人工智能技术丛书

PyTorch 2.0

深度学习从零开始学

王晓华 著

清華大學出版社
北京

内 容 简 介

PyTorch 是一个开源的机器学习框架，它提供了动态计算图的支持，让用户能够自定义和训练自己的神经网络，目前是机器学习领域中最受欢迎的框架之一。本书基于 PyTorch 2.0，详细介绍深度学习的基本理论、算法和应用案例，配套示例源代码、PPT 课件。

本书共分 15 章，内容包括 PyTorch 概述、开发环境搭建、基于 PyTorch 的 MNIST 分类实战、深度学习理论基础、MNIST 分类实战、数据处理与模型可视化、基于 PyTorch 卷积层的分类实战、PyTorch 数据处理与模型可视化、实战 ResNet 卷积网络模型、有趣的 Word Embedding、基于循环神经网络的中文情感分类实战、自然语言处理的编码器、站在巨人肩膀上的预训练模型 BERT、自然语言处理的解码器、基于 PyTorch 的强化学习实战、基于 MFCC 的语音唤醒实战、基于 PyTorch 的人脸识别实战。

本书适合深度学习初学者、PyTorch 初学者、PyTorch 深度学习项目开发人员学习，也可作为高等院校或高职高专学校计算机技术、人工智能、智能科学与技术、数据科学与大数据技术等相关专业的教材。

图书在版编目（CIP）数据

PyTorch 2.0 深度学习从零开始学 / 王晓华著. —北京：清华大学出版社，2023.7
（人工智能技术丛书）
ISBN 978-7-302-64108-7

Ⅰ. ①P… Ⅱ. ①王… Ⅲ. ①机器学习 Ⅳ. ①TP181

中国国家版本馆 CIP 数据核字（2023）第 131187 号

责任编辑：夏毓彦
封面设计：王　翔
责任校对：闫秀华
责任印制：曹婉颖

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-83470000　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 装 者：三河市东方印刷有限公司
经　　销：全国新华书店
开　　本：190mm×260mm　　印　　张：18.25　　字　　数：492 千字
版　　次：2023 年 8 月第 1 版　　印　　次：2023 年 8 月第 1 次印刷
定　　价：69.00 元

产品编号：102831-01

前　言

PyTorch 是一个开源的机器学习框架，它提供了动态计算图的支持，让用户能够自定义和训练自己的神经网络。它由 Facebook 的研究团队开发，并于 2017 年首次发布，从那时起，PyTorch 迅速成为机器学习领域最受欢迎的框架之一。

PyTorch 在学术界和产业界都得到了广泛的应用，被用于完成各种任务，例如图像分类、自然语言处理、目标检测等。在 2019 年，PyTorch 被 Google 和 OpenAI 等机构评选为机器学习框架的首选，这也进一步证明了 PyTorch 在机器学习领域的重要性。

关于本书

本书是一本以 PyTorch 2.0 为框架的深度学习实战图书，以通俗易懂的方式介绍深度学习的基础内容与理论，并以项目实战的形式详细介绍 PyTorch 框架的使用。本书从单个 API 的使用，到组合架构完成进阶的项目实战，全面介绍使用 PyTorch 2.0 进行深度学习项目实战的核心技术和涉及的相关知识，内容丰富而翔实。

同时，本书不仅仅是一本简单的项目实战性质的图书，本书在讲解和演示实例代码的过程中，对 PyTorch 2.0 的核心内容进行深入分析，重要内容均结合代码进行实战讲解，围绕深度学习的基本原理介绍大量案例，读者通过这些案例可以深入掌握深度学习和 PyTorch 2.0 的相关技术及其应用，并能提升使用深度学习框架进行真实的项目实战的能力。

本书特点

（1）重实践，讲原理。本书立足于深度学习，以实战为目的，以新版的 PyTorch 2.0 为基础框架，详细介绍深度学习基本原理以及示例项目的完整实现过程，并提供可运行的全套示例代码，帮助读者在直接使用代码的基础上掌握深度学习的原理与应用方法。

（2）版本新，易入门。本书详细讲解 PyTorch 2.0 的安装和使用，包括 PyTorch 2.0 的重大优化和改进方案，以及官方默认使用的 API 和官方推荐的编程方法与技巧。

（3）作者经验丰富，代码编写优雅细腻。作者是长期奋战在科研和工业界的一线算法设计和程序编写人员，实战经验丰富，对代码中可能会出现的各种问题和“坑”有丰富的处理经验，使得读者能够少走很多弯路。

（4）理论扎实，深入浅出。在代码设计的基础上，本书还深入浅出地介绍深度学习需要掌握的一些基本理论知识，作者以大量的公式与图示相结合的方式进行理论讲解，是一本难得的好书。

（5）对比多种应用方案，实战案例丰富。本书采用了大量的实例，同时也提供了实现同类功能的多种解决方案，覆盖使用 PyTorch 2.0 进行深度学习开发常用的知识。

配套示例源代码、PPT 课件等资源下载

本书配套示例源代码、PPT 课件，需要用微信扫描下面的二维码获取。如果阅读中发现问题或疑问，请联系 booksaga@163.com，邮件主题写“PyTorch 2.0 深度学习从零开始学”。

本书读者

- 深度学习初学者
- PyTorch 初学者
- PyTorch 深度学习项目开发人员
- 计算机技术、人工智能、智能科学与技术、数据科学与大数据技术等专业的师生

作　者
2023 年 5 月

目　　录

第 1 章

PyTorch 2.0——一个新的开始

PyTorch 是一个开源的机器学习框架，提供了动态计算图的支持，让用户能够自定义和训练自己的神经网络。它由Facebook的研究团队开发，并于2017年首次发布，从那时起，PyTorch迅速成为机器学习领域最受欢迎的框架之一。

PyTorch 在学术界和产业界都得到了广泛的应用，被用于实现各种任务，例如图像分类、自然语言处理、目标检测等。2019年，PyTorch被Google和OpenAI等机构评选为机器学习框架的首选，这也进一步证明了PyTorch在机器学习领域的重要性。

1.1 燎原之势的人工智能

人工智能作为当今信息科技最炙手可热的研究领域之一，近年来得到了越来越多的关注。然而，人工智能并不是一蹴而就的产物，而是在不断发展、演变的过程中逐渐形成的。人工智能从无到有是一个漫长而又不断迭代的过程。

1.1.1 从无到有的人工智能

人工智能的发展可以追溯到20世纪50年代，当时的计算机技术还非常落后，但是人们已经开始思考如何让计算机具备人类的智能。于是，在20世纪50年代末期，人工智能这个概念正式被提出。最早的人工智能技术主要是基于规则的，即通过编写一些规则来让计算机进行推理和决策。然而，这种方法很快就被证明是不够灵活的，无法应对各种复杂的情况。

为了解决这个问题，人们开始研究机器学习。机器学习是一种让计算机从数据中学习规律的方法。最早的机器学习算法主要是基于统计学的方法，如线性回归、逻辑回归等。这些算法主要用于解决一些简单的问题，如分类、回归等。

随着数据量的不断增大和计算能力的提升，人们开始研究更加复杂的机器学习算法，如神经网络。神经网络是一种模仿人脑神经元的网络结构，可以用于解决各种复杂的问题，如图像识别、语音识别、自然语言处理等。然而，在早期的研究中，神经网络还存在训练时间长、容易陷入局部最优解等问题。

为了解决这个问题，人们开始研究深度学习。深度学习是一种多层神经网络的方法，通过层层抽象可以获取更加高级、更加抽象的特征表示。深度学习在图像识别、语音识别、自然语言处理等领域都取得了非常显著的成果。

除了深度学习外，人们还在不断探索其他的人工智能算法，如遗传算法、强化学习等。这些算法在不同的领域都取得了一定的成果。

随着互联网的普及和大数据时代的到来，数据变得越来越容易获取。同时，计算能力的提升也为人工智能的发展提供了坚实的基础。GPU 的出现使得人们能够更加高效地训练深度学习模型，云计算的发展使得人们能够更加轻松地部署和运行人工智能应用。

如今，人工智能已经渗透到了各个领域，如医疗、金融、交通、制造等。人工智能已经成为许多企业的核心竞争力，也成为推动社会进步的重要力量。

然而，人工智能的发展依然面临着许多挑战。一方面，人工智能的算法和技术还有很大的提升空间，例如如何让人工智能具备更好的理解能力、创造能力、推理能力等。另一方面，人工智能的应用还需要面对许多社会、伦理、法律等方面的问题。例如如何保障人工智能的安全、隐私和公正性，如何处理人工智能和人类的关系等。

总之，人工智能从无到有是一个漫长而又不断迭代的过程。虽然人工智能已经取得了许多显著的成果，但是还有很多挑战需要我们去面对。相信在不久的将来，人工智能会继续发展，成为推动人类社会进步的重要力量。

1.1.2　深度学习与人工智能

深度学习作为人工智能领域的一种重要技术，正在引领人工智能的发展。它利用多层神经网络模拟人脑的处理方式，可以实现很多人类难以完成的任务，如图像识别、语音识别、自然语言处理等。在过去的几年中，深度学习在各个领域取得了巨大的成功，成为人工智能领域的一颗璀璨的明珠。

深度学习的核心是神经网络，它可以被看作是由许多个简单的神经元组成的网络。这些神经元可以接收输入并产生输出，通过学习不同的权重来实现不同的任务。深度学习的“深度”指的是神经网络的层数，即多层神经元的堆叠。在多层神经网络中，每一层的输出都是下一层的输入，每一层都负责提取不同层次的特征，从而完成更加复杂的任务。

深度学习在人工智能领域的成功得益于其强大的表征学习能力。表征学习是指从输入数据中学习到抽象的特征表示的过程。深度学习模型可以自动学习到数据的特征表示，并从中提取出具有区分性的特征，从而实现对数据的分类、识别等任务。

深度学习在图像识别、语音识别、自然语言处理等领域都取得了很好的效果。例如，在图像识别领域，深度学习已经取代了传统的机器学习方法成为主流，可以实现对复杂场景中的物体进行精确识别和定位。在自然语言处理领域，深度学习可以实现文本的情感分析、机器翻译等任务。在语音识别领域，深度学习可以将语音转换为文本，实现语音助手等应用。

1.1.3　应用深度学习解决实际问题

深度学习是一种机器学习技术，它可以通过大量的数据来训练模型，以解决现实生活中的实际问题。深度学习技术在多个领域都有广泛应用，例如自然语言处理、计算机视觉、智能控制等。接

下来将介绍几个使用深度学习解决实际问题的案例，并探讨深度学习技术的优势。

1. 人脸识别

人脸识别是一种广泛应用的技术，它可以用于安全认证、人脸支付、人脸考勤等。深度学习技术在人脸识别中具有优势，因为它可以从大量的数据中学习特征，使得识别准确率更高。例如，FaceNet 是一个基于深度学习的人脸识别系统，它可以在不同光照和角度下准确地识别同一个人的脸。

2. 自然语言处理

自然语言处理是一种重要的人工智能应用，它可以用于文本分类、情感分析、机器翻译等。深度学习技术在自然语言处理中也有广泛应用。例如，Google 的翻译系统就使用了深度学习技术来提高翻译的准确率。此外，深度学习技术还可以用于语音识别、语音合成等方面，以提高人机交互的体验。

3. 智能控制

智能控制是一种应用广泛的技术，它可以用于自动化控制、机器人控制、智能家居等。深度学习技术在智能控制中的应用也越来越多。例如，深度强化学习可以用于机器人控制，通过从环境中不断学习来优化机器人的决策。此外，深度学习技术还可以用于自动驾驶、航空航天等方面，以提高自主决策的能力。

1.1.4　深度学习技术的优势和挑战

深度学习技术的优势主要体现在以下几个方面。

1. 准确性更高

深度学习技术可以通过大量的数据来训练模型，从而学习到更多的特征，以提高识别准确率。例如，传统的人脸识别技术可能只能识别人脸的一部分特征，而深度学习技术可以学习到更多的细节，从而提高人脸识别的准确率。

2. 可以处理更复杂的问题

传统的机器学习技术在处理复杂问题时往往受限于特征工程的能力，而深度学习技术可以通过学习大量的数据来自动提取特征，从而可以处理更复杂的问题。

3. 可以逐步优化

深度学习技术可以通过逐步优化来提高模型的性能。例如，可以通过改变模型的结构、增加数据量、调整超参数等方法来提高模型的性能。

4. 可以适应不同的场景

深度学习技术可以根据不同的场景进行适应性调整。例如，对于不同的语音、图像等数据，可以通过不同的网络结构和训练方式进行处理。

虽然深度学习技术有着诸多优势，但也存在一些挑战。例如，深度学习技术需要大量的数据

来训练模型，这需要消耗大量的计算资源和存储空间。此外，深度学习模型的解释性也相对较差，难以解释模型内部的决策过程。

总的来说，深度学习技术在解决实际问题方面具有广泛的应用前景。随着计算资源的不断提升和算法的不断优化，深度学习技术在未来将会发挥更加重要的作用。

1.2　为什么选择 PyTorch 2.0

在 PyTorch Conference 2022 上，PyTorch 官方正式发布了 PyTorch 2.0，整场活动含“compiler”率极高，跟先前的 1.x 版本相比，2.0 版本有了颠覆式的变化。

PyTorch 2.0 中发布了大量足以改变 PyTorch 使用方式的新功能，它提供了相同的 Eager 模式和用户体验，同时通过 torch.compile 增加了一个编译模式，在训练和推理过程中可以对模型进行加速，从而提供更佳的性能以及对 Dynamic Shapes 和 Distributed 的支持。

1.2.1　PyTorch 的前世今生

PyTorch 是一个 Python 开源机器学习库，它可以提供强大的 GPU 加速张量运算和动态计算图，方便用户进行快速实验和开发。PyTorch 由 Facebook 的人工智能研究小组于 2016 年发布，当时它作为 Torch 的 Python 版，目的是解决 Torch 在 Python 中使用的不便之处。

Torch 是另一个开源机器学习库，它于 2002 年由 Ronan Collobert 创建，主要基于 Lua 编程语言。Torch 最初是为了解决语音识别的问题而创建的，但随着时间的推移，Torch 开始被广泛应用于其他机器学习领域，包括计算机视觉、自然语言处理、强化学习等。

尽管 Torch 在机器学习领域得到了广泛的应用，但是它在 Python 中的实现相对麻烦，导致它在 Python 社区的使用率不如其他机器学习库（如 TensorFlow）。这也就迫使了 Facebook 的人工智能研究小组开始着手开发 PyTorch。

在 2016 年，PyTorch 首次发布了其 Alpha 版本，但是该版本的使用范围比较有限。直到 2017 年，PyTorch 正式发布了其 Beta 版本，这使得更多的用户可以使用 PyTorch 进行机器学习实验和开发。在 2018 年，PyTorch 1.0 版本正式发布，这也标志着 PyTorch 开始成为机器学习领域最受欢迎的开源机器学习库之一。

PyTorch 在国际学术界和工业界都得到了广泛的认可，并在实践得到广泛的应用。同时，PyTorch 持续更新和优化，使得用户可以在不断的技术发展中获得更好的使用体验。

1.2.2　更快、更优、更具编译支持——PyTorch 2.0 更好的未来

PyTorch 2.0 的诞生使得 PyTorch 的性能进一步提升，并开始将 PyTorch 的部分内容从 C++阵营拉到 Python 阵营中。而其中最为人津津乐道的新技术包括 TorchDynamo、AOTAutograd、PrimTorch 以及 TorchInductor。

1. TorchDynamo

TorchDynamo 可以借助 Python Frame Evaluation Hooks（Python 框架评估钩子），安全地获取 PyTorch 程序，这项重大创新是 PyTorch 过去 5 年来在安全图结构捕获（Safe Graph Capture）方面研发成果的汇总。

2. AOTAutograd

AOTAutograd 重载 PyTorch Autograd Engine（PyTorch 自动微分引擎），作为一个 Tracing Autodiff，用于生成超前的 Backward Trace（后向追溯）。

3. PrimTorch

PrimTorch 将 2 000 多个 PyTorch 算子归纳为约 250 个 Primitive Operator 闭集（Closed Set），开发者可以针对这些算子构建一个完整的 PyTorch 后端。PrimTorch 大大简化了编写 PyTorch 功能或后端的流程。

4. TorchInductor

TorchInductor 是一个深度学习编译器，可以为多个加速器和后端生成快速代码。对于 NVIDIA GPU，它使用 OpenAI Triton 作为关键构建模块。

TorchDynamo、AOTAutograd、PrimTorch 和 TorchInductor 是用 Python 编写的，并且支持 Dynamic Shape（无须重新编译就能发送不同大小的向量），这使得它们灵活且易学，降低了开发者和供应商的准入门槛。

除此之外，PyTorch 2.0 官宣了一个重要特性——torch.compile，这一特性将 PyTorch 的性能推向了新的高度，并将 PyTorch 的部分内容从 C++移回 Python。torch.compile 是一个完全附加的（可选的）特性，因此 PyTorch 2.0 是 100%向后兼容的。

当然，PyTorch 2.0 目前只推出了改革的第一个版本，随着后续 PyTorch 社区以及维护团队的修正和更新，PyTorch 一定会迎来更好的未来。

1.2.3　PyTorch 2.0 学习路径——从零基础到项目实战

学习 PyTorch 的步骤可能因个人情况而有所不同，以下是一般的学习步骤：

（1）安装 PyTorch 和 Python：用户可以在 PyTorch 官方网站上找到安装说明，并选择合适的版本和平台。

（2）学习 Python 基础知识：如果用户还不熟悉 Python，那么需要先学习 Python 基础知识，包括变量、数据类型、控制流语句、函数和模块等。

（3）了解深度学习基础知识：学习深度学习前，需要先了解神经网络、损失函数、优化算法和梯度下降等基础概念。

（4）学习 PyTorch 基础知识：学习 PyTorch 张量（Tensor）、自动微分（Autograd）、模块（Module）和优化器（Optimizer）等 PyTorch 基础知识。

（5）实践 PyTorch：尝试使用 PyTorch 进行一些简单的任务，比如创建张量、定义神经网络模型、计算损失函数和梯度、优化模型参数等。

（6）学习 PyTorch 高级知识：深入学习 PyTorch 的高级知识，包括 PyTorch 的 GPU 加速、数据处理、模型保存和加载、模型微调和迁移学习等。

（7）完成深度学习实战项目：使用 PyTorch 完成多个深度学习项目，例如图像分类、物体检测、文本生成等。这些项目可以来自实际问题或公开的数据集，目的是将前面所学的知识应用于实际情况中。

最终总结一句话，前途是光明的，道路是曲折的。实践出真知，学习 PyTorch 2.0 是一个需要仔细钻研的过程，不仅要学习理论，还需要自己动手实践才能了解其中的奥义。感谢读者选择本书来进行 PyTorch 2.0 的学习之旅，希望沉下心来，认认真真地掌握这个领域的知识。

1.3 本 章 小 结

本章介绍了人工智能技术发展简史、深度学习与人工智能的关系、深度学习能解决的实际问题，以及 PyTorch 的发展历程和最新技术，并给出了 PyTorch 2.0 的学习路径——从零基础到项目实战。

第 2 章

Hello PyTorch 2.0
——深度学习环境搭建

工欲善其事，必先利其器。第 1 章介绍了 PyTorch 与深度学习神经网络之间的关系，本章将正式进入 PyTorch 2.0 的学习过程。

首先读者需要知道，无论是构建深度学习应用程序，还是应用已完成训练的项目到某项具体项目中，都需要使用编程语言完成设计者的目的，本书使用 Python 语言作为开发的基本语言。

Python 是深度学习的首选开发语言，很多第三方提供了集成大量科学计算库的 Python 标准安装包，常用的是 Miniconda 和 Anaconda。Python 是一个脚本语言，如果不使用 Miniconda 或者 Anaconda，那么第三方库的安装会较为困难，各个库之间的依赖性就很难连接得很好。因此，这里推荐使用 Miniconda，当然对 Python 语言非常熟悉的读者也可以直接使用原生 Python。

本章首先介绍 Miniconda 的完整安装，然后完成一个练习项目——生成可控的手写体数字，这是一个入门程序，可以帮助读者了解完整的 PyTorch 项目的工作过程。

2.1 安装 Python

2.1.1 Miniconda 的下载与安装

1. 下载和安装

进入 Miniconda 官方网站，打开下载页面，如图 2-1 所示。

读者可以根据自己使用的操作系统选择不同平台的 Miniconda 下载，目前官网提供的是最新集成了 Python 3.10 版本的 Miniconda。如果读者目前使用的是以前的 Python 版本，例如 Python 3.7，也是完全可以的。从图中可以看到，使用 3.7 ~ 3.10 版本的 Python 都能支持 PyTorch 的使用。读者可以根据自己的操作系统选择下载相应的文件。

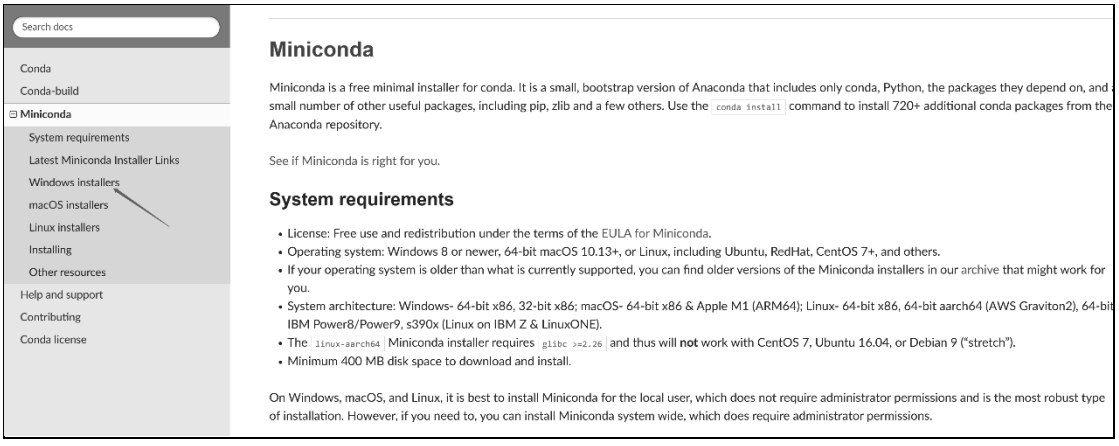

图 2-1　下载页面

这里推荐使用 Windows Python 3.9 版本，相对于 3.10 版本，3.9 版本经过一段时间的训练，具有一定的稳定性。当然，读者可根据自己的喜好选择集成 Python 3.10 版本的 Miniconda。下载页面如图 2-2 所示。

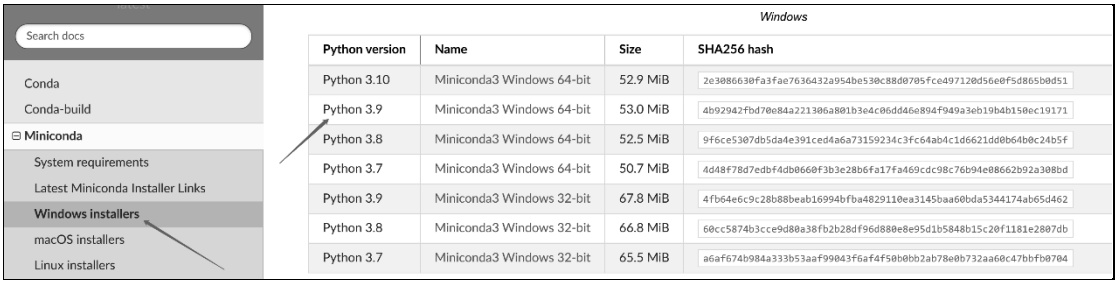

图 2-2　集成 Python 3.10 版本的官方网站 Miniconda3 下载页面

注意：如果读者使用的是 64 位操作系统，那么可以选择以 Miniconda3 开头、以 64 结尾的安装文件，不要下载错了。

下载完成后得到的安装文件是 exe 版本，直接运行即可进入安装过程，安装比较简单，按界面提示进行操作即可。安装完成后，出现如图 2-3 所示的目录结构，说明安装正确。

图 2-3　Miniconda3 安装目录

2. 打开控制台

依次单击“开始”→“所有程序”→Miniconda3→Miniconda Prompt，打开 Miniconda Prompt 窗口，它与 CMD 控制台类似，输入命令就可以控制和配置 Python。在 Miniconda 中最常用的是 conda 命令，该命令可以执行一些基本操作。

3. 验证Python

接下来在控制台中输入 python，若安装正确，则会打印版本号和控制符号。在控制符号下输入以下代码：

```
print("hello Python")
```

输出结果如图 2-4 所示，可以验证 Miniconda Python 已经安装成功。

```
(base) C:\Users\xiaohua>python
Python 3.9.10 | packaged by conda-forge | (main, Feb  1 2022, 21:22:07) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello")
hello
>>>
```

图 2-4　安装成功

4. 使用pip命令

使用 Miniconda 的好处在于，它能够很方便地帮助读者安装和使用大量第三方类库。查看已安装的第三方类库的命令如下：

```
pip list
```

注意：如果此时命令行还处于>>>状态，那么可以输入 exit()退出。

在 Miniconda Prompt 控制台输入 pip list 命令，结果如图 2-5 所示。

```
(base) C:\Users\xiaohua>pip list
WARNING: Ignoring invalid distribution -qdm (c:\miniforge3\lib\site-packages)
WARNING: Ignoring invalid distribution -harset-normalizer (c:\miniforge3\lib\site-packages)
WARNING: Ignoring invalid distribution -ensorflow-gpu (c:\miniforge3\lib\site-packages)
Package                       Version
----------------------------- ---------------------
absl-py                       1.0.0
aiofiles                      0.8.0
aiohttp                       3.8.1
aiosignal                     1.2.0
alabaster                     0.7.12
altair                        4.2.0
altgraph                      0.17.2
anyio                         3.5.0
argon2-cffi                   21.1.0
arrow                         1.1.1
```

图 2-5　已安装的第三方类库

Miniconda 中使用 pip 进行的操作还有很多，其中最重要的是安装第三方类库，命令如下：

```
pip install name
```

这里的 name 是需要安装的第三方类库名，假设需要安装 NumPy 包（这个包已经安装过），那么输入的命令如下：

```
pip install numpy
```

结果如图 2-6 所示。

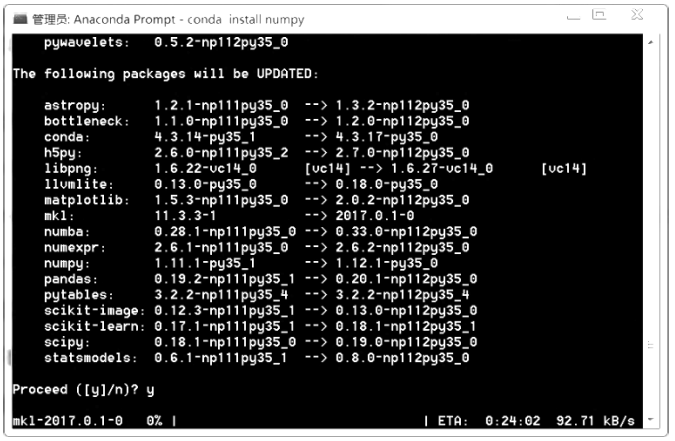

图 2-6　安装 NumPy 包

使用 Miniconda 的一个好处是默认安装了大部分深度学习所需要的第三方类库，这样可以避免使用者在安装和使用某个特定的类库时出现依赖类库缺失的情况。

2.1.2　PyCharm 的下载与安装

和其他语言类似，Python 可以使用 Windows 自带的控制台进行程序编写。但是这种方式对于较为复杂的程序工程来说，容易混淆相互之间的层级和交互文件，因此在编写程序工程时，建议使用专用的 Python 编译器 PyCharm。

1. PyCharm的下载和安装

（1）进入 PyCharm 官网的 Download 页面后，可以选择不同的版本，如图 2-7 所示，有收费的专业版和免费的社区版。这里读者选择免费的社区版即可。

图 2-7　选择 PyCharm 的免费版

（2）双击运行后进入安装界面，如图 2-8 所示。直接单击 Next 按钮，采用默认安装即可。

（3）如图 2-9 所示，在安装 PyCharm 的过程中需要选择安装的位数，这里建议读者选择与已安装的 Python 相同位数的文件。

图 2-8　安装界面

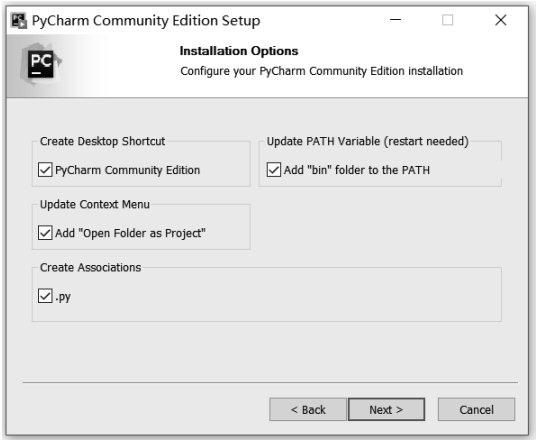

图 2-9　选择安装的位数

（4）安装完成后，单击 Finish 按钮，如图 2-10 所示。

图 2-10　安装完成

2. 使用PyCharm创建程序

（1）单击桌面上新生成的图标进入 PyCharm 程序界面，首先是第一次启动的定位，如图 2-11 所示。这里是对程序存储的定位，一般建议选择第 2 个 Do not import settings。

（2）单击 OK 按钮后进入 PyChrarm 配置窗口，如图 2-12 所示。

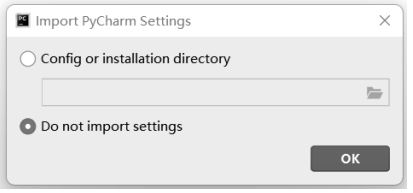

图 2-11　由 PyCharm 自动指定

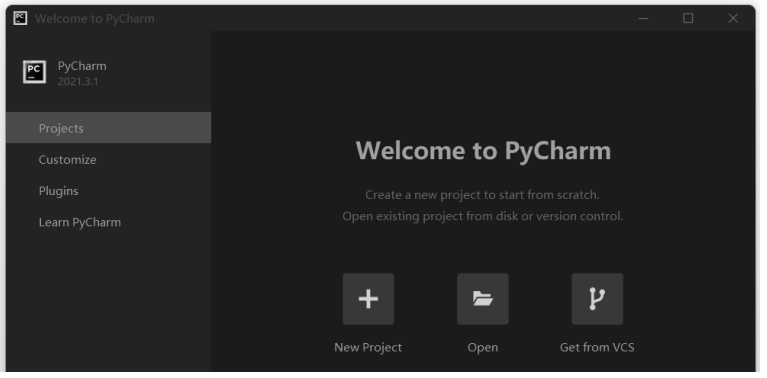

图 2-12　界面配置

（3）在配置窗口上可以对 PyCharm 的界面进行配置，选择自己的使用风格。如果对其不熟悉，直接使用默认配置即可，如图 2-13 所示。

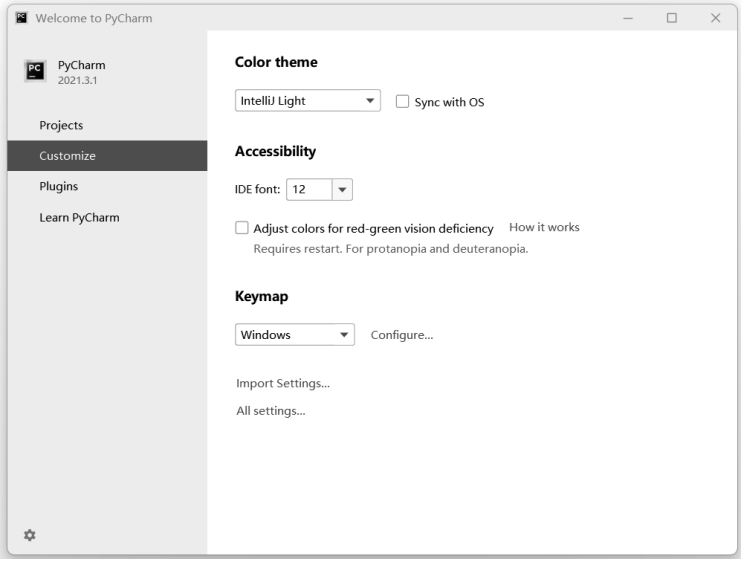

图 2-13　对 PyCharm 的界面进行配置

（4）创建一个新的工程，如图 2-14 所示。

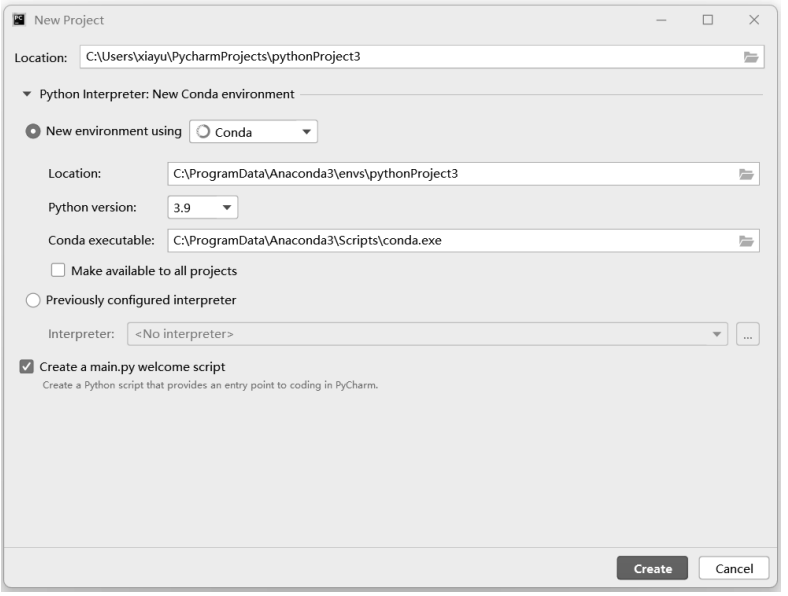

图 2-14　创建一个新的工程

这里，建议新建一个 PyCharm 的工程文件，如图 2-15 所示。

之后右击新建的工程名 PyCharm，选择 New→Python File 菜单，新建一个 helloworld.py 文件，内容如图 2-16 所示。

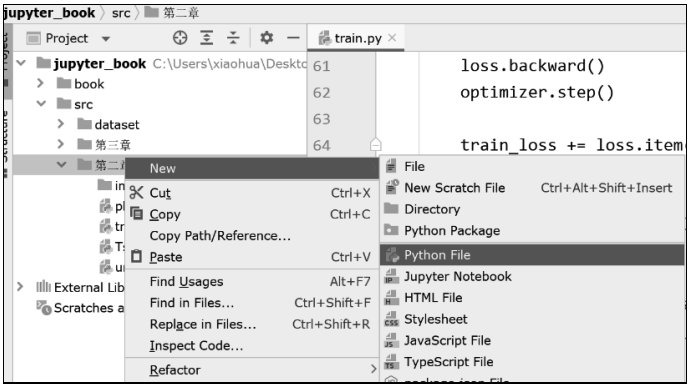

图 2-15　新建一个 PyCharm 的工程文件

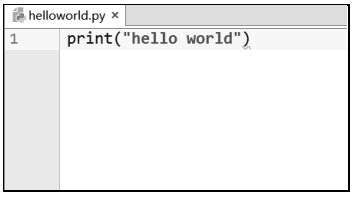

图 2-16　helloworld.py

输入代码并单击菜单栏的 Run→run…运行代码，或者直接右击 helloworld.py 文件名，在弹出的快捷菜单中选择 run。如果成功输出 hello world，那么恭喜你，Python 与 PyCharm 的配置就完成了。

2.1.3　Python 代码小练习：计算 Softmax 函数

对于 Python 科学计算来说，最简单的想法就是将数学公式直接表达成程序语言，可以说，Python 满足了这个想法。本小节将使用 Python 实现一个深度学习中最为常见的函数——Softmax 函数。至于这个函数的作用，现在不加以说明，笔者只是带领读者尝试实现其程序的编写。

Softmax 的计算公式如下：

$$S_i = \frac{e^{V_i}}{\sum_{0}^{j} e^{V_i}}$$

其中，V_i是长度为 j 的数列 V 中的一个数，代入 Softmax 的结果其实就是先对每一个 V_i取 e 为底的指数计算变成非负，然后除以所有项之和进行归一化，之后每个 V_i就可以解释成：在观察到的数据集类别中，特定的 V_i属于某个类别的概率，或者称作似然（Likelihood）。

提示：Softmax 用以解决概率计算中概率结果大而占绝对优势的问题。例如函数计算结果中的两个值 a 和 b，且 $a>b$，如果简单地以值的大小为单位来衡量，那么在后续的使用过程中，a 永远被选用，而 b 由于数值较小而不会被选择，但是有时也需要使用数值小的 b，Softmax 就可以解决这个问题。

Softmax 按照概率选择 a 和 b，由于 a 的概率值大于 b，在计算时 a 经常会被取得，而 b 由于概率较小，被取得的可能性也较小，但是也有概率被取得。

Softmax 的代码如下：

```
import numpy
def softmax(inMatrix):
    m,n = numpy.shape(inMatrix)
    outMatrix = numpy.mat(numpy.zeros((m,n)))
    soft_sum = 0
    for idx in range(0,n):
        outMatrix[0,idx] = math.exp(inMatrix[0,idx])
        soft_sum += outMatrix[0,idx]
    for idx in range(0,n):
        outMatrix[0,idx] = outMatrix[0,idx] / soft_sum
    return outMatrix
a = numpy.array([[1, 2, 1, 2, 1, 1, 3]])
print(softmax(a))
```

可以看到，当传入一个数列后，分别计算每个数值所对应的指数函数值，之后将其相加后计算每个数值在数值和中的概率。结果请读者自行打印验证。

2.2　安装 PyTorch 2.0

Python 运行环境调试完毕后，接下来的重点就是安装本书的主角 PyTorch 2.0。由于 CPU 版本的 PyTorch 相对 GPU 版本的 PyTorch 来说，运行速度较慢，我们推荐安装 GPU 版本的 PyTorch。

2.2.1　Nvidia 10/20/30/40 系列显卡选择的 GPU 版本

由于 40 系显卡的推出，目前市场上会有 Nvidia 10、20、30、40 系列显卡并存的情况。对于需要调用专用编译器的 PyTorch 来说，不同的显卡需要安装不同的依赖计算包，作者在此总结了不同显卡的 PyTorch 版本以及 CUDA 和 cuDNN 的对应关系，如表 2-1 所示。

表 2-1　10/20/30/40 系列显卡的版本对比

显卡型号	PyTorch GPU 版本	CUDA 版本	cuDNN 版本
10 系列及以前	PyTorch 2.0 以前的版本	11.1	7.65
20/30/40 系列	PyTorch 2.0 向下兼容	11.6+	8.1+

注意：这里的区别主要在于显卡运算库 CUDA 与 cuDNN 的区别，当在 20/30/40 系列显卡上使用 PyTorch 时，可以安装 11.6 以上版本以及 cuDNN 8.1 以上版本的计算包，而在 10 系列版本的显卡上，建议优先使用 2.0 版本以前的 PyTorch。

下面以 CUDA 11.7+cuDNN 8.2.0 组合为例，演示完整的 PyTorch 2.0 GPU Nvidia 运行库的安装步骤，其他不同版本 CUDA+cuDNN 组合的安装过程基本一致。

2.2.2　PyTorch 2.0 GPU Nvidia 运行库的安装——以 CUDA 11.7+cuDNN 8.2.0 为例

从 CPU 版本的 PyTorch 开始深度学习之旅完全是可以的，但不是作者推荐的方式。相对于 GPU 版本的 PyTorch 来说，在运行速度方面 CPU 版本存在着极大的劣势，很有可能会让读者的深度学习止步于前。

如果读者的电脑不支持 GPU，可以直接使用 PyTorch 2.0 CPU 版本的安装命令：

```
    pip install numpy --pre torch torchvision torchaudio --force-reinstall
--extra-index-url https://download.pytorch.org/whl/nightly/cpu
```

如果读者的电脑支持 GPU，则继续下面本小节的重头戏，PyTorch 2.0 GPU 版本的前置软件的安装。对于 GPU 版本的 PyTorch 来说，由于调用了 NVIDA 显卡作为其代码运行的主要工具，因此额外需要 NVIDA 提供的运行库作为运行基础。

对于 PyTorch 2.0 的安装来说，最好根据官方提供的安装代码进行安装，如图 2-17 所示。在这里 PyTorch 官方提供了两种安装模式，分别对应 CUDA 11.7 与 CUDA 11.8。

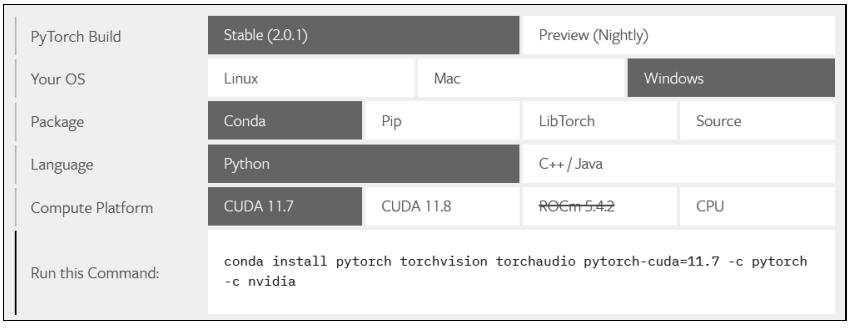

图 2-17　PyTorch 官网提供的配置信息

从图中可以看到，这里提供了两种不同的 CUDA 版本的安装，作者经过测试，无论是使用 CUDA 11.7 还是 CUDA 11.8，在 PyTorch 2.0 的程序编写上没有显著的区别，因此读者可以根据安装配置自行选择。下面以 CUDA 11.7 为例讲解安装的方法。

（1）CUDA 的安装。在百度搜索 CUDA 11.7 download，进入官方下载页面，选择合适的操作系统安装方式（推荐使用 local 本地化安装方式），如图 2-18 所示。

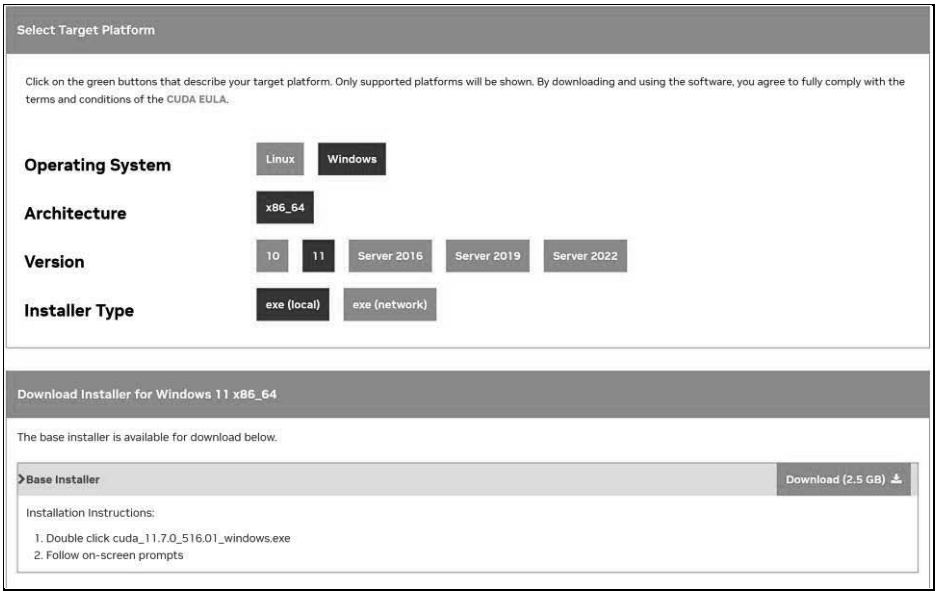

图 2-18　CUDA 下载页面

此时下载的是一个.exe 文件，读者自行安装，不要修改其中的路径信息，使用默认路径安装即可。

（2）下载和安装对应的 cuDNN 文件。cuDNN 的下载需要先注册一个用户，相信读者可以很快完成，之后直接进入下载页面，如图 2-19 所示。注意：不要选择错误的版本，一定要找到对应的版本号，另外，如果使用的是 Windows 64 位的操作系统，那么直接下载 x86 版本的 cuDNN 即可。

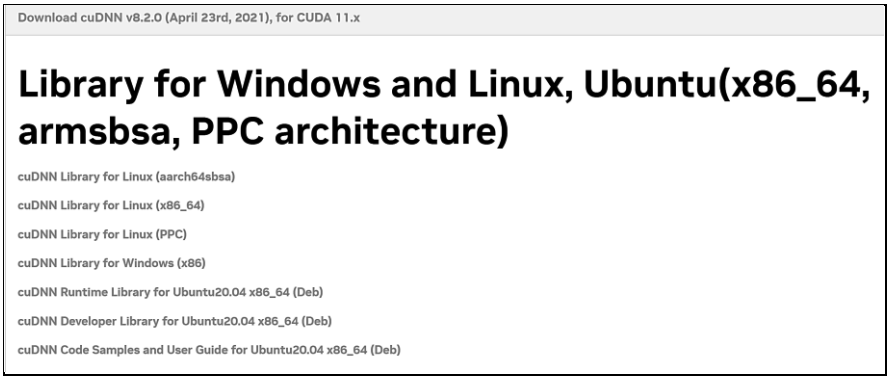

图 2-19　cuDNN 下载页面

下载的 cuDNN 是一个压缩文件，将其解压到 CUDA 安装目录，如图 2-20 所示。

（3）配置环境变量，这里需要将 CUDA 的运行路径加到环境变量 Path 的值中，如图 2-21 所示。如果 cuDNN 是使用.exe 文件安装的，那这个环境变量自动就配置好了，读者只要验证一下即可。

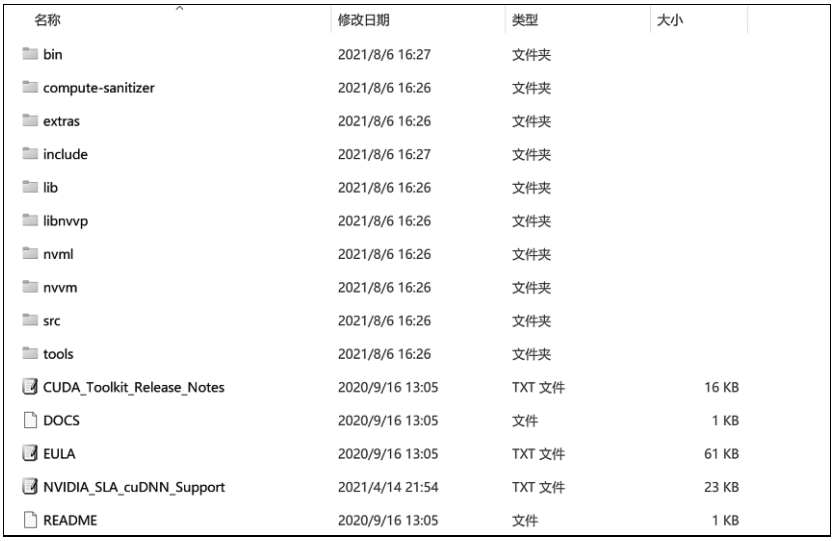

图 2-20　解压 cuDNN 文件

图 2-21　配置环境变量

（4）安装 PyTorch 及相关软件。从图 2-17 可以看到，对应 CUDA 11.7 的安装命令如下：

```
    conda install pytorch torchvision torchaudio pytorch-cuda=11.7 -c pytorch -c
nvidia
```

如果读者直接安装 Python，没有按 2.1.1 节安装 Miniconda，则 PyTorch 安装命令如下：

```
    pip3 install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu117
```

完成 PyTorch 2.0 GPU 版本的安装后，接下来验证一下 PyTorch 是否安装成功。

2.2.3　PyTorch 2.0 小练习：Hello PyTorch

打开 CMD 窗口依次输入如下命令可以验证安装是否成功，代码如下：

```
    import torch
    result = torch.tensor(1) + torch.tensor(2.0)
    result
```

结果如图 2-22 所示。

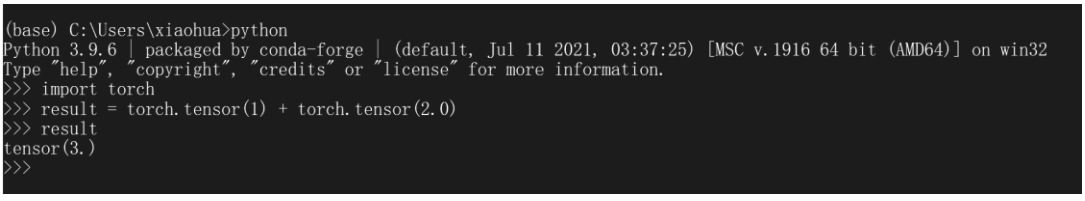

图 2-22　验证结果

或者打开前面安装的 PyCharm IDE，新建一个项目，再新建一个 hello_pytorch.py 文件，输入如下代码：

```
    import torch
    result = torch.tensor(1) + torch.tensor(2.0)
    print(result)
```

最终结果请读者自行验证。

2.3　实战：基于 PyTorch 2.0 的图像去噪

为了给读者提供一个使用 PyTorch 进行深度学习的总体印象，这里准备了一个实战案例，向读者演示进行深度学习任务所需要的整体流程，读者可能不熟悉这里的程序设计和编写，但是只要求了解每个过程需要做的内容以及涉及的步骤即可。

2.3.1　MNIST 数据集的准备

HelloWorld 是任何一门编程语言入门的基础程序，读者在开始学习编程时，打印的第一句话往

往就是 HelloWorld。在前面的章节中，我们也带领读者学习了 PyCharm 打印出来的第一个程序 HelloWorld。

在深度学习编程中也有其特有的 HelloWorld，其编程对象是一个图片数据集 MNIST，要求对数据集中的图片进行分类，因此难度比较大。

对于好奇的读者来说，一定有一个疑问：MNIST 究竟是什么？

实际上，MNIST 是一个手写数字的图片数据库，它有 60 000 个训练样本集和 10 000 个测试样本集。打开来看，MNIST 数据集如图 2-23 所示。

读者可直接使用本书源码库提供的 MNIST 数据集，文件在 dataset 文件夹中，如图 2-24 所示。

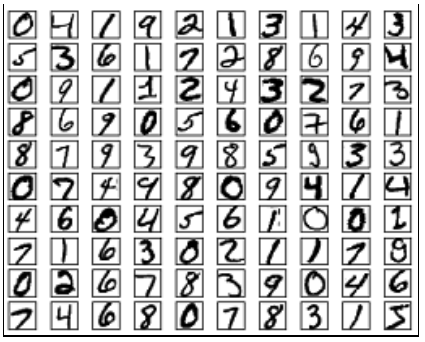

图 2-23　MNIST 数据集

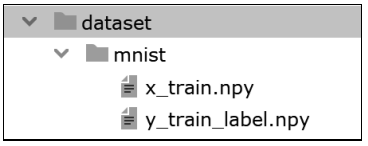

图 2-24　dataset 文件夹

之后使用 NumPy 工具库进行数据读取，代码如下：

```
import numpy as np
x_train = np.load("./dataset/mnist/x_train.npy")
y_train_label = np.load("./dataset/mnist/y_train_label.npy")
```

读者也可以在百度搜索 MNIST 的下载地址，直接下载 train-images-idx3-ubyte.gz、train-labels-idx1-ubyte.gz 等，如图 2-25 所示。

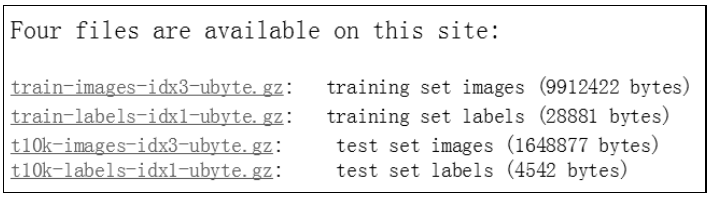

图 2-25　下载页面

下载 4 个文件，分别是训练图片集、训练标签集、测试图片集、测试标签集，这些文件都是压缩文件，解压后，可以发现这些文件并不是标准的图像格式，而是二进制文件，其中训练图片集的部分内容如图 2-26 所示。

MNIST 训练集内部的文件结构如图 2-27 所示。

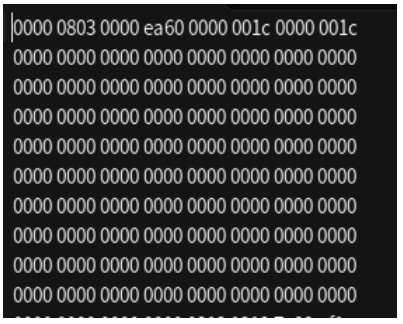

图 2-26　训练图片集的部分内容

TRAINING SET IMAGE FILE (train-images-idx3-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
....			
xxxx	unsigned byte	??	pixel

图 2-27　训练集内部的文件结构

训练集中有 60 000 个实例，也就是说这个文件包含 60 000 个标签内容，每一个标签的值为一个 0~9 的数。这里先解析文件中每一个属性的含义。首先，该数据是以二进制格式存储的，我们读取的时候要以 rb 方式读取；其次，真正的数据只有[value]这一项，其他[type]之类的项只是用来描述的，并不是真正放在数据文件里面的信息。

也就是说，在读取真实数据之前，要读取 4 个 32 位 integer。由[offset]可以看出，真正的 pixel 是从 0016 开始的，一个 int 32 位，所以在读取 pixel 之前要读取 4 个 32 位 integer，也就是 magic number、number of images、number of rows、number of columns。

继续对图片进行分析。在 MNIST 图片集中，所有的图片都是 28×28 的，也就是每幅图片都有 28×28 个像素。如图 2-28 所示，在 train-images-idx3-ubyte 文件中偏移量为 0 字节处，有一个 4 字节的数为 0000 0803，表示魔数。这里补充一下什么是魔数，其实它就是一个校验数，用来判断这个文件是不是 MNIST 里面的 train-images-idx3-ubyte 文件。

接下来是 0000 ea60，值为 60 000，代表容量；从第 8 字节开始有一个 4 字节数，值为 28，也就是 0000 001c，表示每幅图片的行数；从第 12 字节开始有一个 4 字节数，值也为 28，也就是 0000 001c，表示每幅图片的列数；从第 16 字节开始才是我们的像素值。

这里使用每 784 字节代表一幅图片。

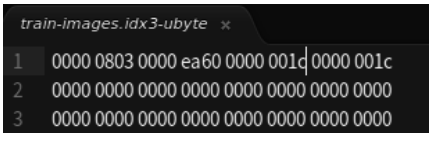

图 2-28　魔数

2.3.2　MNIST 数据集的特征和标签介绍

前面向读者介绍了两种不同的 MNIST 数据集的获取方式，在这里推荐使用本书配套源码中的 MNIST 数据集进行数据读取，代码如下：

```
import numpy as np
x_train = np.load("./dataset/mnist/x_train.npy")
y_train_label = np.load("./dataset/mnist/y_train_label.npy")
```

这里 numpy 函数会根据输入的地址对数据进行处理，并自动将其分解成训练集和验证集。打印训练集的维度如下：

```
(60000, 28, 28)
(60000,)
```

这是使用数据处理的第一个步骤，有兴趣的读者可以进一步完成数据的训练集和测试集的划分。

回到 MNIST 数据集，每个 MNIST 实例数据单元也是由两部分构成的，包括一幅包含手写数字的图片和一个与其对应的标签。可以将其中的标签特征设置成 y，而图片特征矩阵以 x 来代替，所有的训练集和测试集中都包含 x 和 y。

图 2-29 用更为一般化的形式解释了 MNIST 数据实例的展开形式。在这里，图片数据被展开成矩阵的形式，矩阵的大小为 28×28。至于如何处理这个矩阵，常用的方法是将其展开，而展开的方式和顺序并不重要，只需要将其按同样的方式展开即可。

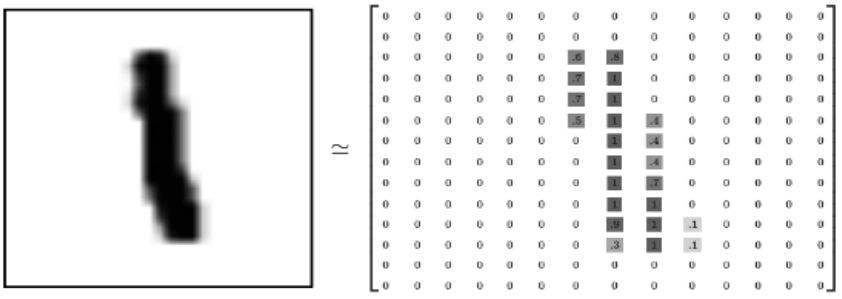

图 2-29　MNIST 数据实例的展开形式

下面回到对数据的读取，前面已经介绍了，MNIST 数据集实际上就是一个包含着 60 000 幅图片的 60000×28×28 大小的矩阵张量[60000,28,28]。

矩阵中行数指的是图片的索引，用以对图片进行提取。而后面的 28×28 个向量用以对图片特征进行标注。实际上，这些特征向量就是图片中的像素点，每幅手写图片的大小是[28,28]，每个像素转换为一个 0~1 的浮点数，构成矩阵，如图 2-30 所示。

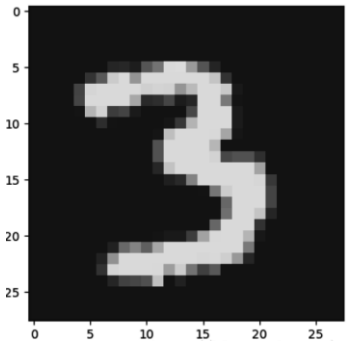

图 2-30　手写图片示例

2.3.3　模型的准备和介绍

对于使用 PyTorch 进行深度学习的项目来说，一个非常重要的内容是模型的设计，模型决定了深度学习在项目中采用哪种方式达到目标的主体设计。在本例中，我们的目的是输入一个图像之后

对其进行去噪处理。

对于模型的选择，一个非常简单的思路是，图像输出的大小应该就是输入的大小，在这里我们选择使用 Unet 作为设计的主要模型。

注意：对于模型的选择，读者现在不需要考虑，随着对本书学习的深入，见识到更多处理问题的手段后，对模型的选择自然会心领神会。

我们可以整体看一下 Unet 的结构（读者目前只需要知道 Unet 输入和输出大小是同样的维度即可），如图 2-31 所示。

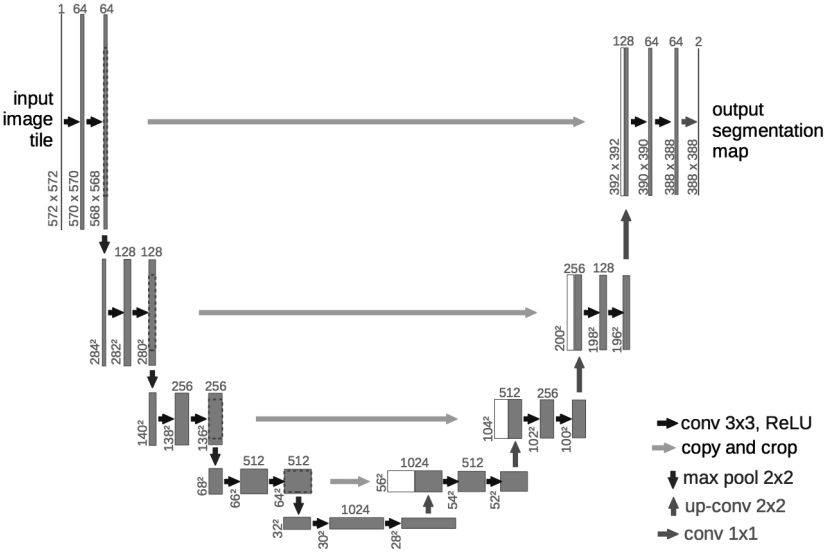

图 2-31　Unet 的结构

可以看到对于整体模型架构来说，其通过若干个“模块”（block）与“直连”（residual）进行数据处理。这部分内容我们在后面的章节中会讲到，目前读者只需要知道模型有这种结构即可。Unet 的模型整体代码如下：

```
    import torch
    import einops.layers.torch as elt
    class Unet(torch.nn.Module):
        def __init__(self):
            super(Unet, self).__init__()
            #模块化结构，这也是后面常用到的模型结构
            self.first_block_down = torch.nn.Sequential(
                torch.nn.Conv2d(in_channels=1,out_channels=32,kernel_size=3,
padding=1),torch.nn.GELU(),
                torch.nn.MaxPool2d(kernel_size=2,stride=2)
            )
            self.second_block_down = torch.nn.Sequential(
                torch.nn.Conv2d(in_channels=32,out_channels=64,kernel_size=3,
padding=1),torch.nn.GELU(),
```

```
            torch.nn.MaxPool2d(kernel_size=2,stride=2)
        )
        self.latent_space_block = torch.nn.Sequential(
            torch.nn.Conv2d(in_channels=64,out_channels=128,kernel_size=3,
padding=1),torch.nn.GELU(),
        )
        self.second_block_up = torch.nn.Sequential(
            torch.nn.Upsample(scale_factor=2),
            torch.nn.Conv2d(in_channels=128, out_channels=64, kernel_size=3,
padding=1), torch.nn.GELU(),
        )
        self.first_block_up = torch.nn.Sequential(
            torch.nn.Upsample(scale_factor=2),
            torch.nn.Conv2d(in_channels=64, out_channels=32, kernel_size=3,
padding=1), torch.nn.GELU(),
        )
        self.convUP_end = torch.nn.Sequential(
            torch.nn.Conv2d(in_channels=32,out_channels=1,kernel_size=3,
padding=1), torch.nn.Tanh()
        )
    def forward(self,img_tensor):
        image = img_tensor
        image = self.first_block_down(image)
        #print(image.shape)
        #torch.Size([5, 32, 14, 14])
        image = self.second_block_down(image)
        #print(image.shape)
        #torch.Size([5, 16, 7, 7])
        image = self.latent_space_block(image)
        #print(image.shape)
        #torch.Size([5, 8, 7, 7])
        image = self.second_block_up(image)
        #print(image.shape)
        #torch.Size([5, 16, 14, 14])
        image = self.first_block_up(image)
        #print(image.shape)
        #torch.Size([5, 32, 28, 28])
        image = self.convUP_end(image)
        #print(image.shape)
        #torch.Size([5, 32, 28, 28])
        return image

if __name__ == '__main__': #main 是 Python 进行单文件测试的技巧，请读者记住这种写法
    image = torch.randn(size=(5,1,28,28))
    Unet()(image)
```

在这里通过一个 main 架构标识了可以在单个文件中对文件进行测试，请读者记住这种写法。

2.3.4　模型的损失函数与优化函数

除了深度学习模型外，完成一个深度学习项目设定模型的损失函数与优化函数也很重要。这两部分内容对于初学者来说可能并不是很熟悉，在这里读者只需要知道有这部分内容即可。

（1）对损失函数的选择，在这里选用 MSELoss 作为损失函数，MSELoss 损失函数的中文名字为均方损失函数（Mean Squared Error Loss）。

MSELoss 的作用是计算预测值和真实值之间的欧式距离。预测值和真实值越接近，两者的均方差就越小，均方差函数常用于线性回归模型的计算。在 PyTorch 中使用 MSELoss 的代码如下：

```
loss = torch.nn.MSELoss(reduction="sum")(pred, y_batch)
```

（2）优化函数的设定，在这里我们采用 Adam 优化器，对于 Adam 优化函数，请读者自行学习，在这里只提供使用 Adam 优化器的代码，如下所示：

```
optimizer = torch.optim.Adam(model.parameters(), lr=2e-5)
```

2.3.5　基于深度学习的模型训练

在介绍了深度学习的数据准备、模型以及损失函数和优化函数后，下面使用 PyTorch 训练一个可以实现去噪性能的深度学习整理模型，完整代码如下（本代码参看配套资源的第 2 章，读者可以直接在 PyCharm 中打开文件运行并查看结果）：

```
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '0' #指定 GPU 编码
import torch
import numpy as np
import unet
import matplotlib.pyplot as plt
from tqdm import tqdm
batch_size = 320                          #设定每次训练的批次数
epochs = 1024                             #设定训练次数
#device = "cpu"  #PyTorch 的特性，需要指定计算的硬件，如果没有 GPU，就使用 CPU 进行计算
device = "cuda"  #在这里默认使用 GPU，如果读者运行出现问题，可以将其改成 CPU 模式
model = unet.Unet()                       #导入 Unet 模型
model = model.to(device)                  #将计算模型传入 GPU 硬件等待计算
model = torch.compile(model)              #PyTorch 2.0 的特性，加速计算速度
optimizer = torch.optim.Adam(model.parameters(), lr=2e-5)   #设定优化函数
#载入数据
x_train = np.load("../dataset/mnist/x_train.npy")
y_train_label = np.load("../dataset/mnist/y_train_label.npy")
x_train_batch = []
for i in range(len(y_train_label)):
    if y_train_label[i] < 2:          #为了加速演示，这里只运行数据集中小于 2 的数字，
```

```
也就是 0 和 1，读者可以自行增加训练个数
            x_train_batch.append(x_train[i])

    x_train = np.reshape(x_train_batch, [-1, 1, 28, 28])   #修正数据输入维度：
([30596, 28, 28])
    x_train /= 512.
    train_length = len(x_train) * 20                       #增加数据的单词循环次数
    for epoch in range(epochs):
        train_num = train_length // batch_size             #计算有多少批次数
        train_loss = 0                                     #用于损失函数的统计
        for i in tqdm(range(train_num)):                   #开始循环训练
            x_imgs_batch = []                              #创建数据的临时存储位置
            x_step_batch = []
            y_batch = []
            # 对每个批次内的数据进行处理
            for b in range(batch_size):
                img = x_train[np.random.randint(x_train.shape[0])]#提取单幅图片内容
                x = img
                y = img
                x_imgs_batch.append(x)
                y_batch.append(y)
            #将批次数据转换为 PyTorch 对应的 tensor 格式并将其传入 GPU 中
            x_imgs_batch = torch.tensor(x_imgs_batch).float().to(device)
            y_batch = torch.tensor(y_batch).float().to(device)
            pred = model(x_imgs_batch)                     #对模型进行正向计算
            loss = torch.nn.MSELoss(reduction=True)(pred, y_batch)/batch_size   #
使用损失函数进行计算
            #这里读者记住下面就是固定格式，一般这样使用即可
            optimizer.zero_grad()                  #对结果进行优化计算
            loss.backward()                        #损失值的反向传播
            optimizer.step()                       #对参数进行更新
            train_loss += loss.item()              #记录每个批次的损失值
        #计算并打印损失值
        train_loss /= train_num
        print("train_loss:", train_loss)
        #下面对数据进行打印
        image = x_train[np.random.randint(x_train.shape[0])]#随机挑选一条数据计算
        image = np.reshape(image,[1,1,28,28])          #修正数据维度
        image = torch.tensor(image).float().to(device) #挑选的数据传入硬件中等待计算
        image = model(image)                           #使用模型对数据进行计算
        image = torch.reshape(image, shape=[28,28])    #修正模型输出结果
        image = image.detach().cpu().numpy()  #将计算结果导入 CPU 中进行后续计算或者展示
        #展示或存储数据结果
        plt.imshow(image)
        plt.savefig(f"./img/img_{epoch}.jpg")
```

在代码中展示了完整的模型训练过程。首先传入数据，然后使用模型对数据进行计算，计算结果与真实值的误差被回传到模型中，之后 PyTorch 框架根据回传的误差对整体模型参数进行修正。训练结果如图 2-32 所示。

图 2-32　训练结果

从图 2-32 中可以很清楚地看到，随着训练过程的进行，模型逐渐能够学会对输入的数据进行整形和输出，此时模型的输出结果表示已经能够很好地对输入的图形细节进行修正，读者可以自行完成这部分代码的运行。

2.4　本 章 小 结

本章是PyTorch实战程序设计的开始。本章介绍了PyTorch程序设计的环境与相关软件的安装，并演示了第一个基于 PyTorch 的程序的整体设计过程，以及部分 PyTorch 组件的使用。

实际上可以看到，深度学习程序设计就是由一个个小组件组合来完成的，本书的后续章节将会针对每个组件进行深入讲解。