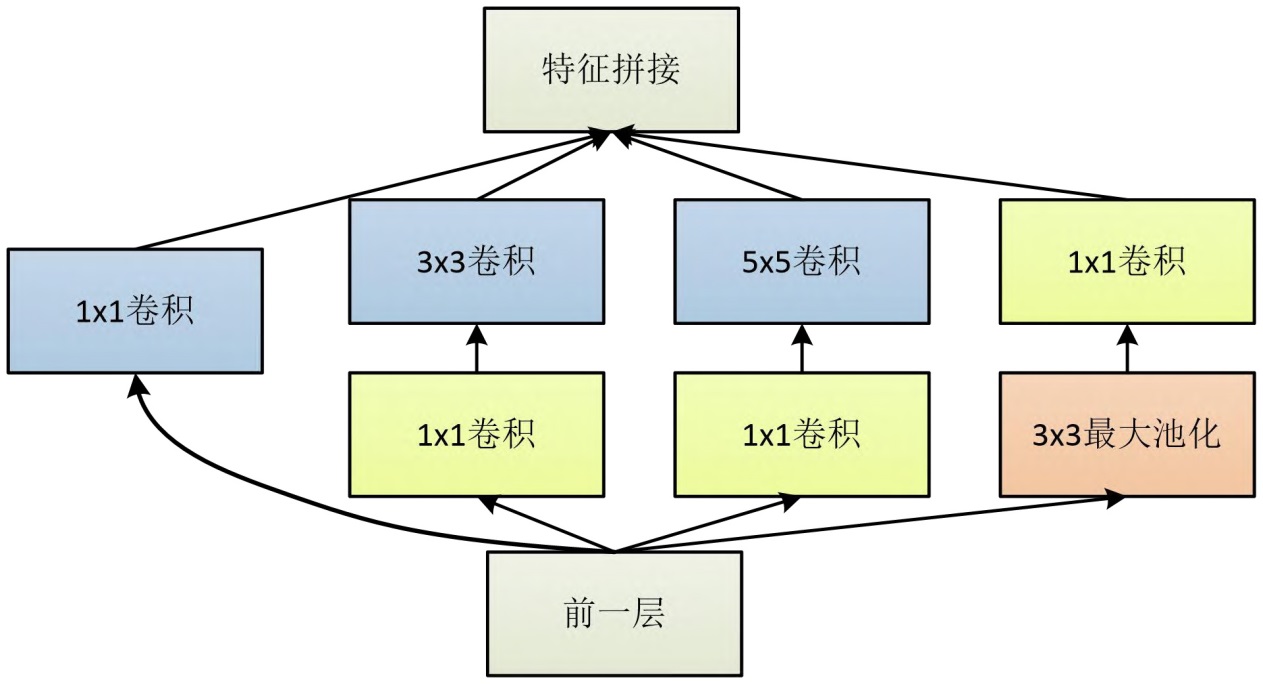
GoogLeNet的创新点在于提出了Inception Module的模组结构，如图2.9所示。我们知道不同尺度的卷积过滤器可以提取不同抽象程度的特征，在以往的模型中主要是通过串联的方式将不同的卷积层连接在一起，这种方法会导致卷积层的特征表达能力较弱。而Inception Module则同时使用了1×1、3×3和5×5三种不同尺寸的过滤器，最后再将不同卷积层的输出通过并联的方式结合在一起，即把不同的输出数据在深度这个维度上拼接。Inception Module同时采用了不同尺寸的过滤器，这意味着同时使用了不同大小的感受野，而最后的拼接则意味着不同尺度特征的融合。Inception Module大大提高了卷积层的特征表达能力，使得网络在同一层就可以提取稀疏或非稀疏的特征。Inception Module既增加了网络的宽度和深度，也增加了网络对不同尺度的适应性

图2.9 Inception Module结构

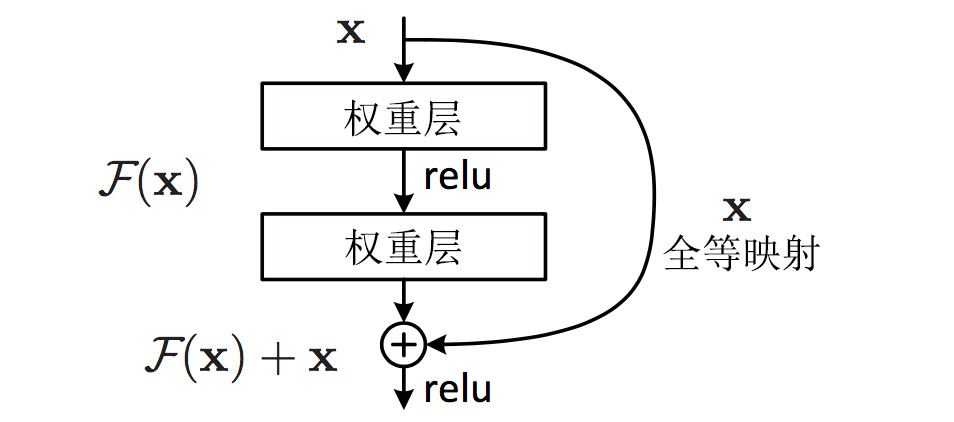
除了Inception Module外，GoogLeNet还去除了最后的全连接层，用全局平均池化层来取代。这是因为在卷积神经网络中全连接层占据了大部分的参数量，但只产生很少的计算量，因此可以把最后的全连接层换成全局平均池化层来减少参数个数，同时也可以加快模型的训练速度。

GoogLeNet也被称为Inception V1。在Inception V1夺冠后，谷歌陆续提出了各种版本的Inception Net，包括Inception V2[9]（Top-5错误率4.8%）、Inception V3[15]（Top-5错误率3.5%）和Inception V4[17]（Top-5错误率3.08%）。Inception V2借鉴了VGG-Net,用两个3×3的小卷积代替5×5的大卷积，此外还提出了Batch Normalization方法。而Inception V3则是引入了Factorization into small convolutions的思想，将一个较大的二维卷积分解成两个较小的一维卷积，比如将3×3卷积分解成1×3卷积和3×1卷积，或者将7×7卷积分解成1×7卷积和7×1卷积，并优化了Inception Module的结构。而Inception V4则是借鉴了微软的ResNet，提出了一个更深更优化的模型。

2.3.5 ResNet

ResNet[16]（Residual Neural Network）是2015年ILSVRC挑战赛的冠军（Top-5错误率3.57%），由微软亚洲研究院的Kaiming He等提出。我们知道层数和深度越深的网络，特征种类越丰富，分类效果越好，但在不断加深网络的层数和深度时，会出现degradation现象，直观的表现即随着层数和深度的不断加深，训练集和测试集的正确率开始饱和，甚至出现下降。ResNet主要采用了一种Residual Unit结构，它消除了degradation现象，成功训练了152层深的深度卷积神经网络。

ResNet采用的Residual Unit结构如图2.10所示。Residual Unit借鉴了Highway Network的思想，即原始输入信息可以不经过变换直接传输到后面的层中。假设某一段神经网络的输入是 *x* ，经过网络层的变换后输出为 *F*(*x*) ，而使用Residual Unit的全等映射后输出就变为 *F*(*x*) + *x* 。假设这一段网络的期望输出是 *H*(*x*) ，在没有使用Residual Unit时学习目标是 *F*(*x*) = *H*(*x*) , 而使用了Residual Unit后学习目标就变成了 *F*(*x*) = *H*(*x*) – *x* ，即残差。

图2.10 Residual Unit结构

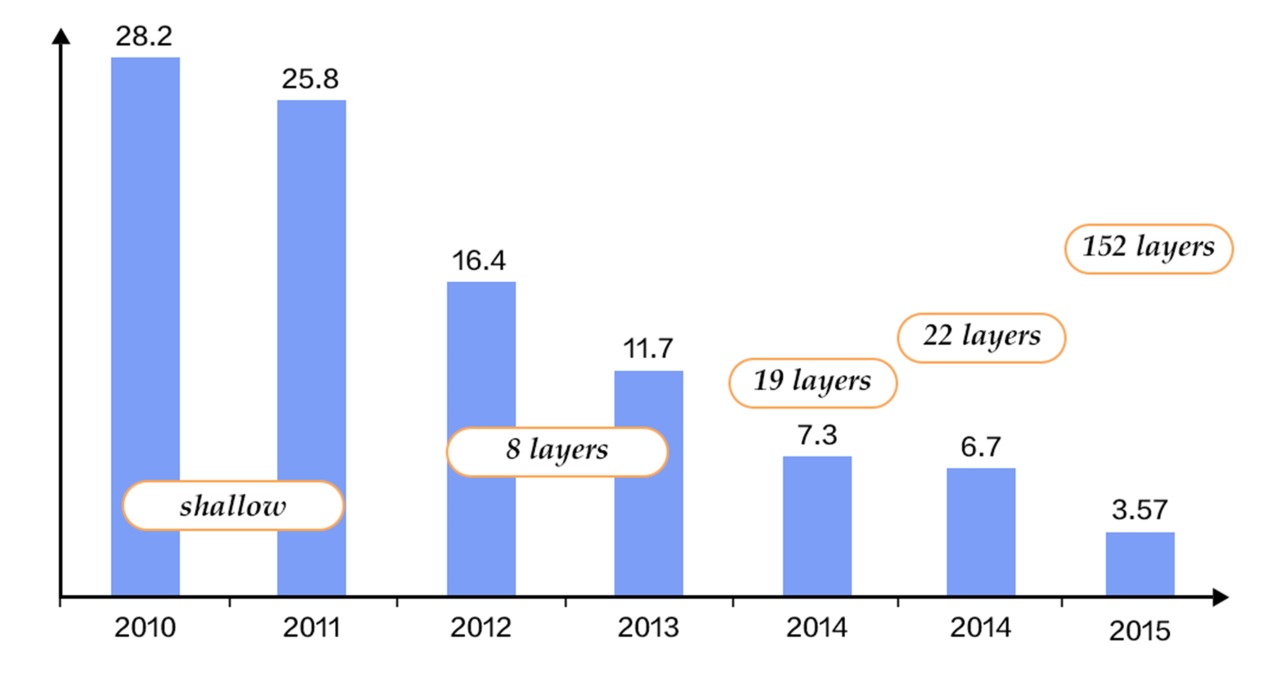
ResNet通过Residual Unit直接将输入信息绕道传到输出，使得整个网络只需要学习目标输出和输入的差别（残差）那一部分，简化了学习目标和难度，使得超深卷积神经网络的构建和训练变成了可能。ResNet的Residual Unit在某种程度上还保护了信息传输的完整性，解决了以往的网络层在信息传递时或多或少存在的信息丢失、损耗等问题。

ResNet的拓展性非常好，在ResNet提出后不久，谷歌就借鉴了ResNet的精髓Residual Unit，提出了Inception V4和Inception-ResNet-V2[17]，并通过融合这两个模型，在ILSVRC数据集上取得了3.08%的Top-5错误率。ResNet可以说是神经网络中里程碑式的突破，它的提出真正意义上支持了超深神经网络的构建和训练。

2.4 轻量级卷积神经网络

2.4.1 MobileNet-V1

如图2.11所示，从2012年的冠军AlexNet（8层）、2013年的冠军ZF-Net（8层）、2014年的亚军VGG-Net（19 层）、2014年的冠军GoogLeNet（22层）到 2015年的冠军ResNet（152层），每一年ILSVRC挑战赛的总趋势是使用更深层、更复杂的网络来实现更高的准确率。然而，这些准确率的提升并不一定会使网络在尺寸和速度方面更有效率。此外，在许多现实应用场合中，例如AR技术、机器人和自动驾驶汽车，识别任务需要及时地在计算资源有限的平台上进行。



NEC-UIUC XRCE AlexNet ZF-Net VGG-Net GoogLeNet ResNet

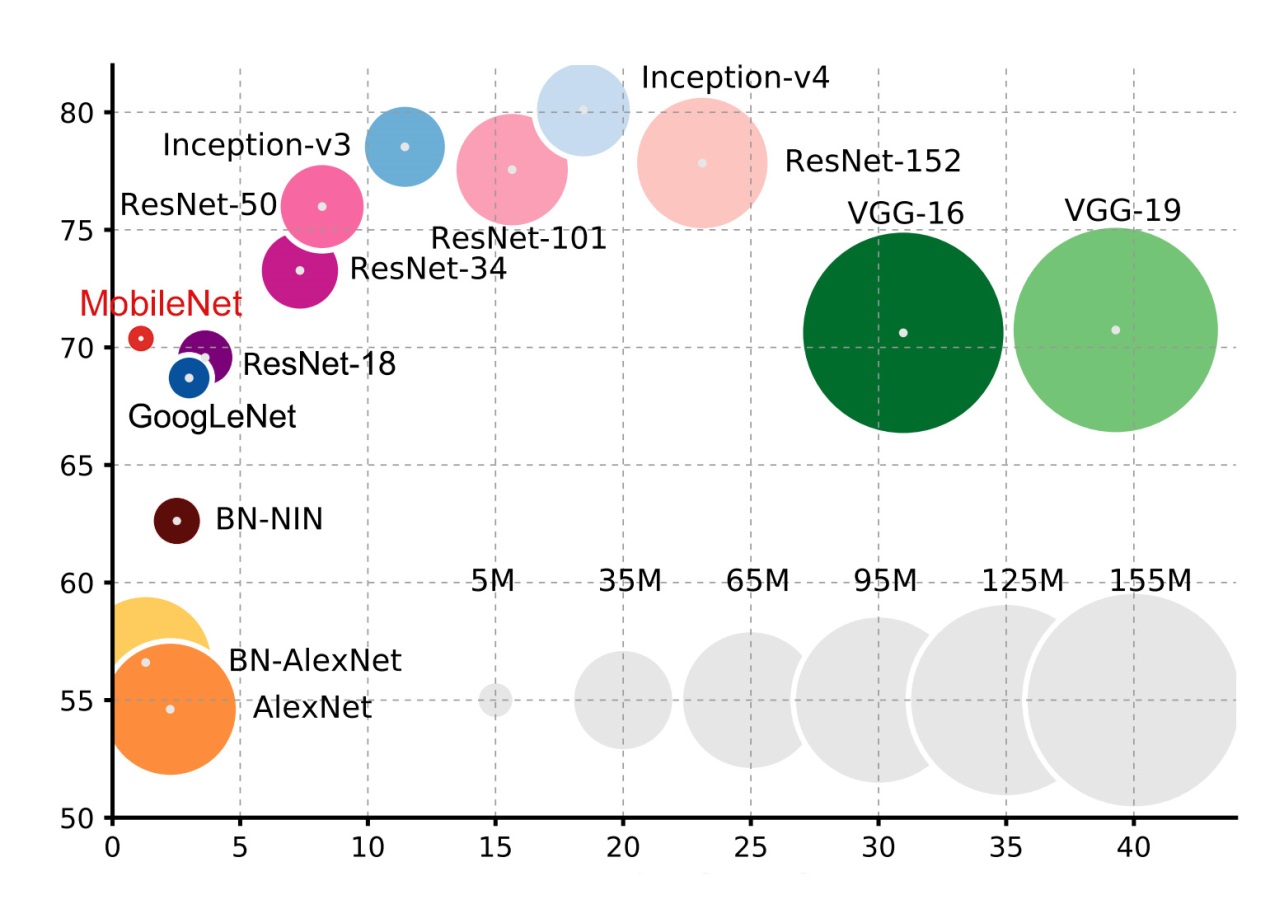
Top-5 错误率（%）

年份

图2.11 历届ILSVRC挑战赛代表性模型的成绩及其网络深度

针对上述情况，谷歌于2017年提出了一种专门为移动设备和嵌入式平台设计的轻量级CNN MobileNet-V1[18]。MobileNet-V1作为一种小型、低延迟且准确率较高的CNN，非常适合部署在类似手机这种计算资源有限的平台。图2.12为MobileNet-V1与最近几年ILSVRC挑战赛各网络架构的Top-1正确率，浮点数乘加运算总次数和网络参数数量的对比情况，其中气泡大小与网络参数数量成正比。可以看出MobileNet-V1在尺寸、速度和精度方面取得了较好的权衡，是一种优秀的轻量级CNN模型。

图2.12 Top-1正确率vs浮点数乘加运算总次数，模型大小∝参数数量

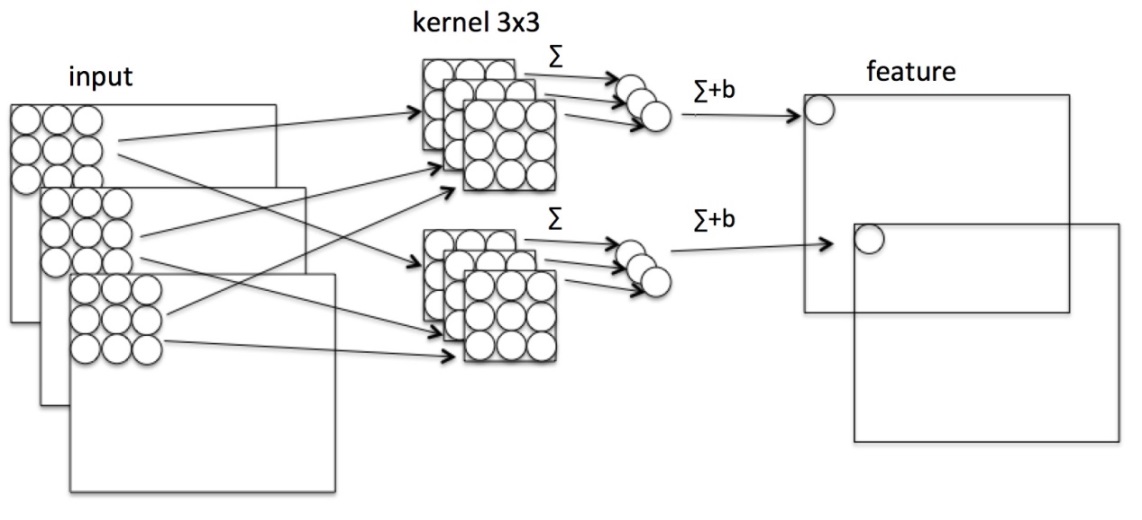


浮点数乘加运算总次数（G- FLOPs）

Top-1正确率（%）

2.4.2 标准卷积与深度可分解卷积的复杂度分析

MobileNet系列主要采用深度可分解卷积[18]（depthwise separable convolutions），将标准化卷积分解为深度卷积（depthwise convolution）和1×1卷积（pointwise convolution）来有效减少计算量。深度可分解卷积的示意图如图2.14、图2.15所示：

图2.13 标准卷积示意图

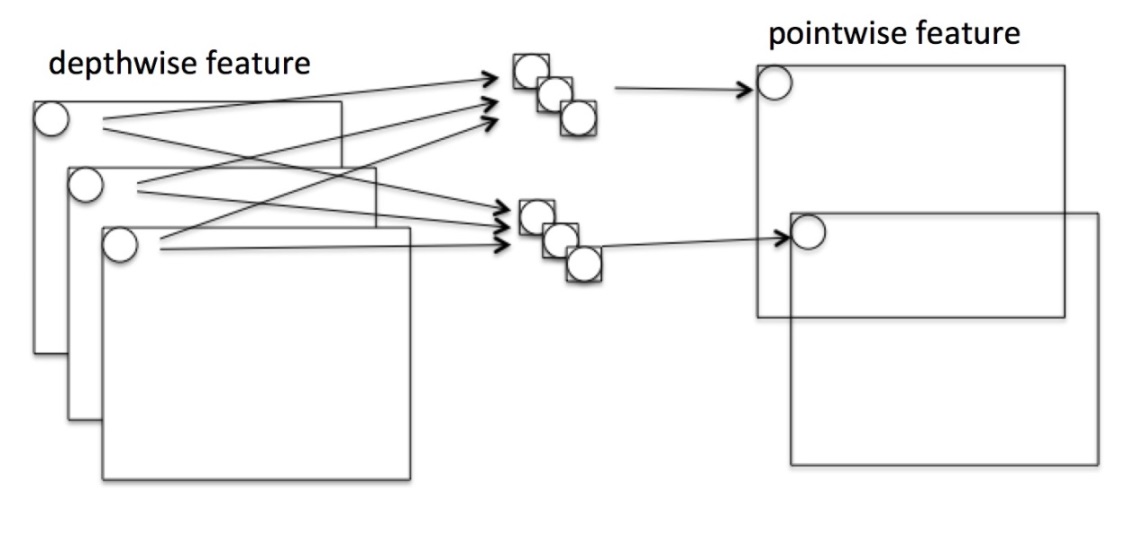
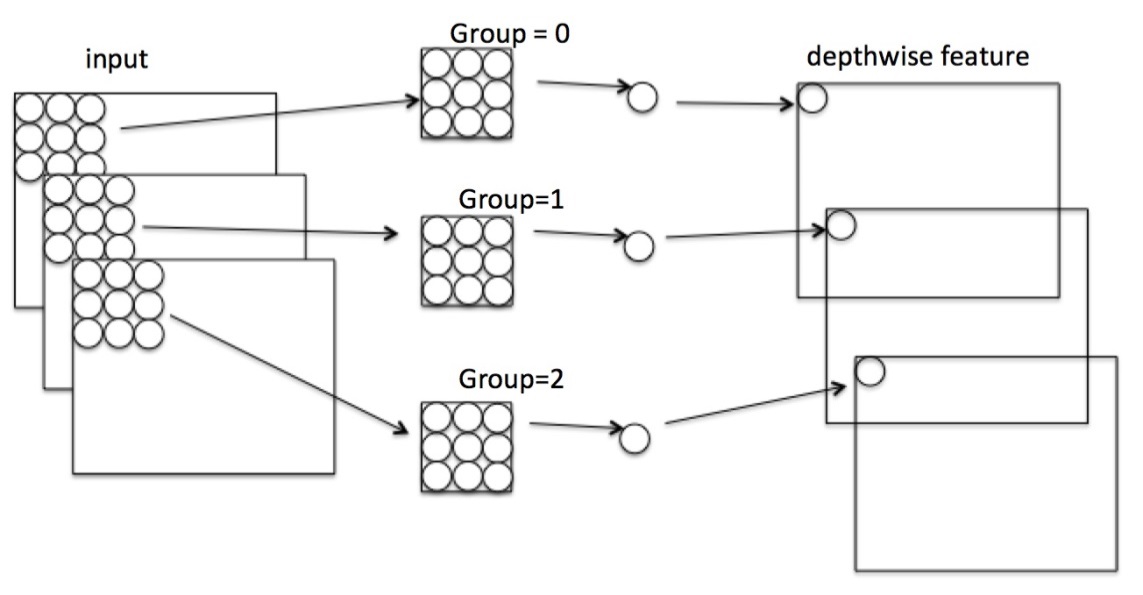
图2.14 深度卷积示意图

图2.15 1×1卷积示意图

在CNN中，我们通常用模型的浮点数的乘加运算总次数（FLOPs）来衡量时间复杂度，用模型的参数个数来衡量空间复杂度。假设一个标准卷积层的输入特征图谱 *F* 的维度为 *DF*×*DF*×*M* ，输出特征图谱 *G* 的维度是 *DG*×*DG*×*N* （其中 *DF* 是正方形输入特征图谱的宽和高， *M* 是输入通道的数量， *DG* 是正方形输出特征图谱的宽和高， *N* 是输出通道的数量），卷积核 *K* 的维度为 *DK*×*DK*×*M*×*N*（*DK* 是正方形卷积核的宽和高）。

我们可以算出标准卷积的时间复杂度为：

|  |  |  |
| --- | --- | --- |
|  |  | （2.4） |

标准卷积的空间复杂度为：

|  |  |  |
| --- | --- | --- |
|  |  | （2.5） |

而深度可分解卷积则将标准卷积分成两层，一层用于滤波，将单个滤波器应用到每一个输入通道；一层用于组合，用1×1卷积来组合不同深度卷积的输出。

深度可分解卷积的时间复杂度为：

|  |  |  |
| --- | --- | --- |
|  |  | （2.6） |

深度可分解卷积的空间复杂度为：

|  |  |  |
| --- | --- | --- |
|  |  | （2.7） |

深度可分解卷积和标准卷积的时间复杂度之比为：

|  |  |  |
| --- | --- | --- |
|  |  | （2.8） |

深度可分解卷积和标准卷积的空间复杂度之比为：

|  |  |  |
| --- | --- | --- |
|  |  | （2.9） |

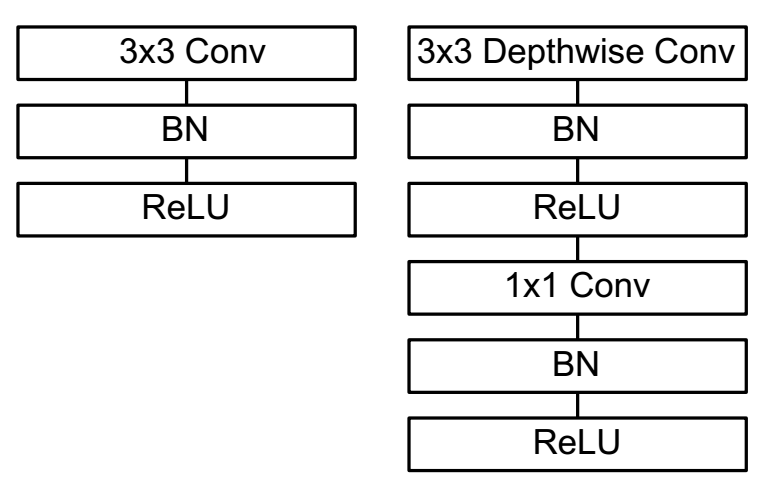
MobileNet使用了大量的3×3的深度可分解卷积，与标准的卷积相比减少了8到9倍的计算复杂度。MobileNet在1×1卷积花费了95%计算复杂度，而1×1卷积不需要在内存中重新排序，可以直接通过高度优化的通用矩阵乘法（GEMM，最优化的数值线性代数算法之一）功能来实现，进一步缩短了运行时间。此外，将将标准化卷积分解为深度卷积（depthwise convolution）和1×1卷积（pointwise convolution）后，网络可以增加一层Batch Normalization层和一层ReLU激活函数层，如图2.16所示，多层Batch Normalization层ReLU激活函数层通常可以加强卷积神经网络的特征学习能力。

图2.16 通过深度可分解卷积网络额外增加一层Batch Normalization层和ReLU激活函数层

2.4.3 MobileNet-V2

MobileNet-V2[8]是MobileNet-V1的改进版本。MobileNet-V2同样采用了深度可分解卷积来代替传统的卷积操作，实现空间和通道之间的解耦，达到模型加速的目的。MobileNet-V2同时也借鉴了ResNet的Residual Unit来增强模型的特征表达能力。MobileNet-V2主要的改进有两点：Linear Bottleneck和Inverted Residual block。接下来将分别介绍这两部分内容。

2.4.3.1 Linear Bottleneck

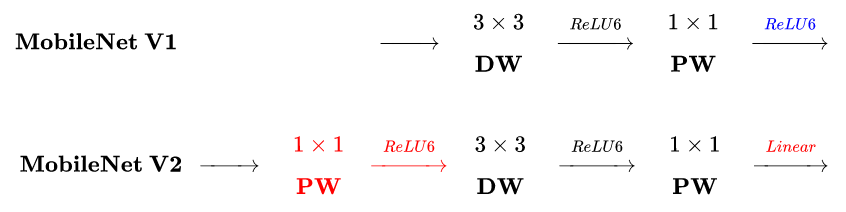
MobileNet-V2与MobileNet-V1的微结构对比如图2.17所示。两者的相同点为：都采用了depthwise (DW) 卷积搭配 pointwise (PW) 卷积的方式（即深度可分解卷积）来提取特征。该方式同时也成倍地减少了卷积层的时间复杂度和空间复杂度。

图2.17 MobileNet-V2与MobileNet-V1的微结构对比

MobileNet-V2与MobileNet-V1的不同点在于：MobileNet-V2在DW卷积之前新加了一个PW卷积。这么做的原因在于DW卷积自身并没有改变通道数的能力，上一层给它多少通道，它就只能输出多少通道。所以如果上一层给的通道数很少的话，DW卷积也就只能在低维空间提特征，因此效果不够好。而MobileNet-V2则给每个DW卷积之前都配备了一个PW卷积，专门用来升维，这样不管输入通道数是多少，经过第一层PW卷积升维之后，DW卷积都是在相对的更高的维度上提取特征。

此外，MobileNet-V2去掉了第二个PW卷积的激活函数。这么做的原因，是因为激活函数在高维空间能够有效的增加非线性，而在低维空间时则会破坏特征，不如线性的效果好。由于第二个PW卷积的主要功能就是降维，因此降维之后就不宜再使用ReLU激活函数。

2.4.3.2 Inverted Residual Block

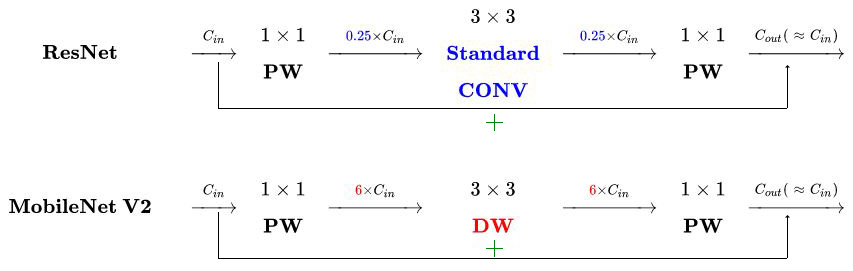
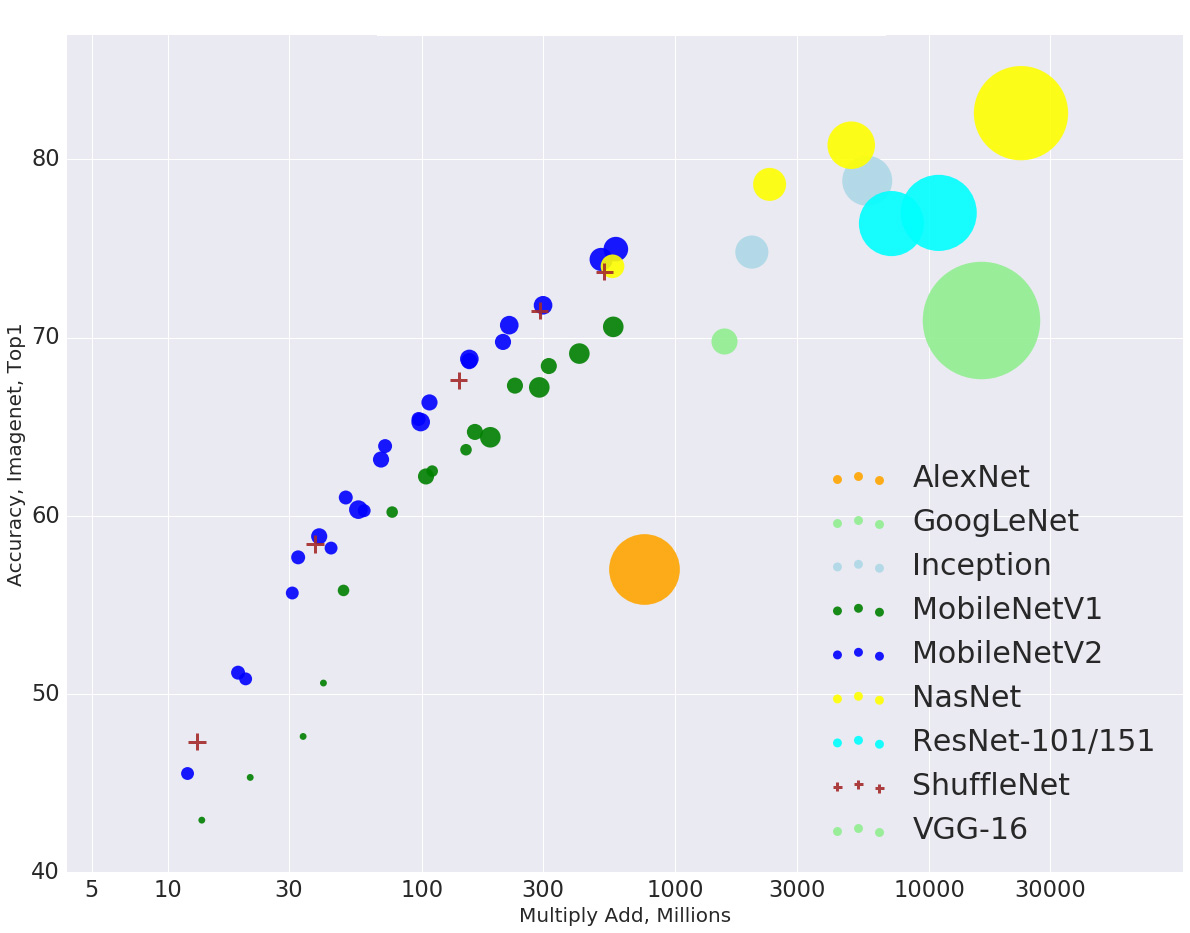
ResNet与MobileNet-V2的微结构对比如图2.18所示。两者的相同点为：都采用了1×1卷积→3×3卷积→1×1卷积的模式；都使用shortcut将输出与输入相加。

图2.18 ResNet与MobileNet-V2的微结构对比如图

ResNet与MobileNet-V2的不同点在于：ResNet使用标准卷积提取特征，MobileNet-V2始终使用DW卷积提特征；ResNet先降维（0.25倍）、卷积、再升维，而MobileNet-V2则是先升维（6倍）、卷积、再降维。直观的形象上来看，ResNet的微结构是沙漏形，而MobileNet-V2则是纺锤形，刚好相反，因此将 MobileNet-V2的这种结构称为Inverted Residual Block。这么做也是因为使用DW卷积而作的适配，希望特征提取能够在高维进行。

2.4.3.3 与主流CNN的对比

图2.19为MobileNet-V2与主流CNN在ImagetNet数据集上的Top-1正确率、浮点数乘加运算总次数和网络参数数量的对比情况，其中气泡大小与网络参数数量成正比。可以看出MobileNet-V2在尺寸、速度和精度方面取得了更好的权衡，比MobileNet-V1更加优秀。

图2.19 MobileNet-V2与主流CNN的对比图

2.5 本章小结

本章介绍了神经网络的前向传播的相关知识，包括人工神经网络、卷积神经网络和经典卷积网络的架构。人工神经网络中最普遍的是全连接神经网络，而卷积神经网络是在全连接神经网络的基础上发展而来的，它主要由卷积层、池化层和全连接层等各种网络层交替组合而成。经典的卷积神经网络有LeNet-5、AlexNet、VGG-Net、GoogLeNet和ResNet等，它们都在各自的年代展现了独特的创新设计，并启发了后续更多的技术创新。此外本章还介绍了专门为移动设备和嵌入式平台设计的轻量级卷积神经网络MobileNet-V1和MobileNet-V2。这些优秀的网络结构对于我们设计一个优秀的卷积神经网络模型具有良好的借鉴意义。

第3章 深度神经网络的反向训练

本章主要介绍深度神经网络的训练（training）过程。深度神经网络属于参数学习算法，参数学习算法的特点是使用之前要先对网络进行训练来寻找最优参数（俗称调参）。深度神经网络的一个特点是通过多个隐函层的方法来实现非线性的函数。多层非线性隐含层的好处是理论上可以模拟任何数据分布，但同时也带来了参数增多以及参数难以学习的问题。本章围绕深度神经网络的训练展开介绍，首先介绍了特征学习方面的一些理论知识，然后介绍了深度神经网络的具体训练流程，紧接着介绍了训练过程中的过拟合现象和如何解决过拟合，以及加快训练速度的一些常用方法。

3.1 特征学习

3.1.1 卷积核的作用

CNN的核心其实就是卷积核的作用。比如我们想做一个最简单的方向滤波器，那就是一个二维卷积核。这个核其实就是一个模板，利用这个模板再通过卷积计算的定义就可以计算出一幅新的图像。新的图像会把这个卷积核所体现的特征突出显示出来，比如这个卷积核可以侦测水平纹理，那卷积出来的图就是原图水平纹理的图像。

假设现在要处理一个图像分类问题，比如辨别一张图像里是否有一辆汽车，我们可以先判断是否有汽车的车身，汽车的车窗、汽车的车轮等等，如果这些特征都具备，那么我就判定这应该是一辆汽车（这一部分其实就是CNN最后的分类层，属于传统的神经网络的范畴）。关键在于这些特征是高级的语义特征，这种特征怎么用卷积核提取呢？

原来的卷积核都是人工事先定义好的，是经过算法设计人员精心设计的。他们往往是根据经验，发现这样或那样的设计卷积核通过卷积运算可以突出一个什么样的特征，于是便拿去做卷积了。但是现在我们所需要的这种特征太高级了，而且随任务的不同而不同，人工设计这样的卷积核非常困难。于是，我们可以采用端到端的机器学习方式，让网络自己去学习出卷积核来，也就是通过样本训练，让网络对特征进行自动提取和抽象。

如前所述，判断是否是一辆汽车，只有一个特征不够，比如仅仅有汽车的车身是不够的，我们需要多个高级语义特征的组合，比如汽车的车窗、汽车的车轮等等，所以我们需要多个卷积核来提取特征，让每个卷积核提取一类语义特征。每个卷积核卷积后会计算出一幅新的图像，我们把所有的卷积核计算出的图像在深度这个维度上拼接后便得到了所谓的特征图（feature map）。

3.1.2 多层特征学习

为什么CNN要设计这么多层呢？还是汽车识别的问题，我们很自然地想到要用汽车的车身、汽车的车窗、汽车的车轮这些高阶特征来对图像进行分类。但是对于充斥着像素点的图像来说，用几个卷积核直接提取汽车的车身、车窗、车轮这些高阶特征还是太困难，怎么办？很简单，任何高阶特征都是由底层特征组合而成的，比如车轮橡胶轮胎由许多黑色的同心圆组成，而这些同心圆也由许多圆弧曲线组成，圆弧曲线都由像素组成。我们将前面的过程逆过来，将一张图片的原始像素慢慢抽象，从像素组成点、线，再将点、线组合成小零件，再将小零件组成车轮、车窗、车身等高阶特征，这样就构成了多层的深度卷积神经网络。

深度神经网络在训练过程中可以对特征逐层抽象提取，先从简单的微观的特征开始，随着层数的增加不断抽象特征的层级，直到最后复杂的宏观特征。这一点类似人的学习过程，先理解简单概念，再逐渐递进到复杂概念。深度神经网络可以一层一层地将简单特征逐步转化成更加复杂的特征，从而使得不同类别的图像更加可分。

多层复杂特征提取

基础特征提取

输入

权重学习

预测结果

图3.1 深度神经网络的学习流程

3.2 深度神经网络的训练流程

3.2.1 损失函数

训练一个神经网络时，会定义一个损失函数(loss function)，用来量化输出值与真实值之间的一致性。对于分类问题，通常采用交叉熵(cross entropy)，对于回归问题（具体数值的预测，比如房价预测、销售预测等），通常采用均方误差(mean squared error)。

假设神经网络的输出值为 ，所对应的真实值为 ，则交叉熵的表达式为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.1） |

均方误差的表达式为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.2） |

损失函数衡量了我们对结果的不满意程度。模型输出值与真实值之间差异越大，则损失函数越大，反之越小。训练神经网络的目标是找出能最小化损失函数的参数。一个神经网络通常有上百万个参数，目前还没有一个可以对任意损失函数直接求解最佳参数值的通用的方法，因此在实践中，梯度下降(gradient decent)算法是训练神经网络最常用的方法。

3.2.2 梯度下降算法

假设用 *θ* 表示神经网络中的参数，用 *L*(*θ* ) 表示在给定的参数取值下损失函数的大小，假设 *θ* 移动一个小量 Δ*θ* ，则由微分法可知 *L*(*θ* ) 的变化为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.3） |

我们将设法选择 Δ*θ* 以确保 Δ*L*(*θ* )为负数，如果我们选择

|  |  |  |
| --- | --- | --- |
|  |  | （3.4） |

这里 *η* 是一个正的小参数，称为学习率，这样表达式3.3将化为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.5） |

由此可知在表达式3.3成立的范围内（即 Δ*θ* 足够小）， Δ*L*(*θ* ) 的值会一直减少，不会增加，直到参数 *θ* 的梯度为0，此时参数 *θ* 将达到局部最优解，不再改变。参数 *θ* 的更新公式为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.6） |

将 Δ*θ* 用表达式3.4代入，最终梯度下降算法的参数更新公式为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.7） |

采用梯度下降算法训练一个神经网络时，算法会迭代更新参数 *θ* ，让参数 *θ*不断沿着梯度的反方向更新，使总损失值 *L*(*θ* ) 更小，而学习率 *η* 则是我们需要手工设置的超参数，它控制着参数更新的幅度。

神经网络的学习算法一般是梯度下降法，梯度下降法需要用到损失函数*L*关于网络所有参数 *θ* 1、*θ* 2、… *θ n*的梯度（偏导数） 。若采用直接计算网络梯度的方法，即：

|  |  |  |
| --- | --- | --- |
|  |  | （3.8） |

其中 是一个非常小的正数， 是*i* 方向上的单位向量。若采用该方法计算神经网络的梯度，运算量会很大。假设网络中有一百万个参数，那么对于每一个参数 ，我们需要计算 来计算 。这意味着我们需要计算一百万次 来算出 的梯度。此外我们还需要计算 *L* ，网络总共需要进行一百万零一次前向传播，计算开销非常大。因此我们需要一种高效的算法来快速计算网络的梯度。反向传播算法[19]（backpropagation）是一种高效计算梯度的算法。

3.2.3 反向传播算法

如图3.2所示，假设用 来表示从网络第 层中第 *k*个神经元指向第 *l* 层中第 *j* 个神经元的连接权重，比如 表示从第二层中第四个神经元指向第三层中第二个神经元的权重。假设用 来表示第 *l* 层中第 *j* 个神经元的偏差，用 来表示第 *l* 层中第 *j* 个神经元的激活输出。

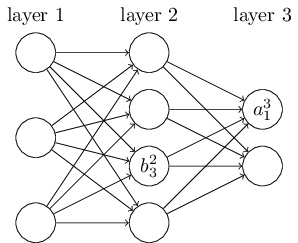
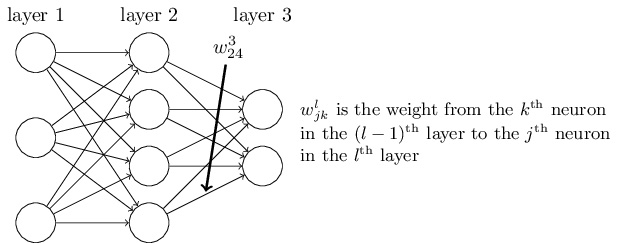


图3.2 神经网络符号示意图

假设激活函数为 *σ* ，则第 *l* 层中第 *j* 个神经元的输出为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.9） |

令 ，我们将 命名为第 *l* 层中第 *j* 个神经元的加权输入，则表达式3.9可化简为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.10） |

假设网络的损失函数为 *C* ，定义损失函数 *C* 对第 *l* 层中第 *j* 个神经元的偏导为 ，即：

|  |  |  |
| --- | --- | --- |
|  |  | （3.11） |

对于网络最后一层（*L*层）的第 *j* 个神经元，偏导可得BP1的元素级表达式：

|  |  |  |
| --- | --- | --- |
|  |  | （3.12） |

向量化可得BP1的表达式：

|  |  |  |
| --- | --- | --- |
|  |  | （3.13） |

其中⊙为Hadamard乘积，操作为把两个向量对应元素相乘组成新的元素。

对于网络第 *l* 层的第 *j* 个神经元，采用多元微积分的链式求导法则可得：

|  |  |  |
| --- | --- | --- |
|  |  | （3.14） |

对于 ：

|  |  |  |
| --- | --- | --- |
|  |  | （3.15） |

对 进行偏导可得：

|  |  |  |
| --- | --- | --- |
|  |  | （3.16） |

代回3.14可得BP2的元素级表达式：

|  |  |  |
| --- | --- | --- |
|  |  | （3.17） |

向量化可得BP2的表达式：

|  |  |  |
| --- | --- | --- |
|  |  | （3.18） |

对 *C* 关于 求偏导可得BP3的元素级表达式：

|  |  |  |
| --- | --- | --- |
|  |  | （3.19） |

向量化可得BP3的表达式：

|  |  |  |
| --- | --- | --- |
|  |  | （3.20） |

对 *C* 关于 求偏导可得BP4的元素级表达式：

|  |  |  |
| --- | --- | --- |
|  |  | （3.21） |

向量化可得BP4的表达式：

|  |  |  |
| --- | --- | --- |
|  |  | （3.22） |

综上所述，反向传播算法的四个基础等式为：

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | （BP1） | （3.23） |
|  |  | （BP2） |
|  |  | （BP3） |
|  |  | （BP4） |

反向传播的四个基础等式给我们提供了一种高效计算损失函数梯度的方法，算法3-1展示了反向传播算法的计算流程：

|  |
| --- |
| **算法3-1** 反向传播算法的计算流程 |
| 1. 输入*x*：为输入层设置对应的激活 ***a*1**。 2. 前向传播：对于每一层 *l* = 2，3，…，*L* ，计算 和 。 3. 输出层误差 ：计算向量  **。** 4. 反向传播误差：对于每一层 *l* = *L* -1，*L* -2，…，2，计算 。 5. 输出：计算损失函数关于每一个参数的梯度 和 。 |

反向传播算法的高效在于它只需要网络向前计算一次，向后计算一次。它允许我们同时计算所有参数的梯度。相比表达式3.8直接计算梯度的方法，反向传播算法可以极大地减少运算量。

3.2.4 改进的优化算法

3.2.4.1 BGD

梯度下降算法要在全部训练数据上最小化损失，所以损失函数 *L* 是在所有训练数据上的损失和。然而在每一轮迭代中，计算全部训练数据的梯度是非常耗时的，为了加速训练过程，实践中常常使用批量的梯度下降法（Batch Gradient Descent，简称BGD）。

BGD算法每轮从训练数据中随机抽取*m*个小批量，通过计算它们的梯度均值来更新参数。算法3-2展示了BGD每轮迭代训练的参数更新流程：

|  |
| --- |
| **算法3-2** 批量梯度下降法（BGD）在第*k*轮迭代训练的更新 |
| **Require：**学习率 *η*  **Require：**初始参数 *θ*  **while** 停止条件未满足 **do**  从训练集中随机抽取 *m* 个样本 的小批量，其中 对应的真实值为 。  计算梯度：  应用更新：  **end while** |

3.2.4.2 Momentum

BGD 算法的训练过程有时会很慢，而Momentum算法是一种加快训练速度的方法。Momentum算法来自物理类比，根据牛顿运动定律，负梯度是在参数空间中移动粒子的力， Momentum在物理中的定义为质量乘以速度。在Momentum算法中，我们假设粒子是单位质量，因此速度 *v* 可视为粒子的Momentum，超参数 决定了之前Momentum的衰减率。

算法3-3展示了使用Momentum的批量梯度下降的参数更新流程：

|  |
| --- |
| **算法3-3** 使用Momentum的批量梯度下降（BGD） |
| **Require：**学习率 *η* ，动量系数 *α*  **Require：**初始参数 *θ* ，初始速度 *v*  **while** 停止条件未满足 **do**  从训练集中随机抽取 *m* 个样本 的小批量，其中 对应的真实值为 。  计算梯度：  计算速度更新：  应用更新：  **end while** |

其中超参数 一般设置为0.5、0.9或者0.99。

3.2.4.3 AdaGrad

BGD算法中，对于每一个参数的学习率都是相同的，这种做法并不合理。试想，如果训练数据是稀疏的，而且特征数据出现的频率变化很大，那么比较合理的做法是，对于出现频率低的特征参数使用较大的学习速率，对于出现频率较高的特征参数使用较小的学习速率。

AdaGrad算法在每一步迭代训练过程中，每个参数都会累加各自的历史平方梯度，并使用当前累加的平方梯度去除以当前的学习速率。净效果是，具有较大历史梯度的参数有一个较小的学习率，具有较小历史梯度的参数有一个较大的学习率。AdaGrad算法对于训练数据比较稀疏的情况比较适用，可以提高梯度下降法的鲁棒性。

算法3-4展示了AdaGrad每轮迭代训练的参数更新流程：

|  |
| --- |
| **算法3-4** AdaGrad算法 |
| **Require：**学习率 *η*  **Require：**初始参数 *θ*  **Require：**小常数 ，为了数值稳定大约设为10-7  初始化梯度累加变量 *r* = 0  **while** 停止条件未满足 **do**  从训练集中随机抽取 *m* 个样本 的小批量，其中 对应的真 实值为 。  计算梯度：。  累加平方梯度：。  计算更新：。  应用更新：。  **end while** |

AdaGrad算法在参数空间中更为平缓的倾斜方向会取得更大的进步，缺点是学习率的调整太激进，常常过早结束了训练过程。

3.2.4.4 RMSProp

RMSProp算法对AdaGrad算法稍作改进, 使得参数更新过程不再像 AdaGrad那么激进。RMSProp使用滑动平均平方梯度（moving average of squared gradients），取代了AdaGrad的累加平方梯度。

算法3-5展示了RMSProp每轮迭代训练的参数更新流程：

|  |
| --- |
| **算法3-5** RMSProp算法 |
| **Require：**学习率 *η* ，衰减速率 *ρ*  **Require：**初始参数 *θ*  **Require：**小常数 ，为了数值稳定大约设为10-6  初始化梯度累加变量 *r* = 0  **while** 停止条件未满足 **do**  从训练集中随机抽取 *m* 个样本 的小批量，其中 对应的真实值为 。  计算梯度：  累加平方梯度：  计算更新：  应用更新：  **end while** |

当RMSProp算法结合Momentum之后，可大大加快训练速度。算法3-6展示了结合Momentum的RMSProp每轮迭代训练的参数更新流程：

|  |
| --- |
| **算法3-6** 结合Momentum的RMSProp算法 |
| **Require：**学习率 *η* ，衰减速率 *ρ* ，动量系数 *α*  **Require：**初始参数 *θ* ，初始速度 *v*  **Require：**小常数 ，为了数值稳定大约设为10-6  初始化梯度累加变量 *r* = 0  **while** 停止条件未满足 **do**  从训练集中随机抽取 *m* 个样本 的小批量，其中 对应的真实值为  计算梯度：  累加平方梯度：  计算速度更新：  应用更新：  **end while** |

3.2.4.5 Adam

Adam算法是另一种自适应学习率的优化算法，来源于适应性矩估计（adaptive moment estimation）。Adam算法通过计算每个参数梯度的一阶矩估计 *m* 和二阶矩估计 *v* ，为每个参数设计独立的自适应性学习率。

算法3-7展示了Adam每轮迭代训练的参数更新流程：

|  |
| --- |
| **算法3-7** Adam算法 |
| **Require：**学习率 *η*  **Require：**矩估计的指数衰减速率 *β*1 和 *β*2 （*β*1一般设为0.9，*β*2一般设为0.999）  **Require：**初始参数 *θ*  **Require：**用于数值稳定的小常数 （一般设为10-6）  初始化一阶和二阶矩估计变量*s* = 0，*r* = 0  初始化时间步 *t* = 0  **while** 停止条件未满足 **do**  从训练集中随机抽取 *m* 个样本 的小批量，其中 对应的真实值为 。  计算梯度：  计算时间步更新：  更新有偏一阶矩估计：  更新有偏二阶矩估计：  修正一阶矩估计的偏差：  修正二阶矩估计的偏差：  计算更新：  应用更新：  **end while** |

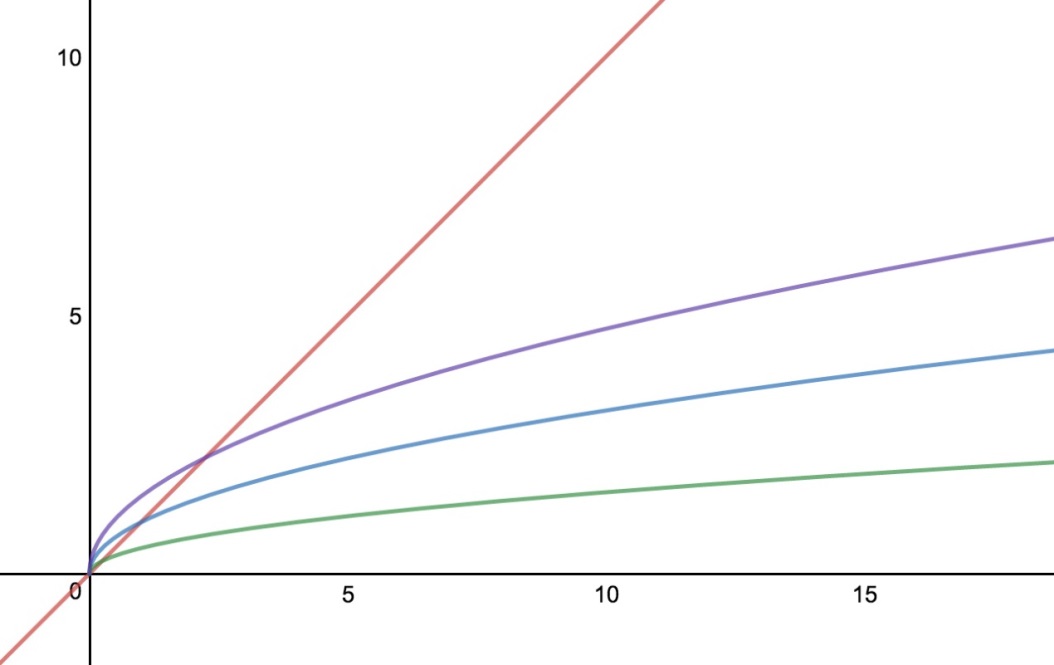
3.2.5 训练流程概述

总的来说深度神经网络的训练流程总共分两步，第一步先通过前向传播得到预测值，并将预测值和真实值做对比得出两者之间差距，第二步通过反向传播算法计算损失函数对每一个参数的梯度，然后根据梯度和学习率使用相应的优化算法更新每一个参数。常用的优化算法有BGD、结合Momentum的BGD、AdaGrad、RMSProp、结合Momentum的RMSProp和Adam等。由没有免费午餐定理可知，并不存在一种最优的优化算法，选择哪一种优化算法通常取决于具体的神经网络类型和实际的训练数据。

3.3 防止过拟合

3.3.1 大数据算法与过拟合现象

深度学习非常看重数据量的大小。深度学习是一种大数据算法，它的一个特点是对海量数据的利用效率非常高。如图3.3所示，Traditional Machine learning算法在获取了一定量的数据后，准确率上升曲线就接近瓶颈，而Neural Network则可以持续上升到更高的准确率才接近瓶颈。规模越大越复杂的Neural Network（从Small Neural Network、Medium Neural Network到Large Neural Network），可以达到的准确率也越高，但也需要更多的数据量才能训练好。可以看出，Large Neural Network在数据量较少的时候，效果并不比其它算法好，直到数据量持续扩大才慢慢超越了Traditional Machine learning、Small Neural Network和Medium Neural Network，并在最后达到一个非常高的准确率。



Large Neural Network

Medium Neural Network

Small Neural Network

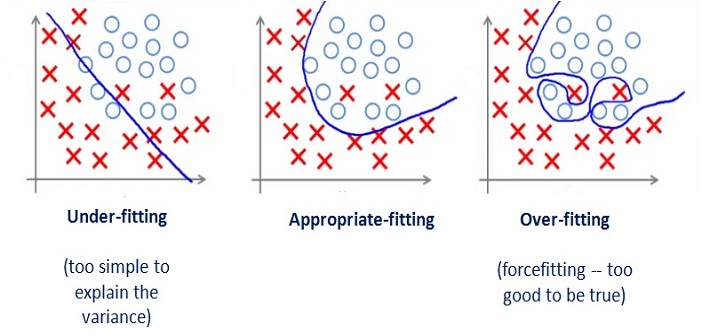
Traditional Machine learning

数据量

准确率

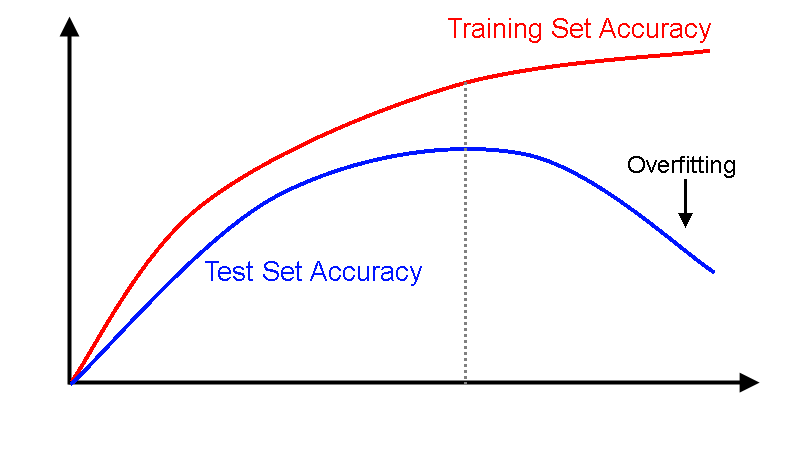
图3.3 传统机器学习算法和深度神经网络在不同数据量下的表现

在训练深度神经网络时，如果没有充分的训练数据，网络极易陷入过拟合中。所谓过拟合，指的是网络完整地记忆了每一个训练样本中的随机噪音部分，而没有去学习训练样本中的通用特征部分。如图3.4所示，第一种状态属于欠拟合（Under-fitting），网络没有学习出样本种类的划分趋势；第二种状态属于恰拟合（Appropriate-fitting），网络既不会过于关注样本中的噪音部分，又能够比较好地对样本种类进行划分；第三种状态属于过拟合（Over-fitting），虽然网络完美地划分了不同形状的点，但这样的划分并不能对未知样本的分布规律做出判断，因为它过度拟合了样本中的噪音部分而忽视了问题的整体规律。对于深度神经网络来说，如果网络中的参数比训练数据的总数还多，那么该网络完全可以记住所有的训练数据从而使得损失函数为0。然而在实际应用中，我们想要的并不是让网络尽量学习训练数据，而是希望通过学习训练数据然后对未知的数据做出判断。过度拟合训练数据中的随机噪音虽然可以在训练集上表现得非常好，但在未知数据（测试集）上可能表现得很糟糕。

图3.4 深度神经网络训练的三种状况

过拟合在迭代训练过程中的直观表现如图3.5所示。在早期迭代训练中（虚线左边），网络逐步学习到训练样本中的通用特征部分，所以训练集和测试集的正确率逐步上升。然而随着迭代轮数的增加（虚线右边），由于训练样本较少，网络开始拟合训练数据中的随机噪音部分，因此虽然训练集上的正确率仍逐步上升，但测试集上的正确率却开始逐步下降。

图3.5 过拟合在迭代训练过程中的直观表现



迭代轮数

正确率

3.3.2 数据增强

训练深度卷积神经网络，必须拥有充分的训练数据才能有效避免过拟合，发挥深度学习的优势。然而在实际应用中，我们能够使用的标注数据通常是有限的。想要收集到额外的标注数据，通常需要耗费大量的人力、物力和财力，而且一些高质量的数据集的构建往往还需要相关的专业知识。为了解决训练数据的数据量的问题，本文采用的是数据增强(data augmentation)的方法。

数据扩增即给单张图片增加多个副本，通过对原始图片进行各种变换来得到更多的数据，防止深度神经网络过度学习某一张图片的随机噪声部分。常用的变换方法有：

1. 旋转/反射变换（Rotation/Reflection）：随机旋转图像的角度；随机改变图像内容的朝向。
2. 平移变换（shift）：在图像平面上对图像以一定范围和步长进行平移，改变图像内容的位置。
3. 尺度变换（scale）：对图像按照指定的尺度因子，进行放大或缩小。
4. 缩放变换（zoom）：按照一定的比例对图像进行放大或者缩小。
5. 翻转变换（flip）：沿着水平方向或者垂直方向翻转图像。
6. 颜色变换（color）：在图像的RGB颜色空间内进行PCA变换。
7. 对比度变换（contrast）：在图像的HSV颜色空间内，保持色调H不变，改变饱和度S分量和亮度V分量。
8. 噪声扰动（noise）：对图像的每个像素在RGB颜色空间内进行随机扰动，常用的噪声模式包括椒盐噪声和高斯噪声。
9. 裁剪（crop）：对图像的不同位置截取不同大小的图像块。

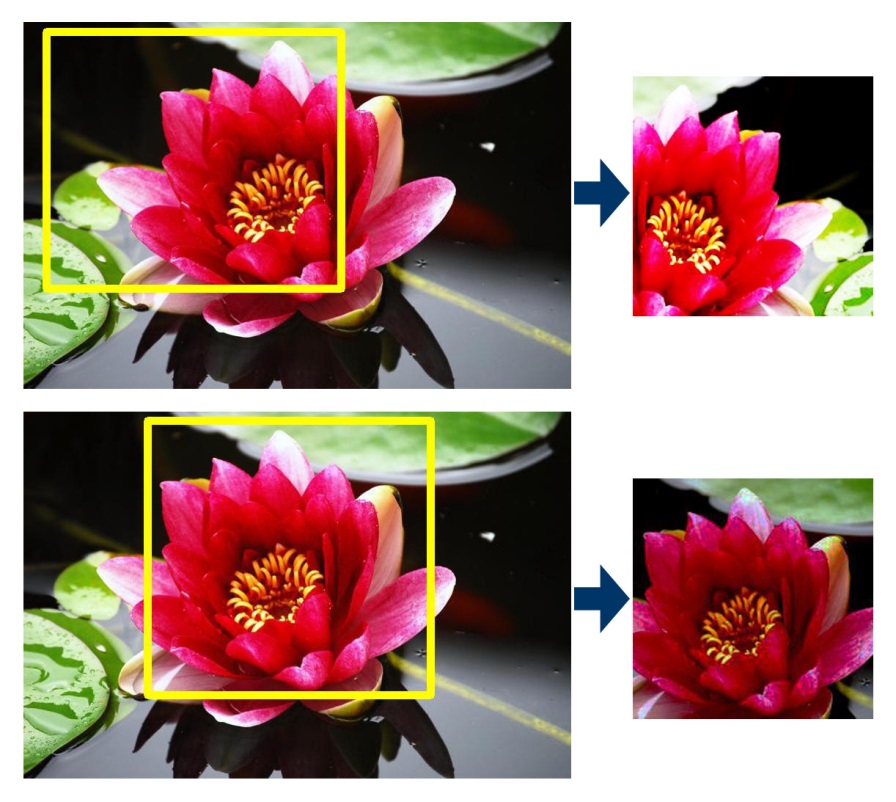
本文采用的数据增强方法如图3.6所示，包括对一张图片随机裁剪、随机的水平翻转、设置随机的亮度和对比度等。通过这些操作，一张图片变成多张图片，相当于拥有不同结构信息的训练样本，扩大了训练样本的多样性，大大提高了网络的泛化性能。

图3.6 数据增强示例

3.3.3 正则化

训练深度神经网络还经常使用正则化（regularization）来避免过拟合问题。正则化通过约束神经网络的参数的范数使其不要过大，以此降低网络的复杂度，从而减小噪声输入的扰动，具体的方法就是在损失函数中增加一个描述网络复杂程度的函数。假设网络的损失函数为 *L*0 ，我们在训练过程中并不是最小化 *L*0 ，而是最小化 *L* = *L*0 + *λR*(*w*) ，其中 *R*(*w*) 刻画的是网络的复杂程度，而 *λ* 表示正则化损失在总损失中的比例。常用的描述网络复杂程度的函数 *R*(*w*) 有两种，一种是L1正则化，一种是L2正则化。

3.3.3.1 L1正则化

L1正则化的表达式为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.24） |

网络的总损失函数的表达式为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.25） |

计算总损失函数对每一个参数的梯度：

|  |  |  |
| --- | --- | --- |
|  |  | （3.26） |

权重 的更新规则为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.27） |

比原始的更新规则多出了 这一项。因为 *η* 和 *λ* 为很小的正数，所以当 为正时，更新后的 变小；当 为负时，更新后的 变大。L1正则化的效果就是使 尽可能往0靠，使网络中的参数变得更稀疏（使更多的参数尽可能为0），相当于减小了网络的复杂度，防止过拟合。

3.3.3.2 L2正则化

L2正则化的表达式为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.28） |

网络的总损失函数的表达式为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.29） |

计算总损失函数对每一个参数的梯度：

|  |  |  |
| --- | --- | --- |
|  |  | （3.30） |

权重 的更新规则为：

|  |  |  |
| --- | --- | --- |
|  |  | （3.31） |

在不使用L2正则化时， 前的系数为1，使用L2正则化后， 前的系数为 。因为 *η* 和 *λ* 为很小的正数，所以 。L2正则化的效果是减小 ，这也就是权重衰减（weight decay）的由来。当 为正时，更新后的 变小；当 为负时，更新后的 变大，其效果也是使网络中的参数尽可能为0，减小网络的复杂度，防止过拟合。

3.3.4 dropout

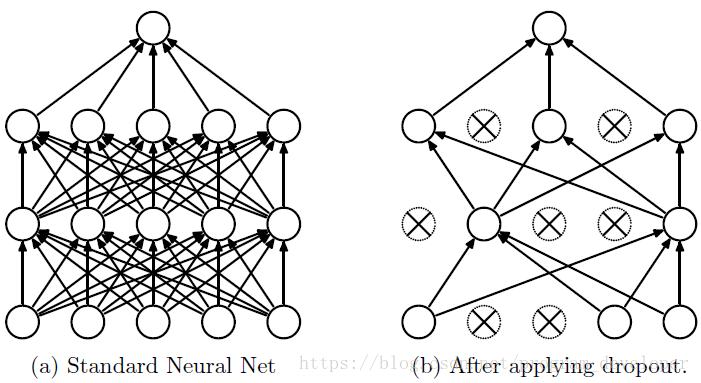
dropout是一种与L1和L2正则化完全不同的技术。dropout并不会修改损失函数，而是修改网络本身。dropout是指在深度神经网络的训练过程中，对于部分神经网络单元，按照一定的概率将其暂时从网络中丢弃（即令其输出值为0），如图3.7所示。

图3.7 dropout示意图。左边为标准神经网络，右边为应用dropout后的神经网络

Dropout能够避免过拟合的原因如下：

(1) 取平均的作用

先回到正常的模型（没有dropout），我们用相同的训练数据去训练5个不同的神经网络，一般会得到5个不同的结果，此时我们可以采用 “5个结果取均值”或者“多数取胜的投票策略”去决定最终结果。（例如 3个网络判断结果为数字9，那么很有可能真正的结果就是数字9，其它两个网络给出了错误结果）。这种“综合起来取平均”的策略通常可以有效防止过拟合问题。因为不同的网络可能产生不同的过拟合，取平均则有可能让一些“相反的”拟合互相抵消。dropout掉不同的隐含层神经元就类似在训练不同的网络（随机删掉一半隐藏神经元导致网络结构已经不同），整个dropout过程就相当于对很多个不同的神经网络取平均。不同的网络产生不同的过拟合，一些互为“反向”的拟合相互抵消就可以达到整体上减少过拟合。

(2) 减少神经元之间复杂的共适应关系

因为dropout导致两个神经元不一定每次都在一个dropout网络中出现，（这样权值的更新不再依赖于有固定关系的隐含节点的共同作用，阻止了某些特征仅仅在其它特定特征下才有效果的情况）。 迫使网络去学习更加鲁棒的特征（这些特征在其它的神经元的随机子集中也存在）。换句话说假如我们的神经网络是在做出某种预测，它不应该对一些特定的线索片段太过敏感，即使丢失特定的线索，它也应该可以从众多其它线索中学习一些共同的模式（鲁棒性）。

3.4 加快收敛速度

即便有高效计算梯度的反向传播算法和快速收敛的训练算法（如Adam、RMSProp等），深度神经网络的训练仍需很长的时间。本小节介绍了一些高效的训练技巧，它们能够加快深度神经网络的收敛速度，缩短训练时间。

3.4.1 迁移学习

在深度神经网络的训练中，采用参数随机初始化的方式从头开始训练一个CNN，即便有多路GPU做并行计算，收敛到一个较好的结果仍需好几天的时间（VGG-Net训练时使用了4块Geforce GTX Titan GPU做并行计算，耗时2~3周训练完）。因此在实际应用中，通常采用迁移学习(transfer learning)的方法来快速训练CNN。

迁移学习通俗来讲，就是运用已有的知识来学习新的知识，用成语来说就是举一反三。由于直接对目标域从头开始学习成本太高，我们故而转向运用已有的相关知识来辅助尽快地学习新知识。比如，已经会编写C语言，就可以类比着来学习C++；已经学会汉语，就可以类比着来学习日语；已经学会下中国象棋，就可以类比着来学习国际象棋等等。世间万事万物皆有共性，迁移学习的核心是构建出已有知识和新知识之间的桥梁，进而找到它们之间的相似性。

深度神经网络中的迁移学习，就是将一个数据集上训练好的网络通过简单的调整使其适用于一个新的数据集。目前一些流行的深度学习平台（Tensorflow、Caffe、Keras、CNTK和Theano等），通常会在它们的官网开源ImageNet数据集上训练好的CNN参数。由于ImageNet数据集中的1000种类别已基本包含日常生活中的常见物体，我们可以认为在ImageNet数据集上训练好的CNN已经提取了非常丰富的特征，具备很强的表达能力，因此在此基础上针对特定数据集进行训练，比起从头开始训练的方式，训练时间将大大缩短。深度神经网络中的迁移学习通常采用以下两种方法：

(1) 把训练好的CNN当作特征提取器

Donahue等[20]提出可以保留在ImageNet训练好的网络中的所有层的参数，只是替换掉最后一层全连接层。在这最后这一层全连接层之前的网络层称为瓶颈层(bottleneck)。因为将瓶颈层的输出再通过一个全连接层可以很好地区分ImageNet中的1000种类别的图像，所以有理由认为瓶颈层输出的节点向量可以被作为任何图像的一个更加精简且表达能力更强的特征向量。

(2) 对训练好的CNN进行微调(fine-tuning)

该方法除了替换和重新训练CNN的最后一层全连接层，还以一个小学习率微调瓶颈层的权值。CNN的图像预测过程可以看成是从简单特征到复杂特征的特征提取的过程。Yosinski等[21]提出在ImageNet训练好的网络中，CNN前几层提取的是一些通用特征（如线条、边、角、简单形状），最后几层提取的是区分1000种类别的细节特征。因此，可以根据具体的数据集种类，微调CNN的所有层，或保持前几层的参数不变（防止过拟合），只微调CNN的最后几层。

除了加快收敛速度，缩短训练时间，迁移学习还能有效避免过拟合问题。由于在ImageNet数据集上训练好的CNN已经提取了非常丰富的特征，具备很强的表达能力，因此在此基础上针对特定数据集进行训练时，网络很快便会收敛，即损失函数会很快降到一个很小的数值，网络不会过度拟合图像中的噪音部分。

3.4.2 多GPU并行训练

我们知道利用GPU的并行计算特点，可以用来加速深度神经网络的训练过程。我们自然而然地想到可以利用多个GPU并行计算来进一步缩短网络的训练时间。常用的并行化深度神经网络训练方式有两种：同步模式和异步模式。

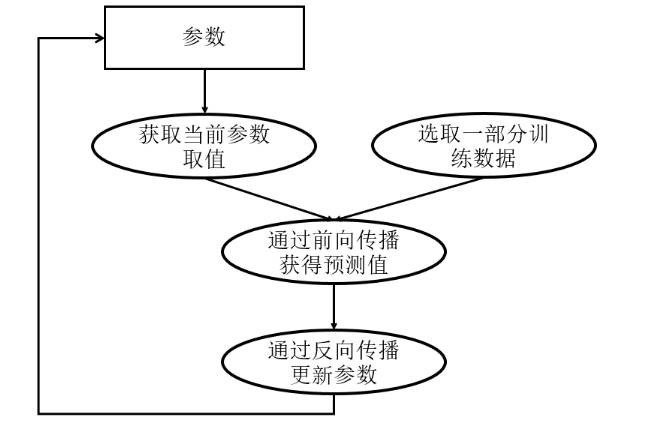
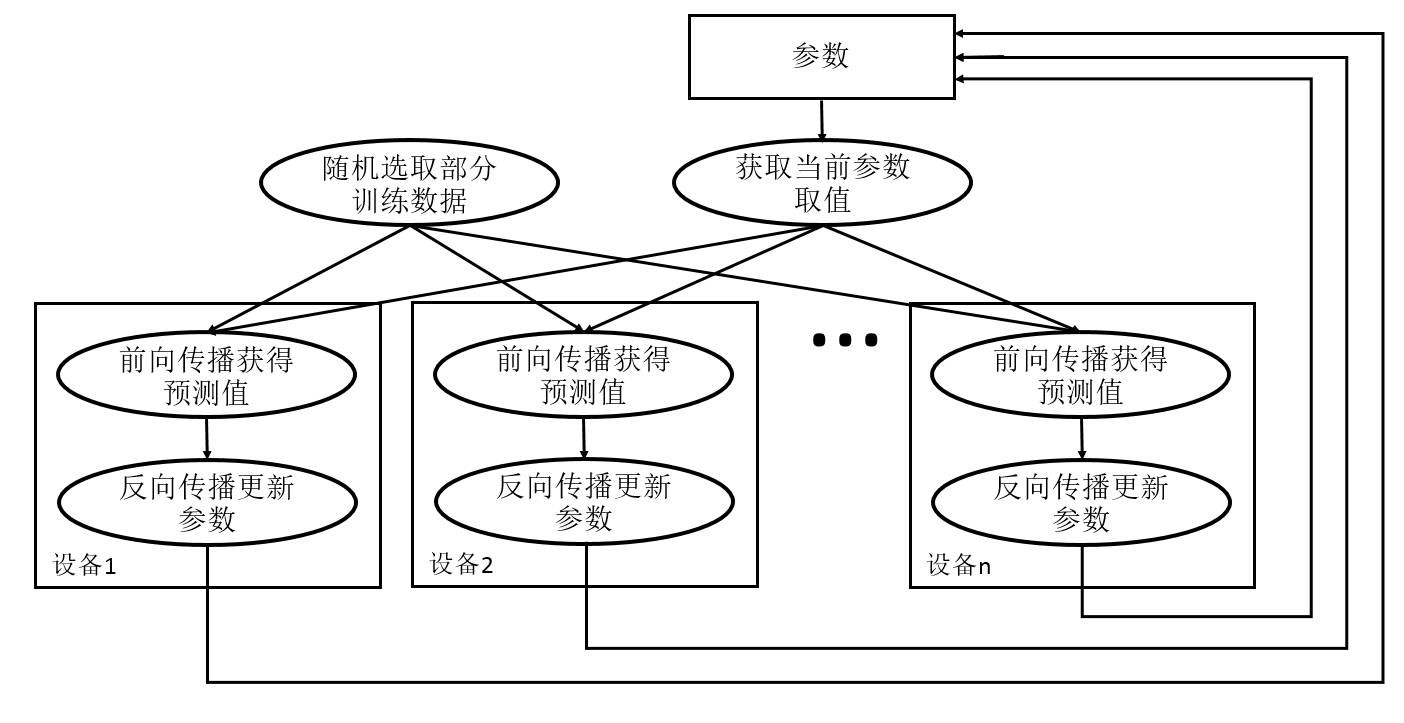
我们首先简单回顾一下如何训练深度神经网络。图3.8展示了深度神经网络的训练流程。深度神经网络的训练是一个迭代的过程。在每一轮迭代中，前向传播算法会根据当前参数的取值，计算出在一小部分训练数据上的预测值，然后反向传播算法再根据损失函数计算参数的梯度并更新参数。在并行化地训练深度神经网络时，不同设备（CPU或GPU），可以在不同训练数据上运行这个迭代的过程，而不同并行模式的区别在于不同的参数更新方式。

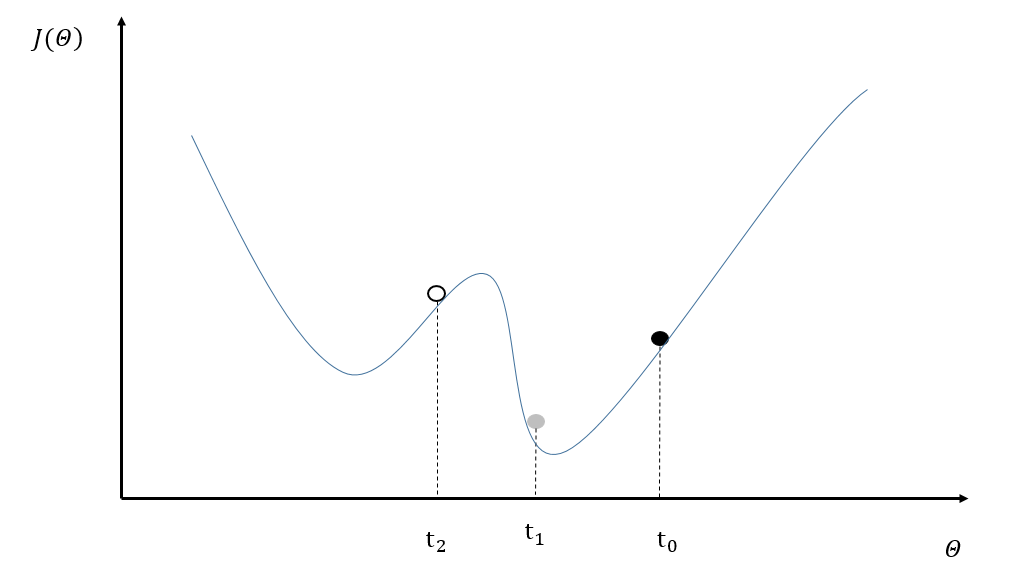
图3.8 深度神经网络训练流程图

3.4.2.1 异步模式训练

图3.9展示了异步模式的训练流程图。我们从图中可以看到，在每一轮迭代时，不同设备会读取参数最新的取值，但因为不同设备读取参数取值的时间不一样，所以得到的值也有可能不一样。根据当前参数的取值和随机获取的一小部分训练数据，不同设备各自运行反向传播的过程并独立地更新参数。可以简单地认为异步模式就是单机模式复制了多份，每一份使用不同的训练数据进行训练。在异步模式下，不同设备之间是完全独立的。

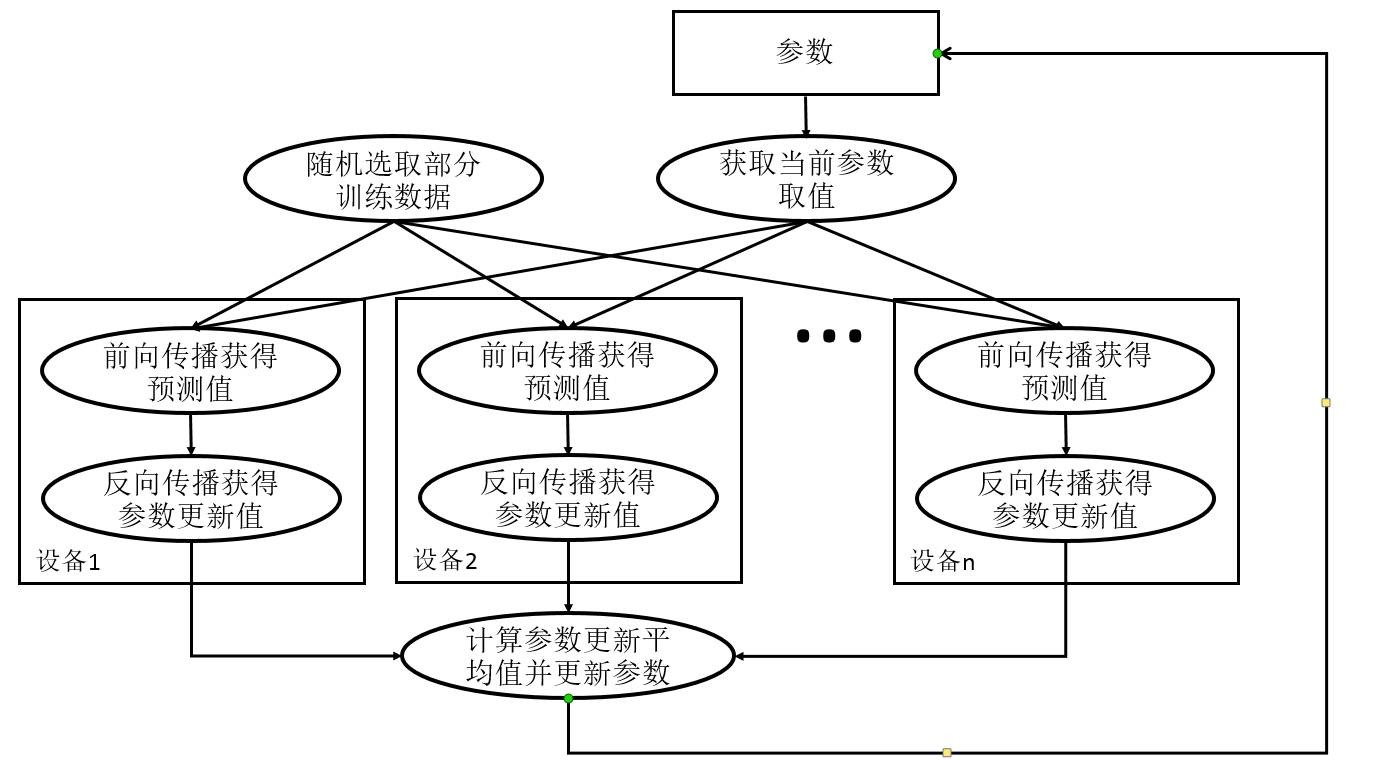
图3.9 异步模式深度神经网络训练流程图

然而使用异步模式训练深度神经网络有可能无法达到全局最优点。图3.10给出了一个具体的样例来说明异步模式的问题。图中曲线展示了模型的损失函数，黑色小球表示在t0时刻的参数所对应的损失函数的大小。假设两个设备device0和device1在同时在t0时刻读取了参数的取值，那么device0和device1计算出来的梯度都会使小黑球向左移动。假设device0在t1时刻已经完成了反向传播的计算并更新了参数，修改后的参数处于图3.10中小灰球的位置。然而这时device1并不知道参数已经被更新了，会继续将小球向左移动，使得小球的位置达到t2时刻小白球的地方。从图3.10中可以看出，当参数被调整到小白球的位置时，将无法达到最优点。

图3.10 异步模式训练深度神经网络存在的问题

3.4.2.2 同步模式训练

同步模式解决了异步模式下参数更新不同步的问题。在同步模式下，所有的设备同时读取参数的取值，并且当反向传播算法完成之后同步更新参数的取值。单个设备不会单独对参数进行更新，而会等待所有设备都完成反向传播之后再统一更新参数 。图3.11展示了同步模式的训练过程。从图3.11可以看到，在每一轮迭代时，不同设备首先统一读取当前参数的取值，并随机获取一小部分数据，然后在不同设备上运行反向传播过程得到在各自训练数据上参数的梯度。注意虽然所有设备使用的参数是一致的，但是因为训练数据不同，所以得到参数的梯度就可能不一样。当所有设备完成反向传播的计算之后，需要计算出不同设备上参数梯度的平均值，最后再根据平均值对参数进行更新。

图3.11 同步模式深度神经网络训练流程图

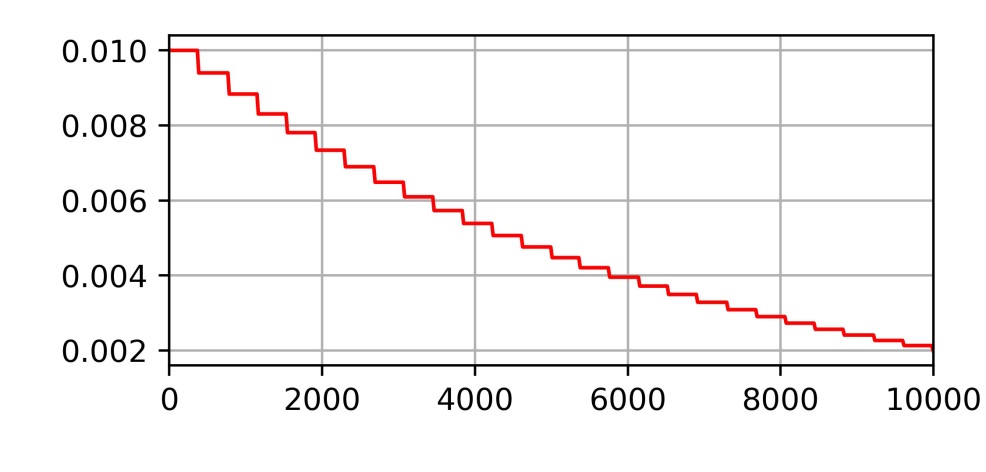
虽然同步模式解决了异步模式下参数更新不同步的问题，但是同步模式的效率却有可能低于异步模式。在同步模式下，每一轮迭代都需要设备统一开始、统一结束。如果设备的运行速度不一致，那么每一轮训练都需要等待最慢的设备结束才能开始更新参数，于是很多时间将被花在等待上。虽然理论上异步模式存在缺陷，但因为训练深度神经网络使用的随机梯度下降本身就是梯度下降的一个近似解法，而且即使是梯度下降也无法保证达到全局最优值，所以在实际应用中，在相同时间内，使用异步模式训练的模型不一定比同步模式差。所以这两种训练模式在实践中都有非常广泛的应用。

3.4.3 指数学习率衰减

在训练深度神经网络时，需要人工设置学习率来控制参数更新的速度。学习率决定了参数每次更新的幅度。如果学习率过大，将导致参数在极优值的两侧来回移动。相反，如果学习率过小，虽然能保证网络的收敛性，但是这会大大降低参数的更新速度，需要更多轮的迭代训练才能达到一个比较理想的优化效果。

我们可以采用一种灵活的学习率设置方法——指数衰减法，先使用较大的学习率让网络在训练前期快速接近较优解，然后随着迭代的继续逐步减小学习率，使得网络在训练后期不会有太大的波动，从而更加接近局部最优解。

图3.12 指数学习率衰减



迭代轮数

学习率

3.5 本章小结

深度神经网络的特点是通过超大规模样本训练让模型做到对特征的自动提取和抽象，特别是卷积核参数的学习。训练过程中我们会定义一个损失函数，然后使用相应的优化算法来最小化这个损失函数，从而寻找到最优参数。常用的优化算法有BGD、结合Momentum的BGD、AdaGrad、RMSProp、结合Momentum的RMSProp和Adam等。

深度神经网络通过多个隐函层的方法来实现非线性的函数，多层非线性隐含层的好处是理论上可以模拟任何数据分布，但也带来了过拟合现象以及参数难以学习的问题。我们可以采用数据增强、正则化、dropout的方式来防止过拟合，采用迁移学习、多GPU并行训练、指数学习率衰减的方式来缩短训练时间。

第4章  高效8-bit整数运算神经网络

卷积神经网络以其强大的特征表示能力，已经在许多应用领域中体现出了不俗的性能，例如人脸识别、图像检索和智能驾驶等。然而深度学习不仅仅是理论创新，更重要的是应用于工程实际，将高效的算法落地应用。随着芯片产业和硬件技术的发展，卷积神经网络已经逐渐地应用在移动设备和嵌入式平台中，例如智能手机、自动驾驶汽车、无人机、AR/VR设备和机器人。然而这些周边应用中的设备一般只有比较低的计算能力，而且也会受限于内存和电量消耗。因此，将模型量化和压缩，使其模型尺寸更小、推断更快、耗电更低是非常有必要的。本章首先简要概述了神经网络的量化与压缩技术，然后详细介绍了高效8-bit整数运算神经网络的一整套量化实现方案。

4.1 神经网络的量化与压缩技术概述

对于神经网络的量化与压缩，主要分为两种方法：一种方法是从头构建一个高效的神经网络模型，例如MobileNet[18]，SqueezeNet[22]，ShuffleNet[23]和DenseNet[24]。另一种方法是通过量化、裁剪和压缩来降低模型尺寸。降低模型的复杂度的一种简单有效的方法是降低权重和激活输出的精度，例如Ternary weight networks （TWN[25]），Binary Neural Networks（BNN[26]），XNOR-net[27]。降低权重和激活输出的精度这种方法往往更有效，这是因为：

(1) 它适用于绝大多数模型和使用场景

我们不需要为了提高速度去开发一个新的模型结构。在许多情况下，我们甚至不需要重新训练模型，只需要使用已有的浮点模型，就可以很快将其量化为定点型模型。目前许多硬件平台和库都支持利用量化的权重和激活输出来进行快速推断，因此这种方法更加符合实际工程应用场景。

(2) 更小的模型尺寸

使用8-bit量化，我们可以将模型的尺寸降低4倍。这种方法不需要任何数据，只需要将权重量化就可实现。这对于模型更新来讲，也可以减少模型下载时间。

(3) 更少的内存和缓存用于激活输出

在CNN中，中间计算结果为了网络的后续层重用，一般会缓存在cache中，如果精度降低那么这块数据就会占用更少的缓存。也就是说，更低精度的权重和激活输出有利于缓存更好地重用。

(4) 更快的计算

通常大多数处理器支持8bit数据的更快处理功能。此外，一些高效计算平台还支持8-bit神经网络的快速推理，其中包括GEMMLOWP[28]、Intel MKL-DNN[29]、ARM CMSIS[30]、Qualcomm SNPE[31]、Nvidia TensorRT[32]以及用于快速推断的定制硬件[33][34][35]。

(5) 更低的功耗

移动8-bit定点型数据与移动32-bit浮点型数据相比，在效率上前者比后者高4倍。对于许多深度神经网络结构，内存的使用量一定程度上正比于功耗[18]。因此减少数据移动量对降低功耗有非常重大的影响。

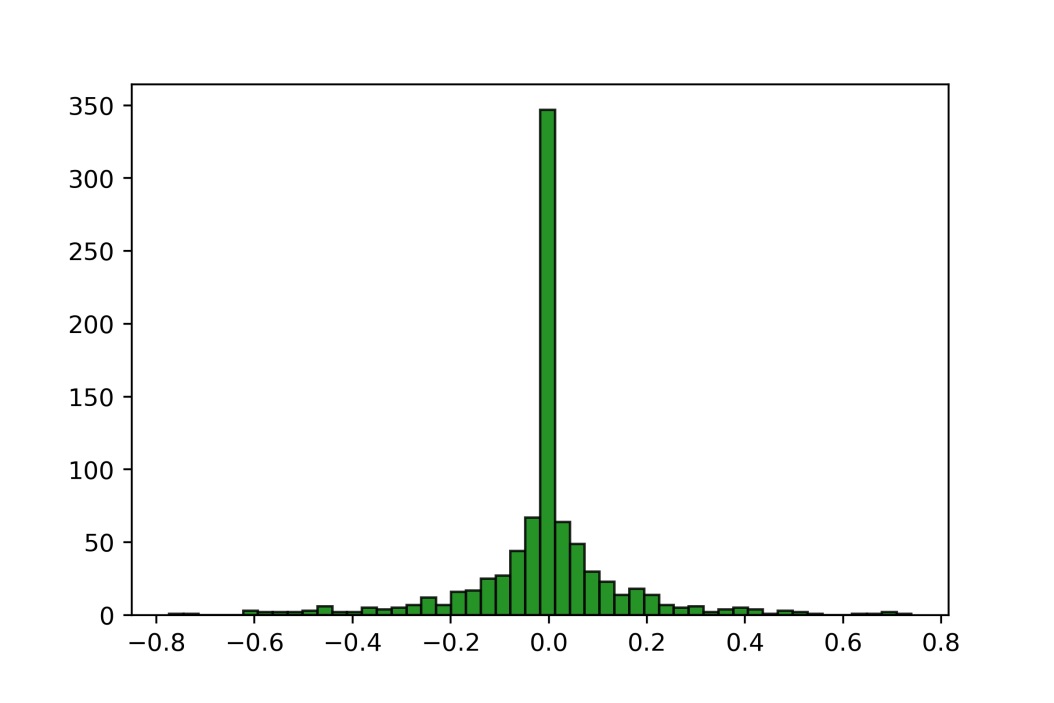
4.2 高效8-bit整数运算神经网络量化方案

在这一部分中，我们描述了我们的通用量化方案，也就是量化值（quantized value，简称q）与它们所对应的真实值（real value，简称r）之间的关系。我们的量化方案在网络的前传过程中采用纯整数算术运算，在网络训练过程中采用浮点算数运算，两种实现保持高度的对应关系。我们首先为我们的量化方案提供了一个严格的数学定义，并在整数算术推理与浮点训练过程中分别采用该方案。

4.2.1 神经网络权值分布特征

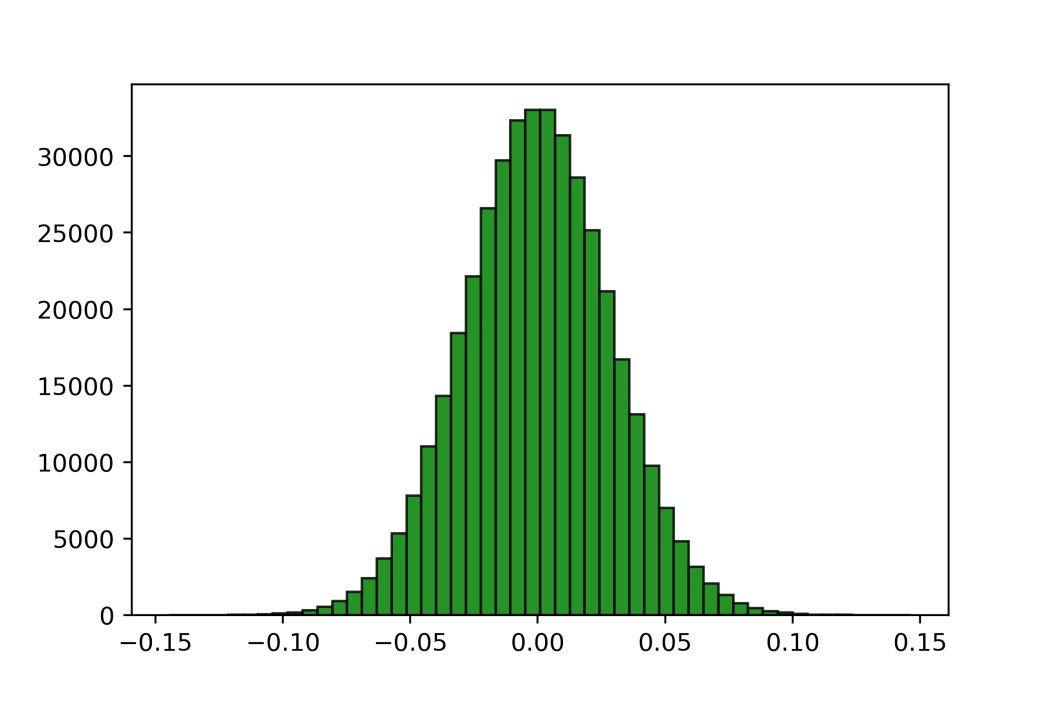
虽然神经网络有非常多的参数，但我们会发现每个卷积层的权重分布并非是凌乱的，而是有一定规律的。我们以MobileNet-V2的第一个卷积层和第二个卷积层为例分析权重的分布特征，如图4.1和图4.2所示：

图4.1 MobileNet-V2的Conv层权重分布



数值

参数个数



数值

参数个数

图4.2 MobileNet-V2的Conv\_1层权重分布

通过实验我们发现不仅是第一层和第二层，每一层权重都有相类似的分布特征。大多数权重都为0或者接近0，所有权重都被限制在一个很小的数值范围内，呈现出以0对称分布的趋势。

之所以会出现这样的分布，主要是正则化的作用。我们知道在机器学习中，不管是分类还是回归任务，都有可能因为特征过多而导致过拟合，因此我们通过减少特征或者惩罚不重要特征的权重来缓解这个问题。正则化通过约束参数的范数使其不要过大，以此降低模型的复杂度，减少特征权重防止过拟合。这也即是奥卡姆剃刀法则，越简单的东西越有效。一般来说，L1正则化会制造稀疏的特征，大部分无用特征的权重会被置为0，而L2正则化会让特征的权重不过大，使得特征的权重比较平均。

总的来说，这种分布特征为我们的量化方案提供了可能。而且，由于神经网络往往是鲁棒的处理噪声，通过量化一小组值而引入的误差将使总体结果在可接受的阈值范围内保持精度。

4.2.2 均匀仿射量化

假设有一个浮点型的变量，它的取值范围为 ，现在我们要把它量化到 取值范围，其中对于8-bit精度来讲 256。我们使用2个参数将浮点型值映射为整型值，尺度（scale，简称*S*）和零点（zero-point，简称*Z*）。*S*指定了量化的步长，而浮点值0则会映射到*Z*。对于单边分布，范围 需要进一步放宽去包含0点。例如，范围为 [2.1, 3.5] 的浮点型变量将会放宽为 [0, 3.5] ，然后再量化。

常数*S*和常数*Z*是我们的量化参数。对于权重，我们使用实际的最大和最小值；对于激活输出，我们使用跨批（batches）的最大和最小值的滑动平均值。*S*和*Z*的计算过程分别如下：

|  |  |  |
| --- | --- | --- |
|  |  | （4.1） |
|  |  | （4.2） |

我们的量化方案对同一层的激活数组和权重数组使用相同的量化参数，不同层的数组使用不同的量化参数。一旦*S*和*Z*定义好了之后，量化过程如下：

|  |  |  |
| --- | --- | --- |
|  |  | （4.3） |
|  |  | （4.4） |

其中

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

而逆量化过程如下：

|  |  |  |
| --- | --- | --- |
|  |  | （4.5） |

对于8位量化，*q*被量化为8位整数。一些数组，通常是偏置向量，被量化为32位整数。常数*S*是一个任意的正数，它和*r*一样是浮点数。常数*Z*与量化值*q*的类型相同，实际上是当*r*为0时所对应的量化值*q*。

4.2.3 卷积计算转化为矩阵乘法

传统的卷积采用卷积核窗口依次滑动的方法来计算输出，这样的计算方式速度往往会很慢。为了加速运算，通常都会将卷积转化为矩阵乘法，这样我们就可以调用各种线性代数运算库来实现加速（例如CUDA里面的矩阵乘法）。这些矩阵乘法都是极限优化过的，比卷积核窗口依次滑动这种暴力计算方法往往快很多倍。

卷积计算转化为矩阵乘法过程如图4.3、图4.4、图4.5所示。我们首先将输入图像矩阵和卷积核矩阵变换为两个大的矩阵x与y，然后x与y进行矩阵相乘得到结果z（利用GPU进行矩阵相乘的高效性）。

假设输入图像的通道数为C，高为H，宽为W，当卷积步长为1且对输入图像的边界进行0填充时，如图4.3所示我们对输入图像进行变形可得特征矩阵x，其维度为（H×W）×（C×K×K）。假设滤波器个数为Cout，滤波器通道数为C，高和宽都为K，我们采用同样的方式对滤波器进行变形可得滤波器矩阵y，其维度为Cout×（C×K×K）。当进行卷积运算时，我们用特征矩阵x乘以滤波器矩阵y的转置，得到输出矩阵z，其维度为（H×W）×Cout，之后再进行还原便可得到原始的特征图。

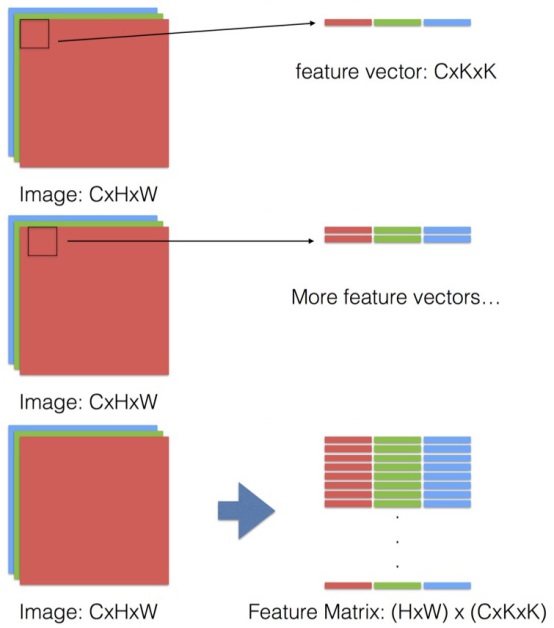
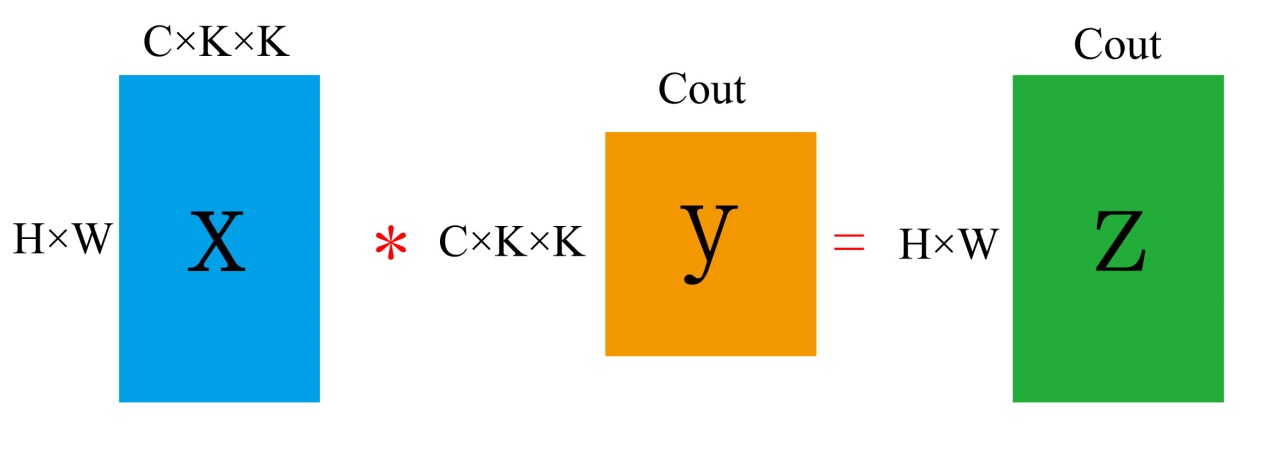
图4.3 卷积计算转化为矩阵乘法示意图I

图4.4 卷积计算转化为矩阵乘法示意图II

图4.5 卷积计算转化为矩阵乘法示意图III

4.2.4 量化的整数运算矩阵乘法

我们接下来将讨论如何利用表达式4.5将浮点运算的矩阵乘法转换为量化的整型运算矩阵乘法，以及如何将后者设计为只涉及整数运算，即使*S*不是整数。

假设2个维度都为 的浮点数矩阵  和 相乘，它们的乘积为 ，我们将这些矩阵 （为1、2或3）的每个元素命名为 （1≤*i, j*≤*N*），每个元素所对应的量化值为 ，每个元素所对应的量化参数为 、，则表达式4.5可以表示为：

|  |  |  |
| --- | --- | --- |
|  |  | （4.6） |

从矩阵乘法的定义，我们有：

|  |  |  |
| --- | --- | --- |
|  |  | （4.7） |

可以重写为：

|  |  |  |
| --- | --- | --- |
|  |  | （4.8） |

而乘法因子*M*可以表示为：

|  |  |  |
| --- | --- | --- |
|  |  | （4.9） |

在表达式4.8中，唯一的非整数项为*M*，作为一个仅依赖于量化尺度 、和 的常数，它可以离线计算出来。通过实验，我们发现*M*总位于区间(0,1)之间，因此*M*可以表示为：

|  |  |  |
| --- | --- | --- |
|  |  | （4.10） |

其中 为整数，而*n*为正整数。这样做的优点是，乘以 可以作为定点乘法来处理，而乘以 可以通过有效的移位并取整来实现。

量化后的矩阵乘法的一个特点是，它仅使用量化值上的整数运算就可以有效地实现所有算术运算。采用这种方式后，我们避免了需要查找表的实现，因为与SIMD硬件上的纯算术相比，查找表的性能往往会差很多。

4.2.5 典型的融合层实现

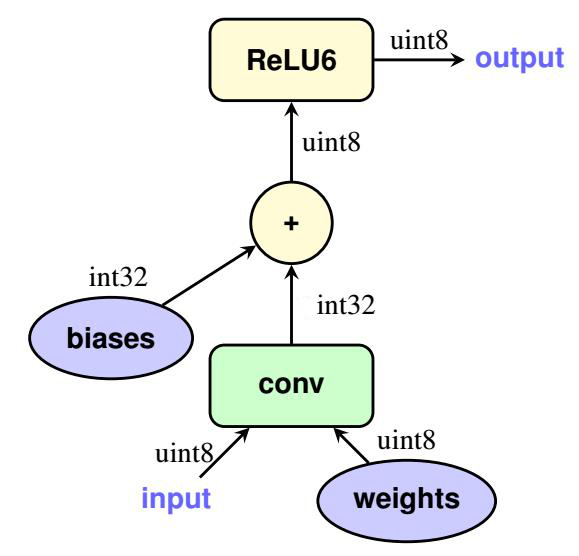
目前主流且高效的CNN都是由Convolution + Batch normalization + ReLu的基本层组成。在整数算术运算神经网络中，这样的基本层将会变成一个如图4.6所示的典型的融合层。我们将会继续讨论表达式4.8，但是现在明确地定义了所有量化运算所涉及到的数据类型，并修改量化矩阵乘法，将偏差加法和激活函数直接合并到其中，将整个层融合成单个运算。

图4.6 整数运算神经网络中的典型融合层

我们取 矩阵作为权重， 矩阵作为上一层的激活输出。所有权重和激活输出都是uint8类型。我们选择int32来作为乘积累加器，来累加uint8值的乘积。

为了使量化的偏置加法是将int32偏置加到int32累加器中，该偏置向量被量化如下：它使用int32作为量化的数据类型；它使用0作为量化零点；它的量化尺度 与累加器相同，即权重尺度和输入激活尺度的乘积。表达式如下：

|  |  |  |
| --- | --- | --- |
|  | ， | （4.11） |

虽然偏置向量被量化为32位值，但它们只占神经网络参数的一小部分。此外，利用较高精度的偏置向量满足实际需要：由于每个偏置向量都被添加到多个激活输出中，偏置向量中的任何量化误差都会充当过偏置 （即具有非零均值的误差项），为了保持端到端神经网络的良好精度，我们必须避免这种误差项。

对于int32累加器的最终值，还剩三件事要做：缩小到8位输出激活所使用的最终规模，转换为uint8，并应用激活函数来产生最终的8位激活输出。缩小操作对应表达式4.8中乘以常数*M*，正如表达式4.10所示，它可以通过定点乘法和移位取整来实现。之后，我们对输出进行饱和转换，并对其数值进行钳位，使其位于区间 [0，255] 之间。我们对输出进行钳位操作时，相当于对输出进行激活操作。完成这些操作后，我们便实现了一个典型的融合层运算。

我们接下来将讨论在预测过程中，如何将Batch normalization层整合进典型的融合层中。

我们知道在CNN中，卷积层后面通常会加上Batch normalization层，将每层输出的各个通道统计量进行归一化，在很好地提升模型精度的同时，也可以降低层间依赖。Batch normalization定义如下：

|  |  |  |
| --- | --- | --- |
|  |  | （4.12） |

在整数运算神经网络中，Batch normalization将按照表达式4.13、表达式4.14所定义的那样折叠进权重中，因此，典型的融合层实现时将没有显式的Batch normalization操作。Batch normalization将按如下方式拆分整合为权值和偏置：

|  |  |  |
| --- | --- | --- |
|  |  | （4.13） |
|  |  | （4.14） |

其中 为典型的融合层的卷积的权重，为典型的融合层的偏置。

4.2.6 模拟量化训练

一种常见的量化方法是用浮点训练网络，然后对训练好的浮点网络进行整型量化。我们发现，这种方法对于具有相当大的表示能力的大型模型非常有效，但对于小型模型则会导致显著的精度下降。而产生这种差异的主要原因在于：不同输出通道的权重范围相差很大（超过100×），而我们的量化方式是要求将同一层的所有通道量化到相同的分辨率（即同一层使用相同的量化参数*S*和*Z*），这使得数值范围较小的通道，量化到相同的分辨率后权重会产生更高的相对误差；另一个是量化后一些不太精确的离群权重值会对小模型的精度产生较大影响。

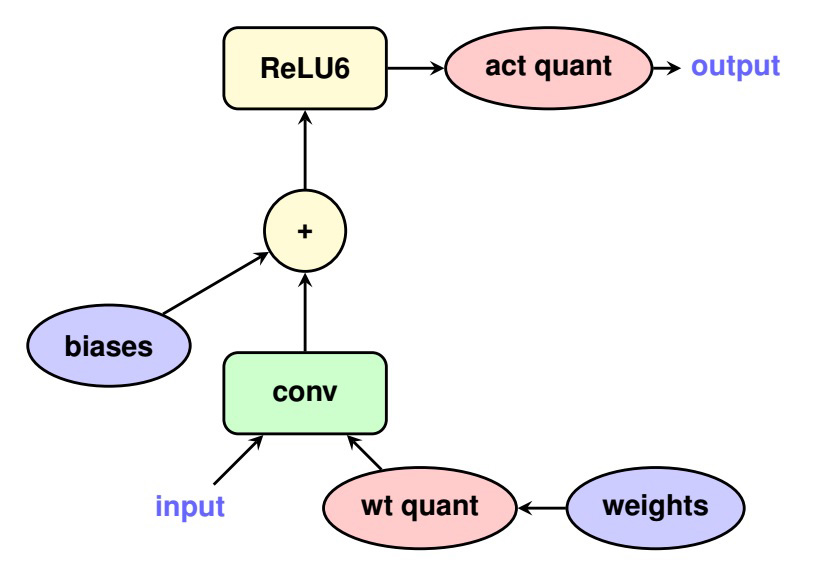
本文采用了一种模拟量化训练的方法，即在原始的浮点计算图中的权重读取和激活输出后插入模拟量化操作，来模拟前向传播过程中量化效应，如图4.7所示。这种训练时量化方法相比于训练后量化，往往能够得到更高的精度。

图4.7 模拟量化训练

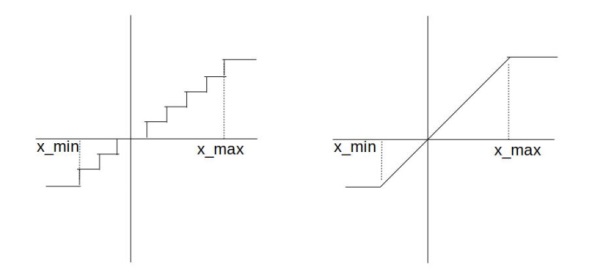
模拟量化操作具体形式如下：

在网络的前向传播过程中，我们采用量化后再紧跟一个逆量化的运算，即：

|  |  |  |
| --- | --- | --- |
|  |  | （4.15） |

在网络的反向传播过程中，由于模拟量化方程的导数几乎在各个位置均为0，因此我们需要在反向传播中构建一个近似量化。一种效果比较好的近似方法是将量化指定为表达式4.16的形式，这样可以方便定义导数，如图4.8所示

|  |  |  |
| --- | --- | --- |
|  |  | （4.16） |

图4.8 左边为模拟量化方程，为*y*=*x*量化后的输出值；右边为用于导数计算的近似方程

采用该近似方法后，我们便可以按往常一样计算该模拟量化操作的反向传播，从而可以用传统的浮点模型优化算法对网络进行训练。

当进行模拟量化训练时，在网络的前向传播和反向传播计算过程中，我们使用的都是模拟量化操作后的权重值和激活输出值。我们同时也保留了浮点型权重，并且在梯度更新的过程中更新它们。这样可以确保使用较小的梯度更新逐步更新权重，而不会造成梯度满溢。更新的权重被量化，然后用于后续的前向和反向传播计算。当使用随机梯度下降（SGD）优化算法时，权重更新方式如下：

|  |  |  |
| --- | --- | --- |
|  |  | （4.17） |

模拟量化训练过程中，我们可以看出偏差没有被模拟量化，因为它们在推理过程中被表示为32位整数，与8位权重和激活相比，其范围和精度要高得多。此外，偏差的量化参数是从权重的量化参数和激活输出的量化参数计算出来的。

采用模拟量化训练后，量化一个模型完整步骤如下：

1. 创建一个浮点类型的模型。
2. 添加量化运算，即在原始的浮点计算模型中的权重读取和激活输出后插入模拟量化操作。
3. 训练模型直到收敛。这个过程结束后，我们通过计算，便可以得到一个对于权重和激活输出都带有各自量化信息（尺度、零点）的模型。
4. 根据均匀仿射量化的转换规则，将浮点模型转化为整数模型。

4.3 本章小结

即使是目前最先进的卷积神经网络（例如MobileNet-V2），也并不适合直接部署到移动设备和嵌入式平台。这是因为绝大多数CNN都采用了float32的数据格式，而只有集成了FPU（浮点处理单元）的硬件才具有处理float32数据的能力。此外，移动设备和嵌入式平台的计算能力较低，计算资源也较少，因此要求CNN要有较低的时间复杂度和空间复杂度。本文采用的高效8-bit整数运算神经网络量化方案能很好地解决这一问题，将CNN的浮点运算转化为纯整数运算，同时减少模型大小，缩短了模型预测时间。另外本文还采用了一种模拟量化训练的方法，该方法能极大减少浮点到整数转换过程中产生的精度下降。

第5章 实验结果与分析

本章将对本文中提出的算法进行实验展示以及分析，主要分为以下几部分：相关数据集及实验环境、模型训练相关实验、8-bit模型量化相关实验以及与当今学术界的对比。通过严谨的实验验证以及详细的分析，验证了本文算法的有效性，并得到了一些有用的结论。

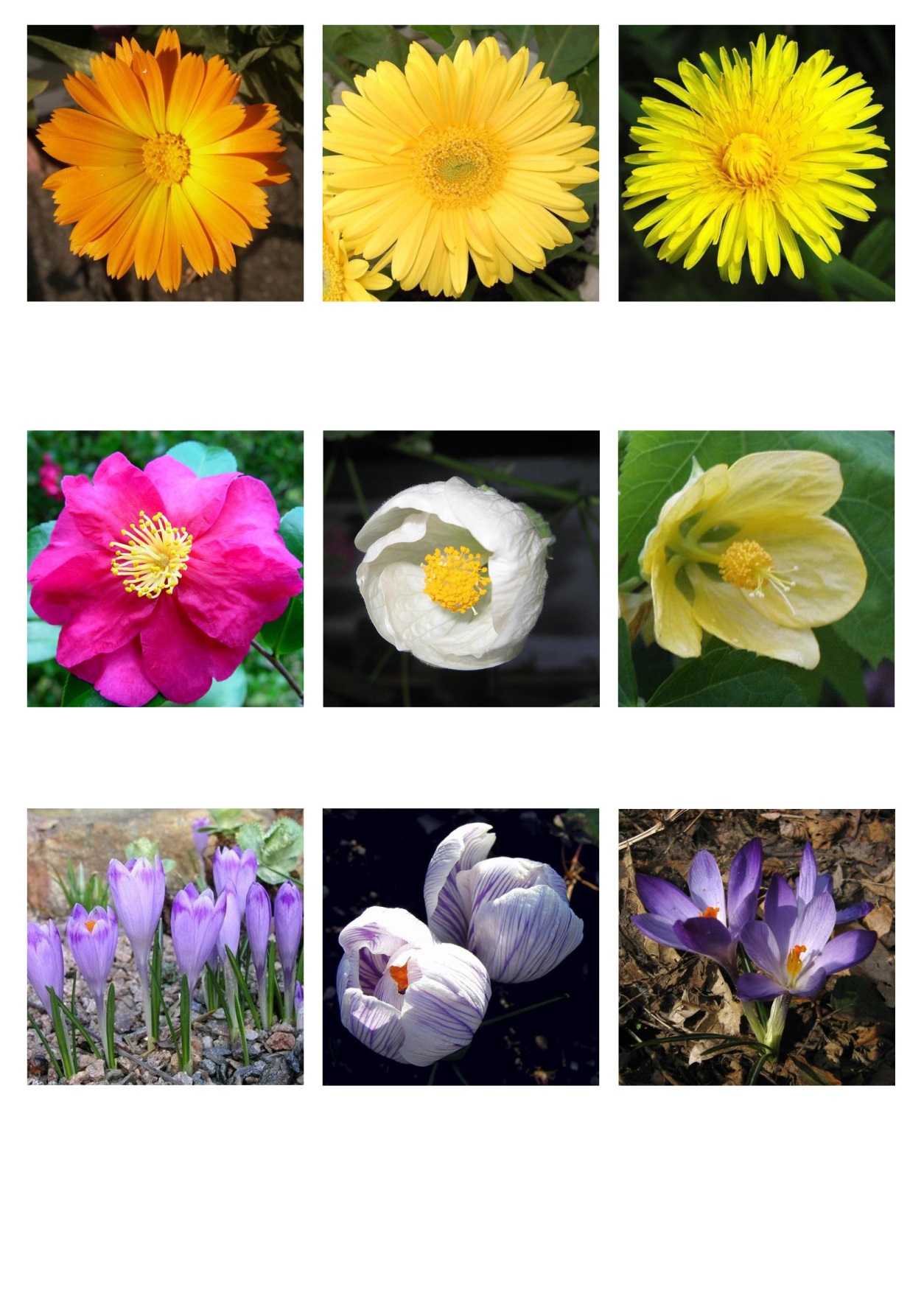
5.1 数据集及实验环境

5.1.1 数据集

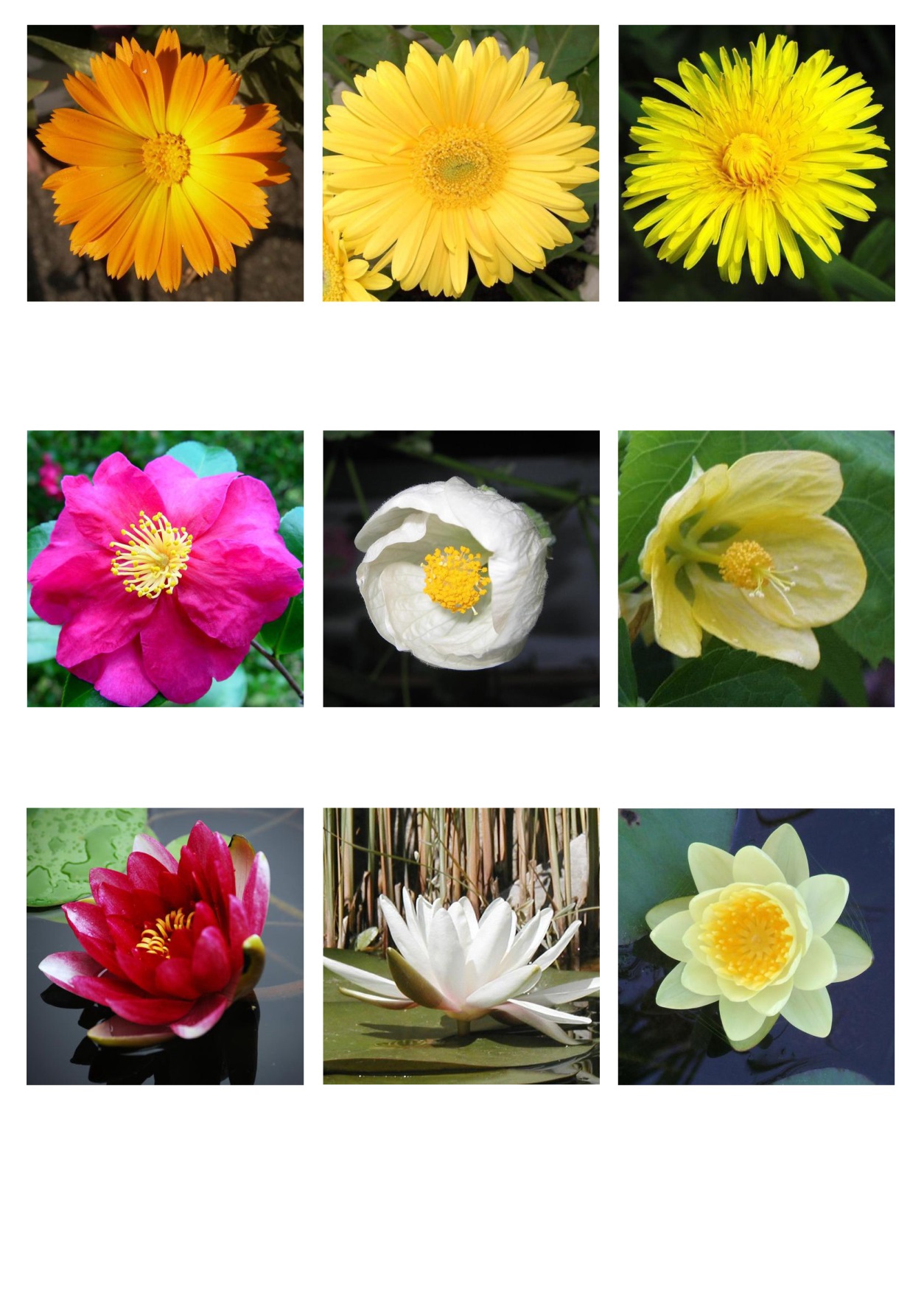
本文采用的数据集是英国牛津大学Visual Geometry Group（VGG）组所创建的Oxford-102 Flower数据集。该数据集包含102类花卉图像，每类图像有40~250张，总共8189张图像。该数据集不仅种类繁多，而且每一类花卉图像之间的数量很不平衡，具有很大的挑战性。此外，该数据集还兼顾了花卉图像识别中所有的难点，包括：

1. 光照变化：花卉图片是在白天的各个不同时段、不同的光照条件拍摄的；
2. 视角变化：拍摄时包括了各种不同的拍摄角度，有的花卉图片是远景拍摄而有一些是近景拍摄；
3. 部分遮挡：部分花卉受其他物体或者叶片边缘的遮挡；
4. 背景复杂：每张图片都采用不同的拍摄场景；
5. 丰富的形态变化：花卉图片包括了不同阶段的花期，有的含苞待放、有的半开，有的全开；
6. 颜色变化：同一类别的花具有不同的颜色；
7. 类间相似性与类内差异性：不同类别的花卉之间具有很大的相似度，同一类别的花卉又有很大的差异性。

图5.1 视觉变化和颜色变化

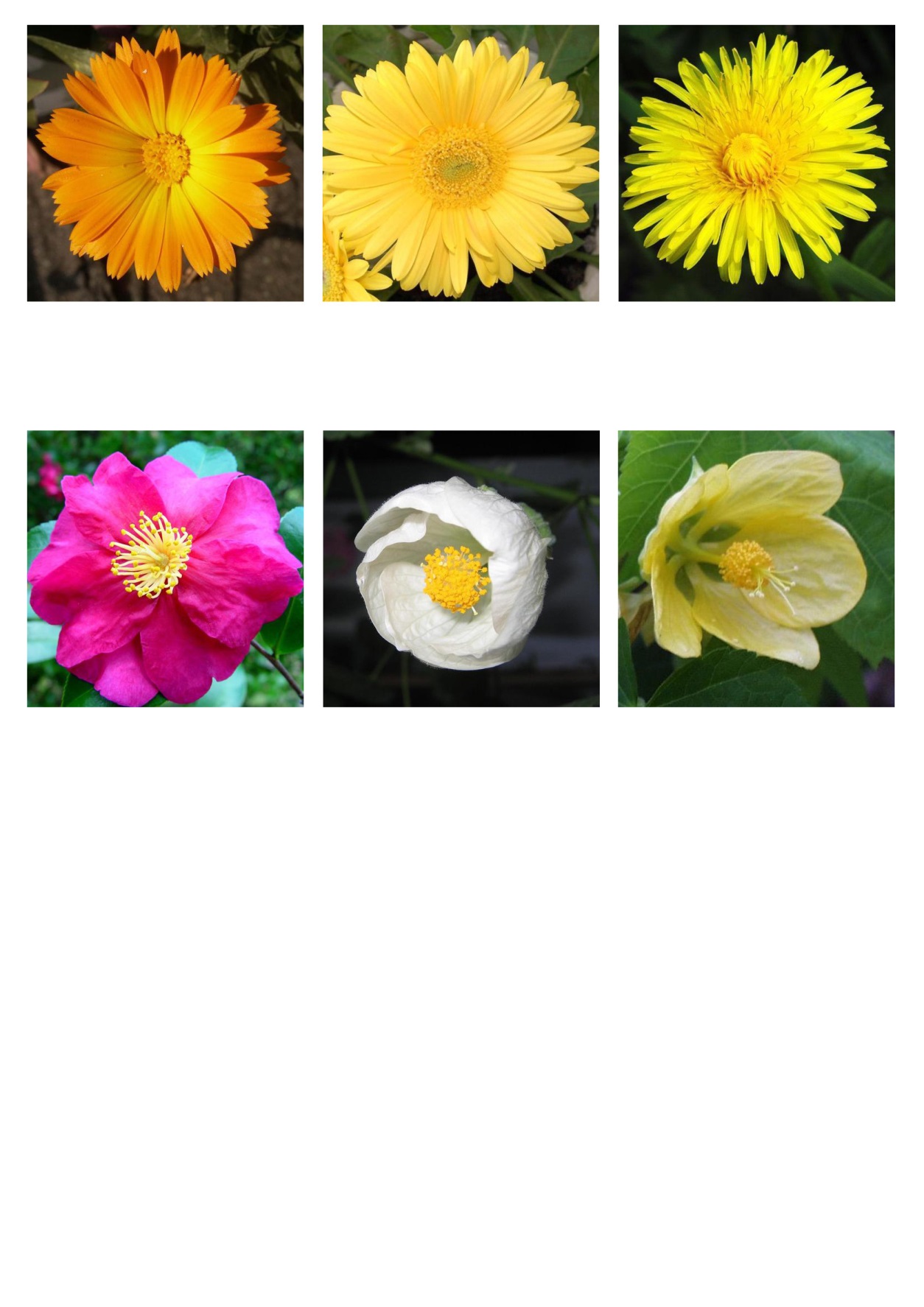


藏红花



荷花

图5.2 形态变化（不同的花期）



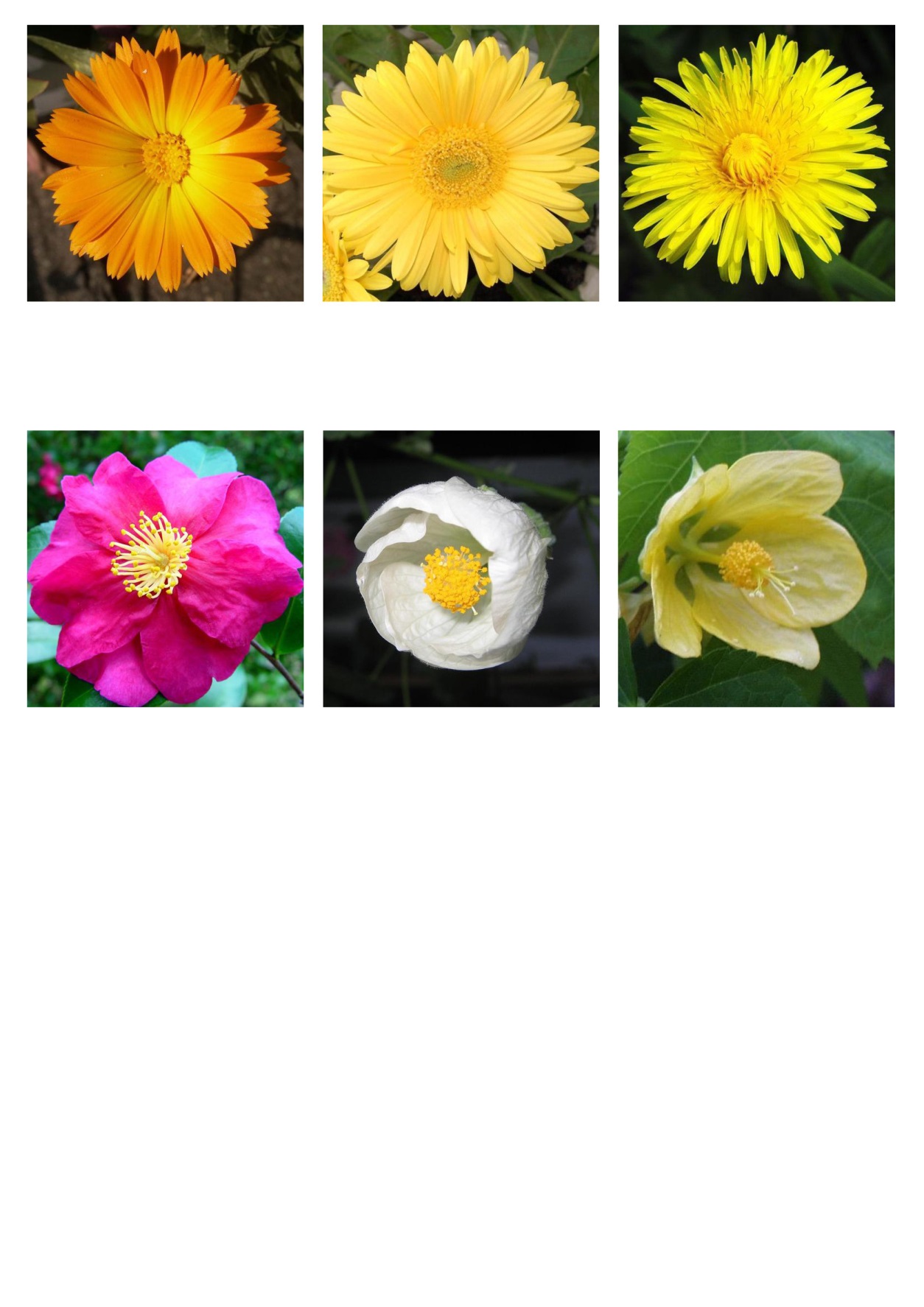
西洋蒲公英

非洲菊

英格兰万寿菊

图5.3 类间相似性

图5.4 类内差异性



山茶

该数据集既充分考虑到了一般图像分类的基本特性，同时也充分考虑到了花卉这一类别自身的一些特性，在花卉图像的分类研宄中具有重要的意义。该数据集被固定分为三个集合，分别为训练集（6149张）、验证集（1020张）和测试集（1020张）。

5.1.2 实验环境

5.1.2.1 主机平台

实验采用的台式机配置为：AMD Ryzen5 1600处理器，主频3.2GHz，16GB 2400MHz DDR4内存，双路Nvidia Geforce GTX 1070（8GB）显卡。

我们选用TensorFlow作为主机的模型训练和预测框架。TensorFlow前端负责构造计算图，后端负责执行计算图。TensorFlow前端支持Python、C++、Go、Java等多种开发语言，后端使用C++、CUDA等写成。TensorFlow实现的算法可以在众多异构的系统上方便地移植，比如Android手机、iPhone、普通的CPU服务器，乃至大规模GPU集群，如图5.6所示。

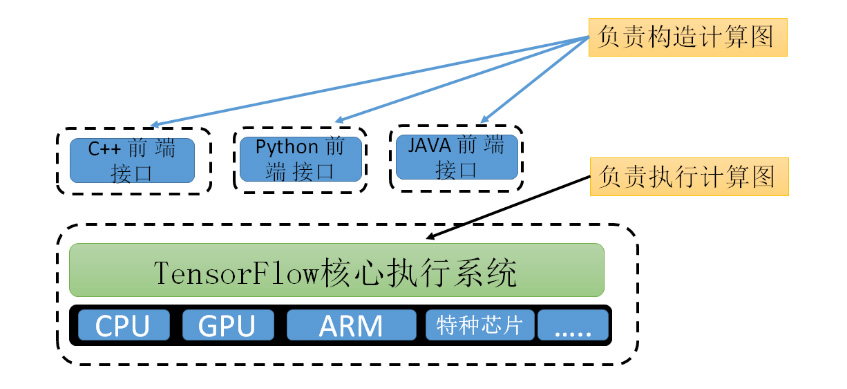


图5.6 TensorFlow 基础架构

TensorFlow使用数据流式图来规划计算流程，它可以将计算映射到不同的硬件和操作系统平台。凭借着统一的架构，TensorFlow可以方便地部署到各种平台，大大简化了真实场景中应用机器学习的难度。使用TensorFlow我们不需要给大规模的模型训练和小规模的应用部署开发两套不同的系统，避免了同时维护两套程序的成本，TensorFlow给训练和预测的共同部分提供了一个恰当的抽象。TensorFlow的计算可以表示为有状态的数据流式图，对于大规模的神经网络训练，TensorFlow可以让用户简单地实现并行计算，同时使用不同的硬件资源进行训练，同步或异步地更新全局共享的模型参数和状态。将一个串行的 TensorFlow算法改造成并行的成本也是非常低的，通常只需要对小部分代码进行改写。虽然绝大多数的TensorFlow应用都在机器学习及深度学习领域，但TensorFlow抽象出的数据流式图也可以应用在通用数值计算和符号计算上，比如分形图计算或者偏微分方程数值求解。表5.1展示了TensorFlow的主要技术特性。

表5.1 TensorFlow的主要技术特性。

|  |  |
| --- | --- |
| 编程模型 | 数据流模型 |
| 语言 | Python、C++、Go、Java |
| 部署 | 一次编写，各处运行 |
| 计算资源 | CPU、GPU、TPU |
| 实现方式 | 单机实现、分布式实现 |
| 平台支持 | 谷歌云平台、Hadoop分布式文件系统 |
| 数学表达 | 数学计算图表达、自动微分 |
| 优化 | 共同子图消除、异步核优化、通信优化、模型并行、数据并行、流水线 |

5.1.2.2 移动端平台

在移动端平台测试中，我们使用小米6手机，处理器为高通骁龙835，主频2.45GHz，内存6GB。

移动端开发我们使用Google提供的Bazel构建工具。Bazel是一个类似于Make的编译工具，是Google为其内部软件开发的特点量身定制的工具，如今Google使用它来构建内部大多数的软件，包括TensorFlow本身。相比传统的Makefile、Ant或者Maven，Bazel在速度、可伸缩性、灵活性以及对不同程序语言和平台的支持都要更加出色。

Bazel的基本概念是项目空间（workspace）。一个项目空间可以简单地理解为一个文件夹，在这个文件夹中包含了编译一个软件所需要的源代码以及输出编译结果的软连接地址。一个项目空间所对应的文件夹是这个项目的根目录，在这个根目录中需要有一个WORKSPACE文件，此文件定义了对外部资源的依赖关系。在一个项目空间内，Bazel通过BUILD文件来找到需要编译的目标，而BUILD文件采用一种类似于Python的语法来指定每一个编译目标的输入、输出以及编译方式。

使用Bazel开发一个APK的简要流程如下：

1. 下载TensorFlow源码。
2. 下载并安装Bazel。
3. 下载并安装开发Android APK所需的Android NDK和Android SDK依赖。
4. 编辑WORKSPACE文件，将Android条目中的NDK和SDK路径改为本地的NDK和SDK路径。
5. 将训练好的固化权重的pb模型、所有类别名称的txt文本复制到该项目空间的assets文件中，并修改相应java文件中的路径。
6. 使用Bazel命令编译生成APK文件。

5.2 模型训练相关实验

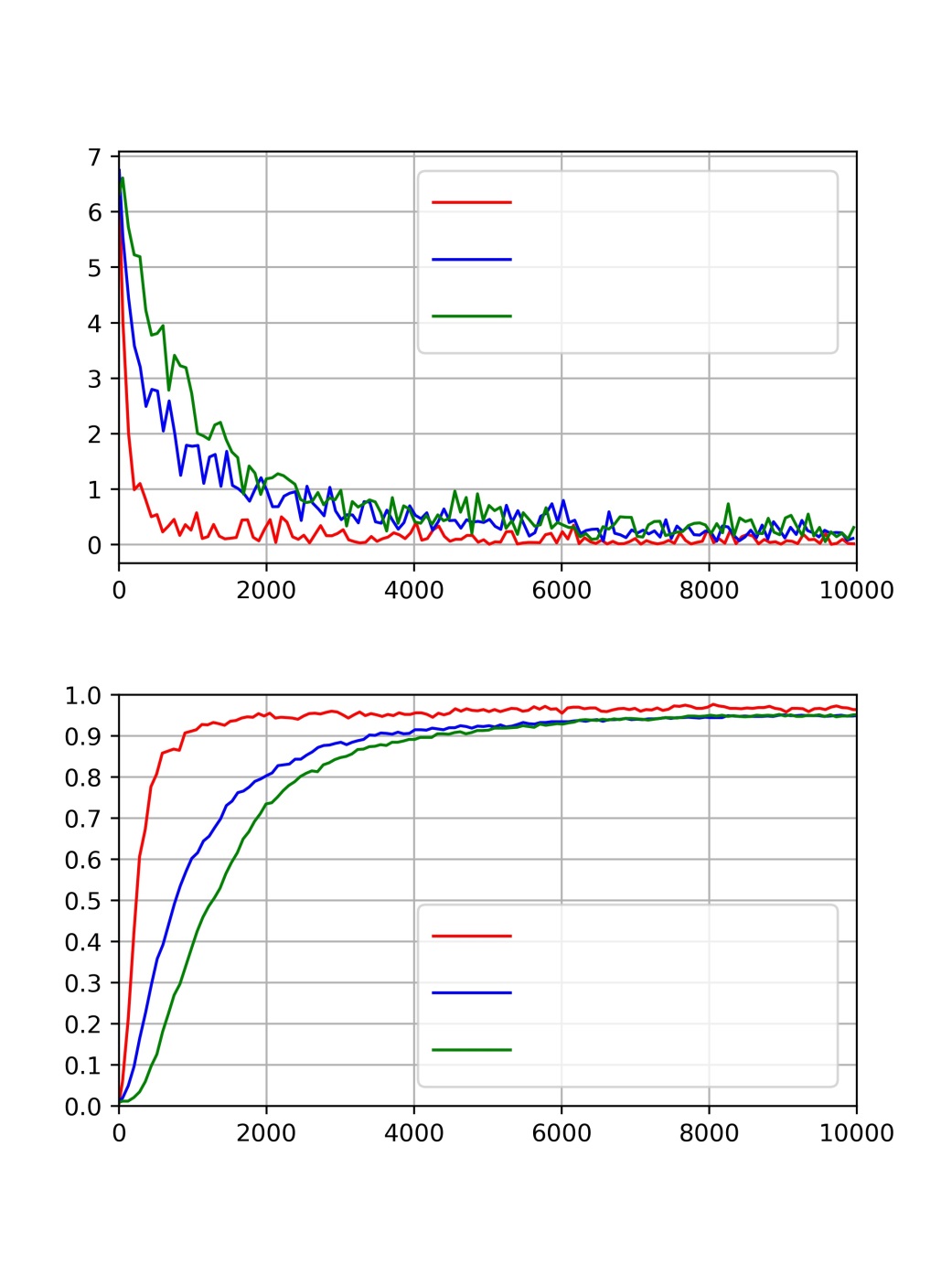
本文采用的CNN为MobileNet-V2，采用TensorFlow作为模型训练框架。模型训练相关实验包括：结合Momentum的RMSProp优化算法与其他优化算法对比、双GPU并行训练与单GPU训练对比以及不同训练策略对最终精度的影响。

5.2.1 结合Momentum的RMSProp优化算法与其他优化算法对比

模型训练过程中，本文采用的是结合Momentum的RMSProp优化算法，对比算法有BGD优化算法和Adam优化算法。

三种优化算法的学习率 *η* 都固定为 0.001，batch\_size都为32。结合Momentum的RMSProp优化算法衰减速率 *ρ* = 0.9，动量系数 *α* = 0.9； Adam优化算法的矩估计指数衰减速率 *β*1 = 0.9， *β*2 = 0.999。

三种优化算法都采用迁移学习的方法（对ImageNet训练好的CNN进行微调），同时采用了数据增强和正则化的训练策略，采用双GPU并行训练的方式。训练过程中，每训练100轮记录损失函数值和在验证集上的Top-1正确率，总共进行10000轮迭代训练。实验结果如图5.7所示。



结合Momentum的RMSProp

Adam

损失函数

BGD

训练迭代轮数

结合Momentum的RMSProp

Adam

BGD

训练迭代轮数

Top-1 正确率

图5.7 结合Momentum的RMSProp优化算法与其他优化算法对比图

从图5.7可以看出，在相同的迭代轮数内，结合Momentum的RMSProp优化算法相比Adam算法和BGD算法，损失函数下降得更快，Top-1正确率上升得更高。此外，在10000轮迭代训练结束后，结合Momentum的RMSProp优化算法相比其他两种优化算法，更能达到一个更高的Top-1正确率。

我们挑选验证集Top-1正确率最高的参数来作为我们最终的模型参数。模型在测试集上的最终Top-1正确率为95.8%。

5.2.2 双GPU并行训练与单GPU训练对比

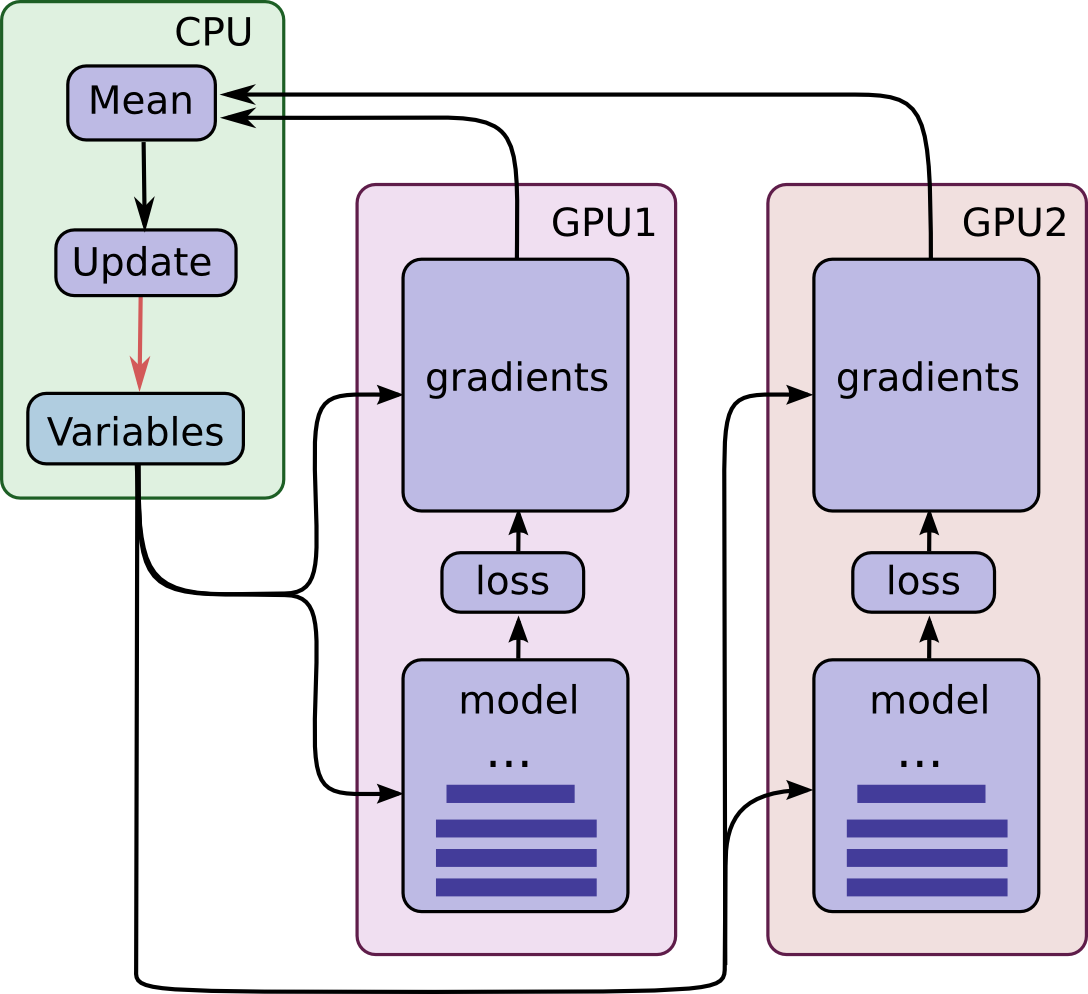
本文采用同步模式的双GPU并行训练来缩短模型训练时间，如图5.8所示：

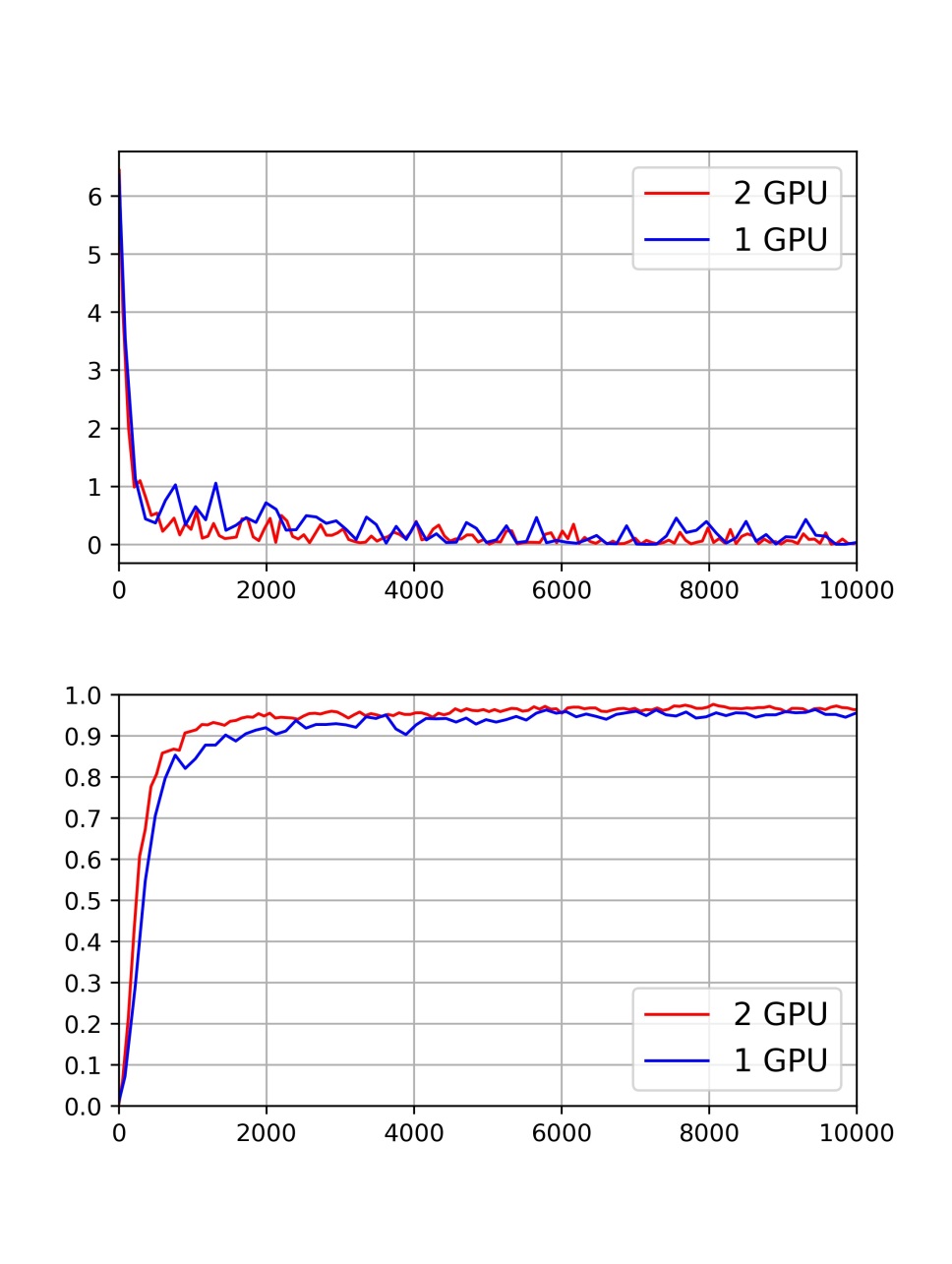
图5.8 双GPU并行训练

计算过程如下：

1. 先将模型定义在CPU上，然后将模型的副本分别定义在两个GPU上；
2. 对于每一个GPU，先从CPU获得数据（模型参数和训练数据），然后前向传播进行计算，得到loss，再进行反向传播计算出各自的梯度；
3. CPU接到GPU的梯度，取平均值，然后进行梯度更新。

本文进行的是单GPU和双GPU的训练对比实验。两者都采用结合Momentum的RMSProp优化算法，学习率 *η* 固定为 0.001，batch\_size为32，衰减速率 *ρ* = 0.9，动量系数 *α* = 0.9；都采用了迁移学习、数据增强和正则化的训练策略。每训练100轮记录损失函数值和在验证集上的Top-1正确率，总共进行10000轮迭代训练。实验对比图如5.9所示：

图5.9 双GPU并行训练与单GPU训练对比图



训练迭代轮数

损失函数

训练迭代轮数

Top-1 正确率

从图5.9可以看出，在相同的迭代轮数内，双GPU对比单GPU损失函数下降得更快，Top-1正确率上升得更高。

5.2.3 不同训练策略对最终精度的影响

本文进行了几个对比实验，来验证不同的训练策略对最终精度的影响，如表5.2所示。所有实验都采用结合Momentum的RMSProp优化算法，学习率 *η* 固定为 0.001，batch\_size为32，衰减速率 *ρ* = 0.9，动量系数 *α* = 0.9。训练过程中，我们都挑选验证集Top-1正确率最高的参数来作为我们最终的模型参数。

表5.2 不同训练策略对最终精度的影响

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验轮数 | 迁移学习 | L2正则化 | 数据增强 | 测试集Top-1正确率(%) |
| 1 | 🗸 | 🗸 | 🗸 | 95.8 |
| 2 | 🗴 | 🗸 | 🗸 | 63 |
| 3 | 🗸 | 🗴 | 🗸 | 95 |
| 4 | 🗸 | 🗸 | 🗴 | 92.8 |

5.3 8-bit模型量化相关实验

本小节将进行一系列模型量化实验，来验证高效8-bit整数运算神经网络量化方案的有效性。采用的模型参数为5.2小节中所选用的最终参数（在测试集上的Top-1正确率为95.8%）。

本小节将进行Top-1正确率和预测时间方面的实验。其中Top-1正确率实验在主机端进行，主要步骤为采用本文提出的8-bit模型量化方案将模型量化后，测试其在测试集上的Top-1正确率。预测时间实验在移动端进行，主要步骤为将最终模型编译成APK然后安装到手机，测试其在手机端的预测时间。

5.3.1 未模拟量化训练

若未采用模拟量化训练，直接对浮点模型进行8-bit量化，其在测试集上的Top-1正确率和移动端的预测时间如下表所示：

表5.3 未采用模拟量化训练的模型指标

|  |  |  |  |
| --- | --- | --- | --- |
| 模型数据类型 | 模型大小(APK文件大小) | 移动端预测时间 | 测试集Top-1正确率(%) |
| float32 | 17.8 MB | 142 ms | 95.8 |
| uint8 | 7.95 MB | 74 ms | 80.6 |

我们可以看出未采用模拟量化训练，直接对浮点模型进行8-bit量化，模型的Top-1正确率直接从95.8%下降到80.6%，精度下降得比较严重。

5.3.2 模拟量化训练

为了减少对浮点模型进行8-bit量化产生的精度下降，我们先在主机端对模型进行模拟量化训练。我们采用5.2小节选用的最终参数（在测试集上的Top-1正确率为95.8%）来作为我们的初始化参数，然后对模型进行模拟量化训练。我们挑选验证集Top-1正确率最高的参数来作为最终的模型参数，再对浮点模型进行8-bit量化。最终模型在测试集上的Top-1正确率和移动端的预测时间如下表所示：

表5.4 采用模拟量化训练后模型的指标

|  |  |  |  |
| --- | --- | --- | --- |
| 模型数据类型 | 模型大小(APK文件大小) | 移动端预测时间 | 测试集Top-1正确率(%) |
| float32 | 17.8 MB | 142 ms | 95.8 |
| uint8 | 7.95 MB | 74 ms | 94.7 |

我们可以看出先采用模拟量化训练再对浮点模型进行8-bit量化，相比直接对浮点模型进行8-bit量化的方式，Top-1正确率仅仅下降1%左右，而两种方式的模型大小和移动端预测时间则相同。先采用模拟量化训练再对浮点模型进行8-bit量化，能使模型在速度、尺度和精度都达到较好的效果。

5.4 与当今学术界的对比

从上面的实验结果，我们可以看出本文的移动端花朵分类模型在速度、尺度和精度方面都具有较好的效果。为了进一步验证本文模型的有效性，本文与当今学术界的前沿方法进行了Top-1正确率的对比，包括传统图像分类方法和CNN方法，如表5.5所示。可以看出本文的移动端花朵分类模型在Oxford-102 Flower数据集上的Top-1正确率高于学术界的前沿方法。

表5.5 不同的花卉识别方法Top-1正确率对比

|  |  |  |
| --- | --- | --- |
| 类别 | 方法 | Top-1正确率(%) |
| 传统图像分类 | HSV + SIFTi + SIFTb + HOG[1] | 72.8 |
| BiCos[2] | 79.4 |
| Seg + Dense + HOG + Coding + Pooling[3] | 80.7 |
| CNN | AlexNet[4] | 82 |
| GoogLeNet[5] | 88.4 |
| Inception-v3[6] | 94 |
| 基于MobileNet-V2的高效8-bit整数运算CNN | 94.7 |

由于对比文献并未给出模型大小与预测时间方面的数据，本文将对比文献中的CNN方法做到移动端并进行测试（采用的是ImageNet数据集训练好的CNN），实验结果如表5.6所示：

表5.6 不同的花卉识别方法模型大小与预测时间对比

|  |  |  |
| --- | --- | --- |
| CNN模型 | 模型大小(APK文件大小) | 移动端预测时间 |
| AlexNet[4] | 196 MB | 253 ms |
| GoogLeNet[5] | 29.8 MB | 354 ms |
| Inception-v3[6] | 95.4 MB | 759 ms |
| 基于MobileNet-V2的高效8-bit整数运算CNN | 7.95 MB | 74 ms |

可以看出本文的移动端花朵分类模型在模型大小与预测时间方面也有较好的效果。

5.5 移动端效果图

本小节展示了该花卉识别系统在移动端的效果截图，如图5.10、图5.11、图5.12所示。其中左图为移动端预测截图，右图为Oxford-102 Flower数据集中该类别的对应图片。移动端截图中，上方显示了模型的预测类别以及对应的概率，下方显示了单帧预测时间。

图5.10 移动端截图（bougainvillea）

图5.11 移动端截图（poinsettia）

图5.12 移动端截图（camellia）

5.6 本章小结

本章通过一系列对比实验，来验证本文的移动端花朵分类模型的有效性。在模型训练相关实验中，我们首先进行了结合Momentum的RMSProp优化算法与其他优化算法对比实验，来证明该优化算法的优越性；随后进行了双GPU并行训练与单GPU训练对比实验，来证明双GPU并行训练对模型的训练加速效果；然后进行了几个对比实验，来验证不同的训练策略对最终精度的影响。在8-bit模型量化相关实验中，我们分别进行了未采用模拟量化训练方式和采用模拟量化训练方式的实验，最终得出了采用模拟量化训练的方式能使模型在速度、尺度和精度都达到较好的效果。本文在最后与当今学术界的花朵分类方法进行了对比，来证明本文模型在速度、尺度和精度方面的有效性。

总结与展望

作为深度学习的一个重要研究成果， CNN已广泛应用于图像分类、物体检测、语义分割、人脸识别、图像检索、图像超分辨率等多个领域，在学术界和工业界掀起了人工智能的浪潮。CNN与传统机器学习方法不同的是，传统机器学习方法需要人工提取图像的特征，而特征的提取往往是非常耗时耗力的，要求研究人员必须在该领域有非常深入的理解；而CNN则是通过超大规模样本训练让模型对特征进行自动提取和抽象，让模型自己学习并组合出有效的特征，免去了人工特征提取的繁琐。CNN无需进行复杂且重复的图像预处理（分割）工作，可以直接使用图像的原始像素作为输入。此外，CNN还对一定程度内的平移、旋转、缩放、视觉改变、亮度调整等畸变具有不变性，有着很强的泛化性。

目前，移动端上基于CNN的应用绝大部分都是采用“客户端－服务器”模式，但是该模式不仅强依赖于网络性能（如网络稳定性等）而且会导致用户隐私泄露。因此，将CNN应用离线部署到移动端是未来的发展方向。CNN的计算量较大，参数较多，因此它要求硬件设施要有较强的算力与较大的存储资源。然而在周边应用（比如移动端，嵌入式平台）中，设备的计算能力较低，内存也较小，因此部署起来有较大难度。本文以移动端的花卉识别为例，将CNN部署到Android平台，并采用一系列优化策略来降低模型尺寸，缩短推理时间，提高模型正确率。论文的主要工作包括：

(1) 针对移动端计算能力和内存资源有限的缺点，采用了轻量级卷积神经网络架构MobileNet-V2。作为一种在尺度、速度和精度都取得了较好权衡的轻量级CNN，MobileNet-V2主要采用了深度可分解卷积来提取特征，与标准卷积相比，能成倍的减少时间复杂度和空间复杂度。此外，MobileNet-V2采用了反向残差模块与线性瓶颈层，使得网络具有较强的特征提取能力。

(2) 网络训练方面，采用结合Momentum的RMSProp优化方法来训练网络。RMSProp优化方法是一种自适应学习率算法，它能够根据历史梯度和当前梯度自动调整学习率，而Momentum因子可以很好地缓解每次迭代计算过程中产生的梯度噪音，加速网络的学习。另外本文采用同步模式的双GPU并行训练来缩短训练时间，并探索了迁移学习、L2正则化和数据增强等训练策略对最终精度的影响。

(3) 网络压缩与量化方面，采用一种高效8-bit整数运算神经网络量化方案。该方案允许使用纯整数算法进行推理，相比传统的浮点推理更有效。由于直接对训练好的浮点模型进行8-bit量化通常会损失较高的精度，因此本文采用了模拟量化训练的方法，通过在权重读取和激活输出插入一系列伪量化节点，来模拟整型网络的前向传播和反向传播过程。采用模拟量化训练后再对浮点模型进行8-bit量化，整型模型精度和浮点模型精度相当，但模型尺寸与推理时间却大大缩短。

论文在最后一章给出了一系列对比实验来证明本文算法的有效性。实验结果表明，本文设计的基于深度卷积神经网络的移动端花卉识别系统，在尺度、速度和精度等方面都具有较好的效果。

本文在某些方面仍具有进一步提高的空间，具体如下：

(1) 能够识别的花朵类别较少，离真正的产品落地还有一定距离。由于本文选用的是Oxford-102 Flower公共数据集，该数据集仅仅包含了102种英国常见花朵，种类较少；此外该样本数据比较干净，而实际中花朵照片千奇百怪，甚至会出现同一张图像包含多个不同类别的花朵，因此即便在该数据集的正确率做得再高，离真正的实用仍有一定差距。

(2) 未能发挥移动端的GPU并行计算性能。CNN的前向预测过程中存在大量的并行计算，比如每个卷积核的卷积计算其实是相互独立的。而GPU相比CPU的最大特点是有非常多的计算单元。虽然GPU的每个计算单元的计算能力相比CPU弱很多，但如果分配好每个计算单元的计算任务，结合CNN的并行性特点，使用GPU做前向预测相比CPU将会获得更短的预测时间。

参考文献

1. Nilsback M E, Zisserman A. Automated Flower Classification over a Large Number of Classes[C]// Sixth Indian Conference on Computer Vision, Graphics & Image Processing. IEEE Computer Society, 2008:722-729.
2. Chai Y, Lempitsky V, Zisserman A. BiCoS: A Bi-level co-segmentation method for image classification[C]// IEEE International Conference on Computer Vision. IEEE, 2011: 2579-2586.
3. Angelova A, Zhu S. Efficient Object Detection and Segmentation for Fine-Grained Recognition[C]// Computer Vision and Pattern Recognition. IEEE, 2013:811-818.
4. 程浩. 基于深度学习的花朵图像分类算法的研究与实现[D]. 北京航空航天大学, 2014.
5. Hu Z , Zhang Y , Tan L . Research on the Fine-grained Plant Image Classification[C]// International Conference on Machinery. 2017.
6. Xia X, Xu C, Nan B. Inception-v3 for flower classification[C]// Image, Vision and Computing (ICIVC), 2017 2nd International Conference on. IEEE, 2017: 783-787.
7. Canziani A, Paszke A, Culurciello E. An analysis of deep neural network models for practical applications[J]. arXiv preprint arXiv:1605.07678, 2016.
8. Sandler M, Howard A, Zhu M, et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.
9. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[J]. arXiv preprint arXiv:1502.03167, 2015.
10. Russakovsky O, Deng J, Su H, et al. Imagenet large scale visual recognition challenge[J]. International Journal of Computer Vision, 2015, 115(3): 211-252.
11. LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.
12. Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.
13. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
14. Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.
15. Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826.
16. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
17. Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-resnet and the impact of residual connections on learning[C]//AAAI. 2017, 4: 12.
18. Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.
19. Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J]. nature, 1986, 323(6088): 533.
20. Donahue J, Jia Y, Vinyals O, et al. Decaf: A deep convolutional activation feature for generic visual recognition[C]// International conference on machine learning. 2014: 647-655.
21. Yosinski J, Clune J, Bengio Y, et al. How transferable are features in deep neural networks?[C]// Advances in neural information processing systems. 2014: 3320-3328.
22. Iandola F N, Han S, Moskewicz M W, et al. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size[J]. arXiv preprint arXiv:1602.07360, 2016.
23. Ma N, Zhang X, Zheng H T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design[J]. arXiv preprint arXiv:1807.11164, 2018.
24. Huang G, Liu Z, Maaten L V D, et al. Densely Connected Convolutional Networks[C]// IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2017:2261-2269.
25. F. Li, B. Zhang, and B. Liu. Ternary weight networks[J]. arXiv preprint arXiv:1605.04711, 2016. 1, 6。
26. Hubara I, Courbariaux M, Soudry D, et al. Binarized neural networks[C]//Advances in neural information processing systems. 2016: 4107-4115.
27. 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: 525-542.
28. GEMMLOWP, Gemmlowp: a small self-contained low-precision GEMM library[EB/OL]. https://github.com/google/gemmlowp.
29. Intel(R) MKL-DNN, Intel(R) Math Kernel Library for Deep Neural Networks[EB/OL]. https://intel.github.io/mkl-dnn/index.html.
30. ARM, Arm cmsis nn software library[EB/OL]. http://arm-software.github.io/CMSIS 5/NN/html/index.html.
31. Qualcomm onQ blog, How can Snapdragon 845’s new AI boost your smart-phone’s IQ? [EB/OL]. https://www.qualcomm.com/news/onq/2018/02/01/how-can-snapdragon-845s-new-ai-boost-your-smartphones-iq.
32. Nvidia, 8 bit inference with TensorRT[EB/OL].

http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf.

1. Sze V, Chen Y H, Yang T J, et al. Efficient processing of deep neural networks: A tutorial and survey[J]. Proceedings of the IEEE, 2017, 105(12): 2295-2329.
2. Nvidia, The nvidia deep learning accelerator[EB/OL]. http://nvdla.org/.
3. Han S, Liu X, Mao H, et al. EIE: efficient inference engine on compressed deep neural network[C]//Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on. IEEE, 2016: 243-254.

致谢

转眼间研究生三年的学习生涯即将结束，回首过去，离不开老师、同学、朋友、家人对我支持和鼓励。

我首先想感谢我的导师李国刚老师，李老师在学术研究方面给予了我最大的自由度，让我能够追寻自己感兴趣的事物来做研究。其次我想感谢我的实习伙伴刘晨曦和徐向南同学，他们在我做项目过程中给了我大量的指导和帮助。最后我想感谢我的父母，是他们在背后默默地给予我精神与物质上的支持。

附录A Oxford-102 Flower数据集花卉类别

Oxford-102 Flower公共数据集中全部102种花卉种类的英文名称和中文名称对照如表A.1所示。

表A.1 Oxford-102 Flower花卉类别表

|  |  |  |  |
| --- | --- | --- | --- |
| 英文名 | 中文名 | 英文名 | 中文名 |
| alpine sea holly | 高山刺芹 | anthurium | 火鹤花 |
| artichoke | 朝鲜蓟 | azalea | 杜鹃花 |
| ball moss | 铁兰 | balloon flower | 桔梗花 |
| barbeton daisy | 非洲菊 | bearded iris | 鸢尾兰 |
| bee balm | 香蜂花 | bird of paradise | 鹤望兰（天堂鸟） |
| bishop of llandaff | 兰达夫主教 | blackberry lily | 射干 |
| black-eyed susan | 黑心金光菊 | blanket flower | 天人菊 |
| bolero deep blue | 波莱罗月季 | bougainvillea | 簕杜鹃（九重葛） |
| bromelia | 凤梨 | buttercup | 毛茛 |
| californian poppy | 加利福尼亚罂粟花 | camellia | 山茶 |
| canna lily | 美人蕉 | canterbury bells | 风铃草 |
| cape flower | 纳丽石蒜 | carnation | 康乃馨 |
| cautleya spicata | 红苞距药姜 | clematis | 铁线莲 |
| colt's foot | 冬花 | columbine | 耧斗草 |
| common dandelion | 西洋蒲公英 | corn poppy | 虞美人 |
| cyclamen | 仙客来 | daffodil | 水仙 |
| desert-rose | 沙漠玫瑰 | english marigold | 英格兰万寿菊 |
| fire lily | 火百合 | foxglove | 毛地黄 |
| frangipani | 鸡蛋花（赤素馨） | fritillary | 贝母 |
| garden phlox | 草夹竹桃 | gaura | 山桃草属 |

续表A.1 Oxford-102 Flower花卉类别表

|  |  |  |  |
| --- | --- | --- | --- |
| gazania | 勋章菊 | geranium | 天竺葵 |
| giant white arum lily | 马蹄莲 | globe thistle | 蓝刺头 |
| globe-flower | 金莲花 | grape hyacinth | 麝香兰 |
| great masterwort | 大星芹 | hard-leaved pocket orchid | 硬叶兜兰 |
| hibiscus | 木槿（芙蓉） | hippeastrum | 朱顶红（孤挺花） |
| japanese anemone | 秋牡丹（日本银莲花） | king protea | 菩提花（帝王花） |
| lenten rose | 菟葵 | lotus | 莲花 |
| love in the mist | 黑种草 | magnolia | 木兰花 |
| mallow | 锦葵 | marigold | 金盏花 |
| mexican aster | 大波斯菊 | mexican petunia | 碧冬茄 |
| monkshood | 舟形乌头(毒草） | moon orchid | 蝴蝶兰 |
| morning glory | 牵牛花 | orange dahlia | 橙色大丽花 |
| osteospermum | 蓝眼菊 | oxeye daisy | 滨菊 |
| passion flower | 西番莲 | pelargonium | 天竺葵 |
| peruvian lily | 水仙百合 | petunia | 矮牵牛 |
| pincushion flower | 蓝盆花属 | pink primrose | 粉色报春花 |
| pink-yellow dahlia | 粉黄大丽花 | poinsettia | 猩猩木（一品红） |
| primula | 报春花 | prince of wales feathers | 威尔士亲王羽毛 |
| purple coneflower | 紫松果菊 | red ginger | 红姜花 |
| rose | 玫瑰 | ruby-lipped cattleya | 卡特兰 |
| siam tulip | 姜荷花 | silverbush | 银旋花 |
| snapdragon | 金鱼草 | spear thistle | 翼蓟 |
| spring crocus | 藏红花 | stemless gentian | 矮龙胆 |
| sunflower | 向日葵 | sweet pea | 麝香豌豆花 |
| sweet william | 美洲石竹 | sword lily | 剑兰 |

续表A.1 Oxford-102 Flower花卉类别表

|  |  |  |  |
| --- | --- | --- | --- |
| thorn apple | 曼陀罗 | tiger lily | 虎皮百合（卷丹） |
| toad lily | 油点草 | tree mallow | 树锦葵 |
| tree poppy | 马梯里亚罂粟 | trumpet creeper | 凌霄花（紫薇） |
| wallflower | 桂竹香（香紫罗兰） | water lily | 睡莲（荷花） |
| watercress | 豆瓣菜 | wild pansy | 三色堇 |
| windflower | 银莲花（白头翁） | yellow iris | 黄鸢尾 |

附录B 相关实验开源代码

高效8-bit整数运算神经网络：

https://github.com/chh175/uint8\_cnn

移动端花卉识别安卓工程：

https://github.com/chh175/fower\_classify\_AndroidDemo

在学期间的研究成果及发表的学术论文

获奖：

1. “华为杯”第十二届中国研究生电子设计竞赛Intel组全国三等奖. 2017.8