全网学员超过100多万的老师，
带你学会PyTorch，成为数据科学家！

Broadview

PyTorch 深度学习实战

从新手小白到数据科学家

张敏　编著

从基础到实战，带你系统学习PyTorch

中国工信出版集团
电子工业出版社

张敏，2015年开始从事大数据行业，多年来一直致力于数据分析和算法开发。2017年开始接触在线教育，目前全网学员总数破100万，相继在51CTO、腾讯课堂、网易课堂、CSDN等平台开启教学。在51CTO平台上的SPARK课程销量做到行业类目前列，曾在多家大公司从事数据挖掘工作，拥有丰富的大数据经验和底层数据思维。在工作中总结了很多丰富的实战经验，都毫无保留地分享给读者，以帮助读者提升、成长。

微信公众号

版权信息

COPYRIGHT

书名：PyTorch深度学习实战：从新手小白到数据科学家
作者：张敏
出版社：电子工业出版社
出版时间：2020年8月
ISBN：9787121388293
字数：538千字

内容简介

PyTorch作为深度学习领域逐渐崛起的新星，其易用性及Python友好性深受广大算法爱好者的喜爱，无论是在学术领域还是在工业领域，PyTorch都已经成为算法研究的首选。

本书以深度学习为核心，详细讲解PyTorch技术堆栈，力求使用最直白的语言，带领更多的小白学员入门直至精通深度学习。本书包括10章，前5章主要讲解深度学习中的基本算法及概念，通过使用PyTorch实现经典的神经网络并辅以“加油站”补充数学知识，力求使每个知识点、每个章节、每个实验都能在小白学员脑海中留下深刻的印象，做到看了能做、做了能会、会了能用。后5章作为PyTorch的进阶，主要介绍使用PyTorch构建深度神经网络、使用HMM实现中文分词、训练对话机器人等实验。

本书引入了当下非常流行的自然语言预训练模型，如ELMo、BERT等，使读者能够使用自然语言利器AllenNLP及高层框架FastAI。最后讲解当下非常流行的大型图嵌入技术，对知识图谱感兴趣的读者可以做深入研究。

前言

近年来，随着大数据、机器学习、深度学习等技术的发展，以及物理硬件性能的提升，越来越多优秀的大数据人工智能项目落地，为社会带来巨大便利的同时也为从事该行业的技术人员带来了前所未有的机遇。越来越多的公司和个人开始关注并学习AI知识，各大厂商也在AI技术及人才培养上投入巨资，纷纷抢占战略高地。笔者所在的公司也成立了AI研究室，从事算法及大数据相关的研究及落地工作，在此过程中会涉及很多开源的框架及技术，由于笔者有教育行业的从业经历，所以想通过出版图书的途径帮助更多的学生及普通的程序员学习AI技能。时势造英雄，希望更多的人可以把握当前的时势，蓄势待发，冲刺人生的高峰。

为了迎合各个层次读者的需求，笔者在编写过程中既兼顾内容的质量，也注重趣味性。鉴于此，笔者采用项目驱动的方式编写本书，部分章设有“实验室小试牛刀”环节，以项目实战的形式总结该章所学知识，并引出下一章的内容。“实验室小试牛刀”环节是实验性质的实战项目，兼顾趣味性，主要培养读者的动手能力。“加油站”部分回顾必要的数学知识，在学习新知识的同时，使读者可以夯实数学基础，为走得更轻松、更长远积蓄能量。

本书分为10章，内容循序渐进，由浅入深。

第1章首先介绍AI发展简史、主流的深度学习框架和框架的演变；其次介绍安装PyTorch环境及开发工具；再次介绍核心概念，包括PyTorch核心模块、自动微分等，在“实验室小试牛刀”环节，使用PyTorch进行科学计算和薪酬预测，读者可以由此了解PyTorch关于科学计算的知识及初探使用PyTorch构建算法模型；最后补充数学知识，为读者学习第2章的内容奠定基础。

第2章是机器学习基础章节，讲述机器学习的分类、常见概念，扫清读者学习途径上的认知障碍，帮助读者更好地理解机器学习与深度学习。本章还会引出机器学习和神经网络算法，使读者对算法发展史有清晰的认识，同时引出神经网络的概念。神经网络的灵感来自人类对大脑神经的生物学发现，通过模拟人类大脑的工作原理，构建出能拟合任意复杂度的深度神经网络。神经网络的训练需要借助BP算法，该算法会涉及微分、导数、偏导数等数学知识，读者可以在“加油站之高等数学知识回顾”部分再次进行复习和巩固。

第3章详细介绍PyTorch中的Tensor及科学计算算子，通过详细地介绍每个算子的用法及功能，读者可以掌握使用PyTorch的法宝。本章还介绍了PyTorch中的变量Variable，Variable是Tensor的包装类型，为PyTorch进行深度学习的训练提供支持。PyTorch中的算子和Python科学计算库NumPy提供的算子非常相似，实际上，PyTorch中的Tensor和NumPy能够进行高效互转，在本章的“实验室小试牛刀之轻松搞定图片分类”环节，读者可以使用PyTorch构建神经网络，探索PyTorch构建神经网络的简捷性及调试的方便性。

第4章主要介绍激活函数、损失函数、优化器及数据加载等相关知识点。PyTorch可以提供高层次的API封装，在torch.optim、torch.nn等模块中，提供了更加简捷的方法调用。torch.optim模块封装了常用的优化算法，如随机梯度下降（Stochastic Gradient Descent，SGD）算法、Momentum梯度下降算法等。torch.nn模块提供了常见的神经网络层，如线性层、一维卷积、二维卷积、长短期记忆（Long Short-Term Memory，LSTM）和门控循环单元（Gated Recurrent Unit，GRU）等。所有网络的学习，最终的输出结果都要使用一个度量的函数测评效果，这个函数通常称为损失函数。通过损失函数及BP算法，可以将损失进行逆向传播，每步传播都会计算各个特征权重的偏导数，进而更新特征权重，达到训练的目的。本章会详细讲解各种损失函数及其应用场景，相信读者会有所领悟。所有模型在构建完成后，都将传入数据进行训

练，而数据的加载有时是很麻烦的，PyTorch提供了DataLoader等接口，这些接口实现了生成器模式，因此适用于大数据量的加载，不会出现内存溢出的问题。在“实验室小试牛刀”环节，会探索第一个深度神经网络VGG-16，并且使用PyTorch计算机视觉库简化网络的构建过程，在构建过程中会冻结VGG-16的网络层权重，更改VGG-16最后的输出层，使其适用于特定的应用场景，而这种技术被称为迁移学习。通过实践迁移学习，读者可以掌握复用预训练模型并用于特定应用场景的核心技巧。

第5章主要讲计算机视觉，本章介绍典型的计算机视觉模型，包括DCGAN、ResNet、Inception、DenseNet等；除此之外，LSTM和GRU也是深度神经网络序列模型中经常出现的结构。本章通过TensorBoard和Visdom可视化训练过程，帮助读者理解网络内部正在发生什么。

第6章安排的内容和自然语言处理相关。自然语言处理是人工智能最后需要解决的难题，被誉为“人工智能皇冠上的明珠”。自然语言处理的内容包括语义理解、文本分类、分词、词性标注、语义消歧等。每个问题都有亟须解决的难题，如中文分词、自然语言理解等。虽然自然语言领域出现了很多优秀的算法和思路，如BERT、LSTM、TextRank等，但是效果仍然欠佳，因此这是一个需要持续被关注的研究领域。本书关注的是如何带领读者使用PyTorch进行自然语言处理，使用PyTorch已经实现的算法和工具解决业务中遇到的问题。本章不仅简单介绍了自然语言的发展及现在所处的阶段，还介绍了TextRank算法、HMM在分词等方面的应用，并使用HMM实现中文分词算法。在“实验室小试牛刀之对话机器人”环节，我们会创建一个对话机器人，希望读者可以用自己的语料训练出一个有趣的对话机器人。

第7章是自然语言处理技术的扩展章节，主要介绍当前非常流行的自然语言预训练模型，包括词嵌入、ELMo、GPT、BERT等。通过对比各个模型框架的演变，读者可以了解自然语言处理技术的发展方向，为深入研究自然语言处理奠定基础。

第8章主要介绍AllenNLP。AllenNLP是构建在PyTorch之上的高层架

构，提供了常见且丰富的自然语言处理功能。从事自然语言处理工作的读者都应该关注AllenNLP，因为其提供的接口方便、灵活。

第9章主要介绍FastAI高层深度学习框架，用10行代码解决深度学习难题，其底层基于PyTorch实现。本章在FastAI框架中使用BERT中文预训练模型完成中文文本分类任务，相信读者会喜欢其编码。

第10章介绍PyTorch Big Graph嵌入，带领读者领略图挖掘的前沿思想。“知识图谱”的概念悄悄地走进了人们的视野，各大公司和实验室都在使用深度学习技术构建各个领域的知识图谱，而基于知识图谱的知识挖掘无疑成了下一个研究的热点。

本书涉及的内容不仅包括计算机视觉、自然语言处理等深度学习应用的前沿技术，还涉及基本数学公式的推导，使读者不仅会用，还知其原理，知其然也知其所以然。本书内容丰富，涉及面广，难免有疏忽遗漏之处，望广大读者批评指正。

另外，本书涉及的公式，若不做特殊说明，则对数log的底数默认为自然常数e。

第1章 初识PyTorch

PyTorch是一个经过市场上无数从业者筛选的深度学习框架，提供了健全的神经网络接口，其动态网络结构及Python友好性，获得了大量深度学习从业人员的青睐。本章将简述深度学习及其框架的发展史，而后动手搭建PyTorch环境，讲解PyTorch核心概念、自动微分、核心架构，在“实验室小试牛刀”环节使用PyTorch完成复杂的科学计算，搭建能预测薪酬的回归模型，“加油站之高等数学知识回顾”部分补充介绍了一部分数学知识，为读者学习后续内容奠定基础。

1.1 神经网络发展简史

神经网络的发展同人类发展一样，经历了各种起伏和挫折、淘汰和进化。正是发展过程中的一些意外成就了人工神经网络今天的辉煌，正如人类进化过程中所遇到的意外一样，充满了不确定的意外和惊喜。正是有了不确定才出现了另外一个词，叫希望。

1.1.1 神经网络的前世今生

人工智能的起源可能要追溯到第二次世界大战，其间涌现出很多优秀的科学家，包括神经学家格雷·沃尔特和数学家艾伦·图灵。其中，沃尔特制造了人类历史上的第一个机器人；图灵则提出了一种测试，这种测试旨在验证机器是否可以达到真正的智能，而这种测试后来被称为图灵测试。图灵测试的思想是测试者在与被测试者（一个人和一台机器）隔开的情况下，通过一些装置（如键盘）向被测试者随意提问，进行多次测试后，如果有超过30%的测试者不能确定被测试者是人还是机器，那么这台机器就通过了测试，并且被认为具有人类智能。

1956年，在美国达特莫斯大学举行了两个月的学术讨论会，从不同学科的角度探讨用机器模拟人类智能等问题，并且首次提出了人工智能（AI）的术语，同时出现了最初的成就和最早的一批研究者，这一事件被看作AI诞生的标志。由于计算机的产生与发展，人们开始了具有真正意义的人工智能的研究。

1957年，罗森·布拉特基于神经感知科学的背景提出并设计出第一个计算机神经网络模型，这个模型模拟了人类大脑的生物学特性，被称

为感知机（Perceptron）。

1960年，维德罗首次将Delta学习规则用于感知机的训练步骤，这种方法后来被称为最小二乘方法。最小二乘法和感知机的结合创造了一个良好的线性分类器。

1968年，马文·明斯基将感知机推到顶峰。他提出了著名的异或（XOR）问题和感知机数据线性不可分的情形，在数学上证明了单层或多层网络的堆叠无法解决数据线性不可分的问题（后来人们通过引入激活函数来解决线性不可分的问题）。此后，神经网络的研究长期处于休眠状态。

1970年，林纳因马提出反向传播（Back Propagation, BP）神经的概念，并将其称为自动分化反向模式，但是并未引起足够的关注。

1980年，在美国的卡内基梅隆大学召开了第一届机器学习国际研讨会，标志着机器学习研究已在全世界兴起。此后，机器归纳学习进入应用。经过一些挫折后，多层感知机（Multilayer Perceptron, MLP）由伟博斯在1981年的神经网络BP算法中具体提出。有了反向传播的新思想，神经网络的研究进程逐渐加快。

1983年，J. J. Hopfield和D. W. Tank建立了互相连接的神经网络模型，被称为霍普菲尔德神经网络，并用它成功地探讨了旅行商（TSP）问题的求解。

1986年，D.E.Rumelhart等人实现了多层网络BP算法，把学习结果反馈到中间层次的隐藏单元中，改变它们的联系矩阵，从而达到预期的学习目的。迄今为止，它仍是应用最广泛的神经网络训练更新算法。

20世纪90年代，统计学习出现并迅速占领了历史舞台，支持向量机（Support Vector Machine，SVM）就是这一时代的产物，SVM在数据分类、图像压缩等应用领域取得了非常好的成绩。

2006年，Hinton及其学生发表了利用（Restricted Boltzmann Machine）自编码的深层神经网络的论文——*Reducing the Dimensionality*

of Data with Neural Networks，目前这篇论文的实用性并不强，它的作用是把神经网络又拉回人们的视线中，利用单层的RBM自编码预训练使深层的神经网络训练变成可能，但那时候深度学习（Deep Learning）的争议很多。

2016年被称为人工智能的元年。在这一年，AlphaGo在围棋比赛中赢了专业棋手李世石，打破了人类自尊的最后的堡垒。

目前，深度学习在不经意间就会出现在我们的日常生活中——它是Google的声音和图像识别，是Netflix和Amazon的推荐引擎，是Apple的Siri，是电子邮件和短信的自动回复，是智能聊天机器人，是百度的智能导航，是阿里巴巴的自动风险感知，是字节抖动的智能推荐。神经网络的发展史如图1.1所示。

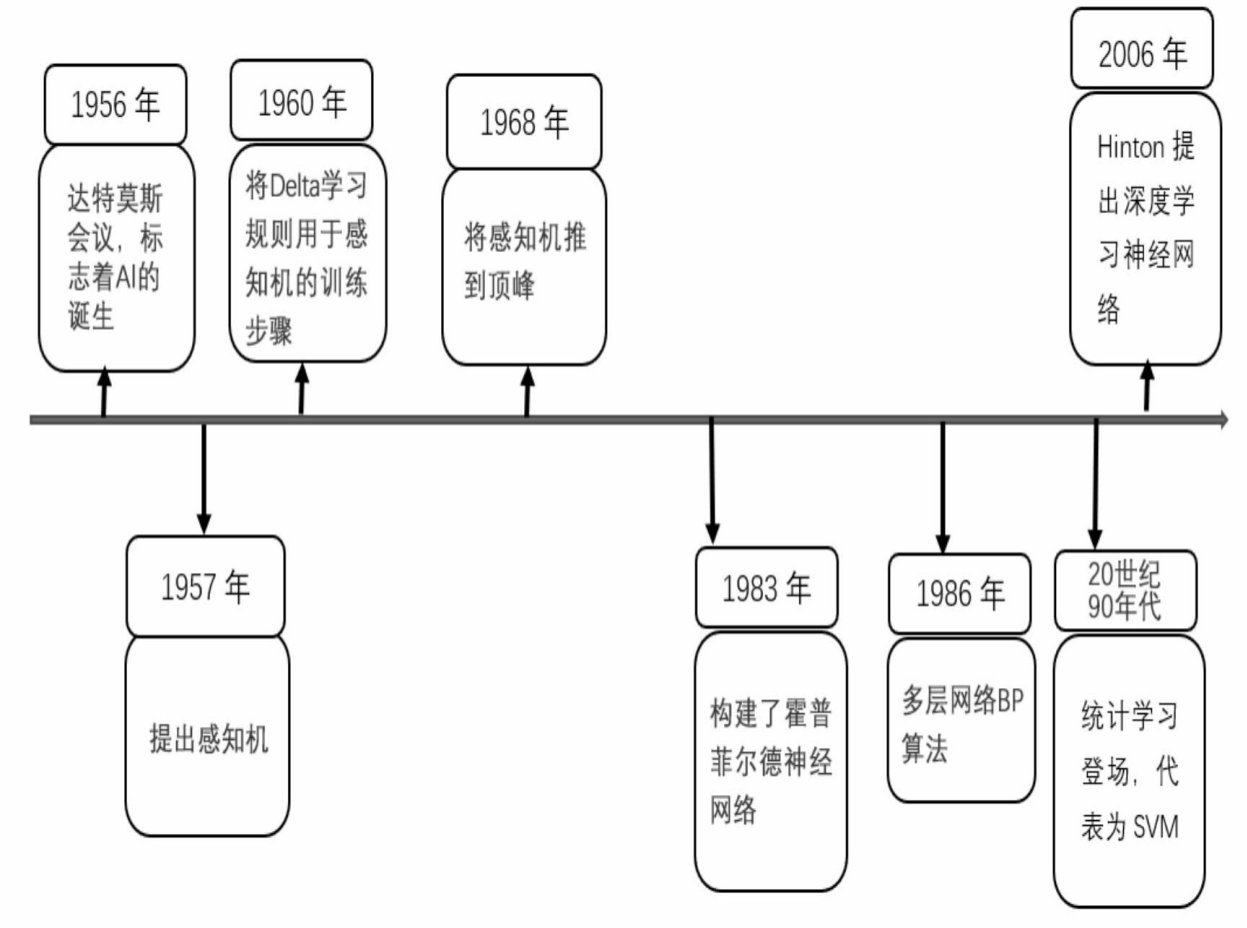


图1.1 神经网络的发展史

1.1.2 深度学习框架对比

深度学习框架发展迅速，各种语言实现的框架层出不穷，最广为人知的是TensorFlow，但最好用且Python友好又便于调试的首推PyTorch。TensorFlow是使用人数最多的深度学习框架，但其是基于静态图定义的框架，使用和调试都非常困难。因为静态图首先需要定义，定义好之后就不能修改。要定义静态图涉及很多特殊的语法及概念，提升了学习门槛，对于初学者来说，无异于学习一门新的语言。而PyTorch是基于动态图构建的，可以使用Python一般的语法，如If/Else/While/For等，天然的Python友好且便于调试，熟悉Python的人们可以快速上手。

测试一个框架热门程度的最直接方式就是搜索指数。通过对比GitHub统计指数，可以直观地发现一个框架的流行程度，如图1.2所示是最常用的几个深度学习框架TensorFlow、Keras、PyTorch、Theano的GitHub统计指数，通过对比可以发现一些趋势。

框架	2018年和2019年GitHub增长率		2019年7月		2017年9月		发布时间	公司	开发语言
	Star	Fork	Star	Fork	Star	Fork			
PyTorch	303%	394%	29635	7186	7361	1456	2016	Facebook	C++和Lua
TensorFlow	87%	121%	130540	75929	69781	34355	2015	Google	C++
PaddlePaddle	71%	72%	9246	2484	5405	1447	2016	Baidu	C++
MXNet	56%	47%	17316	6145	11127	4179	2017	Apache	C++
DL4J	53%	31%	10966	4697	7175	3590	2014	Eclipse	Java
Caffe2	50%	73%	8458	2130	5628	1233	2017	Facebook	C++
Caffe	41%	39%	28495	17204	20155	12371	2014	Berkeley Vision	C++
CNTK	31%	35%	16258	4318	12366	3190	2016	Microsoft	C++
Theano	28%	9%	8834	2493	6902	2290	2007	MILA	Python
Keras	-	-	42504	16187	-	-	2015	Google	Python
Chainer	-	-	4887	1294	-	-	2015	Chainer	Python

图1.2 深度学习框架GitHub统计指数变化趋势

在Star和Fork的数量上，TensorFlow在深度学习领域用得最多，但从趋势来看其地位是否能在接下来继续保持第一位仍然值得关注，因为排名靠后的PyTorch、PaddlePaddle、MXNet等深度学习框架的影响度正在逐年上升。深度学习框架已经变成TensorFlow和PyTorch之争，也是动态图与静态图之间的“斗争”。PyTorch能否登顶非常值得期待。

下面笔者以各个深度学习框架出现的先后顺序进行对比分析。

分析各个框架出现的时间会发现一个很有意思的规律，在2015—2016年，不管是深度学习框架还是其他大数据框架（如Spark等），都呈现爆发式增长。这与2015年和2016年大数据及AI市场得到资本青睐息息相关，被投资的团队开始集中攻克技术难点及开源技术产品，所有的读者都是开源的受益者。

2008年1月，Theano诞生于蒙特利尔大学的LISA实验室。它是一个Python库，结合计算机代数系统CAS和编译器的优化，可以达到与C语言媲美的速度，可以用于定义、计算数学表达式，尤其是计算多维数组。Theano中的多维数组类似于NumPy中的Ndarray。因为Theano诞生于研究机构，具有浓厚的学术气息，并且很难调试，图的构建非常困难，缺乏文档，以及疏于宣传等缺点，所以使用的人很少。2017年9月，LISA实验室负责人Yoshua Bengio宣布“Theano is Dead”，在Theano 1.0正式发布后将不再进行维护。但是Theano作为第一个Python深度学习框架，其退出历史舞台时已完成了它的历史使命，为其他深度学习框架指明了设计方向：以计算图为核心，采用GPU设备加速。后来出现的深度学习框架（如TensorFlow、PyTorch、Caffe/Caffe2等）都采用计算图的方式来定义计算的Pipeline。

2012年1月，Torch项目诞生于纽约大学，Torch采用的是Lua语言，但是使用Lua语言的人不多，因此Torch这个深度学习框架在市场上应用得很少。但是Torch的幕后团队在2017年基于Python语言对Torch的技术架构进行了全面的重构，于是诞生了当下非常流行的动态图框架PyTorch。

2013年12月，使用Java语言开发的深度学习框架Deeplearning4j诞生，Deeplearning4j是使用Java语言开发的深度学习框架，同时支持Clojure接口和Scala接口，由开源科学计算库ND4J驱动，可选择在CPU设备或GPU设备上运行。Deeplearning4j还可以运行在Hadoop/Spark集群中，可以提高运行速度及大数据处理能力。

2013年9月，加利福尼亚大学伯克利分校的贾扬清博士在GitHub上首度发布Caffe，它的全称是Convolutional Architecture for Fast Feature Embedding，是一个清晰、高效的深度学习框架，核心语言采用C++，支持命令行、Python、MATLAB接口，可以选择在CPU设备或GPU设备上运行。Caffe使用简单，代码也易于扩展，运行速度得到了工业界的认可，社区也非常庞大。2017年4月发布的Caffe2是Caffe的升级优化版。

2015年3月诞生的Keras是一个高层的神经网络API，使用纯Python语言编写。它并不是独立的深度学习框架，后端采用TensorFlow、Theano及CNTK作为支持。Keras为实验室而生，能够将想法快速转换为结果，高层的API非常易于使用，因此是所有深度学习框架中最易上手的一个。Keras并不独立，是在其他框架上的层层封装。层层封装导致Keras运行速度慢，扩展性也不好，很多时候BUG被隐藏在封装之中，因此缺少灵活性，使用Keras很快将遇到瓶颈。

2015年5月MXNet诞生了。虽然由一群学生开发，但是MXNet拥有超强的分布式支持，对内存及显存的优化也非常好。2016年11月，MXNet被AWS正式选择为云计算官方深度学习平台。2017年1月MXNet进入Apache基金会，成为Apache孵化器项目。但是MXNet一直不温不火，主要是因为文档维护缓慢而代码更新过快，让老用户面对新接口也不知道如何使用，另外一个原因要归结于推广不力。MXNet是中国人开发的项目，笔者希望它能越来越好。

2015年11月，Google官宣TensorFlow开源。TensorFlow的开源确实在业界引起了不小的轰动，TensorFlow的大火得益于Google在深度学习

领域巨大的影响力及强大的推广能力。目前，TensorFlow是用户数最多的深度学习框架，支持Python及C++编程接口，并且采用C++的Eigen库，所以可以在ARM架构上编译和优化。用户可以在各种服务器和移动设备上部署训练模型而无须执行单独的解码器或Python的解释器。业界对TensorFlow的褒贬不一，其中最大的问题是频繁变动的接口；另外一个问题是复杂的系统设计，TensorFlow源码超过百万行，想通过阅读源码弄懂底层的运行机制是非常痛苦的事情。

2016年1月Microsoft发布的CNTK支持在CPU设备和GPU设备上运行，并且把神经网络描述成一个计算图结构，这一点与其他的深度学习框架是一致的。CNTK的性能比Caffe、Theano、TensorFlow的都好，但是其文档设计晦涩难懂，推广也不给力，因此用户较少。

2016年8月，百度开源PaddlePaddle深度学习框架，PaddlePaddle的前身是百度于2013年自主研发的深度学习平台，并且一直被百度内部工程师研发使用。全球各大科技巨头开源的深度学习平台都极具各自的技术特点，对于百度，由于其自身具有搜索、图像识别、语音语义识别理解、情感分析、机器翻译、用户画像推荐等多领域的业务和技术方向，所以PaddlePaddle的表现更加全面，是一个功能相对较全的深度学习框架。

总之，在这个发生翻天覆地变化的年代，只有不断地学习和进步才能赶上时代的步伐，只有抓住机遇才能更好地迎接挑战。

1.2 环境安装

工欲善其事，必先利其器。道路千万条，环境是第一条，环境是学习的第一步。本节主要介绍Python环境的选择及安装、PyTorch 1.2的安装、开发环境IDE。

1.2.1 Python环境的选择及安装

PyTorch既支持Python 2.7也支持Python 3.5+，由于Python 2在2020年1月1日停止更新，所以笔者推荐使用Python 3以上版本。

安装Python其实有不同的方法，最简单直接的方式是在Python官网上选择合适的版本下载并且安装。本节选择Python 3.6.5，读者也可以根据需求自行选择。可供选择的Python版本如图1.3所示。

下载好之后，直接单击安装。安装时注意勾选“Add Python 3.6 to PATH”选项，如图1.4所示，这样Python安装好之后将被添加到环境变量中，而不用手动添加。

单独安装Python比较简单，但安装常见的库（如Pandas、NumPy等）比较麻烦。最常见的安装方式是通过第三方打包好的软件统一安装，如通过Anaconda。它将常见的Python包打包发布，解决了独立安装时所遇到的版本冲突问题，唯一的麻烦就是安装包较大，约为700MB。在Anaconda官网下载区根据不同的平台选项选择不同的安装包。本书统一使用Windows平台，Anaconda目前提供的最新的Python版本是Python 3.7，如图1.5所示。

Release version	Release date		Click for more
Python 3.7.2	2018-12-24	Download	Release Notes
Python 3.6.8	2018-12-24	Download	Release Notes
Python 3.7.1	2018-10-20	Download	Release Notes
Python 3.6.7	2018-10-20	Download	Release Notes
Python 3.5.6	2018-08-02	Download	Release Notes
Python 3.4.9	2018-08-02	Download	Release Notes
Python 3.7.0	2018-06-27	Download	Release Notes
Python 3.6.6	2018-06-27	Download	Release Notes
Python 2.7.15	2018-05-01	Download	Release Notes
Python 3.6.5	2018-03-28	Download	Release Notes

图1.3 可供选择的Python版本

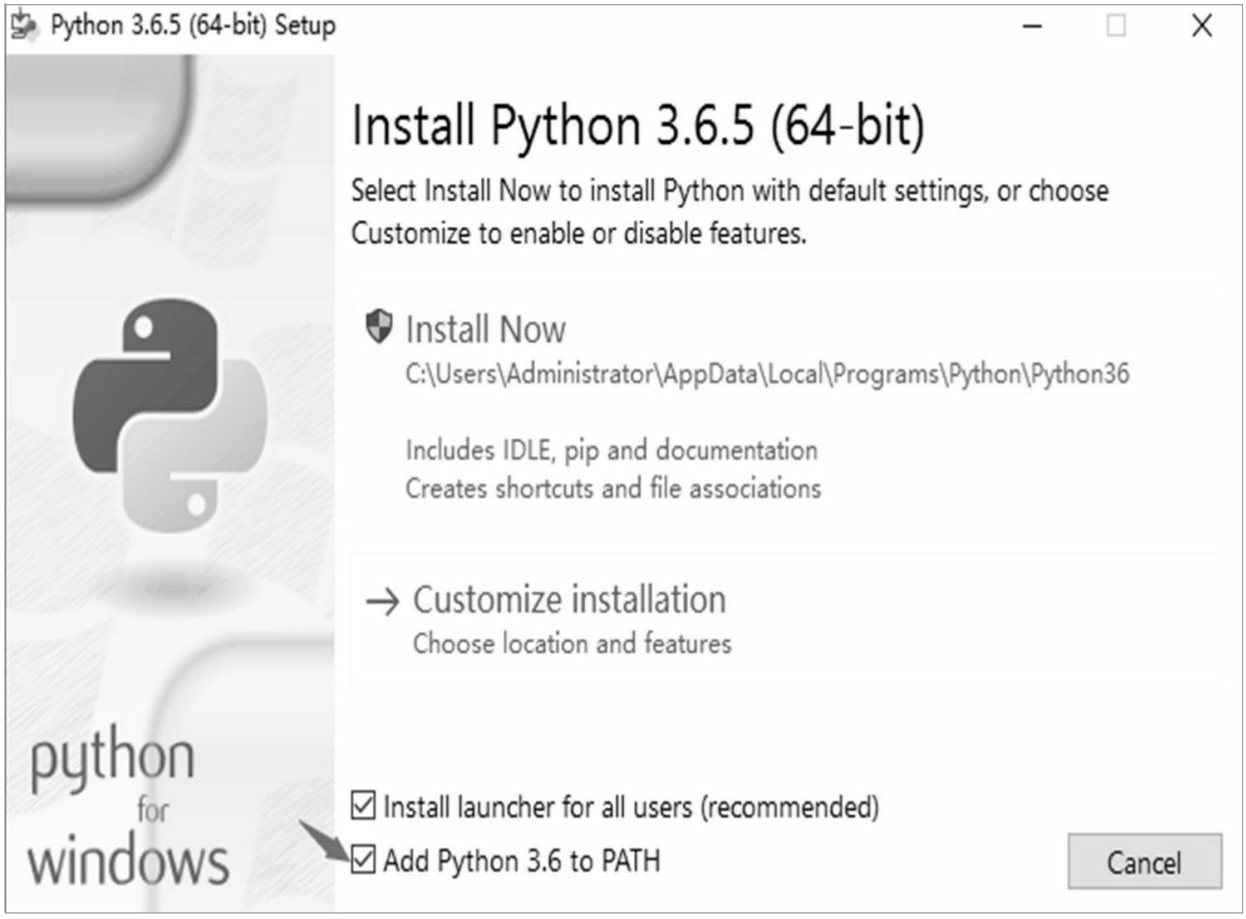


图1.4 勾选“Add Python 3.6 to PATH”

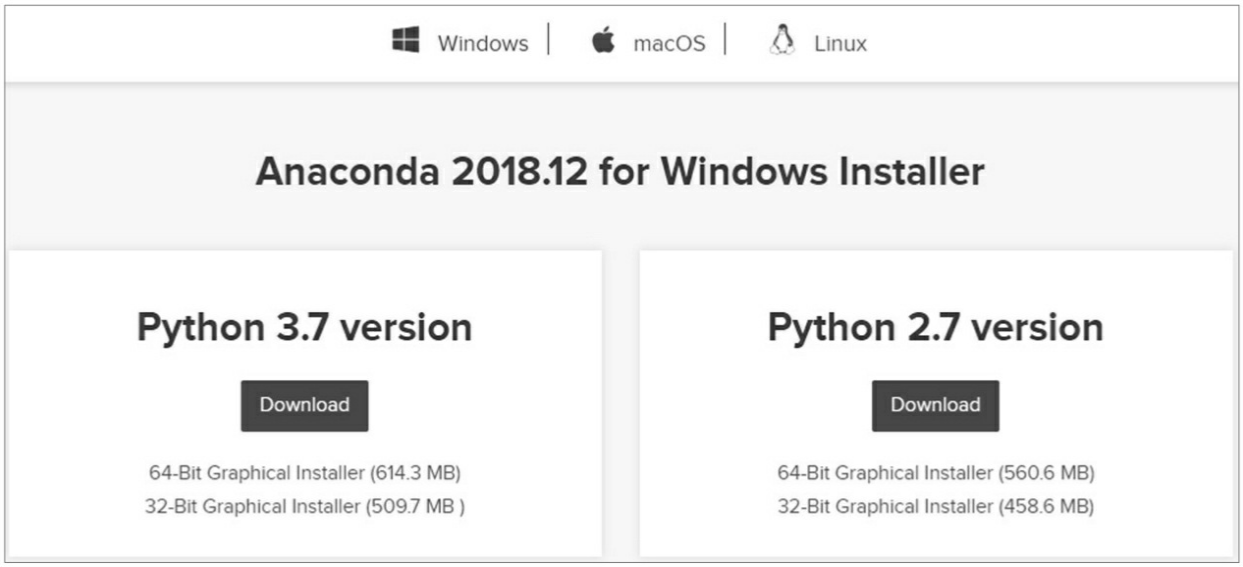


图1.5 下载Anaconda

选择下载64-bit的安装包，等待下载完成后直接单击安装。安装过

程中需要勾选“Add Anaconda to the system PATH environment variable”，如图1.6所示，这样Anaconda中的Python就能够在计算机中任意位置被访问到。

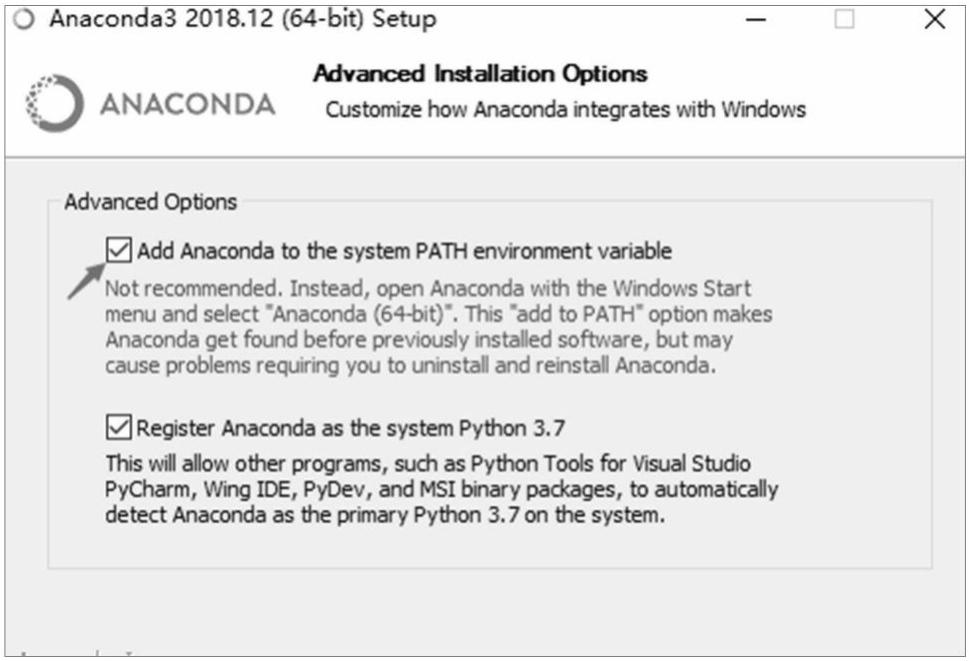


图1.6 勾选“Add Anaconda to the system PATH environment variable”

安装完成之后，在CMD命令行中输入“Python”查看Python版本号。

```
    Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] ::
Anaconda, Inc. on win32
    Type "help", "copyright", "credits" or "license" for more information.
```

采用Anaconda的安装方式，NumPy、Pandas等这些常见的包都已经装好，接下来安装PyTorch。

1.2.2 PyTorch 1.2的安装

基于Python提供的pip工具或使用Anaconda提供的Conda命令，可以非常方便地安装其他Python库。PyTorch官网上提供了安装的选择，可以选择不同的操作系统，采用不同的安装方式，选择不同的

PyTorch/Python版本，以及是否选择CUDA提供的GPU设备支持。PyTorch安装选项卡如图1.7所示。

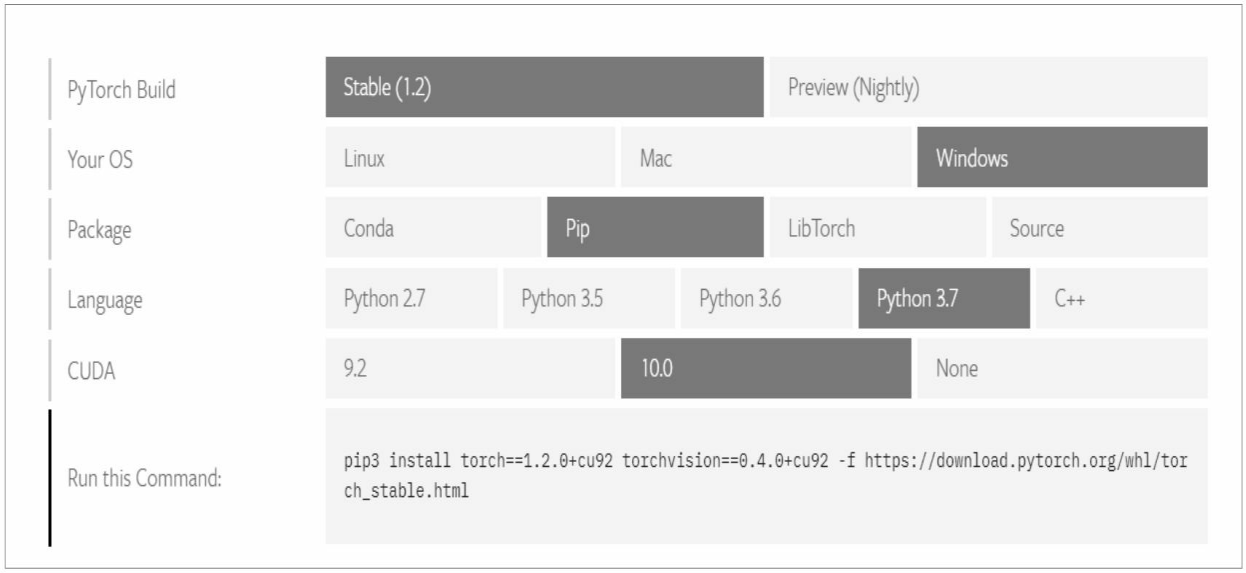


图1.7 PyTorch安装选项卡

在选项卡中选好配置后，选项卡下面的“Run this Command”栏中就会生成相应的安装命令，将其复制到CMD命令行中运行即可完成PyTorch的安装。使用pip3可能会报错，可以将pip3改为pip，到CMD命令行中执行即可。下面CMD命令行中的第二个pip安装的是PyTorch实现并训练好的一些关于计算机视觉处理的模型，如VGG-16、DCGAN等，可以基于这些模型进行微调，通过迁移学习技术快速满足业务需求。

```
    C:\Users\Administrator>pip  install  https://download.pytorch.org/whl/cpu/
torch-1.0.1-cp37-cp37m-win_amd64.whl
    Looking in indexes: http://pypi.douban.com/simple/
    Collecting torch==1.0.1 from https://download.pytorch.org/whl/cpu/torch-
1.0.1-cp37-cp37m-win_amd64.whl
      Downloading https://download.pytorch.org/whl/cpu/torch-1.0.1-cp37-cp37m-
win_amd64.whl (73.6MB)
       100% |████████████████████████████████████████| 73.6MB
229kB/s
    Installing collected packages: torch
    Successfully installed torch-1.0.1
    C:\Users\Administrator>pip install torchvision
    Looking in indexes: http://pypi.douban.com/simple/
    Collecting torchvision
      Downloading http://pypi.doubanio.com/packages/ca/0d/f00b2885711e08bd71242
ebe7b96561e6f6d01fdb4b9dcf4d37e2e13c5e1/torchvision-0.2.1-py2.py3-none-any.whl
(54kB)
       100% |████████████████████████████████████████| 61kB 722kB/s
    Installing collected packages: torchvision
    Successfully installed torchvision-0.2.1
```

pip默认使用国外的镜像，速度较慢，通常推荐使用国内的豆瓣镜像。要使用豆瓣镜像需要配置pip.ini文件。首先进入用户根目录，笔者这里是“C：\Users\Administrator”，然后新建pip文件，在pip文件夹中新建pip.ini配置文件，在配置文件中配置国内的豆瓣镜像，具体配置如下。

```
    [global]
    index-url = http://pypi.douban.com/simple/
    trusted-host = pypi.douban.com
```

国内镜像无论是安装PyTorch还是其他的Python库速度都很快，网络好时速度可以达到5MB/s。对于学习Python的读者，一定要配置Python镜像源。安装好PyTorch之后，在CMD命令行中输入“ipython”，

打开Python终端检验PyTorch是否安装成功。

```
C:\Users\Administrator>ipython
In [1]: import torch
In [2]: torch.Tensor(3,3)
Out[2]:
tensor([[1.3733e-14, 6.4069e+02, 4.3066e+21],
        [1.1824e+22, 4.3066e+21, 6.3828e+28],
        [3.8016e-39,       nan, 9.0595e-14]])
```

至此，PyTorch已安装好，但要完成程序的开发，通常会借助一些IDE环境，这样做的好处很多，如断点调试、语法纠错、自动提示等，因此好的IDE环境可以减少潜在的开发效率。下面为读者推荐几款比较好用的编程环境。

1.2.3 开发环境IDE

第一个是网页版交互式的编辑工具Jupyter Notebook，其支持自动补全、内嵌图表等，不仅可以编辑Python代码，还支持MarkDown语法，是非常理想的教学实验工具。基于Anaconda的安装方式，实际上已经安装好Jupyter。如果没有安装好，可以直接使用pip install jupyter来完成安装。在CMD命令行中输入“jupyter notebook”就可以启动一个网页版的编辑器。其常见的使用技巧请参考随书代码chapter1文件夹中的课件IDE-introduce-xxx.html。

IDE环境有很多，如果读者已经有了熟悉的Python编辑环境，可以直接略过本节。但本节主要介绍本书所使用的IDE环境“Wing IDE 6.1”，这是一款小巧但功能强大的编辑环境，笔者非常喜欢它的断点调试功能，可以在其官网上下载。Wing IDE的常见使用技巧参见chapter1文件夹中的课件IDE-introduce-xxx.html。

另一个比较常用的IDE环境就是PyCharm，它拥有强大的插件选择

和断点调试功能。读者可以自行在PyCharm官网下载，本节不再赘述。

IPython是一个交互式的Python执行环境，支持Tab自动补全，可以非常方便地进行代码的快速尝试和验证。通过安装Anaconda已经自动安装好IPython。如果没有安装好，可以使用pip install ipython进行安装，其提示效果如下。

```
In [3]: a = torch.Tensor(3,3)
In [4]: a.
 abs()     addcmul()     any()        atan2()
 abs_()    addcmul_()    apply_()     atan2_()
 acos()    addmm()       argmax()     atan_()
```

1.3 PyTorch的核心概念

本节主要介绍PyTorch的基本概念（如Tensor和Variable）、自动微分和PyTorch的核心模块。

1.3.1 PyTorch的基本概念

和其他深度学习框架一样，PyTorch在实现过程中也提出了自己的概念，包括张量。PyTorch中的张量用Tensor表示。初学者可能不知道张量是什么。其实，张量可以简单地理解为一个多维数组，类似于NumPy中的ndarray对象。

多维数组可以用相册来形象地解释。假设小米有相册A，相册A包含N张图片，每张图片的宽度为W、高度为H。由于是彩色图片，在计算机中用RGB三通道表示（所有颜色都可以用红、绿、蓝三原色按照不同的比例调配出来，红、绿、蓝三通道不同的像素亮度值叠加在一起就呈现出不同的颜色，所以整张图片呈现出彩色），所以通道数为3。

PyTorch采用四维数组表示这个相册，形如[N，W，H，C]，其中C表示通道数。这种多维数组表示的好处可以从计算机组成原理说起，现代计算机都是多核多处理器的，支持多线程和多进程，非常适合矩阵的并行计算，而且计算一批数据和计算一个数据的调度时间是差不多的，因此科学计算往往都是基于矩阵的计算，并且会指定一个适当的Batch。例如，PyTorch视觉处理中通常将Batch指定为64、128或256，这也是为了充分利用计算机资源而考虑的。基于张量的乘法运算如图1.8所示。

张量就是多维数组，并且提供了CPU设备和GPU设备的支持。如果机器上有GPU设备，就可以在Tensor上调用cuda方法，将Tensor数据加载到GPU设备中运行，提升运行速度。张量上提供了很多有用的方法，这些方法和NumPy中的方法类似，使用过NumPy的读者肯定会觉得这些方法既眼熟又亲切。Torch上的算子多达上百个，对于如此多的算子，读者没有必要全部记住，只需要在使用的时候查字典即可。借助help（function）即可快速查看常用方法的定义及使用。

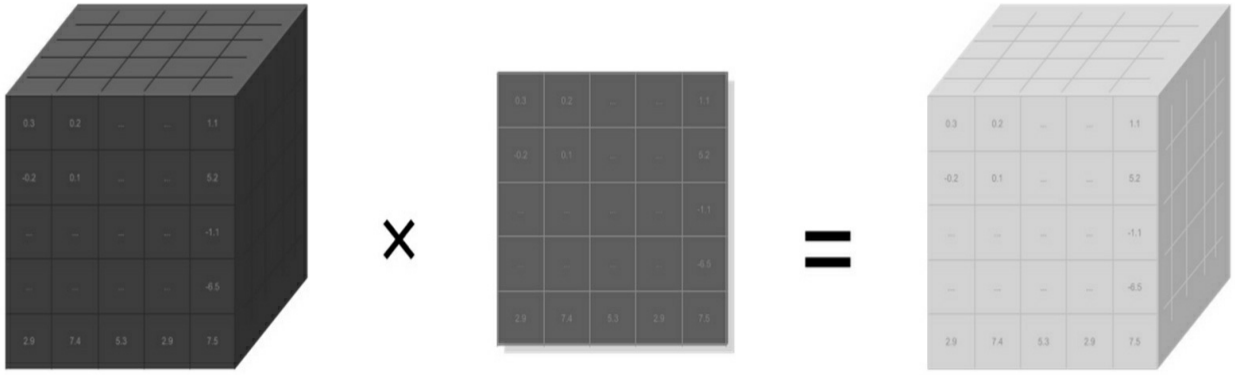
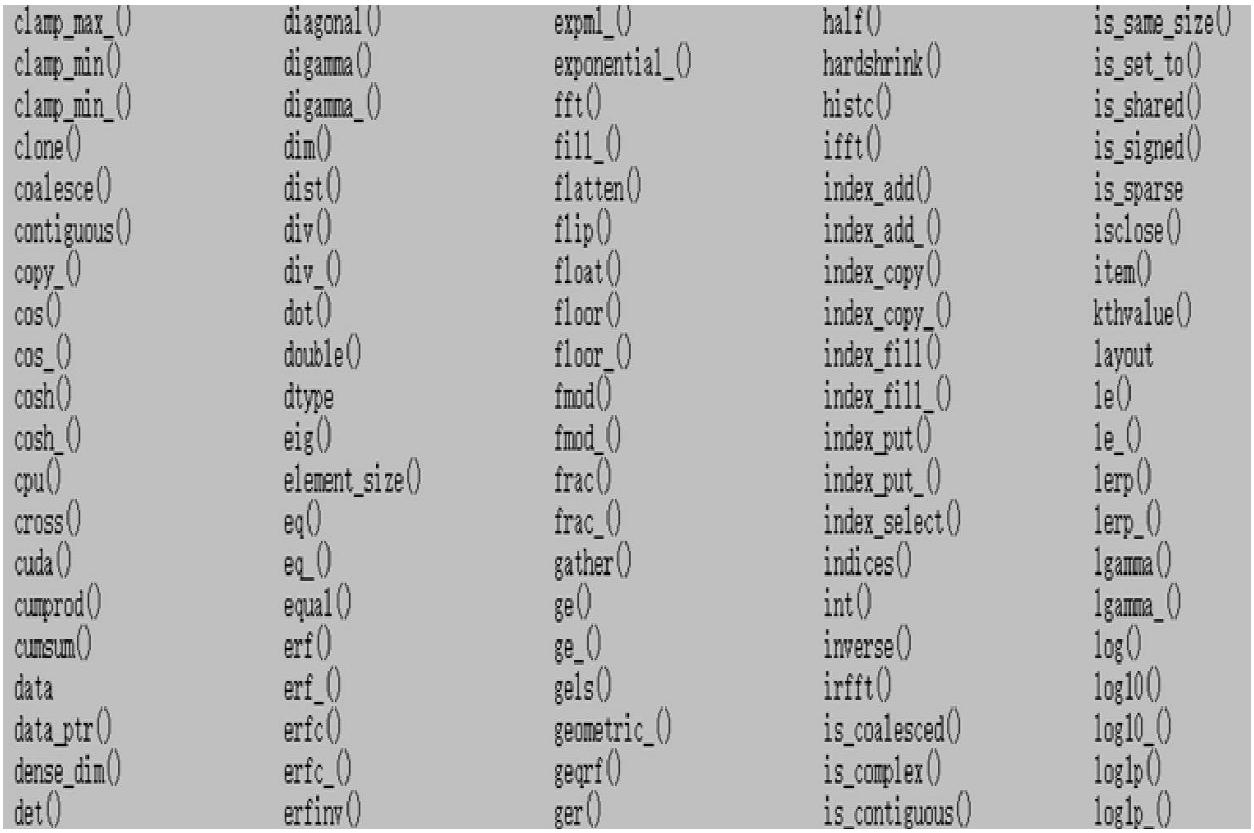


图1.8 基于张量的乘法运算

Tensor可以和NumPy进行无缝转换，在Tensor上可以使用numpy方法将Tensor转换为NumPy中的ndarray对象；ndarray对象也可以通过Torch提供的form_numpy方法转换成Tensor对象。如图1.9所示，Tensor多维数组上常用的算子也很多，读者学会查字典即可。



clamp_max_()	diagonal()	expm1_()	half()	is_same_size()
clamp_min()	digamma()	exponential_()	hardshrink()	is_set_to()
clamp_min_()	digamma_()	fft()	histc()	is_shared()
clone()	dim()	fill_()	ifft()	is_signed()
coalesce()	dist()	flatten()	index_add()	is_sparse
contiguous()	div()	flip()	index_add_()	isclose()
copy_()	div_()	float()	index_copy()	item()
cos()	dot()	floor()	index_copy_()	kthvalue()
cos_()	double()	floor_()	index_fill()	layout
cosh()	dtype	fmod()	index_fill_()	le()
cosh_()	eig()	fmod_()	index_put()	le_()
cpu()	element_size()	frac()	index_put_()	lerp()
cross()	eq()	frac_()	index_select()	lerp_()
cuda()	eq_()	gather()	indices()	lgamma()
cumprod()	equal()	ge()	int()	lgamma_()
cumsum()	erf()	ge_()	inverse()	log()
data	erf_()	gels()	irfft()	log10()
data_ptr()	erfc()	geometric_()	is_coalesced()	log10_()
dense_dim()	erfc_()	geqrf()	is_complex()	log1p()
det()	erfinv()	ger()	is_contiguous()	log1p_()

图1.9 Tensor多维数组上常用的算子

创建张量需要借助Torch提供的Tensor方法或from_numpy方法，理论上可以创建任意维度的多维数组，但最常用的Tensor不超过五维，常见的多维数组如图1.10所示。

除了维度，还可以指定每个维度上的SIZE，如a=torch.Tensor(3,4,5,8)，表示创建了一个四维数组，因为Tensor有4个参数，每个参数表示对应维度的大小，第一维度的大小为3，第二维度的大小为4，以此类推。因此，变量a可以表示Batch数为3、宽度为4、高度为5、通道数为8的特征图（Feature Map，图像卷积运算产生的中间特征）。

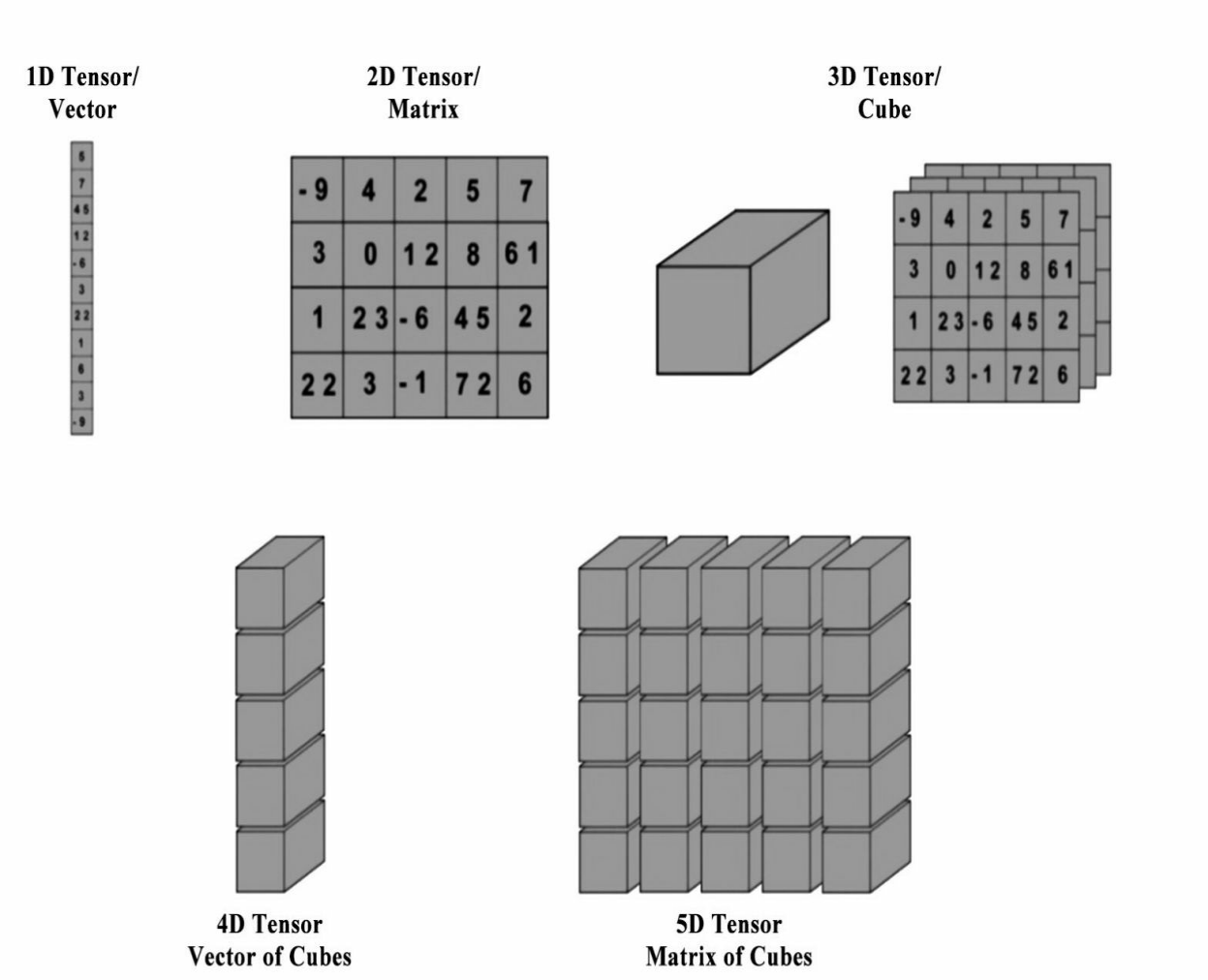


图1.10 常见的多维数组

```
In [1]: import torch
In [2]: a = torch.Tensor(3,4,5,8)
In [3]: a.shape
Out[3]: torch.Size([3, 4, 5, 8])
```

在一个Tensor上可以使用shape方法获取该张量的维度及每个维度的Size。下面依次介绍零维到五维Tensor及其使用场景。

1.零维标量

物理学中对标量的解释是只有大小没有方向，如温度、质量、湿度等。PyTorch中将只包含一个元素的变量称为标量。

```
In [31]: a = np.random.rand(1)
In [32]: a
Out[32]: array([0.52096887])
In [33]: b = torch.Tensor(a)
In [34]: b
Out[34]: tensor([0.5210])
In [35]: b.size()
Out[35]: torch.Size([1])
```

2.一维向量

数学中表示既有大小又有方向的量为一维向量。PyTorch中用Array数组表示向量，如scores=torch.Tensor([88,89,91,91,99,100])，表示考试分数的集合。

```
In [43]: scores = torch.Tensor([88,89,91,91,99,100])
In [44]: scores.size()
Out[44]: torch.Size([6])
In [45]: scores
Out[45]: tensor([ 88.,  89.,  91.,  91.,  99., 100.])
```

3.二维矩阵

矩阵表示多条记录形成的表格，如学生成绩表。它的特点是多行、多列。矩阵是科学计算中最常见的数据结构。例如，Sklearn中的鸢尾花数据集是150条不同品种鸢尾花的数据记录，包含4个特征，分别是花瓣的长度、宽度，以及花萼的长度、宽度。这是一个典型的表格数据，通过load_iris方法可以导入该数据集，可以使用from_numpy方法创建Tensor。

```
In [65]: from sklearn import datasets
In [66]: iris = datasets.load_iris()
In [67]: iris_data = iris["data"]
In [68]: iris_data.shape
Out[68]: (150, 4)
In [69]: iris_tensor = torch.from_numpy(iris_data)
In [71]: iris_tensor.size()
Out[71]: torch.Size([150, 4])
In [72]: iris_tensor[:4,]
Out[72]:
tensor([[5.1000, 3.5000, 1.4000, 0.2000],
        [4.9000, 3.0000, 1.4000, 0.2000],
        [4.7000, 3.2000, 1.3000, 0.2000],
        [4.6000, 3.1000, 1.5000, 0.2000]], dtype=torch.float64)
```

4.三维立方体

将多个二维矩阵叠加在一起就可以形成三维立方体。三维Tensor常用于表示图片数据[width，height，channel]，如图1.11所示。

5.四维多立方体

四维Tensor常见于批量的图片数据，如之前介绍的相册就是多张图片的叠加，最常见的形式为[batch，width，height，channel]。