人工智能技术丛书

谷歌JAX

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

JAX是一个用于高性能数值计算的Python库，专门为深度学习领域的高性能计算而设计。本书详解JAX框架深度学习的相关知识，配套示例源码、PPT课件、数据集和开发环境。

本书共分为13章，内容包括JAX从零开始，一学就会的线性回归、多层感知机与自动微分器，深度学习的理论基础，XLA与JAX一般特性，JAX的高级特性，JAX的一些细节，JAX中的卷积，JAX与TensorFlow的比较与交互，遵循JAX函数基本规则下的自定义函数，JAX中的高级包。最后给出3个实战案例：使用ResNet完成CIFAR100数据集分类，有趣的词嵌入，生成对抗网络（GAN）。

本书适合JAX框架初学者、深度学习初学者以及深度学习从业人员，也适合作为高等院校和培训机构人工智能相关专业的师生教学参考书。

图书在版编目（CIP）数据

谷歌JAX深度学习从零开始学/王晓华著. —北京：清华大学出版社，2022.4
（人工智能技术丛书）
ISBN 978-7-302-60436-5

Ⅰ. ①谷… Ⅱ. ①王… Ⅲ. ①软件工具－程序设计②机器学习 Ⅳ. ①TP311.561②TP181

中国版本图书馆CIP数据核字（2022）第052837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：朱雨萌

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：大厂回族自治县彩虹印刷有限公司
经　　销：全国新华书店
开　　本：190mm×260mm　　**印　　张：**16　　**字　　数：**435千字
版　　次：2022年6月第1版　　**印　　次：**2022年6月第1次印刷
定　　价：79.00元

产品编号：096189-01

前　言

深度学习和人工智能引领了一个新的研究和发展方向，同时正在改变人类固有的处理问题的思维。现在各个领域都处于运用深度学习技术进行重大技术突破的阶段，与此同时，深度学习本身也展现出巨大的发展空间。

JAX 是一个用于高性能数值计算的 Python 库，专门为深度学习领域的高性能计算而设计，其包含丰富的数值计算与科学计算函数，能够很好地满足用户的计算需求，特别是其基于 GPU 或者其他硬件加速器的能力，能够帮助我们在现有的条件下极大地加速深度学习模型的训练与预测。

JAX 继承了 Python 简单易用的优点，给使用者提供了一个“便于入门，能够提高”的深度学习实现方案。JAX 在代码结构上采用面向对象方法编写，完全模块化，并具有可扩展性，其运行机制和说明文档都将用户体验和使用难度纳入考虑范围，降低了复杂算法的实现难度。JAX 的计算核心使用的是自动微分，可以支持自动模式反向传播和正向传播，且二者可以任意组合成任何顺序。

本书由浅到深地向读者介绍 JAX 框架相关的知识，重要内容均结合代码进行实战讲解，读者通过这些实例可以深入掌握 JAX 程序设计的内容，并能对深度学习有进一步的了解。

本书特色

版本新，易入门

本书详细介绍 JAX 最新版本的安装和使用，包括 CPU 版本以及 GPU 版本。

作者经验丰富，代码编写细腻

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。

理论扎实，深入浅出

在代码设计的基础上，本书深入浅出地介绍深度学习需要掌握的一些基本理论知识，并通过大量的公式与图示对理论做介绍。

对比多种应用方案，实战案例丰富

本书给出了大量的实例，同时提供多个实现同类功能的解决方案，覆盖使用 JAX 进行深度学习开发中常用的知识。

本书内容

第 1 章　JAX 从零开始

本章介绍 JAX 应用于深度学习的基本理念、基础，并通过一个真实的深度学习例子向读者展示深度学习的一般训练步骤。本章是全书的基础，读者需要先根据本章内容搭建 JAX 开

发环境，并下载合适的 IDE。

第 2 章　一学就会的线性回归、多层感知机与自动微分器

本章以深度学习中最常见的线性回归和多层感知机的程序设计为基础，循序渐进地介绍 JAX 进行深度学习程序设计的基本方法和步骤。

第 3 章　深度学习的理论基础

本章主要介绍深度学习的理论基础，从 BP 神经网络开始，介绍神经网络两个基础算法，并着重介绍反向传播算法的完整过程和理论，最后通过编写基本 Python 的方式实现一个完整的反馈神经网络。

第 4 章　XLA 与 JAX 一般特性

本章主要介绍 JAX 的一些基础特性，例如 XLA、自动微分等。读者需要了解的是 XLA 是如何工作的，它能给 JAX 带来什么。

第 5 章　JAX 的高级特性

本章是基于上一章的基础比较 JAX 与 NumPy，重点解释 JAX 在实践中的一些程序设计和编写的规范要求，并对其中的循环函数做一个详尽而细致的说明。

第 6 章　JAX 的一些细节

本章主要介绍 JAX 在设计性能较优的程序时的细节问题，并介绍 JAX 内部一整套结构体保存方法和对模型参数的控制，这些都是为我们能编写出更为强大的深度学习代码打下基础。

第 7 章　JAX 中的卷积

卷积可以说是深度学习中使用最为广泛的计算部件，本章主要介绍卷积的基础知识以及相关用法，并通过一个经典的卷积神经网络 VGG 模型，讲解卷积的应用和 JAX 程序设计的一些基本内容。

第 8 章　JAX 与 TensorFlow 的比较与交互

本章主要介绍在一些需要的情况下使用已有的 TensorFlow 组件的一些方法。作为深度学习经典框架，TensorFlow 有很多值得 JAX 参考和利用的内容。

第 9 章　遵循 JAX 函数基本规则下的自定义函数

本章介绍 JAX 创建自定义函数的基本规则，并对其中涉及的一些细节问题进行详细讲解。期望读者在了解和掌握如何利用和遵循这些基本规则后去创建既满足需求又能够符合 JAX 函数规则的自定义函数。

第 10 章　JAX 中的高级包

本章详细介绍 JAX 中高级程序设计子包，特别是 2 个非常重要的模块 jax.experimental 和 jax.nn。这两个包目前仍处于测试阶段，但是包含了建立深度学习模型所必需的一些基本函数。

第 11 章　JAX 实战——使用 ResNet 完成 CIFAR100 数据集分类

本章主要介绍在神经网络领域具有里程碑意义的模型——ResNet。它改变了人们仅仅依

靠堆积神经网络层来获取更高性能的做法，在一定程度上解决了梯度消失和梯度爆炸的问题。这是一项跨时代的发明。本章以手把手的方式向读者介绍 ResNet 模型的编写和架构方法。

第 12 章　JAX 实战——有趣的词嵌入

本章介绍 JAX 于自然语言处理的应用，通过一个完整的例子向读者介绍自然语言处理所需要的全部内容，一步步地教会读者使用不同的架构和维度进行文本分类的方法。

第 13 章　JAX 实战——生成对抗网络（GAN）

本章介绍使用 JAX 完成生成对抗网络模型的设计，讲解如何利用 JAX 完成更为复杂的深度学习模型设计，掌握 JAX 程序设计的技巧。同时也期望通过本章能够帮助读者全面复习本书所涉及的 JAX 的深度学习程序设计内容。

源码下载与技术支持

本书配套源码、PPT 课件、数据集、开发环境、配图文件和答疑服务，需要使用微信扫描右边二维码下载，可按页面提示，把链接转发到自己的邮箱中下载。如果下载有问题或者阅读中发现问题，请联系 booksaga@163.com，邮件主题为“谷歌 JAX 深度学习从零开始学”。

本书读者

- 人工智能入门读者
- 深度学习入门读者
- 机器学习入门读者
- 高等院校人工智能专业的师生
- 专业培训机构的师生
- 其他对智能化、自动化感兴趣的开发者

技术支持、勘误和鸣谢

由于作者的水平有限，加上 JAX 框架的演进较快，书中难免存在疏漏之处，恳请读者来信批评指正。本书的顺利出版，首先要感谢家人的理解和支持，他们给予我莫大的动力，让我的努力更加有意义。此外特别感谢出版社的编辑们，感谢他们在本书编写过程中给予的无私指导。

编　者

2022 年 4 月

深度学习图书推荐

《深度学习案例精粹：基于TensorFlow与Keras》

本书由13个深度学习案例组成，所有案例都基于Python+TensorFlow 2.5+Keras技术，可用于深度学习的实战训练，拓宽解决实际问题的思路和方法。

《TensorFlow知识图谱实战》

大数据时代的到来，为人工智能的飞速发展带来前所未有的数据红利。在大数据背景下，大量知识不断涌现，如何有效地发掘这些知识呢？知识图谱横空出世。本书教会读者使用TensorFlow 2深度学习构建知识图谱，引导读者掌握知识图谱的构建理论和方法。

《TensorFlow人脸识别实战》

深度学习方法的主要优势是可以用非常大型的数据集进行训练，学习到表征这些数据的最佳特征，从而在要求的准确度下实现人脸识别的目标。本书教会读者如何运用TensorFlow 2深度学习框架实现人脸识别。

《TensorFlow语音识别实战》

本书使用TensorFlow 2作为语音识别的基本框架，引导读者入门并掌握基于深度学习的语音识别基本理论、概念以及实现实际项目。全书内容循序渐进，从搭建环境开始，逐步深入理论、代码及应用实践，是学习语音识别技术的首选。

《TensorFlow 2.0深度学习从零开始学》

本书系统讲解TensorFlow 2.0的新框架设计思想和模型的编写，详细介绍TensorFlow 2.0的安装、使用以及Keras编程方法与技巧，剖析卷积神经网络原理及其实战应用。

《深度学习的数学原理与实现》

本书主要讲解深度学习中的数学知识、算法原理和实现方法。内容包括深度学习概述、梯度下降算法、卷积函数、损失函数、线性回归和逻辑回归、时间序列模型和生成对抗网络、TensorFlow框架、推荐算法、标准化正则化和初始化、人脸识别案例、词嵌入向量案例。

《Python深度学习从零开始学》

本书立足实践，以通俗易懂的方式详细介绍深度学习的基础理论以及相关的必要知识，同时以实际动手操作的方式来引导读者入门人工智能深度学习。本书的读者只需具备Python语言基础知识，不需要有数学基础或者AI基础，按照本书的步骤循序渐进地学习，即可快速上手深度学习。

目　录

第 1 章 JAX 从零开始

近年来，人工智能（Artificial Intelligence，AI）在科研领域取得了巨大的成功，影响到了人们生活的方方面面，而其中基于神经网络的深度学习（Deep Learning，DL）发展尤为迅速。深度学习将现实世界中的每一个概念都由表现更为抽象的概念来定义，即通过神经网络来提取样本特征。

深度学习通过学习样本数据的内在规律和表示层次，从而使得人工智能能够像人一样具备分析能力，可以自动识别文字、图像和声音等数据，以便帮助人工智能项目更接近于人的认知形式。

深度学习是一个复杂的机器学习算法，目前在搜索技术、数据挖掘、机器学习、机器翻译、自然语言处理、多媒体学习、语音、推荐和个性化技术，以及其他相关领域都取得了令人瞩目的成就。深度学习解决了很多复杂的模式识别难题，使得人工智能相关技术取得了重大进步。

1.1 JAX 来了

“工欲善其事，必先利其器”。人工智能或者其核心理论深度学习也一样。任何一个好的成果落地并在将来发挥其巨大作用，都需要一个能够将其落地并应用的基本框架工具。JAX 是机器学习框架领域的新生力量，它具有更快的高阶微分计算方法，可以采用先编译后执行的模式，突破了已有深度学习框架的局限性，同时具有更好的硬件支持，甚至将来可能会成为谷歌的主要科学计算深度学习库。

1.1.1 JAX 是什么

JAX 官方文档的解释是：“JAX 是 CPU、GPU 和 TPU 上的 NumPy，具有出色的自动差异化功能，可用于高性能机器学习研究。”

就像 JAX 官方文档解释的那样，最简单的 JAX 是加速器支持的 NumPy，它具有一些便利的功能，可用于常见的机器学习操作。

更具体地说，JAX 的前身是 Autograd，也就是 Autograd 的升级版本。JAX 可以对 Python 和 NumPy 程序进行自动微分，可通过 Python 的大量特征子集进行区分，包括循环、分支、

递归和闭包语句进行自动求导，也可以求三阶导数（三阶导数是由原函数导数的导数的导数，即将原函数进行三次求导）。通过 grad，JAX 完全支持反向模式和正向模式的求导，而且这两种模式可以任意组合成任何顺序，具有一定灵活性。

开发 JAX 的出发点是什么？说到这，就不得不提 NumPy。NumPy 是 Python 中的一个基础数值运算库，被广泛使用，但是 NumPy 不支持 GPU 或其他硬件加速器，也没有对反向传播的内置支持。此外，Python 本身的速度限制了 NumPy 使用，所以少有研究者在生产环境下直接用 NumPy 训练或部署深度学习模型。

在此情况下，出现了众多的深度学习框架，如 PyTorch、TensorFlow 等。但是 NumPy 具有灵活、调试方便、API 稳定等独特的优势，而 JAX 的主要出发点就是将 NumPy 的优势与硬件加速相结合。

目前，基于 JAX 已有很多优秀的开源项目，如谷歌的神经网络库团队开发了 Haiku，这是一个面向 JAX 的深度学习代码库，通过 Haiku，用户可以在 JAX 上进行面向对象开发；又比如 RLax，这是一个基于 JAX 的强化学习库，用户使用 RLax 就能进行 Q-learning 模型的搭建和训练；此外还包括基于 JAX 的深度学习库 JAXnet，该库一行代码就能定义计算图，可进行 GPU 加速。可以说，在过去几年中，JAX 掀起了深度学习研究的风暴，推动了其相关科学研究的迅速发展。

1.1.2 为什么是 JAX

JAX 是机器学习框架领域的新生力量。JAX 从诞生就具有相对于其他深度学习框架更高的高度，并迈出了重要的一步，不是因为它比现有的机器学习框架具有更简洁的 API，或者因为它比 TensorFlow 和 PyTorch 在被设计的事情上做得更好，而是因为它允许我们更容易地尝试更广阔的思想空间。

JAX 把看不到的细节藏在底层内部结构中，而无须关心其使用过程和细节，很明显，JAX 关心的是如何让开发者做出创造性的工作。JAX 对如何使用做了很少的假设，具有很好的灵活性。

JAX 目前已经达到深度学习框架的最高水平。在当前开源的框架中，没有哪一个框架能够在简洁、易用、速度这 3 个方面有两个能同时超过 JAX。

- 简洁：JAX 的设计追求最少的封装，尽量避免重复造轮子。不像 TensorFlow 中充斥着 graph、operation、name_scope、variable、tensor、layer 等全新的概念，JAX 的设计遵循 tensor → variable(autograd) → Module 3 个由低到高的抽象层次，分别代表高维数组（张量）、自动求导（变量）和神经网络（层 / 模块），而且这 3 个抽象之间联系紧密，可以同时进行修改和操作。简洁的设计带来的另外一个好处就是代码易于理解。JAX 的源码只有 TensorFlow 的十分之一左右，更少的抽象、更直观的设计使得 JAX 的源码十分易于阅读。
- 速度：JAX 的灵活性不以速度为代价，在许多评测中，JAX 的速度表现胜过 TensorFlow 和 PyTorch 等框架。框架的运行速度和程序员的编码水平存在着一定关系，

但同样的算法，使用 JAX 实现可能快过用其他框架实现。

- 易用：JAX 是所有的框架中面向对象设计得最优雅的一个。JAX 的设计最符合人们的思维，它让用户尽可能地专注于实现自己的想法，即所思即所得，不需要考虑太多关于框架本身的束缚。

JAX 的设计体现了 Linux 设计哲学——do one thing and do it well。JAX 很轻量级，专注于高效的数值计算，由于提供了调用其他框架的功能，这样 JAX 程序的编写以及数据的加载可以使用其他框架的现成工具。并且 Google 也在基于 JAX 构建生态：包括神经网络库 Haiku、梯度处理和优化的库 Optax、强化学习库 Rlax，以及用来帮助编写可靠代码的 chex 工具库。很多 Google 的研究组利用 JAX 来开发训练神经网络的工具库，比如 Flax、Trax。

1.2 JAX 的安装与使用

JAX 是一个最新的深度学习框架，但是当读者开始使用 JAX 的时候就会发现其非常简单。本节将带领读者安装 JAX 的开发环境，并演示第 1 个 JAX 的实战程序——MNIST 数据集识别。

目前，JAX 开发环境必须搭建在 Linux 系统上，而对于大多数读者来说，安装和使用 Linux 是一个较为困难的操作，因此这里采用 Windows 10 或者更高版本自带的虚拟机形式安装 JAX 开发环境。不用担心，只需要安装一遍并跟随笔者手把手地配置好开发环境和工具，之后就可以像开发传统的 Windows 程序一样进行 JAX 程序的开发。

1.2.1 Windows Subsystem for Linux 的安装

在 Windows 10 操作系统中，借助 Windows Subsystem for Linux（简称 WSL）的功能，后续的 Windows 操作系统都支持和兼容 Linux 程序。而 Windows Subsystem for Linux 是一个为在 Windows 10 上能够原生运行 Linux 二进制可执行文件（ELF 格式）的兼容层。它由微软与 Canonical 公司合作开发，目标是使纯正的 Ubuntu Trusty Tahr 映像能下载和解压到用户的本地计算机，并且映像内的实用工具能在此子系统上原生运行。

当然，我们可以简单地认为就是在 Windows 环境上安装了一个 Linux 虚拟机环境。

1．第一步：启用 Linux 子系统

（1）在开始菜单中选择“设置”命令打开“Windows 设置”窗口，搜索“启用或关闭 Windows 功能”，如图 1.1 所示。

（2）搜索出“启用或关闭 Windows 功能”选项，单击打开“Windows 功能”对话框，如图 1.2 所示。勾选“虚拟机平台”和“适用于 Linux 的 Windows 子系统”选项即可，其他选项为默认。

（3）单击“确定”按钮之后，等待更改完成并重启计算机，如图 1.3 所示。

图 1.1 “Windows 设置”窗口

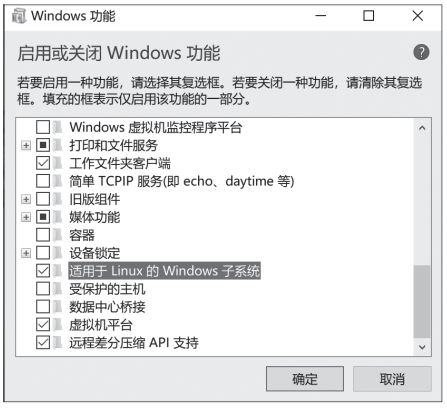

图 1.2 “Windows 功能”对话框

图 1.3 重启计算机

2. 第二步：启用开发者模式

在“Windows 设置”中搜索“使用开发人员功能”，打开“开发者选项”窗口，将“开发人员模式”下的开关打开，如图 1.4 所示。然后单击“是”按钮。

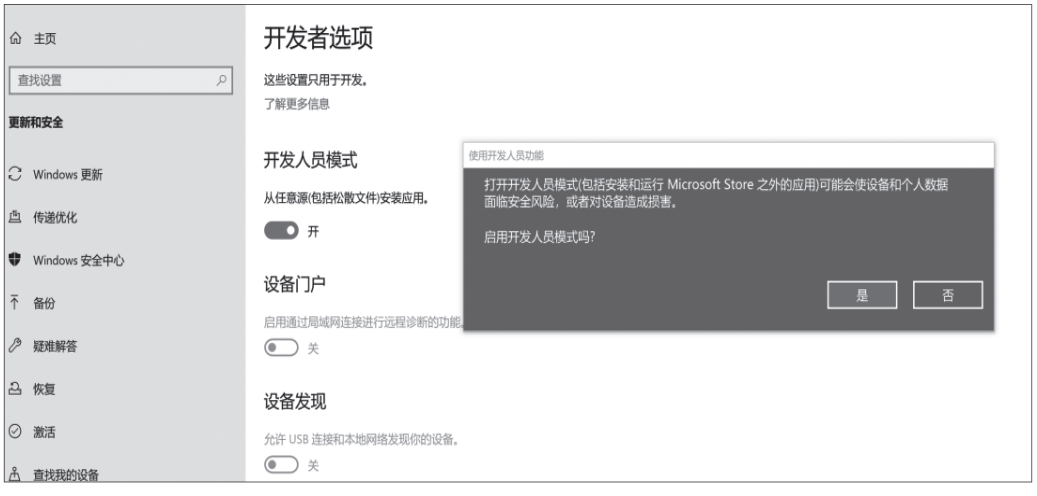

图 1.4 打开“开发人员模式”

在安装 Ubuntu 之前，需要手动设置 WSL 的版本，这里建议在 Windows 终端中以管理员身份运行如下命令：

```
wsl.exe --update
```

等待升级结束后运行如下命令：

```
wsl --set-default-version 2
```

可以通过如下命令查看 WSL 的版本号：

```
wsl --list --verbose
```

显示如图 1.5 所示的结果即可完成这一步的工作。

图 1.5 WSL 的版本号

3. 第三步：从“Microsoft Store”中安装 Ubuntu

打开 Microsoft Store 页面，搜索 Ubuntu，在搜索的结果中选择安装 Ubuntu 20.04 版本的 Linux 虚拟机，如图 1.6 所示。单击“获取”按钮即开始安装，如图 1.7 所示。

图 1.6 选择 Ubuntu 20.04 版本的虚拟机

图 1.7 安装 Ubuntu 20.04

4．第四步：启动 WSL 虚拟机

安装完成后启动 WSL 虚拟机，第一次启动时可能时间稍长，根据不同的计算机配置需要花费若干分钟，请耐心等待一下，如图 1.8 所示。

图 1.8 第一次启动 WSL

以后启动 WSL 可以像启动普通计算机程序一样，在 Windows 开始菜单的所有应用窗口中查找并点击对应的图标即可。。

5．第五步：配置 Ubuntu 虚拟机

图 1.9 所示就是 Ubuntu 的配置界面，首先输入用户名和密码，这里需要注意的是，相对于 Windows 系统，Ubuntu 系统在输入密码时是不会有字符显示的。当出现此界面时，即可认为用户设置成功，另外，根据笔者的学习经验，配置一个最简单的密码是较为方便的选择。

```
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: xiaohua
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 4.4.0-19041-Microsoft x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Sep 14 10:53:18 CST 2021

  System load:    0.52      Processes:             7
  Usage of /home: unknown   Users logged in:       0
  Memory usage:   7%        IPv4 address for eth0: 192.168.1.94
  Swap usage:     0%        IPv4 address for eth1: 172.21.112.1

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable
```

图 1.9 Ubuntu 的配置界面

对于 WSL 的安装，读者需要知道几个小知识：

（1）如果想在 Linux 中查看其他分区，WSL 将其他盘符挂载在“/mnt/”下。

举例说明（下面的语句都是在 WSL 终端中操作输入）：

① 复制 Ubuntu 上 sources.list 到 Windows 上进行修改，可以在终端中输入如下命令：

```
sudo cp /etc/apt/sources.list /mnt/d/sources.list
```

其中 WSL 会把 Windows 上的磁盘挂载到“/mnt/”下，所以 Windows 的 D 盘根目录在 Ubuntu 上的路径为“/mnt/d/”。

② 用 Windows 上修改后的 sources.list 覆盖 Ubuntu 上的 sources.list：

```
sudo mv /mnt/d/sources.list /etc/apt/sources.list
```

这样就可以做到在 Windows 计算机与 WSL 之间互相查看文件。

（2）如果想在 Windows 下查看 WSL 文件位置，可以查看“C:\Users\ 用户名 \AppData\Local\Packages\”文件夹中以“CanonicalGroupLimited.Ubuntu20.04onWindows”为开头的文件夹，而其中的“\LocalState\rootfs”就是对应的 WSL 文件目录。

1.2.2 JAX 的安装和验证

新安装的 WSL 需要更新一次，打开 WSL 终端界面，依次输入如下操作语句：

```
sudo apt update
sudo apt install gcc make g++
sudo apt install build-essential
sudo apt install python3-pip
pip install --upgrade pip
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple jax== 0.2.19
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple jaxlib== 0.1.70
```

因为 JAX 现在仍旧处于调整阶段，可能后面函数会有调整。本书使用的是 0.2.19 版本的 jax 和 0.1.70 版本的 jaxlib，读者一定要注意版本的选择。

在需要输入密码的地方直接输入，并且在需要确认的地方输入字符“y”进行确认。

等全部命令运行完毕后，用户可以打开 WSL 终端运行如下命令：

```
python3
```

这是启动 WSL 自带的 Python 命令，之后键入如下命令：

```
import jax.numpy as np
np.add(1.0,1.7)
```

最终结果如图 1.10 所示。还可以看到 Ubuntu 系统上默认安装了 Python 3.8.10。

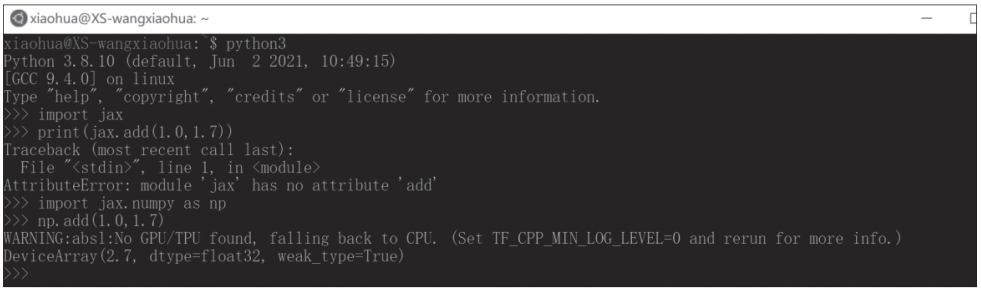

图 1.10 运行结果

可以看到最终结果是 2.7，并且也提示了本机在运行中只使用 CPU 而非 GPU。对于想使用 GPU 版本的 JAX 读者来说，最好的方案是使用纯 Ubuntu 系统作为开发平台，或者可以升

级到 Windows 11 并安装特定的 CUDA 驱动程序，这里不再过多阐述，有兴趣的读者可以参考本书附录。

1.2.3 PyCharm 的下载与安装

上一节笔者做过演示，Python 程序可以在 WSL 控制终端中编写。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。

笔者的做法是，在 Windows 中安装 PyCharm，然后使用 WSL 中的 Ubuntu 环境进行编译。

（1）进入 PyCharm 官网的 Download 页面后可以选择不同的版本，如图 1.11 所示。有收费的专业版和免费的社区版，这里建议读者选择免费的社区版即可。

（2）双击运行后进入安装界面，如图 1.12 所示。直接单击“Next”按钮，采用默认安装即可。

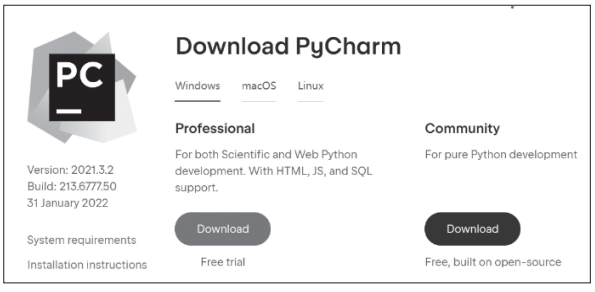

图 1.11 PyCharm 的下载页面

图 1.12 PyCharm 的安装界面

（3）如图 1.13 所示，在安装 PyCharm 的过程中需要对安装的位数进行选择，这里建议选择与已安装的 Python 相同位数的文件。

（4）安装完成后出现“Finish”按钮，单击该按钮即可完成安装，如图 1.14 所示。

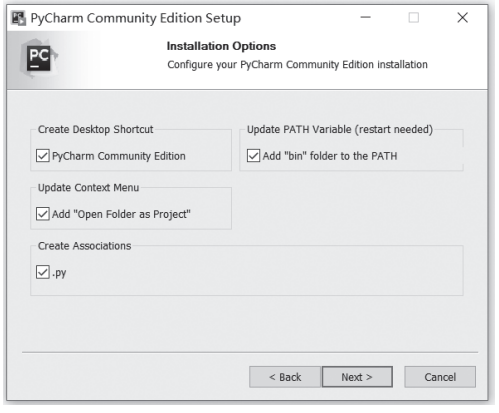

图 1.13 PyCharm 的配置选择（按个人真实情况选择）

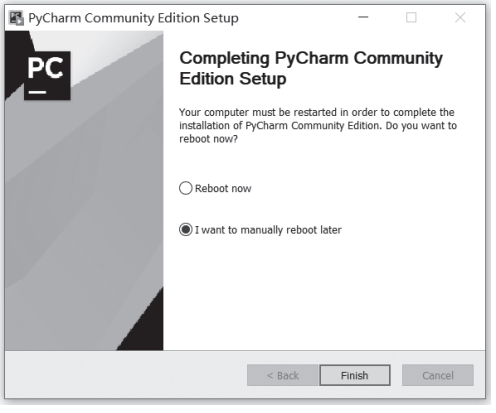

图 1.14 PyCharm 安装完成

1.2.4 使用 PyCharm 和 JAX

下面使用 PyCharm 进行程序设计，在进行程序设计之前，需要在 PyCharm 中加载 JAX 编译环境，编译步骤说明如下。

步骤 01 在桌面新建一个空文件夹，命名为“JaxDemo”。启动 PyCharm 并打开刚才新建的空文件夹“JaxDemo”，结果如图 1.15 所示。

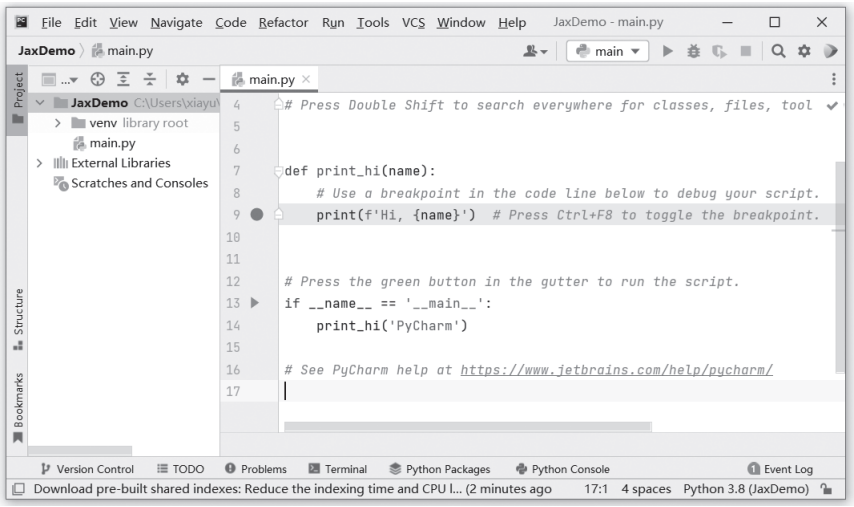

图 1.15 启动 PyCharm

此时并没有使用 WSL 中的 Python 进行解释，请继续进行下一步工作。

步骤 02 在 Windows 环境下启动 PyCharm，依次选择“Setting|Project|ProjectInterpreter|Python Interpreter”选项，然后单击图 1.16 所示窗口右侧的操作按钮，准备更改 Python 的解释器。

步骤 03 单击“Add”按钮，加载 WSL 中的 Python 解释器，如图 1.17 所示。

步骤 04 之后在弹出的界面左侧的类别中选择“WSL”，右侧路径改成如图 1.18 所示的内容。

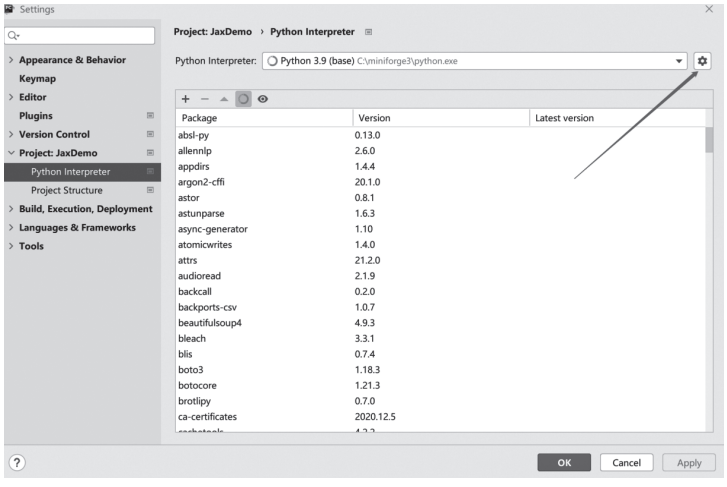

图 1.16 更改 Python 解释器

注 意

这里的默认路径地址是 python，而读者在 WSL 中使用的 Python 版本是 python3。

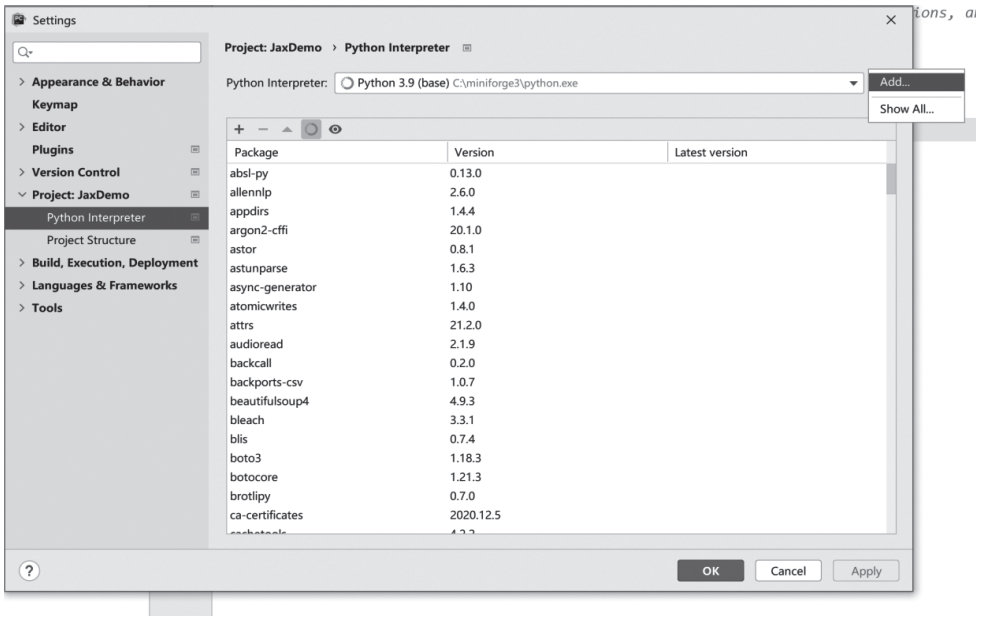

图 1.17 添加 Python 解释器

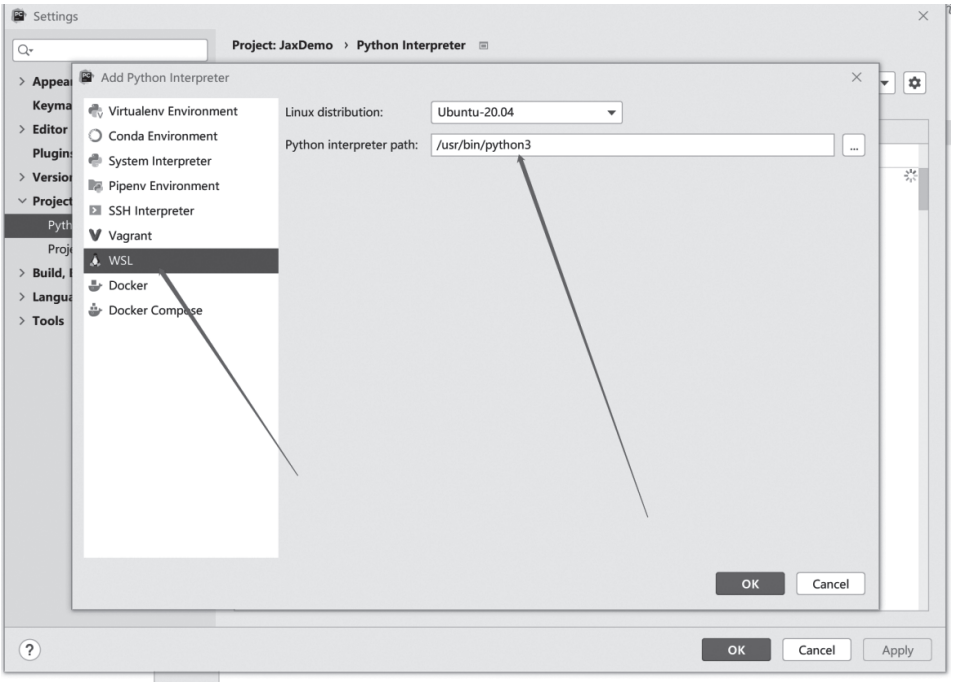

图 1.18 加载 WSL 中的 Python3 解释器

步骤 05 单击“OK”按钮，PyCharm 开始加载文件，界面如图 1.19 所示。

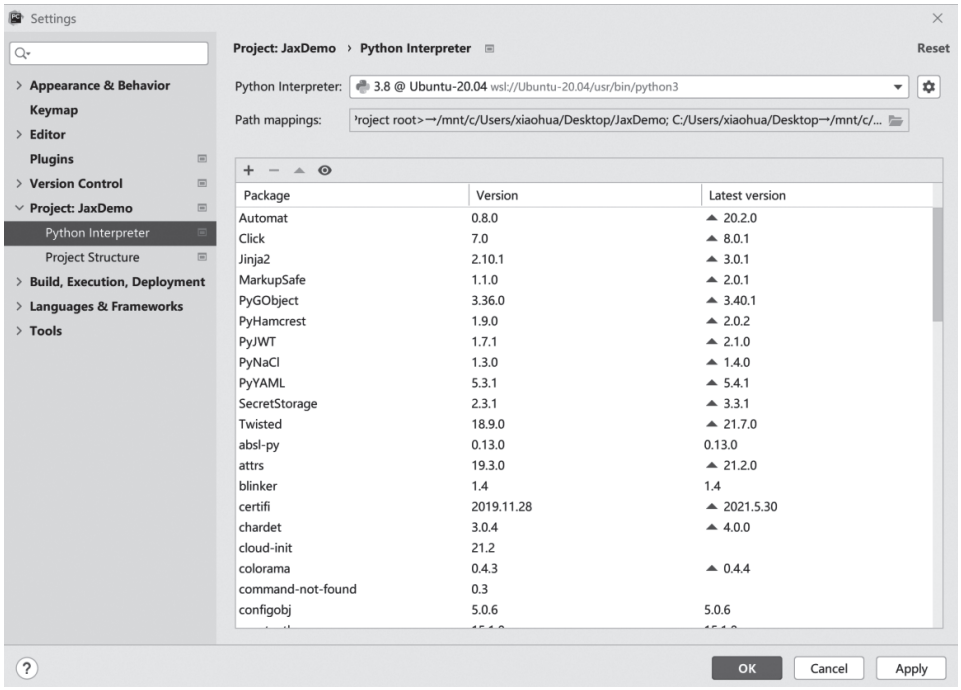

图 1.19 编译器配置完成

等编译结束后就可以进入 PyCharm 的程序代码阶段，至此配置完成。

1.2.5 JAX 的 Python 代码小练习：计算 SeLU 函数

对于科学计算来说，最简单的想法就是可以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 实现和计算一个深度学习中最为常见的函数——SeLU 激活函数。至于这个函数的作用，现在不加以说明，这里只是带领读者尝试实现其程序的编写。

首先 SeLU 激活函数计算公式如下所示：

$$\alpha=\alpha\times(e^{x}-1)\times\theta$$

$$\alpha=1.67$$

$$\theta=1.05$$

e 是自然常数

其中 α 和 θ 是预定义的参数，e 是自然常数，以上 3 个数在这里直接使用即可。SeLU 激活函数的图形如图 1.20 所示。

SeLU 激活函数的代码如下所示。

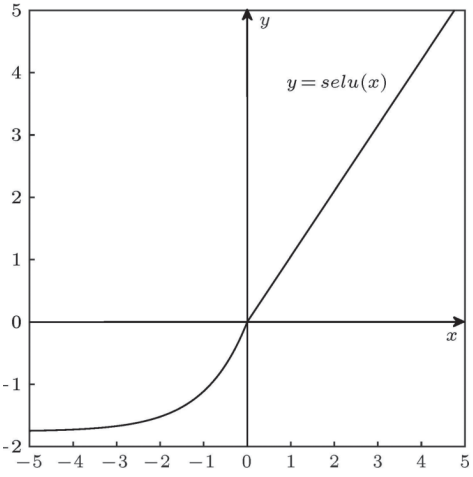

图 1.20 SeLU 激活函数图形

【程序 1-1】

```
import jax.numpy as jnp                          # 导入 NumPy 计算包
from jax import random                           # 导入 random 随机数包
# 完成的 seLU 函数
def selu(x, alpha=1.67, lmbda=1.05):
    return lmbda * jnp.where(x > 0, x, alpha * jnp.exp(x) - alpha)
key = random.PRNGKey(17)                         # 产生了一个固定数 17 作为 key
x = random.normal(key, (5,))                     # 随机生成一个大小为 [1,5] 的矩阵
print(selu(x))                                   # 打印结果
```

可以看到，当传入一个随机数列后，分别计算每个数值所对应的函数值，结果如下：

```
[-1.2464962   0.45437852  1.5749676  -0.8136045   0.27492574]
```

1.3 JAX 实战——MNIST 手写体的识别

MNIST 是深度学习领域常见的数据集。每一个 MNIST 数据单元由两部分组成：一幅包含手写数字的图片和一个对应的标签。我们把这些图片设为“xs”，把这些标签设为“ys”。训练数据集和测试数据集都包含 xs 和 ys，比如训练数据集的图片是 mnist_train_x，训练数据集的标签是 mnist_train_y。

如图 1.21 所示，每一幅图片包含 28×28 个像素点。如果我们把这个数组展开成一个向量，长度是 28×28 = 784。如何展开这个数组（数字间的顺序）不重要，只要保持各幅图片采用相同的方式展开即可。从这个角度来看，MNIST 数据集的图片就是在 784 维向量空间里面的点。

图 1.21 数据集 MNIST

1.3.1 第一步：准备数据集

程序设计的第一步是准备数据，我们使用 tensorflow_datasets 自带的框架解决 MNIST 数据集下载的问题。打开 WSL 终端，输入如下命令：

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple tensorflow_datasets
```

注意 进度条读取完毕后还不能使用，PyCharm 对于 WSL 的支持需要重新加载在 WSL 中的 Python 程序，这里只需要重启计算机即可。

MNIST 数据集下载好了之后，只需要直接使用给定的代码完成 MNIST 数据集的载入即可。代码如下：

```
import tensorflow as tf
import tensorflow_datasets as tfds
```

```
x_train = jnp.load("mnist_train_x.npy")
y_train = jnp.load("mnist_train_y.npy")
```

注意，由于 MNIST 给出的 label 是一个以当前图像值为结果的数据，需要转换成 one_hot 格式，代码如下：

```
def one_hot_nojit(x, k=10, dtype=jnp.float32):
    """ Create a one-hot encoding of x of size k. """
    return jnp.array(x[:, None] == jnp.arange(k), dtype)
```

1.3.2 第二步：模型的设计

我们需要使用一个深度学习网络对 MNIST 数据集进行分类计算，如图 1.22 所示。

在此采用的深度学习网络是使用两个全连接层加激活层进行，代码如下：

```
# {Dense(1024) -> ReLU}x2 -> Dense(10) -> Logsoftmax
init_random_params, predict = stax.serial(
    stax.Dense(1024), stax.Relu,
    stax.Dense(1024), stax.Relu,
    stax.Dense(10), stax.Logsoftmax)
```

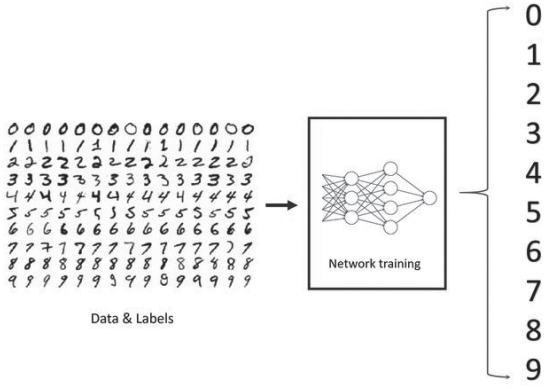

图 1.22 分类计算

其中的 Dense 是全连接层，而参数是当全连接层计算后生成的维度。

1.3.3 第三步：模型的训练

下面就是模型的训练过程，由于深度学习的训练需要使用激活函数以及优化函数，在本例中笔者为了方便起见，只提供具体的使用方法，暂时不提供相应的讲解，完整的模型训练代码如下所示。

【程序 1-2】

```
import tensorflow as tf
import tensorflow_datasets as tfds
import jax
import jax.numpy as jnp
```

```
    from jax import jit, grad, random
    from jax.experimental import optimizers
    from jax.experimental import stax
    num_classes = 10
    reshape_args = [(-1, 28 * 28), (-1,)]
    input_shape = reshape_args[0]
    step_size = 0.001
    num_epochs = 10
    batch_size = 128
    momentum_mass = 0.9
    rng = random.PRNGKey(0)
    x_train = jnp.load("mnist_train_x.npy")
    y_train = jnp.load("mnist_train_y.npy")
    def one_hot_nojit(x, k=10, dtype=jnp.float32):
        """ Create a one-hot encoding of x of size k. """
        return jnp.array(x[:, None] == jnp.arange(k), dtype)
    total_train_imgs = len(y_train)
    y_train = one_hot_nojit(y_train)
    ds_train = tf.data.Dataset.from_tensor_slices((x_train, y_train)).
shuffle(1024).batch(256).prefetch(
        tf.data.experimental.AUTOTUNE)
    ds_train = tfds.as_numpy(ds_train)
    def pred_check(params, batch):
        """ Correct predictions over a minibatch. """
        inputs, targets = batch
        predict_result = predict(params, inputs)
        predicted_class = jnp.argmax(predict_result, axis=1)
        targets = jnp.argmax(targets, axis=1)
        return jnp.sum(predicted_class == targets)
    # {Dense(1024) -> ReLU}x2 -> Dense(10) -> Logsoftmax
    init_random_params, predict = stax.serial(
        stax.Dense(1024), stax.Relu,
        stax.Dense(1024), stax.Relu,
        stax.Dense(10), stax.Logsoftmax)
    def loss(params, batch):
        """ Cross-entropy loss over a minibatch. """
        inputs, targets = batch
        return jnp.mean(jnp.sum(-targets * predict(params, inputs), axis=1))
    def update(i, opt_state, batch):
        """ Single optimization step over a minibatch. """
        params = get_params(opt_state)
        return opt_update(i, grad(loss)(params, batch), opt_state)
    # 这里的 step_size 就是学习率
    opt_init, opt_update, get_params = optimizers.adam(step_size = 2e-4)
    _, init_params = init_random_params(rng, input_shape)
    opt_state = opt_init(init_params)
```

```
    for _ in range(17):
        itercount = 0
        for batch_raw in ds_train:
            data = batch_raw[0].reshape((-1, 28 * 28))
            targets = batch_raw[1].reshape((-1, 10))
            opt_state = update((itercount), opt_state, (data, targets))
            itercount += 1
        params = get_params(opt_state)
        train_acc = []
        correct_preds = 0.0
        for batch_raw in ds_train:
            data = batch_raw[0].reshape((-1, 28 * 28))
            targets = batch_raw[1]
            correct_preds += pred_check(params, (data, targets))
        train_acc.append(correct_preds / float(total_train_imgs))
        print(f"Training set accuracy: {train_acc}")
```

这样就完成了一个模型的训练代码，运行结果如图 1.23 所示。

```
C:\Windows\system32\wsl.exe --distribution Ubuntu-20.04 --exec /bin/sh -c "export PYCHARM_DISPLAY_PORT=63342 && export PYTHONPATH='/mnt/c/Users/xiaohua/De
2021-09-15 11:22:27.671777: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dlerror: l
2021-09-15 11:22:27.671810: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machi
WARNING:absl:No GPU/TPU found, falling back to CPU. (Set TF_CPP_MIN_LOG_LEVEL=0 and rerun for more info.)
2021-09-15 11:22:33.910658: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcuda.so.1'; dlerror: libcud
2021-09-15 11:22:33.910704: W tensorflow/stream_executor/cuda/cuda_driver.cc:326] failed call to cuInit: UNKNOWN ERROR (303)
2021-09-15 11:22:33.910722: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (XS-wangxi
2021-09-15 11:22:33.911468: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Libra
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Training set accuracy: [DeviceArray(0.95171666, dtype=float32, weak_type=True)]
Training set accuracy: [DeviceArray(0.96788335, dtype=float32, weak_type=True)]
Training set accuracy: [DeviceArray(0.98298335, dtype=float32, weak_type=True)]
Training set accuracy: [DeviceArray(0.9792333, dtype=float32, weak_type=True)]
Training set accuracy: [DeviceArray(0.9884833, dtype=float32, weak_type=True)]
Training set accuracy: [DeviceArray(0.98955, dtype=float32, weak_type=True)]
Training set accuracy: [DeviceArray(0.9938167, dtype=float32, weak_type=True)]
```

图 1.23 模型训练结果

可以看到从第 7 个 epoch 开始，模型在训练集上的准确率已经达到了一个较高的水平。

1.4 本章小结

本章介绍了 JAX 的基本概念、JAX 虚拟环境搭建以及 JAX 应用开发方法，还分析了一个 MNIST 手写体识别的例子，告诉读者使用 JAX 时只需要简单的几步就可以。当然，在真正掌握 JAX 处理的步骤和方法之前，还有很长一段路要走，希望本书能够引导大家入门。