第3章 深度学习环境

视频

工欲善其事，必先利其器。深度学习通常用来处理图像、语音和视频等大量非结构化数据，因此，拥有强大的计算资源是进行深度学习开发的必要条件之一。深度学习有其独特的运行环境，对于深度学习研究者而言，搭建能够运行深度学习代码的环境将是艰难的第一步。本章简要介绍与深度学习相关的编程语言、编程环境以及开发框架，并以 PyTorch 为例指导读者搭建一个深度学习环境。

3.1 编程语言和环境

3.1.1 编程语言

编程语言可以理解为人与计算机硬件之间沟通的一种方式。通过构建基本的词汇和语法规则，人们可以向计算机硬件发出指令，使其执行特定任务。目前，根据不同的应用场景，编程语言有 600 多种，学术领域常用的编程语言有 C 语言、MATLAB、R 语言和 Python 等。

1. C 语言

C 语言是一门面向过程的、抽象化的通用程序设计语言，它能以简易的方式编译低级存储器，并不需要任何运行环境支持便能运行。虽然 C 语言可以提供许多低级处理的功能，但仍然保持着跨平台的特性。C 语言只有 32 个关键字、九种控制语句，程序书写形式自由，区分大小写，可把高级语言的基本结构和语句与低级语言的实用性结合起来，运算符和数据类型极为丰富、表达方式灵活实用、程序执行效率高、可移植性好。然而，C 语言也有明显不足，具体表现在 C 语言的语法及变量类型约束不严，对数组下标越界不进行检查，从而导致程序的安全性不强。另外，C 语言比其他高级语言较难掌握，程序设计的门槛较高。

2. MATLAB

MATLAB 是美国 MathWorks 公司推出的商业数学软件，其名字是 Matrix、Laboratory 两个单词的组合，意为矩阵实验室。MATLAB 主要用于数值分析、矩阵计算、算法开发、科学数据可视化以及非线性动态系统的建模和仿真等方面，并集成在一个易于使用的视窗环境中。MATLAB 主要包含两个部分：核心部分和各种可选的工具箱。核心部分中有数百个核心内部函数，而工具箱主要分成两大类：功能性工具箱和学科性工具箱。由于 MATLAB 是解释型语言，其执行速度要比编译型语言慢得多。根据 MATLAB 版本的不同，安装时需要的磁盘空间大小会有所不同，但目前常用的 MATLAB 基本都需要 5GB 以上的磁盘空间，完整安装需要占用 9～10GB 的磁盘空间。MATLAB 在科学计算中的功能十分强大，以致 2020 年中国的部分理工科高校被美国禁

用 MATLAB。

3. R 语言

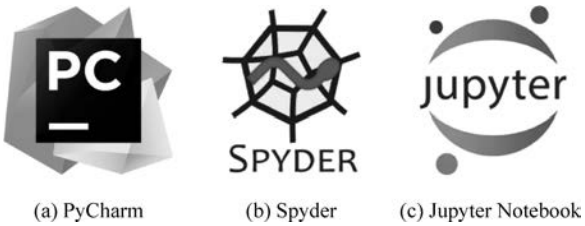
R 语言是属于 GNU(GNU's Not Unix,GNU 不是 Unix)操作系统的一个自由、免费和开源的软件,是一种用来进行数据探索、统计分析和绘图的解释型语言。R 语言具有丰富的数据类型、数量众多的算法功能包以及顶尖的绘图功能。由于 R 语言涵盖基础统计学、社会学、经济学、生态学、空间分析和生物信息学等诸多方面,深受科研工作者喜爱。作为一个开源软件,R 语言背后有一个强大的社区和大量的开放源代码支持,获取帮助非常容易。国外比较活跃的社区有 GitHub 和 Stack Overflow 等,通常 R 包的开发者会先将代码放到 GitHub,再根据世界各地使用者提出的问题进行修改完善。

4. Python

Python 由荷兰数学和计算机科学研究学会的吉多·范罗苏姆(Guido van Rossum)于 20 世纪 90 年代初设计,由于其简洁性、易读性以及可扩展性已成为最受欢迎的编程语言之一。Python 不但可以提供高效的数据结构,而且能够简单有效地面向对象编程,在开发过程中没有编译环节。Python 拥有一个强大的标准库,众多开源的科学计算软件包都提供了 Python 的调用接口,例如计算机视觉库 OpenCV、三维可视化库 VTK、医学影像处理库 ITK。Python 专用的科学计算扩展库则更多,例如 NumPy、SciPy 和 Matplotlib 等。Python 可应用于多种平台,包括 Windows、Linux/Unix 和 macOS 等,而运行 Python 可以使用交互式解释器、命令行脚本以及集成开发环境等方式。

3.1.2 编程环境

集成开发环境(Integrated Developing Environment,IDE)是一个综合性的工具软件,它把程序设计全过程所需的各项功能集成在一起,统一在一个图形化操作界面下,为程序设计人员提供完整的服务。例如 C 语言的 IDE 有 Visual Studio、Eclipse 和 CLion,MATLAB 有自身的 IDE 以及深度学习工具箱,R 语言的 IDE 有 RStudio,Python 的 IDE 有 PyCharm、Spyder 和 Jupyter Notebook 等,本书重点介绍 Python 的三个常用 IDE,其图标如图 3.1 所示。



(a) PyCharm (b) Spyder (c) Jupyter Notebook

图 3.1 集成开发环境图标

1. PyCharm

PyCharm 是使用最广泛的一种适用于 Python 编程语言的集成开发环境，诸如 Twitter、Pinterest、HP、Symantec 和 Groupon 等大型企业正在使用它进行软件算法开发[1]。PyCharm 中有一个 Python Console 交互式开发环境，可通过单击 PyCharm 下方的 Python Console 按钮调出 Python Console 控制台，这个控制台会自动加载 PyCharm 已经调试过的解释器。交互式开发环境使得程序员可以直接在控制台左侧写代码，在不写 print()的情况下也能输出对应数据。总体而言，PyCharm 提供了一整套可以帮助用户在进行 Python 开发时提高效率的工具，无论是第三方库的下载安装，还是插件的自定义配置，再到使用框架来完成程序搭建和发布，PyCharm 都能够轻松实现。无论对于初学者，还是资深开发者，PyCharm 都是一个非常实用的深度学习编程环境。

2. Spyder

Spyder 是一个相对简易的 IDE，支持包括 Windows、Linux 和 macOS 等操作系统，轻量、便捷是其显著特点[2]。此外，Spyder 能够模仿 MATLAB 的“工作空间”功能，可以方便地观察和修改数值。因此，对熟悉 MATLAB 的 Python 初学者非常友好。安装 Spyder 时可以直接从其官网下载，也可以直接安装 Anaconda，这是由于 Anaconda 集成了 Python 以及常用的数据分析环境，具体包括 Numpy、Pandas、Scipy 和 Sklearn 等常用包以及集成开发环境 Spyder，安装完成之后可直接使用。在编程过程中 Spyder 可实时提示文档、交互式运行、调试时显示全部变量表，并可一键可视化等，同时它也支持步进跟踪等一系列 Python 调试器所提供的调试功能。

3. Jupyter Notebook

Jupyter Notebook 是一个开源的 Web 应用程序，允许用户创建和共享包含代码、方程式、可视化和文本的文档，可用于数据清理和转换、数值模拟、统计建模、数据可视化和机器学习等[3]。Jupyter Notebook 支持运行 40 多种编程语言，科学计算方向的开发者一般都会使用 Jupyter Notebook。总体而言，Jupyter 最大的优点在于它特殊的编辑环境能够通过 pip 命令下载安装完成，启动之后会将浏览器打开并进入一个网页进行程序的编写。轻量化、使用简单是 Jupyter Notebook 的特色，其不足是在启动之后无法再打开其他目录下的文件，只有在启动之前才能切换工作目录。Jupyter Notebook 在 Python 安装时不会默认安装，但是在另一个开源 Python 发行版本 Anaconda 中会默认安装。

3.2 深度学习框架

在深度学习的初始阶段，不同研究者需要编写大量相似的程序代码。为了提高工作效率，研究者将这些代码写成了一个框架，放到互联网上进行开源以供所有研究者使用。

深度学习框架就是一套通用积木，各个组件就是某个模型或算法的一部分，研究者可以自行设计如何使用积木去实现符合项目要求的组合。随着时间的推移，最好用的几个框架被大量研究者使用而流行起来。深度学习框架的出现降低了人工智能开发者入门的技术门槛，研究人员不需要从复杂的神经网络开始编写代码，而是可以根据需要使用已有的深度神经网络，并对网络参数进行训练，或者对已有网络的结构进行改进。

不同的深度学习框架各有千秋，没有一个框架是完美无缺的，因此，在开始深度学习项目之前，选择一个合适的框架就显得尤为重要。总体而言，现有的深度学习框架包括Theano、Caffe、Darknet、MXNet、CNTK、飞桨、TensorFlow、Keras 和 PyTorch 等。由于Python 的易用性和可扩展性，众多深度学习框架均提供了与 Python 的接口。

1. Theano

Theano 是由蒙特利尔大学在 2007 年推出的深度学习开发工具，也是第一个使用“符号计算图”描述模型表达式的开源架构[4]。Theano 的不足表现在调试困难，对错误信息可能没有帮助提示，编译时间长，其框架也更偏向底层，在计算机视觉领域没有预训练模型可以使用。自 2017 年之后，Theano 不再提供更新，表明其逐渐退出了历史舞台。

2. Caffe

Caffe 是加州大学伯克利分校贾扬清博士于 2013 年发布的深度学习框架，尤其适合前馈网络和图像处理[5]。模块化的代码使其具有较强的可读性，运算速度也比较快。Caffe 支持基于 GPU 和 CPU(Central Processing Unit，中央处理单元)的加速计算，还提供了与 Python 和 MATLAB 的接口。Caffe 的基本工作流程是基于一个简单假设，即所有计算均以层的形式表示。Caffe 的不足表现在不支持自动求导、代码不够精简以及扩展性稍差，尽管随后发布了性能更优的 Caffe2，但仍然没有流行起来。

3. Darknet

Darknet 是一个基于 C 语言和 CUDA 的开源深度学习框架，其主要优点是容易安装，没有任何依赖项，结构明晰，源代码查看和修改方便，也提供了与 Python 的接口[6]。Darknet 的可移植性较好，支持 CPU 和 GPU 两种计算方式。由于 Darknet 较为轻型，这使得研究人员可以方便地从底层对其进行改进和扩展。遗憾的是，Darknet 没有使用社区，算是一个比较小众的深度学习框架。

4. MXNet

MXNet 是 2014 年由陈天奇和李沐号召开发的，允许用户在多种设备(基础设施或移动设备)上定义、训练和部署深度神经网络[7]。目前，MXNet 是亚马逊云计算的官方深度学习平台。MXNet 框架是一个支持混合命令式和符号式的编程模型，可以更大限

度地提升计算效率。MXNet 支持多种语言，具有高度的可扩展性。然而，由于 MXNet 的快速更新迭代，很多文档长时间没有更新，掌握起来相对困难。

5. CNTK

CNTK 是微软公司开发的一个开源深度学习工具包，支持 CPU 和 GPU。CNTK 将神经网络描述为有向图上的一系列计算[8]。在 CNTK 上可以容易地实现主流模型，在实现随机梯度下降学习时能够自动计算梯度，而且还能通过多个 GPU 实现并行计算。由于 CNTK 是由微软语音团队开源，非常适合进行语音处理，现为 Cortana 数字助理和 Skype 翻译中使用的语音识别系统框架。

6. 飞桨

百度公司研发的飞桨(PaddlePaddle)是中国首个自主研发的开源产业级深度学习框架。飞桨集深度学习核心训练和推理框架、基础模型库、端到端开发套件以及丰富的工具组件于一体，使得深度学习的技术创新和应用更简单[9]。飞桨平台结合了动态图的易用性和静态图的高性能，使开发者可以兼顾两者的优势。飞桨可以支持各种硬件端到端的部署能力，对于部署在移动端的模型，飞桨可以缩减到轻量化。此外，作为中国自己的深度学习框架，飞桨自然对国产硬件的适配性更好。

7. TensorFlow

2015 年，谷歌公司推出了 TensorFlow 框架。作为目前流行的深度学习框架，TensorFlow 可以提供矩阵运算、深度学习相关函数以及众多图像处理函数，其跨平台能力强，支持自动求导[10]。TensorFlow 最开始采用静态图计算，在最新发布的版本中，引入了动态图的计算方式。然而，TensorFlow 也有局限性，具体表现在其接口设计较为复杂、接口变动频繁、新旧版本兼容性较差、抽象概念较多以及需要使用者具有较高的理解能力。同年，谷歌工程师弗朗索瓦·肖莱(Francois Chollet)发布了 Keras 框架，其专注于用户友好、模块化和可扩展性，旨在快速实现深度神经网络，是 TensorFlow 高级集成应用程序界面(Application Program Interface，API)，可以非常方便地和 TensorFlow 进行融合，其高度的模块化使得网络搭建非常简洁。

8. Keras

Keras 是一个由 Python 编写的开源人工神经网络库，可以作为 TensorFlow、Microsoft-CNTK 和 Theano 的高阶应用程序接口，并方便地定义和训练几乎所有的深度网络[11]。在代码结构上完全模块化并具有可扩展性，内置支持卷积网络、循环网络以及二者的任意组合，相同代码能在 CPU 和 GPU 上进行无缝切换运行。Keras 没有单独的模型配置文件类型，模型由 Python 代码描述，使其更易调试。Keras 本身是一个中间层，通过它调

用 TensorFlow 会比单独使用 TensorFlow 要慢。为了提高扩展性，Keras 的层大多数使用 Python 实现，导致其占用较多的 GPU 显存。此外，由于 Keras 高度封装，用户使用时不灵活，难以对其底层进行操作。

9. PyTorch

PyTorch 是由 Facebook 人工智能研究院在 2017 年推出的开源 Python 深度学习库。PyTorch 支持 GPU 加速张量计算，拥有实现深度神经网络自动求导的强大工具[12]。由于 PyTorch 能够创建动态图，与 Python 紧密集成，使得 PyTorch 在众多深度学习框架中脱颖而出，成为构建神经网络灵活、高效的平台，其关注度持续上升。PyTorch 采用命令式编程，与 Python、Java 等计算机编程语言的用法比较类似，代码简洁且容易上手，其设计逻辑比较符合人类思维。尽管 PyTorch 是非常年轻的深度学习框架，但简洁易懂、便于调试和强大高效的优点，使其成为与 TensorFlow 平分秋色的深度学习框架。

3.3 PyTorch 深度学习环境的搭建

PyTorch 是基于 Python 的深度学习框架，网络搭建方便，整体可读性好。与 TensorFlow 相比，PyTorch 在工程化部署方面流程相对简单，易于编译。因此，本书主要以 PyTorch 深度学习框架为例介绍其搭建过程，首先对安装时常被重点提及的名词进行解释。

图形处理单元(Graphics Processing Unit，GPU)：专为数据的并行处理而设计，可以更好地进行图形和视频渲染[13]。GPU 把所有的计算任务都安排好，然后再批处理，对缓存的要求相对很低，适合逻辑简单的并行处理。由于深度学习的核心是参数学习，而各参数之间相对独立可以并行处理，因此多核的 GPU 更适合深度学习。

统一计算设备架构(Compute Unified Device Architecture，CUDA)：CUDA 是由英伟达(NVIDIA)公司推出的通用并行计算架构，该架构使 GPU 能够解决复杂的计算问题，实现更高效的并行计算[14]。CUDA 包含指令集架构(Instruction Set Architectures，ISA)以及 GPU 内部的并行计算引擎，开发人员可以使用 C 语言为 CUDA 架构编写程序，并能在支持 CUDA 的处理器上以超高性能运行。

CUDA 深度神经网络库(CUDA Deep Neural Network library，cuDNN)：cuDNN 是 NVIDIA 打造的针对深度神经网络的 GPU 加速库。如果要用 GPU 训练模型，cuDNN 不是必需的，但是一般会采用这个加速库。cuDNN 的使用使得深度学习研究人员和框架开发人员可以专注于训练神经网络和开发软件应用程序，而不必在底层 GPU 性能优化上耗费时间。

综上所述，要进行深度学习的训练，需要拥有支持 CUDA 使用的 GPU 以及其他较

高性能的计算资源，建议 GPU 配置为 NVIDIA GTX 1070 及以上，RAM（Random Access Memory，随机存取存储器）内存大于 8GB，CPU 大于四核八线程。安装时要确认计算机的硬件配置是否满足需求，在终端输入 nvidia-smi 可查询 GPU 和 CUDA 的相关信息，如图 3.2 所示。需要指出的是，如果使用 Windows 系统，需把 nvidia-smi.exe 所在路径添加到环境变量中才可使用相关命令。最后，根据显示的 GPU 型号安装对应的 CUDA 软件版本，以及适配的 cuDNN 和 PyTorch 版本。这里以 CUDA 11.6、cuDNN v8.8.0、PyTorch 1.13.1、Anaconda 3.0 和 Python 3.9 为例进行介绍。

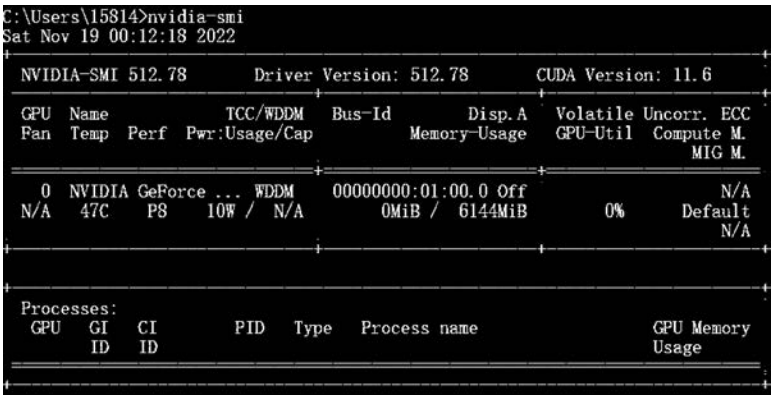


图 3.2　CUDA 查询口令及界面

1. 安装 CUDA

首先需要在英伟达（NVIDIA）官网根据系统信息选择对应的 CUDA 版本。对于 Windows 系统，下载后直接以管理员身份运行即可，如图 3.3 所示。CUDA 完成安装后，需要手动打开环境变量（控制面板→系统和安全→查看该计算机的名称→高级系统设置→环境变量）并进行编辑，将 CUDA 安装路径添加到环境变量中。最后打开终端输入 nvcc -V，若出现 CUDA 版本信息则说明安装成功。

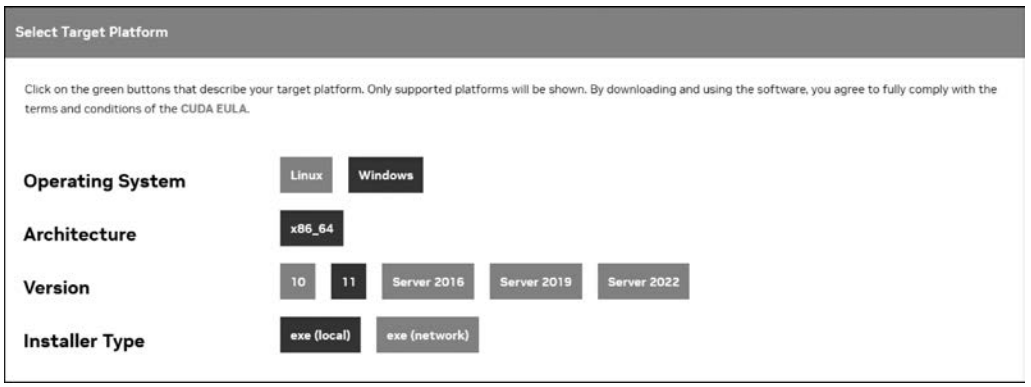


图 3.3　Windows 系统中 CUDA 的安装

如果是 Linux(Ubuntu 20.04)系统,同样根据系统配置选择相应的安装包,如图 3.4 所示,然后根据提示在终端依次输入安装命令。

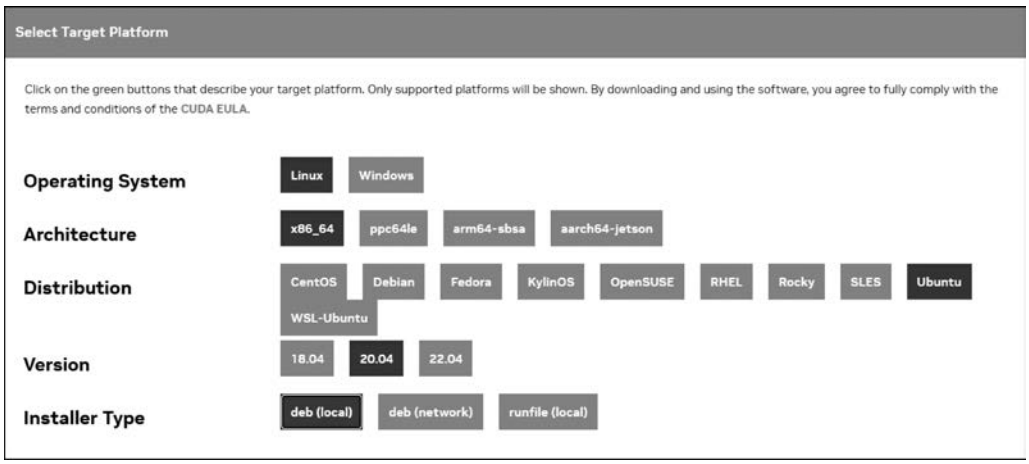


图 3.4 Linux 系统中 CUDA 的安装

CUDA 安装完毕后,需要将 CUDA 的目录设置到 PATH。此时需要打开终端输入:

```
gedit ~/.bashrc
```

需要指出的是,./bashrc 文件中需要添加如下路径信息,添加完成后打开终端,输入 source ~/.bashrc,让路径生效。

```
export LD_LIBRARY_PATH = $LD_LIBRARY_PATH:/usr/local/cuda - 11.6/lib64
export PATH = $PATH:/usr/local/cuda - 11.6/bin
export CUDA_HOME = $CUDA_HOME:/usr/local/cuda - 11.6
```

2. cuDNN 安装

使用 GPU 训练深度网络时,通常会使用 cuDNN 作为加速库[15]。首先需要进入官网注册 NVIDIA 账号,登录后根据自身的 CUDA 版本下载相对应的 cuDNN 版本,如图 3.5 所示。

在 Windows 系统中,对 cuDNN 压缩包解压后进行安装即可看到文件夹中的 bin、include 和 lib,将这几个文件夹替换到安装后的 CUDA 文件夹中即可完成安装。

针对 Linux(Ubuntu 20.04)系统,以 CUDA 11.6 为例下载对应的版本 cuDNN v8.8.0,然后将其解压,进入 cuDNN 文件夹目录后打开终端,运行以下命令进行安装:

```
sudo dpkg - i cudnn - local - repo - ubuntu2004 - 8.8.0.121_1.0 - 1_amd64.deb
```

最后运行如下命令即可验证是否安装成功。

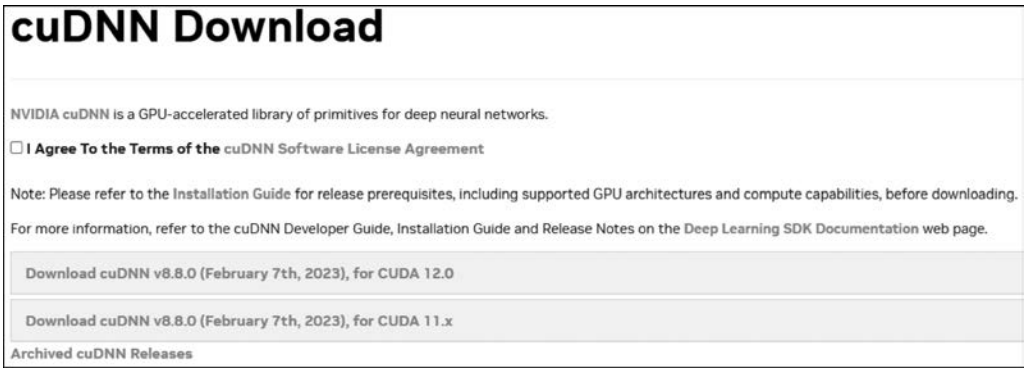


图 3.5　cuDNN 的下载界面

```
Idconfig -v | grep cudnn
```

3. 安装 Anaconda

Python 是一种面向对象的解释型计算机程序设计语言，可以在 Windows、Linux 和 macOS 系统中搭建环境并进行使用。Anaconda 是一个开源的 Python 发行版本，其包含 180 多个科学包和依赖项[16]。Anaconda 可利用 conda（包管理器）帮助计算机安装、卸载和更新第三方包。同时，conda 可帮助操作者为不同的项目建立不同的运行环境。因此，本章主要介绍利用 Anaconda 安装 Python 的过程。

在 Windows 系统中，进入 Anaconda 官网，如图 3.6 所示，单击 Download 按钮即可开始下载对应版本的安装文件，双击下载后的.exe 文件进行安装。测试 Anaconda 是否安装配置成功，可以使用 WIN＋R 组合键调出运行窗口并输入 cmd，最后在终端输入 conda 命令可查看是否安装成功。

图 3.6　Anaconda 的下载界面

在 Linux(Ubuntu 20.04)系统中,在官网下载对应版本的 Anaconda,使用如下的命令(根据本地 Anaconda 版本修改)进行安装:

```
sh Anaconda3-2022.10-Linux-x86_64.sh
```

安装过程中根据提示一直按 Enter 键即可,当最后一步提示是否初始化 Anaconda,输入 yes 后需要重新打开 shell,此时 Anaconda 就安装好了,使用 conda info 可查看是否安装成功。在 Windows 系统完成 Anaconda 安装后,可以继续选择安装 Python 常用的 IDE,例如 PyCharm、Spyder 和 Jupyter Notebook 等。

4. 安装 PyTorch

根据 CUDA 的对应版本执行 PyTorch 的安装工作。Pytorch 的常规安装方法包括 conda 和 pip 安装,本书主要介绍使用 conda 的安装过程。从 PyTorch 官网选择 PyTorch 的版本、操作系统、包管理器、编程语言以及 CUDA 版本,如图 3.7 和图 3.8 所示,每种操作系统对应不同的安装命令。

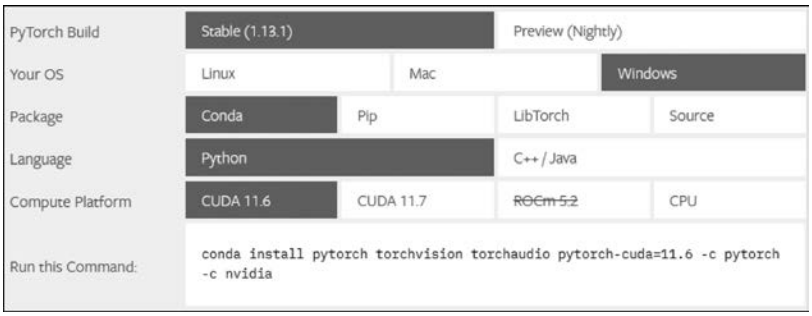


图 3.7 Windows 系统中使用 conda 安装 PyTorch 界面

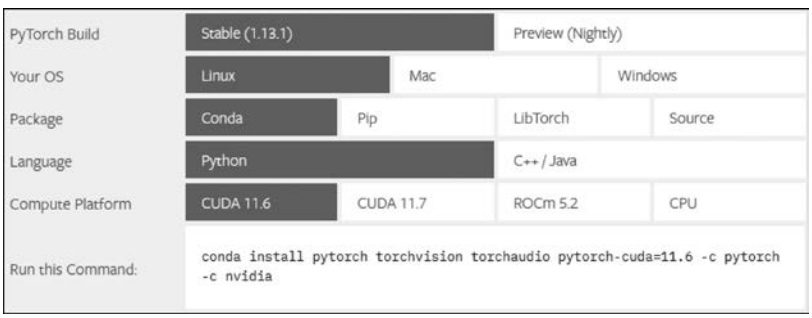


图 3.8 Linux 系统中使用 conda 安装 PyTorch 界面

在 Windows 系统中,打开 Anaconda Prompt 输入如下命令行:

```
conda install pytorch torchvision torchaudio pytorch-cuda=11.6 -c pytorch -c nvidia
```

随后会自动下载所需组件以及 PyTorch，安装完成后进入 Python，输入 import torch 没有报错，说明安装成功。需要指出的是，在 PyTorch 的安装过程中通常存在下载进程慢、进度不变或多次中断的情况，可利用清华镜像源的路径进行下载。

在 Linux(Ubuntu)环境中，首先需要创建虚拟环境、激活环境和检测环境，方法如下：

```
conda creat -n torch python=3.9
conda activate torch
conda info --envs
```

根据系统情况选择对应的 PyTorch 版本，安装命令如下：

```
conda install pytorch torchvision torchaudio pytorch-cuda=11.6 -c pytorch -c nvidia
```

由于 conda 的安装速度较慢，建议国内用户将 conda 源设置为清华 tuna，执行以下命令即可完成修改。

```
conda config - add channels https://mirrors.tuna.tsinghua. edu.cn/anaconda/pkgs/free/
conda config --set show_channel_urls yes
```

PyTorch 安装完成后需进行环境测试，首先激活虚拟环境并进入 torch 查看配置环境是否成功，如果结果显示 True，则配置完成。测试命令如下：

```
conda activate pytorch
python
import torch
torch.cuda.is_available()
```

3.4 本章小结

本章首先回顾近十年深度学习发展过程中流行过的编程语言和编程环境，介绍目前主流的深度学习框架。针对 PyTorch 深度学习框架，详细阐述如何搭建一个深度学习环境，并给出具体操作。需要指出的是，计算机硬件的配置需要根据所需处理的数据规模决定。数据量越大，其对计算机硬件资源的要求就越高。对于想从事大规模医学影像深度学习的研究者来说，这无疑设置了一道很高的门槛。

参考文献

[1] Hu Q, Ma L, Zhao J J, et al. DeepGraph: a PyCharm tool for visualizing and understanding deep learning models [C]. Proceedings of the International Conference on Asia-Pacific Software Engineering, Nara, Japan, 2018: 628-632.

[2] Raybaut P. Spyder-documentation[EB/OL]. Available online at: pythonhosted. org,2009.
[3] Zuniga-Lopez A, Aviles-Cruz C. Digital signal processing course on Jupyter-Python Notebook for electronics undergraduates[J]. Computer Applications in Engineering Education, 2020, 28(5): 1045-1057.
[4] AI-Rfou R, Alain G, Almahairi A, et al. Theano: A Python framework for fast computation of mathematical expressions[J]. arXiv: 1605. 02688,2016.
[5] Jia Y Q, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding[C]. Proceedings of the International Conference on Multimedia, New York, USA, 2014: 675-678.
[6] Fachkha C, Debbabi M. Darknet as a source of cyber intelligence: Survey, taxonomy, and characterization[J]. IEEE Communications Surveys & Tutorials, 2015, 18(2): 1197-1227.
[7] Chen T Q, Li M, Li Y T, et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems[J]. arXiv: 1512. 01274, 2015.
[8] Seide F, Agarwal A. CNTK: Microsoft's open-source deep-learning toolkit[C]. Proceedings of the International conference on Knowledge Discovery and Data Mining, San Francisco, USA, 2016.
[9] Ma Y J, Yu D H, Wu T, et al. PaddlePaddle: An open-source deep learning platform from industrial practice[J]. Frontiers of Data and Domputing, 2019, 1(1): 105-115.
[10] Abadi M. TensorFlow: learning functions at Scale [C]. Proceedings of the International Conference on Functional Programming, Nara, Japan, 2016.
[11] Gulli A, Pal S. Deep learning with Keras[M]. Packt Publishing Ltd, 2017.
[12] Stevens E, Antiga L, Viehmann T. Deep learning with PyTorch[M]. Manning Publications, 2020.
[13] Huang H, Liu X Y, Tong W Q, et al. High performance hierarchical tucker tensor learning using GPU tensor cores[J]. IEEE Transactions on Computers, 2023, 72(2): 452-465.
[14] Sanders J, Kandrot E. CUDA by example: an introduction to general-purpose GPU programming [M]. Addison-Wesley Professional, 2010.
[15] Jorda M, Valero-Lara P, Pena A J. Performance evaluation of cuDNN convolution algorithms on NVIDIA volta GPUs[J]. IEEE Access, 2019, 7: 70461-70473.
[16] Rolon-Mérette D, Ross M, Rolon-Mérette T, et al. Introduction to Anaconda and Python: Installation and setup[J]. The Quantitative Methods for Psychology, 2016, 16(5): S3-S11.