第 1 章

人工智能、机器学习与深度学习简介

目前业界许多图像识别技术与语音识别技术的进步都源于深度学习的发展。深度学习的发展极大地促进了机器学习的地位提高，进一步推动了业界对人工智能梦想的再次重视。深度学习摧枯拉朽一样地实现了各种任务，使得所有的机器辅助功能都变为可能，无人驾驶汽车、预防性医疗保健甚至是更好的电影推荐都将可以实现。“AI 无所不能，马上就要改变世界，取代人类”的领域，基本跟深度学习有关系。

1.1 什么是人工智能

首先，我们先来界定接下来所要讨论的人工智能的定义和范畴。

人工智能（Artificial Intelligence，AI）可以理解为让机器具备类似人的智能，从而代替人类去完成某些工作和任务。

同学们对 AI 的认知可能来自《西部世界》《超能陆战队》《机器人总动员》等影视作品，这些作品中的 AI 都可以定义为“强人工智能”，因为它们能够像人类一样思考和推理，且具备知觉和自我意识。强人工智能是指具有完全人类思考能力和情感的人工智能。而弱人工智能是指不具备完全智慧，但能完成某一特定任务的人工智能。弱人工智能系统是人类已经掌握的技术，能够在特定的任务上，在已有的数据集上进行学习，它往往只擅长某方面的工作，无论是可以预约烧饭的电饭煲，还是会聊天的机器人，都属于此列。这种弱人工智能就在你身边，早已服务在大家生活的方方面面了，已经开始为社会创造价值。

比如语音助手，在手机、音响、车里，甚至你的手表上。最常见的“Hi Siri，帮我查查明天上海的天气”。这里面涉及机器如何听懂、理解人类的意图，并且在互联网上找到合适的数据进行回复。

顺便说一下电话客服的问题，相信大家平时都接到过一些推销电话、骚扰电话，和人类的声音完全一样，甚至能够对答如流，但是你有没有想过，和你进行交流的其实只是一台机器。

这个其实是最接近大家普遍认知里面人工智能的模样，无奈要让机器理解人类的自然语言，

还是路漫漫，特别是人类隐藏在语言里面的情感、隐喻。所以，自然语言处理一直被视为人类征服人工智能的珠穆朗玛峰。

相比于理解自然语言，计算机视觉发展得就顺利多了，它教计算机“看懂”一些人类交给他们的事物。

比如最常见的出行环境中，停车场的牌照识别。以前得雇一个老大爷天天守在门口抄牌子，现在一个摄像头可以搞定所有的事情。

在购物场景中，如 Amazon 的无人超市，能够通过人脸识别知道你是否来过、以前有没有购物过，给你推荐更好的体验。

而除这些身边“有形”的能看能听的人工智能外，帮助人类做决策、做预测，也是人工智能的强项。

比如在网购场景下，能够根据你以前的购物习惯，“猜测”你可能喜欢购买哪件物品。

比如刷抖音的时候，机器会学习你的喜好并推荐更符合你口味的视频。

再比如专业性更高的医疗行业，你有没有想过，自己学医 8 年，从 20 岁到 28 岁，仍然有可能被新技术所取代。笔者一个朋友的儿子是医疗影像专业的，在一家医院工作，有次一起交流的时候，我发现他对自己的前景充满了担忧：他说一个影像科的医生，从学习到出师，需要花费数十年的时间。这些 X 光片或者 CT、核磁共振的片子和诊断结果，让人工智能来进行判断，可能只需要几秒钟就能完成。而且计算机诊断的准确率明显高于人类医生，甚至成本更低。

在家庭生活场景中，每年我们都会看到全球智能家居厂商发布的硬核产品。例如科沃斯发布了第一款基于视觉技术的扫地机器人 DG70，它可以识别家里的鞋子、袜子、垃圾桶、充电线，当然除用到视觉系统外，还需要机身上各种各样的传感器信息融合处理，才能实现在清扫复杂家居环境时合理避障。

1.2　人工智能的本质

举一个简单的例子，如果我们需要让机器具备识别狗的智能：第一种方式是将狗的特征（毛茸茸、四条腿、有尾巴……）告诉机器，机器将满足这些规则的东西识别为狗；第二种方式是完全不告诉机器狗有什么特征，但我们“喂”给机器 10 万幅狗的图片，机器就会自己从已有的图片中学习狗的特征，从而具备识别狗的智能。

其实，AI 本质上是一个函数。AI 其实就是我们“喂”给机器目前已有的数据，机器从这些数据中找出一个最能拟合（即最能满足）这些数据的函数，当有新的数据需要预测的时候，机器就可以通过这个函数来预测这个新数据对应的结果是什么。

对于一个具备某种 AI 智能的模型而言，一般具备以下要素：“数据”+“算法”+“模型”，理解了这三个词，AI 的本质你也就搞清楚了。

我们用一个能够区分猫和狗图片的分类器模型来辅助理解一下这三个词：

- “数据”就是我们需要准备大量标注过是“猫”还是“狗”的图片。为什么要强调大量？因为只有数据量足够大，模型才能够学习到足够多且准确的区分猫和狗的特征，才能在区分猫和狗这个任务上表现出足够高的准确性；当然，在数据量不大的情况下，我们也可以训

练模型，不过在新数据集上预测出来的结果往往就会差很多。

- “算法”指的是构建模型时打算用浅层的网络还是深层的网络，如果是深层的网络，我们要用多少层，每层有多少神经元，功能是什么，等等，也就是网络架构的设计，相当于确定了我们的预测函数大致结构应该是什么样的。

我们用 Y=f(W,X,b)来表示这一函数，X 是已有的用来训练的数据（猫和狗的图片），Y 是已有的图片数据的标签（该图片是猫还是狗）。聪明的你会问：W 和 b 呢？问得好，函数里的 W（权重）和 b（偏差）我们还不知道，这两个参数是需要机器学习后自己找出来的，找的过程也就是模型训练的过程。

- “模型”指的我们把数据代入算法中进行训练，机器就会不断地学习，当机器找到最优W（权重）和b（偏差）后，我们就说这个模型训练成功了，这个时候函数Y=f(W,X,b)就完全确定下来了。

然后就可以在已有的数据集外给模型一幅新的猫或狗的图片，模型就能通过函数 Y=f(W,X,b)计算出这幅图的标签究竟是猫还是狗，这也就是所谓的模型的预测功能。

至此，你应该已经能够理解 AI 的本质了。我们再简单总结一下：无论是最简单的线性回归模型还是较复杂的拥有几十甚至上百个隐藏层的深度神经网络模型，本质都是寻找一个能够良好拟合目前已有数据的函数 Y=f(W,X,b)，并且希望这个函数在新的未知数据上能够表现良好。

前面提到的科沃斯发布的 DG70，只提供一个“眼睛”和有限个传感器，但却要求其可以识别日常家居物品，比如前方遇到的障碍物是拖鞋还是很重的家具脚，可不可以推过去？如果遇到了衣服、抹布这种奇形怪状的软布，机器还需要准确识别出来以避免缠绕。

让扫地机器人完成图像识别大致会经过以下几个步骤：

步骤 01 定义问题：就像刚刚说的，根据扫地机器人的使用场景，识别家居场景里面可能遇到的所有障碍物：家具、桌角、抹布、拖鞋等。有了这些类别定义，我们才可以训练一个多分类模型，针对扫地机器人眼前看到的物体进行分类，并且采取相应的规避动作。对于很多不了解机器学习的同学来说，能够理解到这一步其实已经是巨大的认知突破了。因为机器智能无法像人类一样学习，自我进化，举一反三。当前阶段的机器智能永远只能忠实执行人类交给他的任务。

步骤 02 收集数据 & 训练模型：接下来接着收集数据，并且标注数据。现在的深度神经网络动不动就有几百万个参数，具有非常强大的表达能力。因此需要大量的数据，而且是标注数据。所谓标注数据，就是在收集了有关图片后，需要人工标注员一个一个判断这些图片是否属于上面已定义类别中的某一个。在工业界这个成本非常昂贵，一个任务一年可能要花费几百万美金，仅仅是为了做数据标注。有了高质量的标注数据，才有可能驱动深度神经网络拟合真实世界问题。

步骤 03 这么复杂的人工智能运算在这个具体案例上是在本地机器上运行的。一方面，要保护用户隐私，不能将用户数据上传到云端；另一方面，扫地是一个动态过程，很多运算对时效性要求非常高，稍有延迟可能一不小心就撞到墙壁了。

综上所述，连简单的“识别拖鞋”都需要经过上面这么复杂的过程。因此，扫地机器人虽小，但其涉及的技术领域堪比自动驾驶。而对于自动驾驶汽车来说，其信号收集过程也跟上面差不多。不过为了保证信号的精确程度，现代的自动驾驶汽车除图像视觉信号外，车身还会配备更多的传感器，精确感知周围环境。

1.3　人工智能相关专业人才就业前景

1. 国家鼓励发展新一代人工智能

2017 年 7 月，国务院印发《新一代人工智能发展规划》，确立了未来我国人工智能发展的目标和方向，战略目标分三步走：

第一步，到 2020 年人工智能总体技术和应用与世界先进水平同步，人工智能产业成为新的重要经济增长点，人工智能技术应用成为改善民生的新途径，有力支撑进入创新型国家行列和实现全面建成小康社会的奋斗目标。

第二步，到 2025 年人工智能基础理论实现重大突破，部分技术与应用达到世界领先水平，人工智能成为带动我国产业升级和经济转型的主要动力，智能社会建设取得积极进展。

第三步，到 2030 年人工智能理论、技术与应用总体达到世界领先水平，成为世界主要人工智能创新中心，智能经济、智能社会取得明显成效，为跻身创新型国家前列和经济强国奠定重要基础。

2. 人工智能产业飞速发展引发巨量人才需求

近年来，随着人工智能的飞速发展，生产效率和生活品质都得到大幅提升，各路资本、巨头和创业公司纷纷涌入相关领域，苹果、谷歌、微软、亚马逊和 Facebook 五大巨头都投入了大量资源抢占人工智能市场，甚至将自己整体转型为人工智能驱动型公司。据麦肯锡统计，全球范围内，包括谷歌、苹果、Facebook 等科技巨头在 AI 上的相关投入已经达到 200~300 亿美元，其中 90%用于技术研发和部署，10%用于收购。此外，面向初创公司的 VC 和 PE 投资也快速增长，总计 60~90 亿美元，三年间的外部投资年增长率接近 40%。国内互联网领军者 BAT（即百度、阿里巴巴、腾讯）也将人工智能作为重点战略，凭借自身优势，积极布局人工智能领域，尤其是计算机视觉、服务机器人、语音及自然语言处理、智能医疗、机器学习、智能驾驶等。截至目前，阿里巴巴、腾讯、百度、华为、微软、亚马逊等国内外知名科技企业均已在上海设立了人工智能科研机构。

在此背景下，相关人才的需求量日益增加，尤其是北京、上海、广东、江苏、浙江等地区需求量尤为庞大。北京以领先全国其他地区的政策环境、人才储备、产业基础、资本支持等成为人工智能创业首要阵地；上海、江苏、浙江均有良好的经济基础和科技实力，人工智能应用实力雄厚，也聚集了一批人工智能垂直产业园；浙江计划用 5 年时间引进 10 万名人工智能人才，还将建立全球人工智能人才数据库。广东互联网产业发达，企业对数据需求强烈，依靠大数据产业链有效推动了人工智能产业蓬勃发展。

据工信部调研统计，中国人工智能产业发展与人才需求比为 1:10，预计到 2030 年，人工智能核心产业规模将达到 1 万亿，相关产业规模达到 10 万亿，人工智能人才缺口达到 500 万，需求量最大的是工程应用型人才，其次是技术应用和科技转化中端人才，最后是前沿理论高端研究人才。可以预见，未来 5~10 年随着各类公司对人工智能布局的推广和深入，核心技术和人才的争夺将会越来越激烈。

3. 国内人工智能专业人才供不应求

据教育部公布的信息显示，截至 2018 年年底，全国已经开设人工智能相关专业的院校数量如

下：智能科学与技术专业 155 所，人工智能专业 38 所，机器人工程专业 194 所。经过调研显示，普遍存在着缺乏实验环境、缺乏实验项目、缺乏课程教师、缺乏配套教材、缺乏测评体系等方面的困难，人才培养的速度总体缓慢，规模总体较小，远远不能满足相关产业人才需求。

4. 国家鼓励加强人工智能人才培养

2018 年 4 月，教育部印发《高等学校人工智能创新行动计划》，从科研、教学、成果转化三个方面给高等教育体系下达“任务”：

2020 年，基本完成适应新一代人工智能发展的高校科技创新体系和学科体系的优化布局。

2025 年，取得一批具有国际重要影响的原创成果，部分理论研究、创新技术与应用示范达到世界领先水平。

2030 年，高校成为建设世界主要人工智能创新中心的核心力量和引领新一代人工智能发展的人才高地。

综上所述，当前国家正积极鼓励和引导发展新一代人工智能，国内人工智能产业得到了飞速发展，由此引发巨量人才需求，然而，由于现有人才存量不多，相关院校人才培养速度总体缓慢，相关人才供不应求，迫切需要加大力度培养人工智能专业人才。因此，可以预见，在未来 5~10 年内，人工智能专业人才就业前景乐观。

以上内容的目的是让大家看清人工智能行业目前的发展情况，一个日益增长且正面临全面商业化的行业，需要的人只会越来越多，而不是越来越少。传统行业的智能化已经启动，企业在 AI 时代构建新的竞争优势的核心在于人工智能人才的有效供给。目前我国高等教育对人工智能人才的培养处于较为滞后的状态，高校对人才的培养很难满足企业需求。一些掌握人工智能前沿技术的企业开始寻找新的人才培养模式，未来将有更多的符合岗位需求的人才进入市场。

1.4　机器学习和深度学习

1.4.1　什么是机器学习

要说明什么是深度学习，首先要知道机器学习（Machine Learning，ML）、神经网络、深度学习之间的关系。

众所周知，机器学习是一种通过利用数据训练出模型，然后使用模型预测的方法。与传统的为解决特定任务、硬编码的软件程序不同，机器学习是用大量的数据来“训练”的，通过各种算法从数据中学习如何完成任务。举个简单的例子，当我们浏览网上商城时，经常会出现商品推荐的信息。这是商城根据你往期的购物记录和冗长的收藏清单识别出其中哪些是你真正感兴趣的，并且愿意购买的产品。这样的决策模型可以帮助商城为客户提供建议并鼓励产品消费。

机器学习是人工智能的子领域，机器学习理论主要是设计和分析一些让计算机可以自动学习的算法。

举个例子，假设要构建一个识别猫的程序。传统上，如果我们想让计算机进行识别，需要输入一串指令，例如猫长着毛茸茸的毛、顶着一对三角形的耳朵等，然后计算机根据这些指令执行下去。

但是，如果我们对程序展示一只老虎的照片，程序应该如何反应呢？更何况通过传统方式要制定全部所需的规则，而且在此过程中必然涉及一些较难定义的概念，比如对毛茸茸的定义。因此，更好的方式是让机器自学。我们可以为计算机提供大量的猫的照片，系统将以自己特有的方式查看这些照片。随着实验的反复进行，系统会不断学习更新，最终能够准确地判断出哪些是猫，哪些不是猫。

我们不给机器规则，取而代之，我们“喂”给机器大量的针对某一任务的数据，让机器自己学习，继而挖掘出规律，从而具备完成某一任务的智能。机器学习是通过算法，使用大量数据进行训练，训练完成后会产生模型，将来有新的数据进来能够进行准确的分类或预测。

机器学习的常用方法主要分为监督学习（Supervised Learning）和无监督学习（Unsupervised Learning）。

1. 监督学习

监督学习需要使用有输入和预期输出标记的数据集。例如，指定的任务是使用一种图像分类算法对男孩和女孩的图像进行分类，那么男孩的图像需要带有“男孩”标签，女孩的图像需要带有“女孩”标签。这些数据被认为是一个训练数据集，通过已有的训练数据集（即已知数据及其对应的输出）来训练得到一个最优模型，这个模型就具备了对未知数据进行分类的能力。它之所以被称为监督学习，是因为算法从训练数据集学习的过程就像是一位老师正在监督学习。在我们预先知道正确的分类答案的情况下，算法对训练数据不断进行迭代预测，然后预测结果由“老师”进行不断修正。当算法达到可接受的性能水平时，学习过程才会停止。

在人对事物的认知中，我们从小就被大人教授这是鸟，那是猪，那是房子，等等。我们所见到的景物就是输入数据，而大人对这些景物的判断结果（是房子还是鸟）就是相应的输出。当我们见识多了以后，脑子里就慢慢地得到了一些泛化的模型，这就是训练得到的那个（或者那些）函数，从而不需要大人在旁边指点的时候，我们也能分辨出来哪些是房子，哪些是鸟。

2. 无监督学习

无监督学习（也叫非监督学习）则是另一种研究得比较多的学习方法，它与监督学习的不同之处在于我们事先没有任何训练样本，需要直接对数据进行建模。这听起来有些不可思议，但是在我们认识世界的过程中很多地方都用到了无监督学习。比如我们去参观一个画展，我们对艺术一无所知，但是欣赏完多幅作品之后，也能把它们分成不同的派别（比如哪些更朦胧一点，哪些更写实一些，即使我们不知道什么叫作朦胧派，什么叫作写实派，但是至少能把它们分为两个类别）。

1.4.2　深度学习独领风骚

机器学习有很多经典算法，其中有一个叫作神经网络的算法。神经网络最初是一个生物学的概念，一般是指大脑神经元、触点、细胞等组成的网络，用于产生意识，帮助生物思考和行动，后来人工智能受神经网络的启发，发展出了人工神经网络。人工神经网络是指由计算机模拟一层一层的神经元组成的系统。这些神经元与人类大脑中的神经元相似，通过加权连接相互影响，并且通过改变连接上的权重，可以改变神经网络执行的计算。

最初的神经网络是感知器（Perceptron）模型，可以认为是单层神经网络，但由于感知器算法无法处理多分类问题和线性不可分问题，当时计算能力也落后，因此对神经网络的研究沉寂了一段时

间。2006 年，Geoffrey Hinton 在科学杂志 *Science* 上发表了一篇文章，不仅解决了神经网络在计算上的难度，同时也说明了深层神经网络在学习上的优异性。深度神经网络（Deep Neural Network，DNN）的深度指的是这个神经网络的复杂度，神经网络的层数越多，就越复杂，它所具备的学习能力就越深，因此我们称之为深度神经网络。从此神经网络重新成为机器学习界主流强大的学习技术，同时具有多个隐藏层的神经网络被称为深度神经网络，基于深度神经网络的学习研究称为深度学习。

如图 1-1 所示，神经网络与深度神经网络的区别在于隐藏层级，神经网络一般有输入层→隐藏层→输出层，一般来说隐藏层大于 2 的神经网络叫作深度神经网络，深度学习就是采用像深度神经网络这种深层架构的一种机器学习方法。它的实质就是通过构建具有很多隐藏层的神经网络模型和海量的训练数据来学习更有用的特征，从而最终提升分类或预测的准确性。

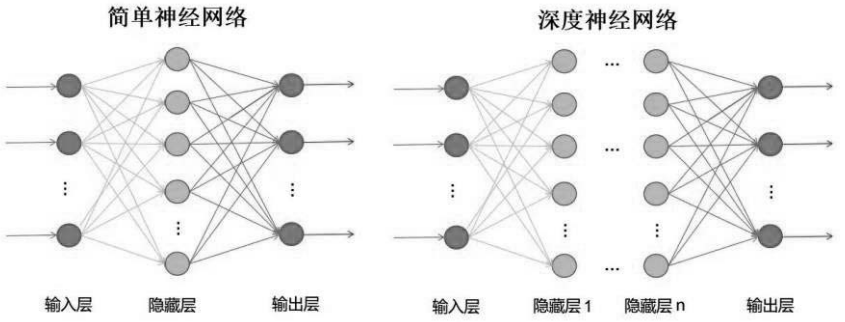


图 1-1

有计算机界诺贝尔奖之称的 ACM A.M.图灵奖（ACM A.M. Turing Award）公布 2018 年获奖者由引起这次人工智能革命的三位深度学习之父——蒙特利尔大学教授 Yoshua Bengio、多伦多大学名誉教授 Geoffrey Hinton、纽约大学教授 Yann LeCun 获得，他们使深度神经网络成为计算的关键。ACM 这样介绍他们三人的成就：Hinton、LeCun 和 Bengio 三人为深度神经网络这一领域建立起了概念基础，通过实验揭示了神奇的现象，还贡献了足以展示深度神经网络实际进步的工程进展。

Google 的 AlphaGo 与李世石九段惊天动地的大战，AlphaGo 以绝对优势完胜李世石九段，击败棋圣李世石的 AlphaGo 所用到的算法实际上就是基于神经网络的深度学习算法。在自然语言处理领域，深度学习技术已经取得了很多重大突破。这些深度学习模型可以对大量的文本数据进行自动学习，自动生成丰富的语言表示，这些表示可以被用来解决多种自然语言处理任务，如火爆的 ChatGPT 的技术架构就是深度学习模型。人工智能、深度学习成为这几年计算机行业、互联网行业最火的技术名词。

1.4.3　机器学习和深度学习的关系和对比

如图 1-2 所示，深度学习（Deep Learning，DL）属于机器学习的子类。它的灵感来源于人类大脑的工作方式，这是利用深度神经网络来解决特征表达的一种学习过程。深度神经网络本身并非是一个全新的概念，可理解为包含多个隐含层的神经网络结构。为了提高深层神经网络的训练效果，人们对神经元的连接方法以及激活函数等做出了调整，其目的在于建立、模拟人脑进行分析学习的神经网络，模仿人脑的机制来解释数据，如文本、图像、声音。

图 1-2

如果是传统机器学习方法，我们会首先定义一些特征，如有没有胡须，耳朵、鼻子、嘴巴的模样，等等。总之，首先要确定相应的“面部特征”作为机器学习的特征，以此来对对象进行分类识别。

而现在，深度学习方法更进一步。深度学习可以自动找出这个分类问题所需要的重要特征，而传统的机器学习则需要人工给出特征。

那么深度学习是如何做到这一点的呢？以猫狗识别的例子来说，按照以下步骤即可：

（1）确定有哪些边和角跟识别出猫和狗关系最大。

（2）根据上一步找出的很多小元素（边、角等）构建层级网络，找出它们之间的各种组合。

（3）在构建层级网络之后，就可以确定哪些组合可以识别出猫和狗。

深度学习的“深”是因为它通常有较多的隐藏层，正是因为有那么多隐藏层存在，深度学习网络才拥有表达更复杂函数的能力，这样才能识别更复杂的特征，继而完成更复杂的任务。

对于机器学习与深度学习，我们从 5 个方面进行比较：数据依赖、硬件依赖、特征工程、运行时间、可理解性。

1. 数据依赖

机器学习能够适应各种数据量，特别是数据量较小的场景。如果数据量迅速增加，那么深度学习的效果将更加突出，如图 1-3 所示。这是因为深度学习算法需要大量数据才能完美理解。随着数据量的增加，二者的表现有很大区别。

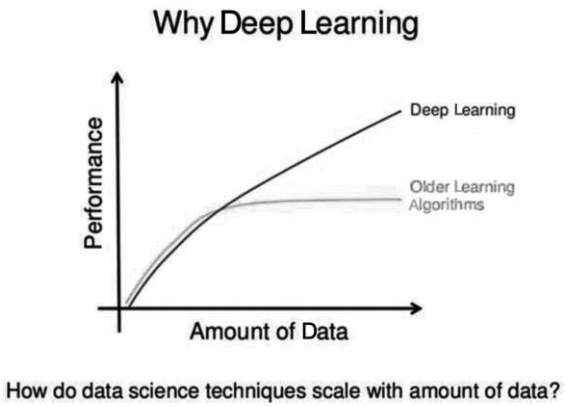


图 1-3

通过数据量对不同方法表现的影响可以发现，深度学习适合处理大数据，而数据量比较小的时候，用传统机器学习方法也许更合适。之前提到的预先训练过的网络在 120 万幅图像上进行了训练。对于许多应用来说，这样的大数据集并不容易获得，并且花费昂贵且耗时。

2. 硬件依赖

深度学习十分依赖高端的硬件设施，因为计算量实在太大了。深度学习中涉及很多矩阵运算，因此很多深度学习都要求有 GPU 参与运算，因为 GPU 就是专门为矩阵运算而设计的。相反，普通的机器学习随便给一台计算机就可以运行。深度学习需要使用高端的 CPU 和 GPU（图形处理器）。与 GPU 相比，CPU 在处理并行计算的速度上略有落后。GPU 的设计初衷是用于图形渲染，这需要大规模的并行计算来处理图像中的像素。深度学习中的许多任务也可以受益于大规模并行计算，因为神经网络中的许多操作可以同时在多个数据点上执行。GPU 的高度并行化结构使其非常适合执行这些操作。

3. 特征工程

特征工程是指我们在训练一个模型的时候，需要首先确定有哪些特征。在机器学习方法中，几乎所有的特征都需要通过行业专家来确定，然后手工对特征进行编码。然而深度学习算法试图自己从数据中学习特征。这也是深度学习十分引人注目的一点，毕竟特征工程是一项十分烦琐、耗费很多人力物力的工作，深度学习的出现大大减少了发现特征的成本。经典的机器学习算法通常需要复杂的特征工程。首先在数据集上执行深度探索性数据分析，然后做一个简单的降低维数的处理。最后，必须仔细挑选一些对结果预测最有用的特征传递给机器学习算法。而当使用深度网络时，不需要这样做，因为只需将数据直接传递到网络，通常就可以实现良好的性能。深度学习算法试图自己从数据中学习特征。这也是深度学习十分引人注目的一点，毕竟特征工程是一项十分烦琐、耗费很多人力物力的工作，深度学习的出现大大减少了发现特征的成本。

4. 运行时间

深度学习需要花费大量时间来训练，因为有太多参数需要学习。顶级的深度学习算法需要花费几周的时间来训练。但是机器学习一般几秒钟，最多几小时就可以训练好。执行时间是指训练算法所需要的时间量。一般来说，深度学习算法需要大量时间进行训练。这是因为该算法包含很多参数，因此训练它们需要比平时更长的时间。相对而言，机器学习算法的执行时间更少。但是深度学习花费这么多时间训练模型肯定不会白费力气的，优势在于模型一旦训练好，在预测任务时就运行得很快。

5. 可理解性

最后一点，也是深度学习的一个缺点。其实也算不上缺点吧，那就是深度学习很多时候我们难以理解。一个深层的神经网络，每一层都代表一个特征，而层数多了，我们也许根本不知道它们代表什么特征，就无法把训练出来的模型用来对预测任务进行解释。例如，我们用深度学习方法来批改论文，也许训练出来的模型对论文评分都十分准确，但是我们无法理解模型的规则，这样那些拿了低分的同学找你问“凭什么我的分这么低？”，你会哑口无言，因为深度学习模型太复杂，内部的规则很难理解。

传统机器学习算法就不一样，比如决策树算法，可以明确地把规则列出来，每一个规则、每一个特征都可以理解。但这不是深度学习的错（指深度学习模型太复杂，内部的规则很难理解），只能说它太厉害了，人类还不够聪明，理解不了深度学习的内部特征。

1.5　小白如何学深度学习

很早之前，听过雷军说的一句话：“站在风口上，猪都可以飞起来”！这句话用来形容现在的深度学习非常贴切。是的，近几年来，深度学习的发展极其迅速。其影响力已经遍地开花，在医疗、自动驾驶、机器视觉、自然语言处理等各个方面大展身手。在深度学习这个世界大风口上，谁能抢先进入深度学习领域，学会运用深度学习技术，谁就能真正地在 AI 时代“飞”起来。

对于每一个想要开始学深度学习的大学生、IT 程序员或者其他想转行的人来说，最迫切的需求就是深度学习该如何入门。下面来谈一谈笔者的看法。

1.5.1　关于两个“放弃”

1. 放弃海量资料

没错，就是放弃海量资料！在我们想要入门深度学习的时候，往往会搜集很多资料，比如 xx 学院深度学习内部资源、深度学习从入门到进阶百吉字节资源、xx 人工智能教程等。很多时候我们拿着十几吉字节、几百吉字节的学习资源，然后踏踏实实地放到了某个云盘里存着，等着日后慢慢学习。殊不知，有 90% 的人仅仅只是搜集资料、保存资料而已，放在云盘里一年半载也没打开学习。躺在云盘的资料很多时候只是大多数人“以后好好学习”的自我安慰和“自我”安全感而已。而且，面对海量的学习资料，很容易陷入一种迷茫的状态，最直接的感觉就是：天啊，有这么多东西要学！天啊，还有这么多东西没学！简单来说，就是选择越多，越容易让人陷入无从选择的困境。

所以，第一步就是要放弃海量资料，选择一份真正适合自己的资料，好好研读下去，消化它。最终会发现收获很大。

2. 放弃从零起步

深度学习的初学者总会在学习路径上遇到困惑。先是那些框架，就让你不知道该从哪里着手。一堆书籍，也让你犹豫如何选择。即便你去咨询专业人士，他们也总会轻飘飘地告诉你一句“先学好数学”。怎样算是学好？深度学习是一门融合概率论、线性代数、凸优化、计算机、神经科学等多方面的复杂技术。学好深度学习需要的理论知识很多，有些人可能基础不是特别扎实，就想着从最底层的知识开始学起，如概率论、线性代数、机器学习、凸优化公式推导等。但是这样做的坏处是比较耗费时间，而且容易造成“懈怠学习”，打消学习的积极性，直到你彻底放弃学习的想法。真要按照他们的要求按部就班地学习，没有几年功夫，你连数学和编程基础都学不完。可到那时候，许多“低垂的果实”还在吗？

因为“啃”书本和推导公式相对来说是比较枯燥的，远不如自己搭建一个简单的神经网络更能激发自己学习的积极性。当然，不是说不需要钻研基础知识，只是说，在入门的时候，最好先从顶层框架上有个系统的认识，然后从实践到理论，有的放矢地查缺补漏机器学习的知识点。从宏观到微观，从整体到细节，更有利于深度学习快速入门。而且从学习的积极性来说，也起到了“正反馈”的作用。

1.5.2　关于三个“必须”

谈完了深度学习入门的两个“放弃”之后，接下来看深度学习究竟该如何快速入门。

1. 必须选择编程语言：Python

俗话说“工欲善其事，必先利其器”。学习深度学习，掌握一门合适的编程语言非常重要。最佳的选择就是 Python。为什么人工智能、深度学习会选择 Python 呢？一方面是因为 Python 作为一门解释型语言，入门简单，容易上手。另一方面是因为 Python 的开发效率高，Python 有很多库很方便实现人工智能，比如 NumPy、SciPy 做数值计算，Sklearn 做机器学习，Matplotlib 将数据可视化，等等。总的来说，Python 既容易上手，又是功能强大的编程语言。可以毫不夸张地说，Python 可以支持从航空航天器系统的开发到小游戏开发的几乎所有领域。

这里笔者强烈推荐 Python，因为 Python 作为一个万能胶水语言，能做的事情实在太多，并且它非常容易上手。笔者大概花了 50 个小时学习了 Python 的基础语法，然后就可以开始动手写神经网络代码了。

总之，Python 是整个过程并不耗精力的环节，但是刚开始记语法确实挺无聊的，需要些许坚持。

2. 必须选择一个或两个最好的深度学习框架（如PyTorch）

对于工业界的人工智能项目，一般都不重复造轮子：不会从零开始写一套人工智能算法，而往往选择采用一些已有的算法库和算法框架。以前，我们可能会选用已有的各种算法来解决不同的问题。现在一套深度学习框架就可以解决几乎所有问题，进一步降低了人工智能项目开发的难度。Facebook 人工智能研究院（FAIR）团队在 GitHub 上开源了 PyTorch 深度学习框架，并迅速占领 GitHub 热度榜榜首。

如果说 Python 是我们手中的利器，那么一个好的深度学习框架无疑给了我们更多的资源和工具，方便我们实现庞大、高级、优秀的深度学习项目。

奥卡姆剃刀定律（Occam’s Razor, Ockham’s Razor）又称“奥康的剃刀”，它是由 14 世纪英格兰的逻辑学家、圣方济各会修士奥卡姆的威廉（William of Occam，约 1285—1349 年）提出的。这个原理称为“如无必要，勿增实体”，即“简单有效原理”。正如他在《箴言书注》2 卷 15 题说的“切勿浪费较多东西去做，用较少的东西，同样可以做好事情。”

深度学习的底层实际结构很复杂。然而，作为应用者，你只需要几行代码，就能实现上述神经网络。加上数据读取和模型训练，也不过寥寥十来行代码。感谢科技的进步，深度学习的用户接口越来越像搭积木。只要你投入适当的学习成本，总是能很快学会。PyTorch 是当前主流的深度学习框架之一，其追求最少的封装、最直观的设计，其简洁优美的特性使得 PyTorch 代码更易理解，对新手非常友好。今年大火的 ChatGPT 是由 OpenAI 使用 Python 编程语言实现的自然语言处理模型，是基于深度学习技术实现的，使用了 Python 中的 PyTorch 等深度学习框架来训练模型。

3. 必须坚持“唯有实践出真知”

现在很多教程和课程都忽视了实践的重要性，将大量精力放在了理论介绍上。我们都知道纸上谈兵的典故，重理论、轻实践的做法是非常不可取的。就像前面讲的第 2 个“放弃”一样，在具备基本的理论知识之后，最好就去实践、编写代码，解决实际问题。从学习的效率上讲，速度是最快的。

对于毫无 AI 技术背景，只会 Python 编程语言，从零开始入门深度学习的同学，不要犹豫开始学习吧，深度学习入门可以很简单！

第 2 章

深度学习框架 PyTorch 开发环境搭建

工欲善其事，必先利其器。开发工具的准备是进行深度学习的第一步。 PyCharm 是目前最流行的 Python IDE。另外，搭建 PyTorch 环境，建议尽量利用 GPU 的算力，如果没有 Nvidia 显卡提供的 GPU，CPU 也可以，但示例代码的运算速度要慢很多。

PyTorch 的前身便是 Torch，Torch 是纽约大学的一个机器学习开源框架，几年前在学术界非常流行，包括 Lecun 等大佬都在使用。但是由于使用的是一种绝大部分人没有听过的 Lua 语言，导致很多人都被吓退。后来随着 Python 的生态越来越完善，Facebook 人工智能研究院推出了 PyTorch 并开源。PyTorch 不是简单地封装 Torch 并提供 Python 接口，而是使用 Python 重新写了很多内容，不仅更加灵活，支持动态图，而且提供了 Python 接口。它是一个以 Python 优先的深度学习框架，不仅能够实现强大的 GPU 加速，同时还支持动态神经网络，这是很多主流深度学习框架（比如 TensorFlow 等）都不支持的。

2.1 PyCharm 的安装和使用技巧

PyCharm 是一款 Python IDE，其带有一整套可以帮助用户在使用 Python 语言开发时提高效率的工具，比如调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制等。

进入 PyCharm 官方下载页面（https://www.jetbrains.com/pycharm/），如图 2-1 所示，读者可按页面提示，选择下载免费的 PyCharm Community Edition。

安装过程很简单，选择安装路径不要选择带中文和空格的目录，跟着安装向导一步一步就可以完成。

安装完成后，双击桌面上的 PyCharm 图标，进入 PyCharm 中。首先创建一个 Python 项目，在项目中创建一个文件，在文件中才可以编写程序，为什么不可以直接创建文件？我们可以这样理解，这个项目相当于一个总文件，我们写程序有很多内容需要运行，要存储到多个文件中，所以可以把它们放在总文件中同步运行，也就是成为一个项目。如图 2-2 所示，创建一个 Python 项目，这里可

以修改项目存放的位置，修改 Python 版本。

图 2-1

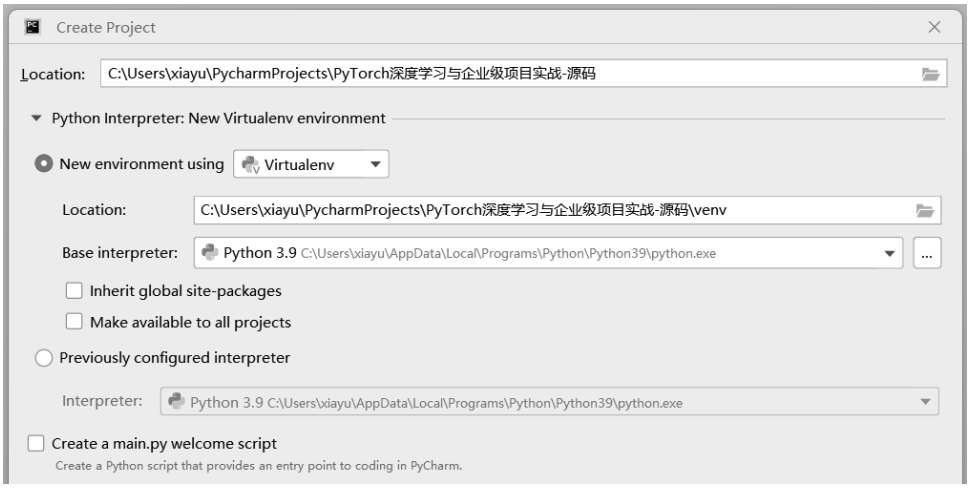


图 2-2

这里我们遇到一个虚拟环境（Virtual Environment）的概念，笔者先阐述一下关于虚拟环境的作用。虚拟环境在 Python 中是相当重要的存在，它起到了项目隔离的作用。前面我们安装的 Python，相当于在本地安装了一个 Python 的全局环境，在任何地方都可以使用这个 Python 的全局环境。

但是大家有没有想过一个问题：笔者同时接手了 Demo A 和 Demo B 两个项目，两个项目用到了同一个模块 X，但是 Demo A 要求使用模块 X 的 1.0 版本，Demo B 要求使用模块 X 的 2.0 版本。全局环境中一个模块只能安装一个版本，这样就遇到问题了，怎样才能让两个项目同时正常运行呢？

这时虚拟环境就能发挥作用了，笔者使用全局的 Python 环境分别创建两个虚拟环境给 Demo A 和 Demo B。相当于两个项目分别有自己的环境，这个时候笔者把各自需要的模块安装到各自的虚拟环境中，就成功实现了项目隔离。假如这个项目笔者不需要了，直接删除就可以（一个虚拟环境相当于一个拥有 Python 环境的文件夹，可以自行指定路径）。

右击刚创建的项目，选择 New 选项，如图 2-3 所示，再选择 Python File 选项，即可创建一个

Python 文件。

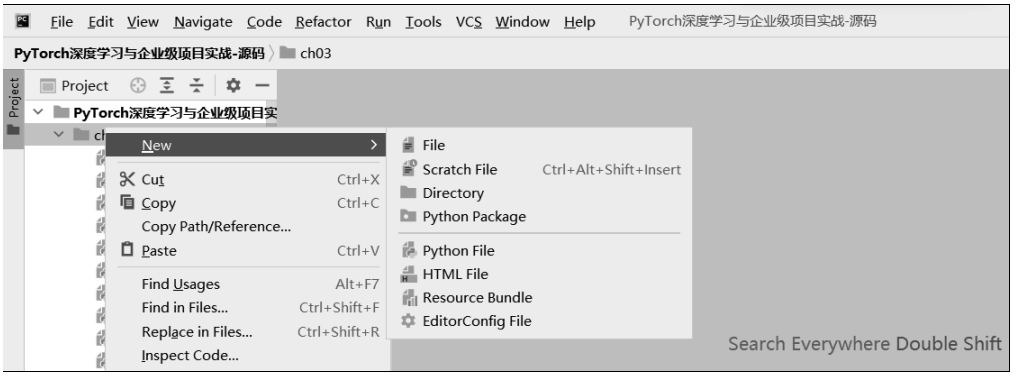


图 2-3

输入代码内容，在空白处右击显示菜单，单击 Run 'helloworld'运行，如图 2-4 所示。

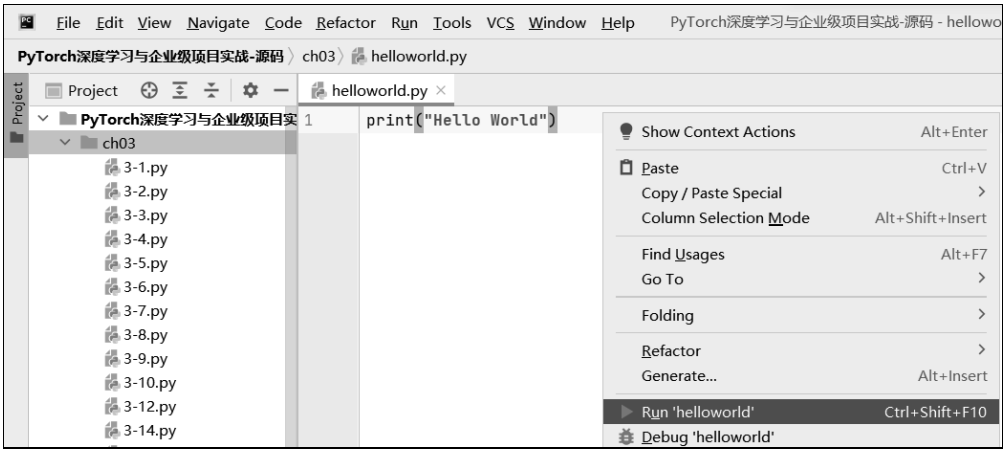


图 2-4

有些人可能不习惯背景，我们可以自己设置背景跟文字大小、颜色等，单击菜单 File→Settings 即可设置，如图 2-5 和图 2-6 所示。

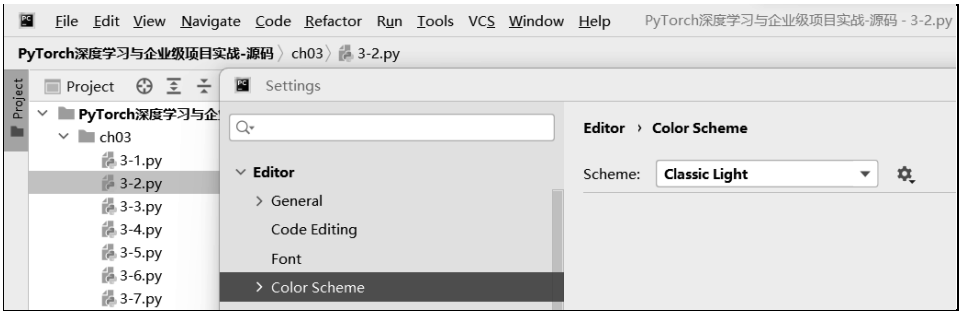


图 2-5

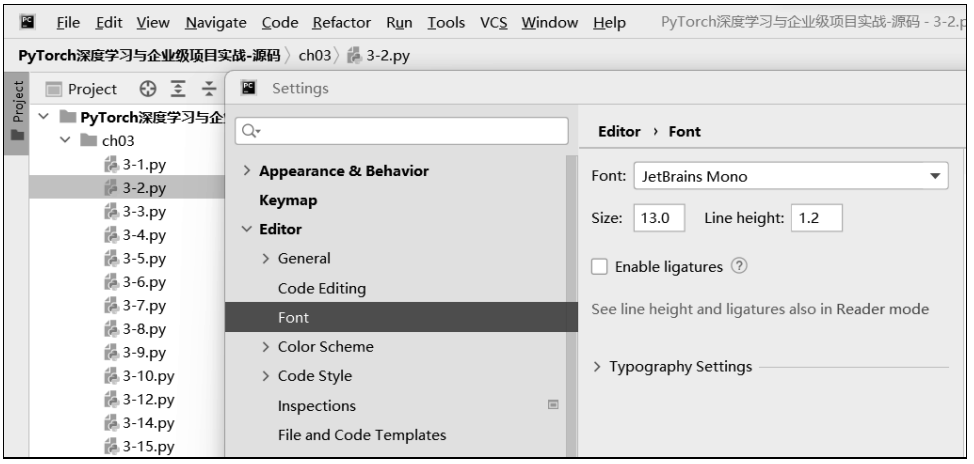


图 2-6

“断点”与“调试”是比较重要的功能，“断点”就是故意停止的地方或者让 Debug 程序停下来。而“调试”是当程序停下来时，我们可以一步一步往下调试，看清程序每一步的结果，让我们发现缺陷或问题。如何添加断点？在代码前面单击就可以了。而调试断点时，在空白处右击显示菜单，单击绿色甲虫 Debug，如图 2-7 所示。单击后会运行到第一个断点位置，下面就会显示断点之前的变量信息或者参数，然后继续往下运行，按 F8 键，可以单步运行到下一个断点，执行到最后就可以看到下面显示了上面的变量信息。

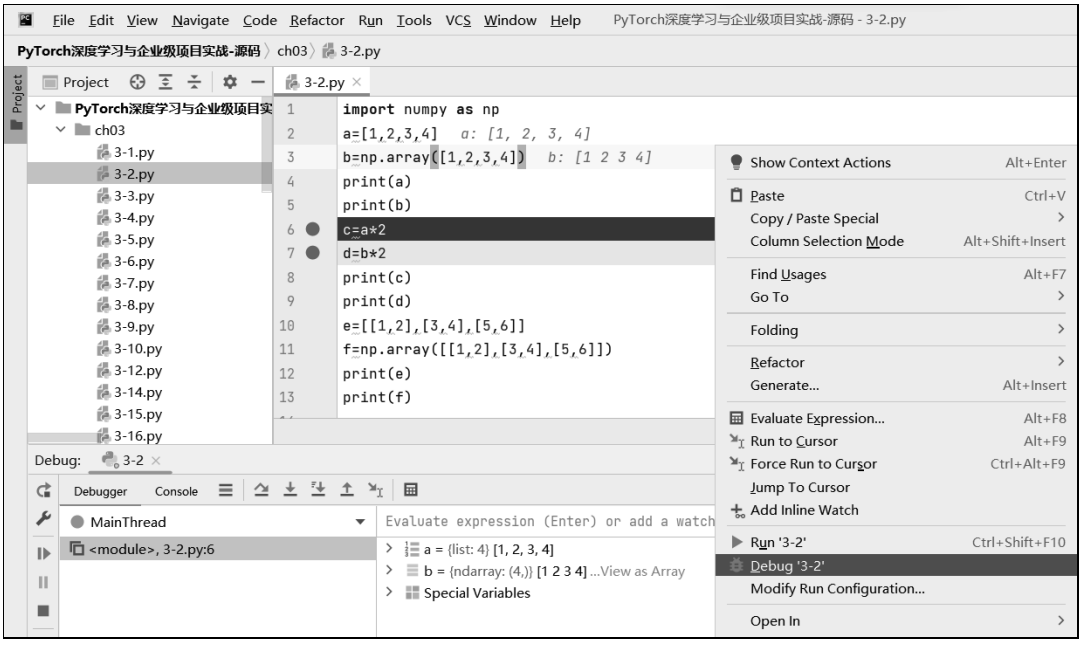


图 2-7

可以看到，这段代码经过调试，从下面的结果可以看到各个变量的值。断点调试很重要，不仅可以让我们知道运行过程，还能减少程序的错误。这便是在 PyCharm 中创建项目、编写及运行代码的过程。

2.2　在 Windows 环境下安装 CPU 版的 PyTorch

PytTorch 是基于 Python 开发的，首先需要安装 Python，Python 的安装很简单，这里不再赘述。而 Windows 用户能直接通过 conda、pip 和源码编译三种方式来安装 PyTorch。

打开 PyTorch 官网（https://pytorch.org/），在主页中根据自己的计算机选择 Linux、Mac 或 Windows 系统，如图 2-8 所示，系统将给出对应的安装命令语句，比如这里为 pip3 install torch torchvision torchaudio。

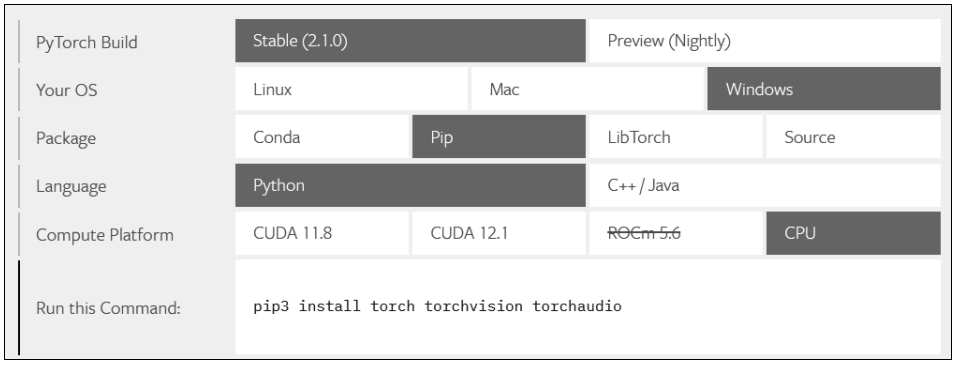


图 2-8

通过安装命令安装 PyTorch 2.1.0，结果如下：

```
    PS C:\Users\xiayu> pip3 install torch torchvision torchaudio
    Collecting torch
      Downloading torch-2.1.0-cp39-cp39-win_amd64.whl.metadata (24 kB)
    Collecting torchvision
      Downloading torchvision-0.16.0-cp39-cp39-win_amd64.whl.metadata (6.6 kB)
    Collecting torchaudio
      Downloading torchaudio-2.1.0-cp39-cp39-win_amd64.whl.metadata (5.7 kB)
    Collecting filelock (from torch)
      Downloading filelock-3.12.4-py3-none-any.whl.metadata (2.8 kB)
    Collecting typing-extensions (from torch)
      Downloading typing_extensions-4.8.0-py3-none-any.whl.metadata (3.0 kB)
    Collecting sympy (from torch)
      Downloading sympy-1.12-py3-none-any.whl (5.7 MB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.7/5.7 MB 14.7 kB/s eta 0:00:00
    Collecting networkx (from torch)
      Downloading networkx-3.2-py3-none-any.whl.metadata (5.2 kB)
    Collecting jinja2 (from torch)
      Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.1/133.1 kB 12.8 kB/s eta 0:00:00
    Collecting fsspec (from torch)
      Downloading fsspec-2023.10.0-py3-none-any.whl.metadata (6.8 kB)
    Requirement already satisfied: numpy in
    c:\users\xiayu\appdata\local\programs\python\python39\lib\site-packages
(from torchvision) (1.26.1)
```

```
    Collecting requests (from torchvision)
      Downloading requests-2.31.0-py3-none-any.whl.metadata (4.6 kB)
    Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
    c:\users\xiayu\appdata\local\programs\python\python39\lib\site-packages
(from torchvision) (10.1.0)
    Collecting MarkupSafe>=2.0 (from jinja2->torch)
      Downloading MarkupSafe-2.1.3-cp39-cp39-win amd64.whl.metadata (3.1 kB)
    Collecting charset-normalizer<4,>=2 (from requests->torchvision)
      Downloading charset normalizer-3.3.1-cp39-cp39-win amd64.whl.metadata (33
kB)
    Collecting idna<4,>=2.5 (from requests->torchvision)
      Downloading idna-3.4-py3-none-any.whl (61 kB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.5/61.5 kB 32.2 kB/s eta 0:00:00
    Collecting urllib3<3,>=1.21.1 (from requests->torchvision)
      Downloading urllib3-2.0.7-py3-none-any.whl.metadata (6.6 kB)
    Collecting certifi>=2017.4.17 (from requests->torchvision)
      Downloading certifi-2023.7.22-py3-none-any.whl.metadata (2.2 kB)
    Collecting mpmath>=0.19 (from sympy->torch)
      Downloading mpmath-1.3.0-py3-none-any.whl (536 kB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 536.2/536.2 kB 17.7 kB/s eta 0:00:00
    Downloading torch-2.1.0-cp39-cp39-win_amd64.whl (192.2 MB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 192.2/192.2 MB 96.2 kB/s eta 0:00:00
    Downloading torchvision-0.16.0-cp39-cp39-win_amd64.whl (1.3 MB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 78.0 kB/s eta 0:00:00
    Downloading torchaudio-2.1.0-cp39-cp39-win_amd64.whl (2.3 MB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.3/2.3 MB 78.5 kB/s eta 0:00:00
    Downloading filelock-3.12.4-py3-none-any.whl (11 kB)
    Downloading fsspec-2023.10.0-py3-none-any.whl (166 kB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 166.4/166.4 kB 121.9 kB/s eta 0:00:00
    Downloading networkx-3.2-py3-none-any.whl (1.6 MB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 81.6 kB/s eta 0:00:00
    Downloading requests-2.31.0-py3-none-any.whl (62 kB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.6/62.6 kB 119.8 kB/s eta 0:00:00
    Downloading typing_extensions-4.8.0-py3-none-any.whl (31 kB)
    Downloading certifi-2023.7.22-py3-none-any.whl (158 kB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 158.3/158.3 kB 103.1 kB/s eta 0:00:00
    Downloading charset_normalizer-3.3.1-cp39-cp39-win_amd64.whl (98 kB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.7/98.7 kB 111.1 kB/s eta 0:00:00
    Downloading MarkupSafe-2.1.3-cp39-cp39-win_amd64.whl (17 kB)
    Downloading urllib3-2.0.7-py3-none-any.whl (124 kB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 124.2/124.2 kB 165.7 kB/s eta 0:00:00
    Installing collected packages: mpmath, urllib3, typing-extensions, sympy,
networkx, MarkupSafe, idna, fsspec, filelock, charset-normalizer, certifi, requests,
jinja2, torch, torchvision, torchaudio
    Successfully installed MarkupSafe-2.1.3 certifi-2023.7.22
charset-normalizer-3.3.1 filelock-3.12.4 fsspec-2023.10.0 idna-3.4 jinja2-3.1.2
mpmath-1.3.0 networkx-3.2 requests-2.31.0 sympy-1.12 torch-2.1.0 torchaudio-2.1.0
torchvision-0.16.0 typing-extensions-4.8.0 urllib3-2.0.7
    WARNING: There was an error checking the latest version of pip.
    PS C:\Users\xiayu>
```

验证 PyTorch 是否安装成功，执行如下命令，注意命令中的双下画线：

```
print(torch.__version__)
print(torch.version.cuda)
print(torch.cuda.is_available())
```

命令执行结果如下：

```
PS C:\Users\xiayu> python
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> print(torch.__version__)
2.1.0+cpu
>>> print(torch.version.cuda)
None
>>> print(torch.cuda.is_available())
False
>>>
```

如果没有报错，则说明 PyTorch 安装成功。

2.3　在 Windows 环境下安装 GPU 版的 PyTorch

2.3.1　确认显卡是否支持 CUDA

在深度学习中，我们经常要对图像数据进行处理和计算，而处理器 CPU 因为需要处理的事情多，并不能满足我们对图像处理和计算速度的要求，显卡 GPU 就是用来帮助 CPU 解决这个问题的，GPU 特别擅长处理图像数据。

为什么 GPU 特别擅长处理图像数据呢？这是因为图像上的每个像素点都有被处理的需要，而且每个像素点处理的过程和方式都十分相似，GPU 就是用很多简单的计算单元来完成大量的计算任务，类似于纯粹的人海战术。GPU 不仅可以在图像处理领域大显身手，它还被用在科学计算、密码破解、数值分析、海量数据处理（比如排序、Map-Reduce）、金融分析等需要大规模并行计算的领域。

而 CUDA（Compute Unified Device Architecture）是显卡厂商 NVIDIA 推出的只能用于自家 GPU 的并行计算架构，只有安装这个软件，才能够进行复杂的并行计算。该架构使 GPU 能够解决复杂的计算问题。它包含 CUDA 指令集架构（Instruction Set Architecture，ISA）以及 GPU 内部的并行计算引擎，安装 CUDA 之后，可以加快 GPU 的运算和处理速度，主流的深度学习框架也都是基于 CUDA 进行 GPU 并行加速的。

想要使用 GPU 加速，则需要安装 CUDA，首先需要自己的计算机显卡支持 CUDA 的安装，也就是查看自己的计算机有没有 NVIDA 的独立显卡。在 NVIDA 官网列表（https://developer.nvidia.com/cuda-gpus）中可以查看自己的显卡型号是否包括在 NVIDA 列表中。

在计算机桌面上右击，在弹出的菜单中如果能找到 NVIDIA 控制面板，如图 2-9 所示，则说明该计算机配有 GPU。

图 2-9

打开 NVIDIA 控制面板窗口，可以查看 NVIDIA 的一些信息，包括显卡的驱动版本，通过单击“帮助”菜单，并选择“系统信息”选项，查看系统信息获取支持的 CUDA 版本。如图 2-10 所示，选择“组件”，在 3D 设置模块找到 NVCUDA64.DLL，在该行可以看到该 NVCUDA 的版本，可以看到图中显示的版本是 11.8。

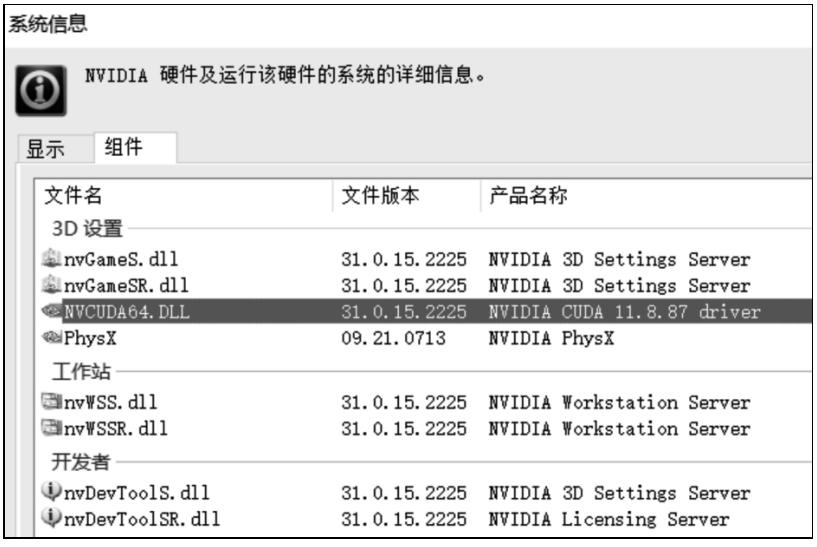


图 2-10

2.3.2　安装 CUDA

我们已经知道，CUDA 是显卡厂商 NVIDIA 推出的通用并行计算架构，能利用英伟达 GPU 的并行计算引擎。目前已经确认系统已有支持 CUDA 的显卡，这时可以到 NVIDIA 网站（https://developer.nvidia.com/cuda-toolkit-archive）下载 CUDA，如图 2-11 所示。注意，安装 CUDA

Driver 时，需要与 Nvidia GPU Driver 的版本驱动一致，CUDA 才能找到显卡。

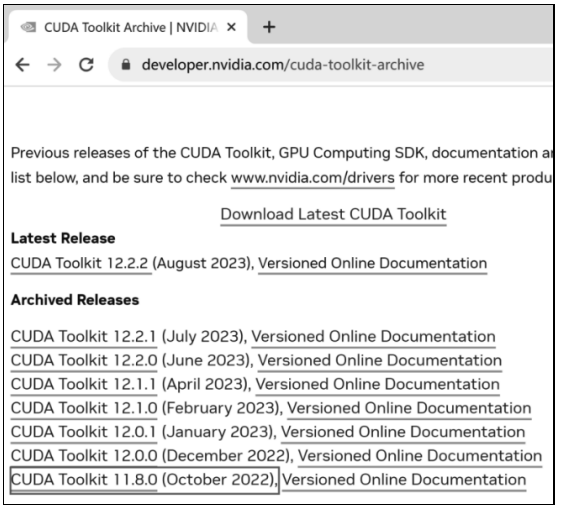


图 2-11

根据实际情况选择合适的版本，比如计算机操作系统是 Windows 10，这里下载 CUDA 11.8.0 的本地安装包，如图 2-12 所示。

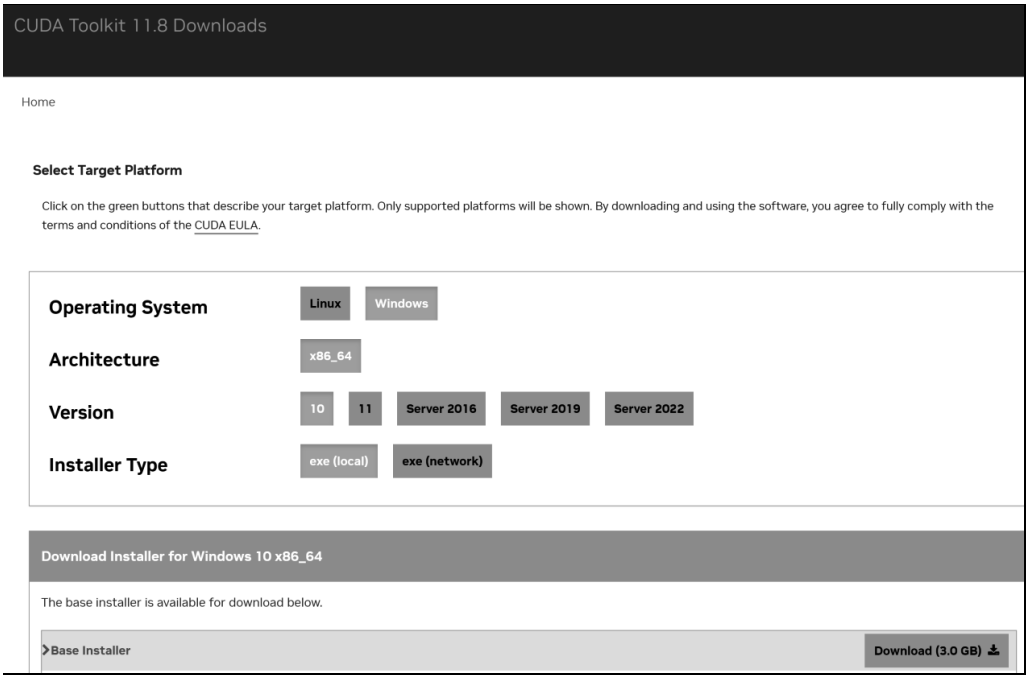


图 2-12

2.3.3　安装 cuDNN

接下来下载与 CUDA 对应版本的 cuDNN，cuDNN 是用于深度神经网络的 GPU 加速库，下载

地址为 https://developer.nvidia.com/rdp/cudnn-archive，页面如图 2-13 所示。

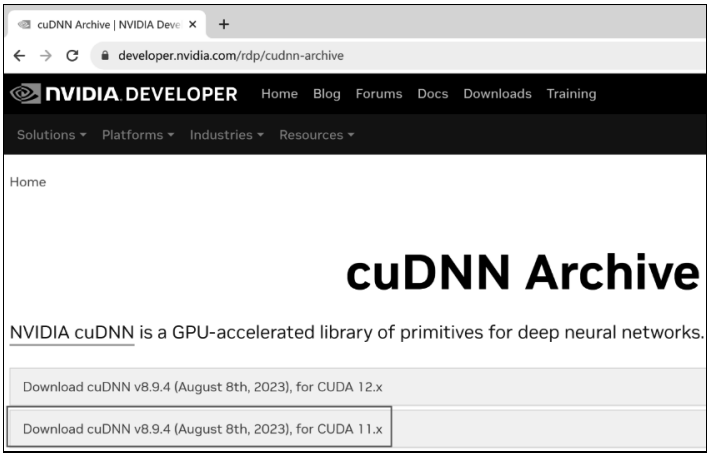


图 2-13

下载 cuDNN，需要注册英伟达开发者计划的会员账号才能下载，如图 2-14 所示，读者请自行注册账户，加入会员。

进入下载页面后，注意不要选择错误的版本，一定要找到对应 CUDA 的版本号。另外，如果使用的是 Windows 64 位的操作系统，那么需要对应下载 x86 版本的 cuDNN。

cuDNN 就是个压缩包，解压会生成 bin、include、lib 三个目录（见图 2-15），里面的文件复制到 CUDA 安装目录（这里是 C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0 目录）下对应的目录即可。注意，不是替换文件夹，而是将文件放入对应的文件夹中。

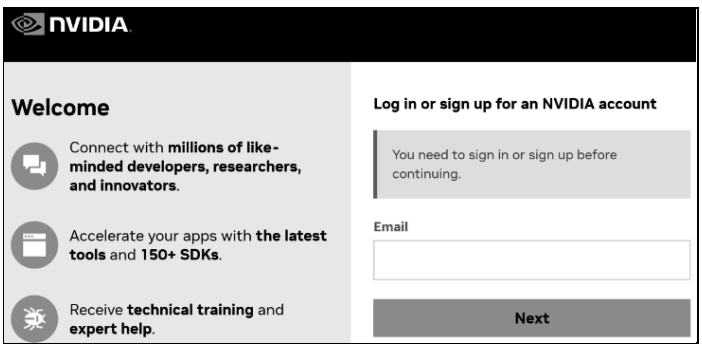
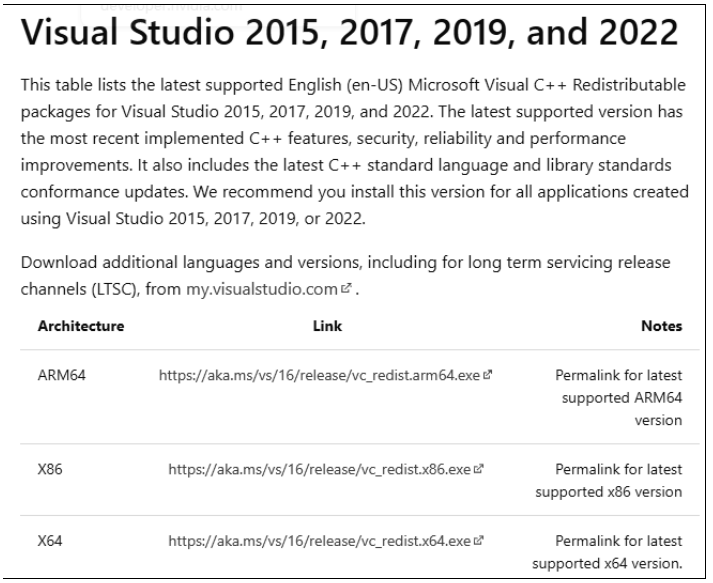


图 2-14

图 2-15

接下来安装 Visual Studio 2015、2017、2019 和 2022 支持库，这个支持库务必安装，否则后面安装 PyTorch 支持库会出现各种“坑”。支持库不大，有十多兆字节，安装完成后重启系统。其下载地址为 https://docs.microsoft.com/zh-CN/cpp/windows/latest-supported-vc-redist?view=msvc-160，下载页面如图 2-16 所示。



Visual Studio 2015, 2017, 2019, and 2022

This table lists the latest supported English (en-US) Microsoft Visual C++ Redistributable packages for Visual Studio 2015, 2017, 2019, and 2022. The latest supported version has the most recent implemented C++ features, security, reliability and performance improvements. It also includes the latest C++ standard language and library standards conformance updates. We recommend you install this version for all applications created using Visual Studio 2015, 2017, 2019, or 2022.

Download additional languages and versions, including for long term servicing release channels (LTSC), from my.visualstudio.com.

Architecture	Link	Notes
ARM64	https://aka.ms/vs/16/release/vc_redist.arm64.exe	Permalink for latest supported ARM64 version
X86	https://aka.ms/vs/16/release/vc_redist.x86.exe	Permalink for latest supported x86 version
X64	https://aka.ms/vs/16/release/vc_redist.x64.exe	Permalink for latest supported x64 version.

图 2-16

2.3.4　安装 GPU 版 PyTorch

安装 GPU 版本的 PyTorch 稍微复杂，前提是需要安装 CUDA、cuDNN 并行计算框架，然后安装 PyTorch。PyTorch 官网（地址为 https://pytorch.org）给出了匹配的版本号及安装命令，如图 2-17 所示。

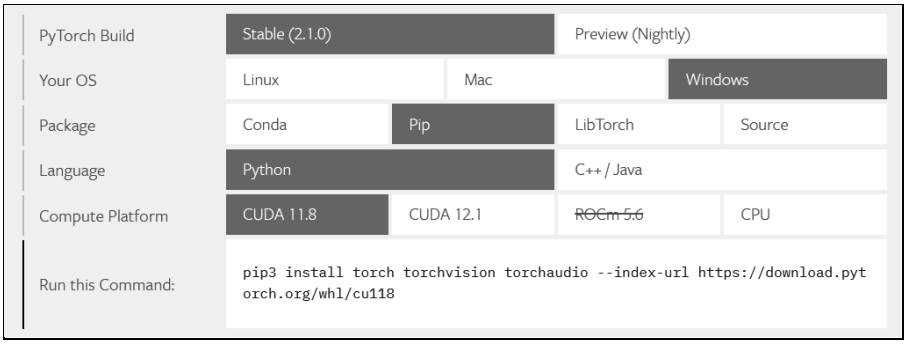


图 2-17

运行命令（Run this command）字段中的命令如下：

```
    pip3 install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu118
```

把此命令复制到管理员终端命令行，按回车键执行即可安装 GPU 版本的 PyTorch。PyTorch 安装好以后，执行下面的示例代码看看结果如何。

```
import torch
print(torch.__version__)
print(torch.cuda.is_available())
print("是否可用：", torch.cuda.is_available())        # 查看 GPU 是否可用
print("GPU 数量：", torch.cuda.device_count())        # 查看 GPU 数量
print("torch 方法查看 CUDA 版本：", torch.version.cuda)# torch 方法查看 CUDA 版本
print("GPU 索引号：", torch.cuda.current_device())    # 查看 GPU 索引号
print("GPU 名称：", torch.cuda.get_device_name(0))    # 根据索引号得到 GPU 名称
```

接下来，我们看 CPU 与 GPU 在模型训练时的性能差异对比图，如图 2-18 所示。

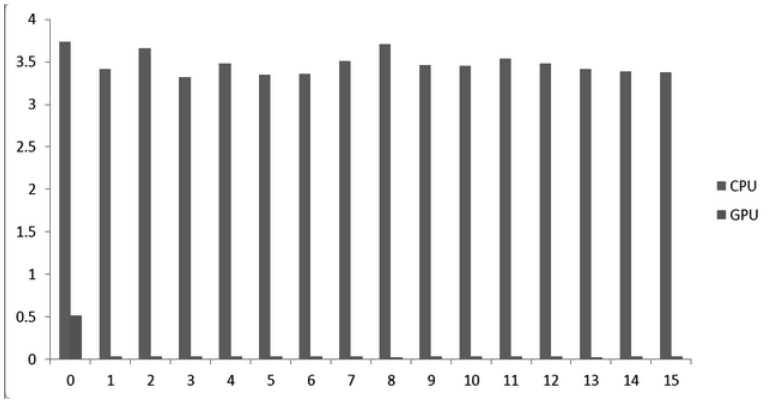


图 2-18　（颜色参见下载资源中的相关图片文件）

可以看出，15 次 bathsize=128 的耗时（秒）对比，GPU 快得非常多，大大减少了运行时间。如果有条件的话，尽量购买大显存的 GPU 显卡，在深度学习训练中能节省大量的计算时间，物超所值。