第3章 深度学习环境搭建

3.1 使用云端交互式计算环境

云端交互式计算环境为深度学习领域的学习者、开发者和研究者提供了高效便捷的开发模式和丰富的计算资源。交互式的开发环境使用户可以基于前序代码的计算结果实时地对后续代码进行开发和修改。云端计算环境使用户可以直接通过浏览器编写和运行代码，而无须在本地进行复杂的安装和配置过程。谷歌和 Kaggle 是两大提供免费计算资源的云端交互式计算环境服务商。谷歌提供了 Colaboratory(Colab)，用户在 Colab 中可以交互式地开发代码并免费使用计算资源。Kaggle 是一个竞赛平台，其中包含了众多深度学习相关的竞赛任务，并允许参赛用户免费使用其交互式计算环境和计算资源。

Colab 和 Kaggle 本质上都是基于 Jupyter 搭建的交互式计算服务。Jupyter 是一个基础的交互式计算环境，提供了基本的代码运行、数据可视化及 Markdown 文本编辑和渲染等功能。本节首先介绍 Jupyter 在本地的部署及其基本操作，然后介绍 Colab 和 Kaggle 等云端 Jupyter 服务的使用。

3.1.1 Jupyter 的本地部署和基本操作

JupyterLab 是一个多功能的交互式计算环境。在传统编程中，通常需要先编写完整的代码，然后一次性运行整个程序，若要调整代码或查看中间结果，则往往需要重新运行整个脚本，过程烦琐且效率较低，而在 Jupyter 中，用户可以创建 Jupyter 笔记本(Jupyter Notebook)，如图 3-1 所示。在笔记本中可以逐行或逐个代码块地运行代码，立即看到执行结果，实现快速高效的交互。除了可以交互式地运行代码，Jupyter 笔记本还支持嵌入 Markdown 文本及数学公式，以便用户向他人展示和分享自己的代码和想法。另外，Jupyter 笔记本可以更加方便地对数据进行可视化，在页面中直接展示图像和图表，对于数据分析及机器学习相关工作提供支持。

Jupyter 本质上是一个 Web 应用程序，它分为客户端和服务器端，用户可以在客户端中管理和访问自己的笔记本，并可在笔记本中进行交互式计算。用户在客户端提交的代码将

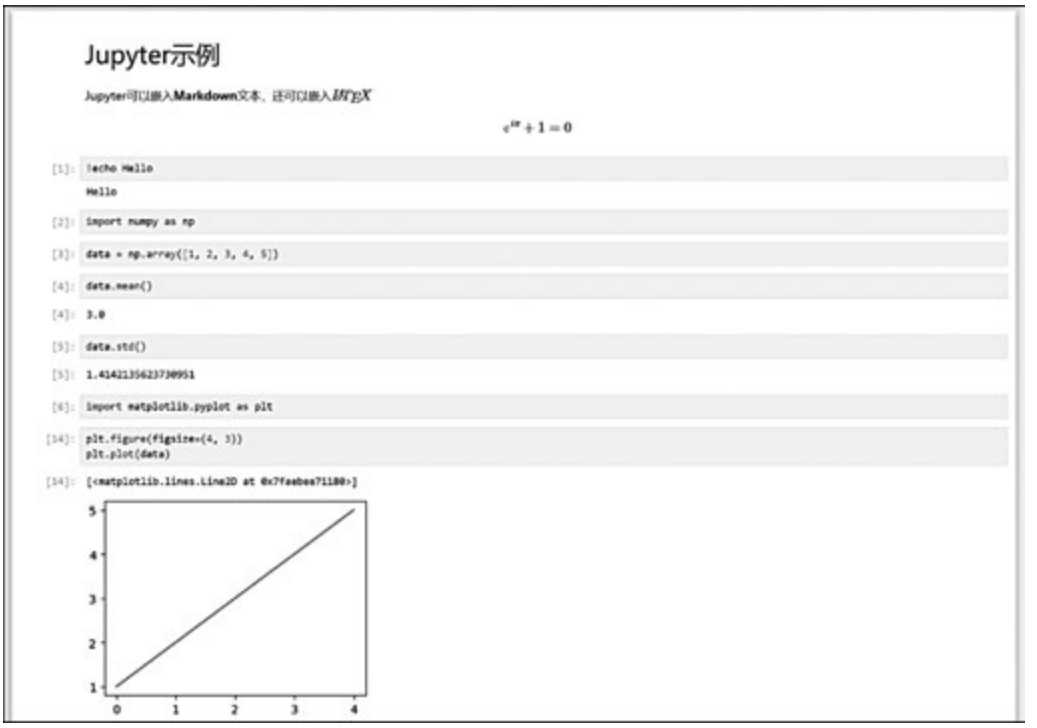


图 3-1　Jupyter 笔记本功能示意图

会在服务器上的 Python 解释器中运行，运行结果将被返回客户端呈现。实践中，可以在本地部署自己的 Jupyter 服务器并运行客户端，也可以直接使用谷歌 Colab 和 Kaggle Notebook 等提供云计算服务的产品。

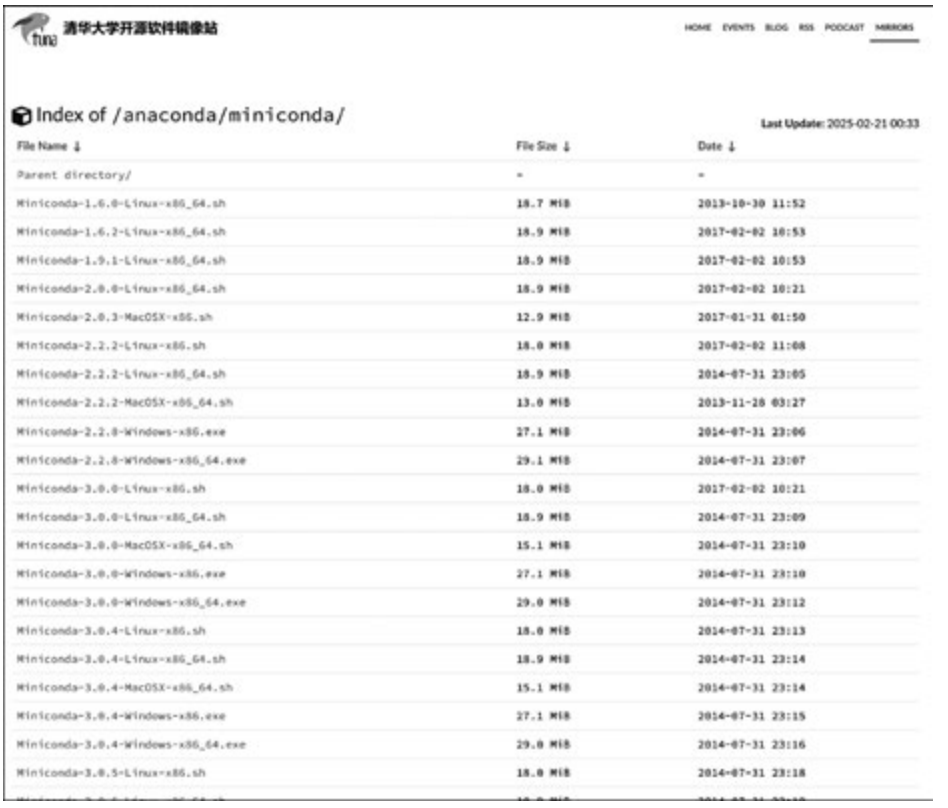
为了更好地理解 Jupyter Notebook 及其系列产品的运行原理，在讲述谷歌 Colab 和 Kaggle Notebook 的使用方法之前，首先介绍如何在本地 PC 上部署 Jupyter 服务器。

1. 安装 Miniconda

部署 Jupyter 服务器需要 Python 解释器。虽然 macOS 和 Linux 系统一般自带 Python 解释器，但是推荐安装 Miniconda 来管理 Python 环境和软件包。Miniconda 是一个跨平台开源软件包和环境管理系统。建议访问 https://mirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/从清华大学开源软件镜像站下载和安装适用于自己 PC 系统的 Miniconda 安装包，如图 3-2 所示。

对于不熟悉各版本区别的入门用户而言，建议下载名为 Miniconda3-py310_24.1.2-0 的安装包，可以在浏览器中按快捷键 Ctrl+F 搜索这一名称，如图 3-3 所示。这里高亮显示的安装包即为该版本适用于各操作系统的安装包。安装包名称中的 Miniconda3 表示是基于 Python 3 而不是 Python 2 的 Miniconda 发行版，py310 表示该发行版中默认的 Python 版本是 3.10，24.1.2 是该发行版的版本号，后缀为操作系统名称和架构。

对于 Windows 系统而言，下载.exe 安装包后运行并按照流程安装，注意勾选将



清华大学开源软件镜像站

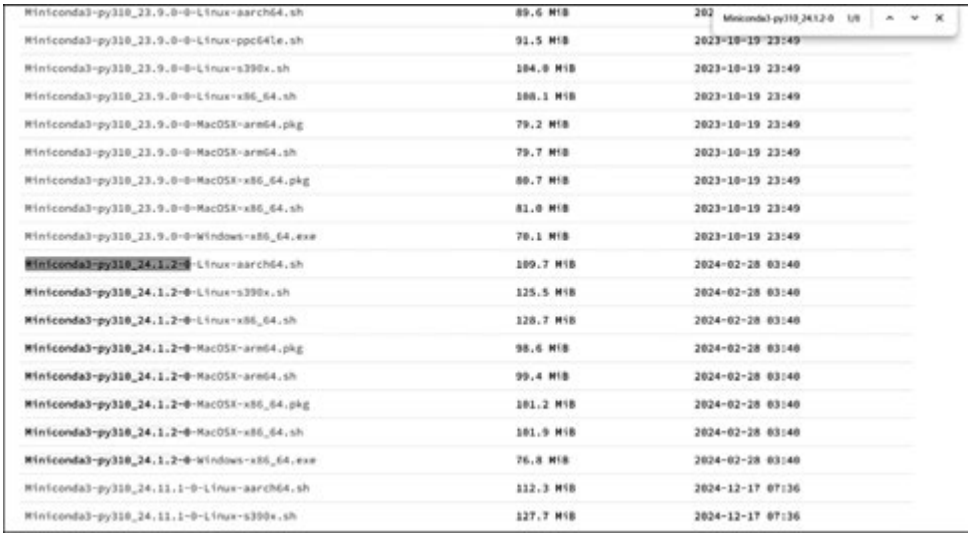
HOME EVENTS BLOG RSS PODCAST MIRRORS

Index of /anaconda/miniconda/

Last Update: 2025-02-21 00:33

File Name ↓	File Size ↓	Date ↓
Parent directory/	-	-
Miniconda-1.6.0-Linux-x86_64.sh	18.7 MiB	2013-10-30 11:52
Miniconda-1.6.2-Linux-x86_64.sh	18.9 MiB	2017-02-02 10:53
Miniconda-1.9.1-Linux-x86_64.sh	18.9 MiB	2017-02-02 10:53
Miniconda-2.0.0-Linux-x86_64.sh	18.9 MiB	2017-02-02 10:21
Miniconda-2.0.3-MacOSX-x86.sh	12.9 MiB	2017-01-31 01:50
Miniconda-2.2.2-Linux-x86.sh	18.0 MiB	2017-02-02 11:08
Miniconda-2.2.2-Linux-x86_64.sh	18.9 MiB	2014-07-31 23:05
Miniconda-2.2.2-MacOSX-x86_64.sh	13.0 MiB	2013-11-28 03:27
Miniconda-2.2.8-Windows-x86.exe	27.1 MiB	2014-07-31 23:06
Miniconda-2.2.8-Windows-x86_64.exe	29.1 MiB	2014-07-31 23:07
Miniconda-3.0.0-Linux-x86.sh	18.0 MiB	2017-02-02 10:21
Miniconda-3.0.0-Linux-x86_64.sh	18.9 MiB	2014-07-31 23:09
Miniconda-3.0.0-MacOSX-x86_64.sh	15.1 MiB	2014-07-31 23:10
Miniconda-3.0.0-Windows-x86.exe	27.1 MiB	2014-07-31 23:10
Miniconda-3.0.0-Windows-x86_64.exe	29.0 MiB	2014-07-31 23:12
Miniconda-3.0.4-Linux-x86.sh	18.0 MiB	2014-07-31 23:13
Miniconda-3.0.4-Linux-x86_64.sh	18.9 MiB	2014-07-31 23:14
Miniconda-3.0.4-MacOSX-x86_64.sh	15.1 MiB	2014-07-31 23:14
Miniconda-3.0.4-Windows-x86.exe	27.1 MiB	2014-07-31 23:15
Miniconda-3.0.4-Windows-x86_64.exe	29.0 MiB	2014-07-31 23:16
Miniconda-3.0.5-Linux-x86.sh	18.0 MiB	2014-07-31 23:18

图 3-2　清华大学开源软件镜像站 Miniconda 安装包页面



Miniconda3-py310_23.9.0-0-Linux-aarch64.sh	89.6 MiB	202
Miniconda3-py310_23.9.0-0-Linux-ppc64le.sh	91.5 MiB	2023-10-19 23:49
Miniconda3-py310_23.9.0-0-Linux-s390x.sh	104.0 MiB	2023-10-19 23:49
Miniconda3-py310_23.9.0-0-Linux-x86_64.sh	108.1 MiB	2023-10-19 23:49
Miniconda3-py310_23.9.0-0-MacOSX-arm64.pkg	79.2 MiB	2023-10-19 23:49
Miniconda3-py310_23.9.0-0-MacOSX-arm64.sh	79.7 MiB	2023-10-19 23:49
Miniconda3-py310_23.9.0-0-MacOSX-x86_64.pkg	80.7 MiB	2023-10-19 23:49
Miniconda3-py310_23.9.0-0-MacOSX-x86_64.sh	81.0 MiB	2023-10-19 23:49
Miniconda3-py310_23.9.0-0-Windows-x86_64.exe	70.1 MiB	2023-10-19 23:49
Miniconda3-py310_24.1.2-0-Linux-aarch64.sh	109.7 MiB	2024-02-28 03:40
Miniconda3-py310_24.1.2-0-Linux-s390x.sh	125.5 MiB	2024-02-28 03:40
Miniconda3-py310_24.1.2-0-Linux-x86_64.sh	128.7 MiB	2024-02-28 03:40
Miniconda3-py310_24.1.2-0-MacOSX-arm64.pkg	98.6 MiB	2024-02-28 03:40
Miniconda3-py310_24.1.2-0-MacOSX-arm64.sh	99.4 MiB	2024-02-28 03:40
Miniconda3-py310_24.1.2-0-MacOSX-x86_64.pkg	101.2 MiB	2024-02-28 03:40
Miniconda3-py310_24.1.2-0-MacOSX-x86_64.sh	101.9 MiB	2024-02-28 03:40
Miniconda3-py310_24.1.2-0-Windows-x86_64.exe	76.8 MiB	2024-02-28 03:40
Miniconda3-py310_24.11.1-0-Linux-aarch64.sh	112.3 MiB	2024-12-17 07:36
Miniconda3-py310_24.11.1-0-Linux-s390x.sh	127.7 MiB	2024-12-17 07:36

图 3-3　搜索建议安装的 Miniconda 版本

Miniconda 添加到环境变量及注册为默认 Python 解释器的选项，如图 3-4 所示。对于 macOS，可以下载.pkg 安装包并按照流程安装，也可以下载相应的.sh 安装脚本进行安装。对于 Linux 系统，可以先下载.sh 安装脚本，然后在终端运行安装脚本。

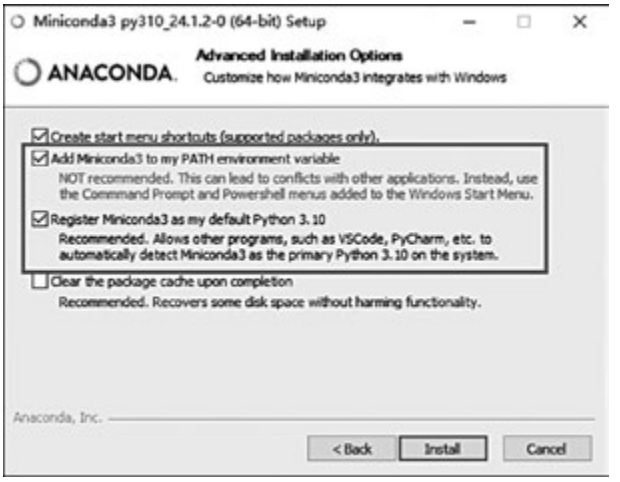


图 3-4　将 Miniconda 添加到环境变量并注册为默认 Python 解释器

2. 启动 Jupyter 服务

安装完 Miniconda 后，打开终端（Windows 系统打开 PowerShell 或命令提示符），运行如下命令对 Miniconda 及 Python 环境进行初始化并安装相关软件包：

```
conda update conda
conda update pip
pip config set global.index-urlhttps://pypi.tuna.tsinghua.edu.cn/simple
pip install numpy matplotlib pandas scikit-learnjupyterlab
pip install jupyter lab-language-pack-zh-CN
```

其中，第 1 条和第 2 条命令对 Conda 和 pip 的版本进行更新，其中 pip 是 Python 包安装器。第 3 条命令将 pip 的下载源设置为清华大学 pip 镜像站，可以加速软件包的下载。第 4 条命令则利用 pip 安装具体的 Python 软件包，其中 numpy 用于数值计算，matplotlib 用于绘制图表，pandas 用于数据操纵和分析，scikit-learn（sklearn）包含了机器学习相关的数据集和算法，而 jupyterlab 则用于部署 Jupyter 服务器。第 5 条命令用于安装 Jupyter 客户端的中文语言包。

接下来，在终端中运行如下命令即可启动 Jupyter 服务器：

```
python -m jupyter notebook
```

启动服务器后，终端中会出现 Jupyter 的运行日志，如图 3-5 所示，其中包含了 Jupyter 服务运行的主机名和端口号。打开浏览器并访问相应的地址（此处为 http://localhost:8888）即可使用 Jupyter 进行开发。

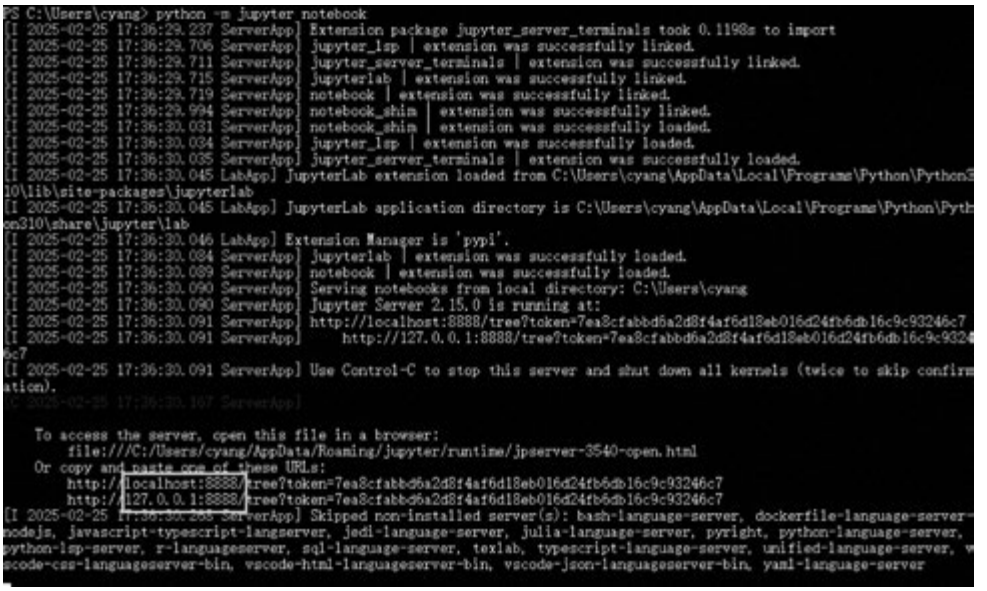


图 3-5 启动 Jupyter 服务器后终端显示的日志

3. 新建笔记本

在浏览器中进入 Jupyter 页面后，单击上方菜单栏中的“设置”，将语言切换为中文。此时所在的页面是当前 Jupyter 服务运行的目录，如图 3-6 所示。

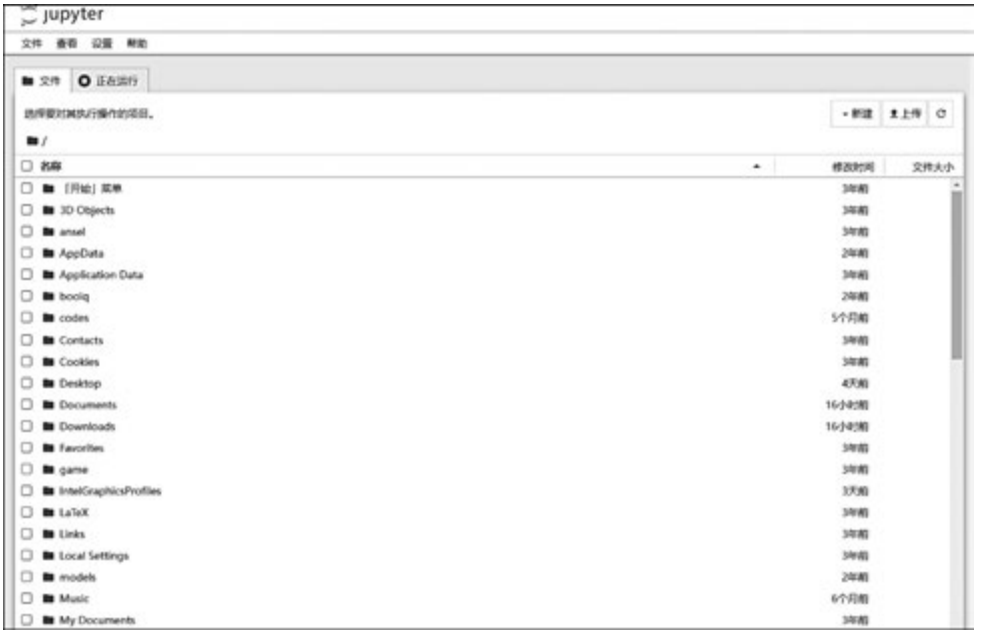


图 3-6 Jupyter 服务运行目录

首先单击右上角的“新建”按钮，然后选择 Python 3 (ipykernel)，即可在当前目录下新建一个笔记本，如图 3-7 所示。每个笔记本是一个.ipynb 文件，其中 Untitled1 是它的文件名，单击文件名可以对笔记本进行重命名。

图 3-7　新建的 Jupyter Notebook

4. 在笔记本中插入并运行代码和命令

新建的笔记本中包含一个代码块，可以在其中编写 Python 代码后，按快捷键 Shift＋Enter 运行，如图 3-8 所示。可以看到，运行的结果打印在相应的代码块下方，并且结果的下方将会出现新的代码块。

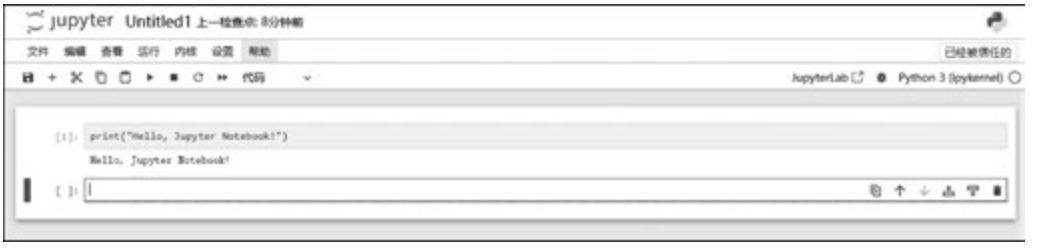


图 3-8　在代码块中运行 Python 代码

除了运行 Python 代码，在笔记本中还可以运行终端命令，例如，安装 Python 软件包时需要在终端运行 pip 命令，此时可以直接在笔记本中运行这一命令，如图 3-9 所示。需要注

图 3-9　在代码块中运行终端命令

意的是，运行终端命令时需要在命令开头加上英文感叹号，告诉笔记本这是一条终端命令而不是 Python 代码。另外，终端命令需要与运行 Jupyter 服务的终端一致，例如，如果 Jupyter 服务部署在 Linux 系统的 Shell 终端上，则代码块中运行的终端命令是 Shell 脚本。如果部署在 Windows 系统的 PowerShell 终端上，则运行的是 PowerShell 脚本。

除了可以顺序地使用笔记本自动创建的代码块，还可以在任意两个代码块之间插入一个代码块。假设现在需要在第 1 个和第 2 个代码块之间插入一个代码块，可以单击第 1 个代码块，此时代码块的右边会出现几个按钮，单击“下方插入单元格”按钮即可，如图 3-10 所示。另外，图中代码块最右边有删除按钮，可以删除不需要的代码块。

图 3-10　在笔记本中插入单元格

5. 在笔记本中插入 Markdown

每个单元格既可以作为代码块运行代码和命令，也可以在其中编写 Markdown 或纯文本内容，作为对代码的说明。对于当前选中的单元格，可以在上方的下拉菜单中选择其功能，如图 3-11 所示。

图 3-11　选择单元格功能

将所选单元格设为 Markdown 单元格，此时可以在单元格中编写 Markdown 代码，然后按快捷键 Shift ＋ Enter 即可显示解析后的 Markdown 文档，如图 3-12 所示。双击 Markdown 文档即可再次进入编辑模式。

(a) 编辑Markdown代码

(b) 显示Markdown文档

图 3-12　Markdown 的编辑与显示

6. 对内核进行操作

笔记本前端提交的所有代码均在 Jupyter 服务器中运行，更具体来讲是在 IPython 内核(IPython Kernel，ipykernel)中运行。IPython 内核是一个交互式的 Python 解释器[21]，相比原生的 Python 解释器，IPython 提供了更强大的编辑和互动功能。代码块中创建的变量将一直保存在内核中，直到内核被重启或关闭。

在笔记本中可以对内核进行一系列操作，如图 3-13 所示，其中“中断内核”表示中止当

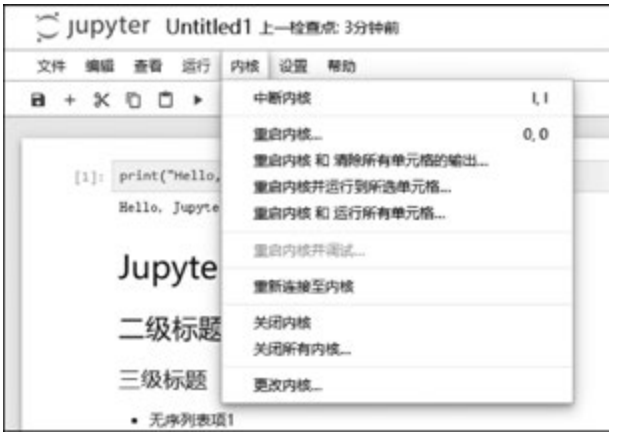


图 3-13　笔记本中针对内核的操作

前正在运行的代码，但是并不会删除内核中留存的变量。“重启内核”则不仅会中断当前运行的代码，还会删除之前运行代码时存储的变量，并启动一个新的内核。“重新连接至内核”适用于前端与内核丢失连接的情况，该操作不会影响内核中代码的运行，只是将前端与内核重新连接起来。“关闭内核”会中断当前运行的代码并删除内存中的变量，但是不会重新启动一个内核。

7. 选择代码运行模式

笔记本还提供了多种代码的运行模式。前面介绍的模式 Shift＋Enter 是最常见的逐单元格运行模式，也可以通过单击菜单栏的运行键实现这一运行模式，如图 3-14(a)所示。在“运行”菜单中可以选择多种运行模式，如图 3-14(b)所示，其中“运行选中的单元格并在下面插入(Alt＋Enter)”与“运行选中的单元格(Shift＋Enter)”略有区别。在运行笔记本中的最后一个单元格时，二者是等价的，即都会在提交运行后在下方插入一个新的单元格。然而，在运行中间单元格时，Shift＋Enter 在提交运行后会跳转到下一个单元格，而 Alt＋Enter 仍然会在下方插入一个新的单元格。“在控制台中运行所选文本或当前行”会跳转到 IPython 控制台中运行对应的代码，一般不需要用到该功能。另外，还有“运行选中单元格上方的所有单元格”、“渲染全部 Markdown”、“运行所有单元格”及“重启内核和运行所有单元格”等运行模式。

(a) 运行一个单元格　　(b) 在菜单中选择运行模式

图 3-14　笔记本中不同的代码运行模式

8. 利用笔记本进行数据可视化

一个简单的数据可视化程序如图 3-15 所示。第 1 个代码块用于导入相关的 Python 依赖包。第 2 个块中的两行代码分别用于加载 Iris 数据集并将其解析为 Pandas 表格数据，存储在变量 df 中。第 3 个块中只有变量 df，表示将该变量的值或其所包含的内容显示到笔记本中，其中 sepal length 表示花萼长度，sepal width 表示花萼宽度，petal length 表示花瓣长度，petal width 表示花瓣宽度。

利用上述表格数据可以绘制散点图，如图 3-16 所示，其中 plt. scatter 是绘制散点图的函数，其一般接收两个参数作为输入，第 1 个参数是 x 轴数据，第 2 个参数是 y 轴数据。这里取用花萼长度作为 x 轴数据，取用花瓣长度作为 y 轴数据，绘制的散点图直观地反映了

花瓣长度与花萼长度的相关关系。

```
[3]: from sklearn.datasets import load_iris
     import pandas as pd
     import matplotlib.pyplot as plt

[4]: iris = load_iris()
     df = pd.DataFrame(iris.data, columns=iris.feature_names)

[5]: df
```

[5]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

图 3-15　在笔记本中查看表格数据

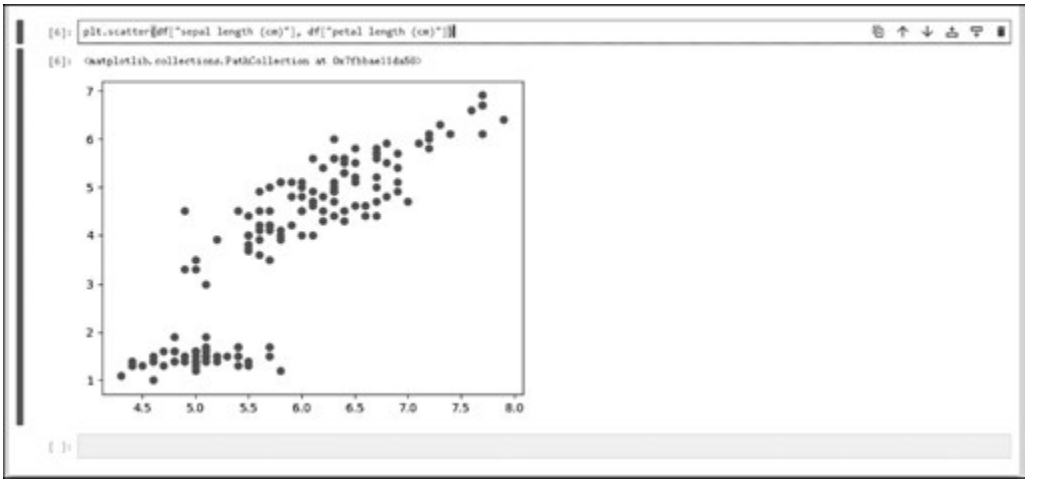


图 3-16　在笔记本中绘制散点图

9. 切换至 JupyterLab

单击菜单栏右上角的 JupyterLab 按钮可以将前端切换到 JupyterLab，如图 3-17 所示。JupyterLab 提供了更多功能，其中最主要的是在左侧提供了文件系统的可视化，在笔记本代码中对文件进行读写时可以先在左侧找到文件，然后打开此文件进行查看。被打开的文件会出现在新的标签页中，如图 3-18 所示。

注意：此处需要取消勾选左下角的“简易界面”功能才能保证标签页的正常显示，否则需要单击菜单栏中的“标签页”进行标签页的切换。

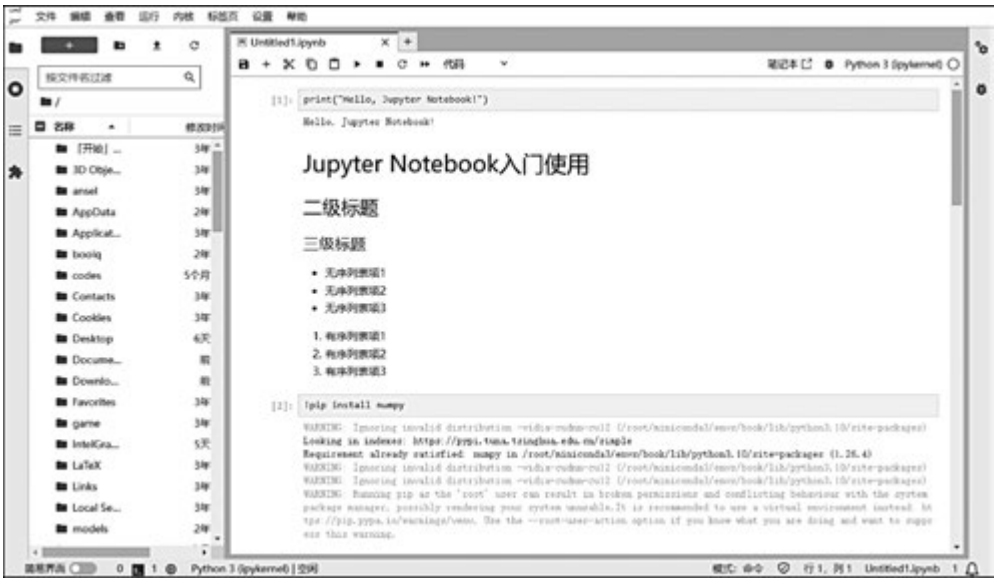


图 3-17 JupyterLab 界面示意图

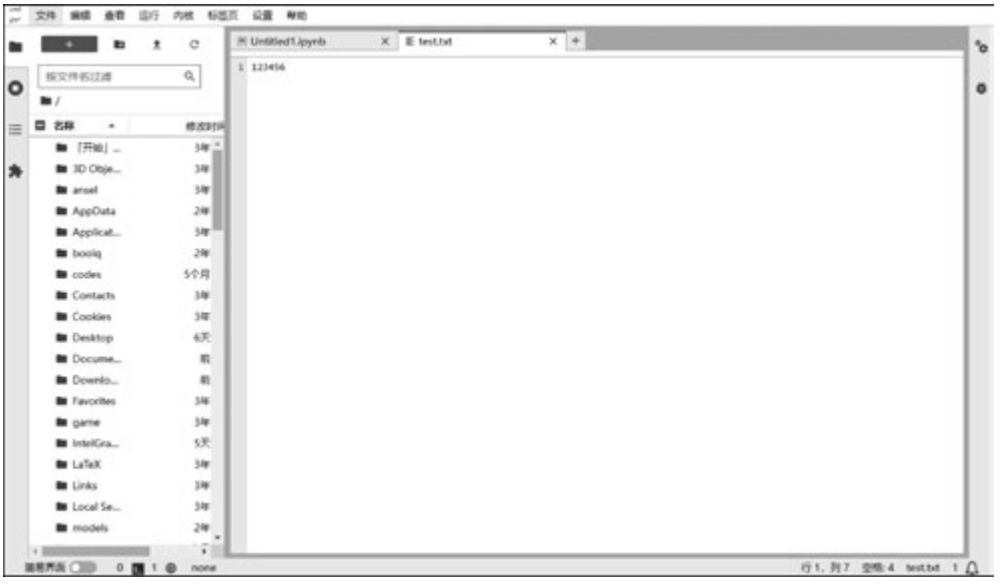


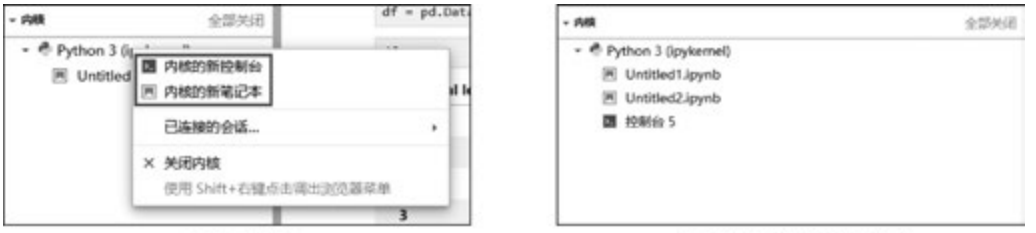
图 3-18 打开的文件出现在新标签页中

除了文件系统的可视化，在 JupyterLab 中还可以对内核、笔记本目录及 Jupyter 插件进行管理，如图 3-19 所示。

在内核管理功能区可以查看当前激活的所有内核，右击相应的内核可以基于该内核新建一个笔记本或控制台，如图 3-20 所示。双击图 3-20(b)中 Python 3 (ipykernel)内核下的任意笔记本或控制台即可在 JupyterLab 右侧标签栏中打开。

(a) 内核与标签页管理　(b) 目录管理　(c) 插件管理

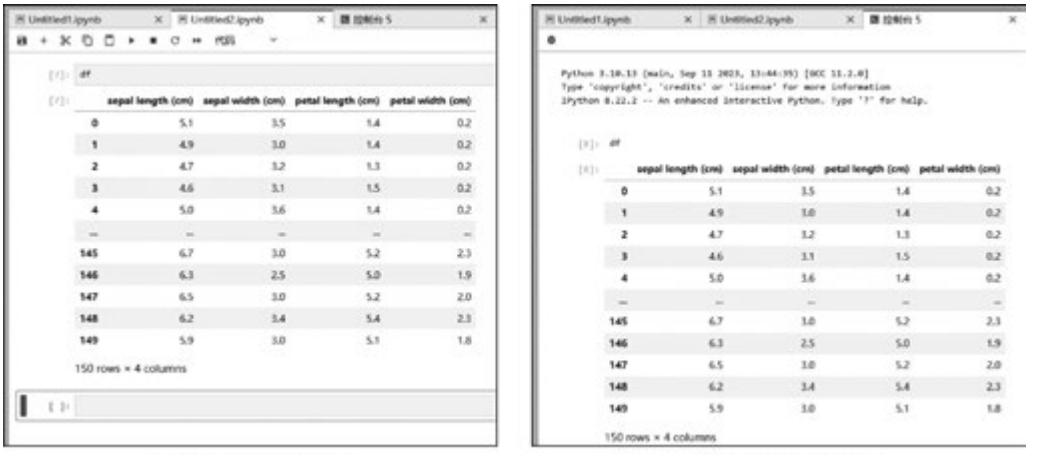
图 3-19　JupterLab 中的新功能



(a) 右击内核　(b) 新建的笔记本和控制台

图 3-20　基于内核新建笔记本或控制台

在新的笔记本或控制台中保留了该内核中存储的所有变量及该内核的运行状态，可以直接访问在原始笔记本中创建的变量，如图 3-21 所示。先前在 Untitled1.ipynb 笔记本中



(a) 同内核下的笔记本　(b) 同内核下的终端

图 3-21　共享内核的笔记本和终端

将鸢尾花数据集存储到了变量 df 中，此时可以在基于同一内核新建的 Untitled2.ipynb 笔记本和控制台中直接对 df 进行访问。

在目录管理功能区可以看到当前激活标签页中笔记本的目录结构，单击相应的标题即可导航至笔记本中对应的区域。

在插件管理功能区可以看到当前已安装的插件及可供安装的插件。这些插件由第三方开发者开发并上传至 Jupyter 项目中。

3.1.2 Colab 的使用方法

谷歌提供的 Colab 是一款方便且免费的云端交互式计算环境，在学术、教学和工程实践中被广泛使用。本节介绍 Colab 的基本使用方法，以及如何利用 Colab 中的免费计算资源。

首先打开浏览器，访问 Colab 的官方网站。打开 Colab 官网后，找到右上角的登录按钮，使用谷歌账号进行登录。若没有账号，则需先注册，可以使用国内的邮箱进行注册。完成登录后，网页中会出现“打开笔记本”对话框，如图 3-22 所示。

图 3-22 登录后的谷歌 Colab 官网页面

1. 新建笔记本

单击左下角的“新建笔记本”按钮，此时会新建并打开一个新的笔记本，如图 3-23 所示。该操作与在本地部署的 Jupyter 中新建笔记本是类似的，主要区别在于 Colab 中创建的笔记本(.ipynb 文件)存储在谷歌云端硬盘(Google Drive)中，而不是本地。

2. 在 Colab 中插入代码和命令

在 Colab 笔记本中编写和运行代码的方法与本地 Jupyter 笔记本大同小异。在插入单元格的操作上，Colab 笔记本可以在左上角的菜单栏中选择插入代码或文本单元格，从而在当前单元格下方插入新的单元格。另外，也可以将鼠标移至已有单元格的上方或下方，单击出现的悬停按钮进行插入，如图 3-24 所示。

图 3-23　在谷歌 Colab 中新建的笔记本

(a) 利用菜单栏按钮在当前单元格下方插入

(b) 利用悬停按钮在当前单元格下方插入

(c) 利用悬停按钮在当前单元格上方插入

图 3-24　在 Colab 笔记本中新建单元格的方法

注意：在 Colab 笔记本中需要在插入单元格时确定单元格是代码块还是文档块，后续无法对现存单元格的功能进行修改，只能插入新的单元格和删除不需要的单元格。

3. Colab 运行环境

Colab 在运行代码之前会创建运行环境(Runtime)，或译作“运行时”。Colab 允许用户连接到不同类型的运行环境。在菜单栏中单击“代码执行程序”下的“更改运行时类型”会弹出“更改运行时类型”对话框，如图 3-25 所示，其中“运行时类型”支持 Python 3 和 R 语言，“硬件加速器”支持选择 GPU 和 TPU 等免费计算资源及其他付费计算资源。

(a) 更改运行环境菜单　　(b) 可选的运行环境

图 3-25　在 Colab 中更改运行环境

单击右上角菜单栏中的“连接”即可将当前笔记本连接到指定的运行环境，连接成功后将显示计算资源的使用情况，如图 3-26 所示。事实上，用户不需要主动单击“连接”按钮，在

图 3-26　连接到指定运行环境

初次运行笔记本中的代码块时将自动连接到事先设定的运行环境。

Colab 中的运行环境更类似于虚拟机的概念，而非 Jupyter 笔记本中内核的概念。Jupyter 笔记本在运行代码之前会先创建一个内核，并在内核中运行代码，而 Colab 在运行代码前会先创建运行环境，然后在运行环境中创建会话(Session)，随后在会话中运行代码。在“代码执行程序”菜单中包括了针对运行环境和会话的操作及各种不同的代码运行模式，如图 3-27 所示，其中大部分功能与 Jupyter 笔记本中“运行”和“内核”菜单中的功能类似，在此不赘述。需要特别注意的是“重启会话”和“断开连接并删除运行时”，其中前者只重置内核，类似于 Jupyter 笔记本中的“重启内核”，后者则会将整个运行环境删除，下次运行代码时需要重新创建并连接到运行环境。

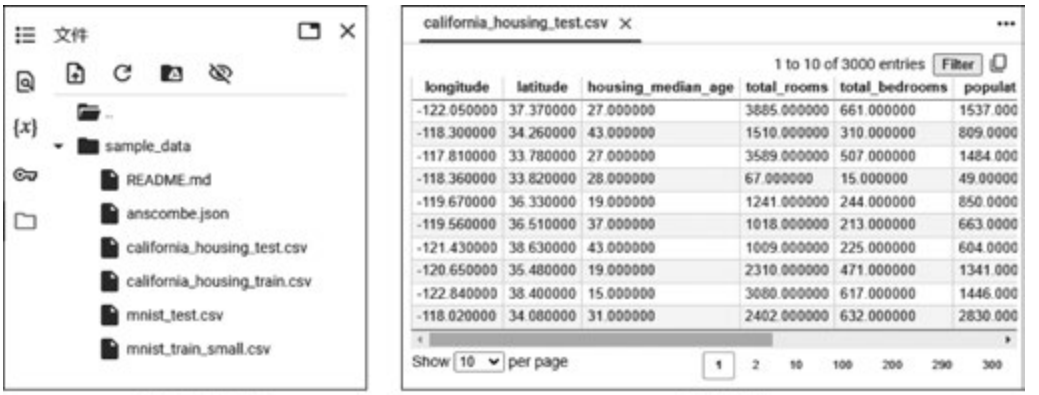
图 3-27　代码执行程序菜单选项

4. Colab 文件系统

与 JupyterLab 类似，在 Colab 中也可以对文件系统进行访问。Colab 中每个新建的运行环境都有一个临时的文件系统，其中名为 sample_data 的目录包含了一系列样例数据，如图 3-28(a)所示。双击数据文件即可在右侧进行预览，如图 3-28(b)所示。

注意：“断开连接并删除运行时”会导致临时文件系统中存储的所有文件丢失，“重启会话”不会导致文件丢失。然而，长时间与运行环境失去连接也会导致运行环境被删除，因此如果在临时文件系统中保存了文件，则应及时右击文件，将文件下载并保存到本地，以防止文件丢失。

如果需要永久保存文件，则可以在 Colab 中装载谷歌云端硬盘。一种方法是单击页面



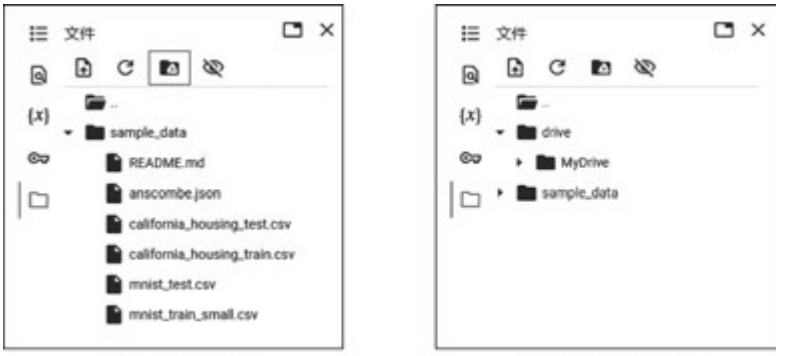
(a) 临时文件系统　　(b) 样例数据

图 3-28　Colab 中的临时文件系统

左侧临时文件系统中的“装载谷歌云端硬盘”按钮，如图 3-29(a)所示。按照指引进行操作后，Colab 笔记本中会出现一段用于装载云端硬盘的代码：

```
from google.colab import drive
drive.mount('/content/drive')
```

上述代码的含义是将用户的谷歌云端硬盘挂载到/content/drive 路径下。运行上述代码后，在文件系统中将出现一个名为 drive 的目录，其中包含一个名为 MyDrive 的子目录(用户的云端硬盘目录)，如图 3-29(b)所示。此后编写的代码可对/content/drive/MyDrive 目录中的文件进行永久性读写。



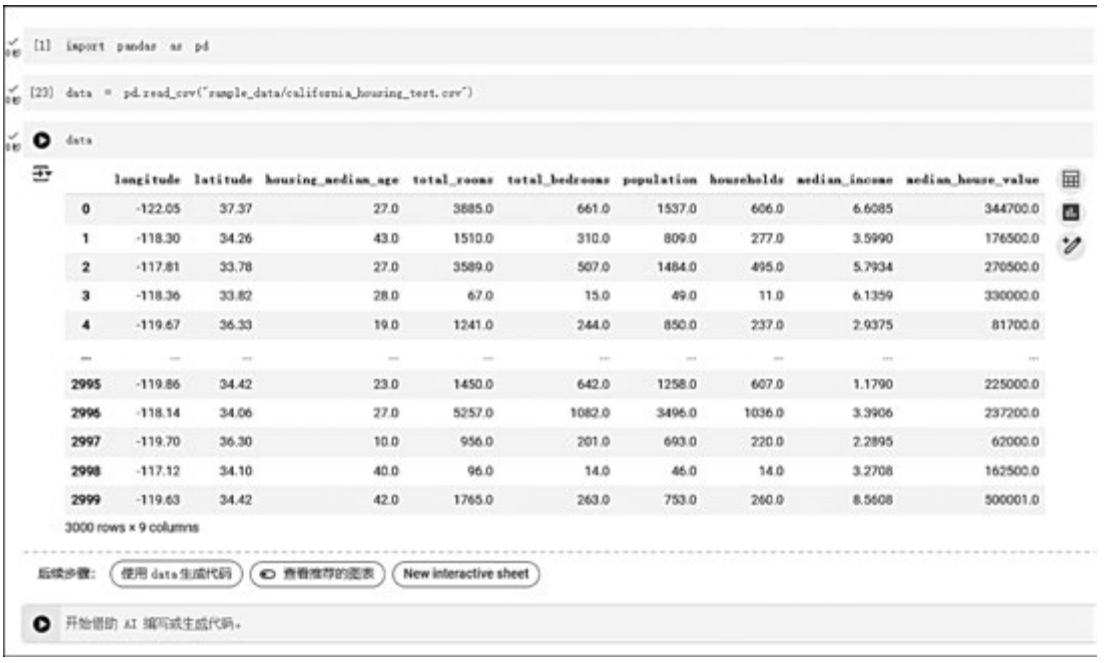
(a) 单击装载按钮　　(b) 装载成功示意图

图 3-29　装载谷歌云端硬盘

5. 利用 Colab 进行数据可视化

在 Colab 中编写代码和文档的方法与 Jupyter 笔记本中的方法类似。可以编写代码加载 Colab 中样例数据并对其进行可视化，如图 3-30 所示。

此外，Colab 中还提供了基于数据利用 AI 编写代码的功能，如图 3-31 所示。单击图中



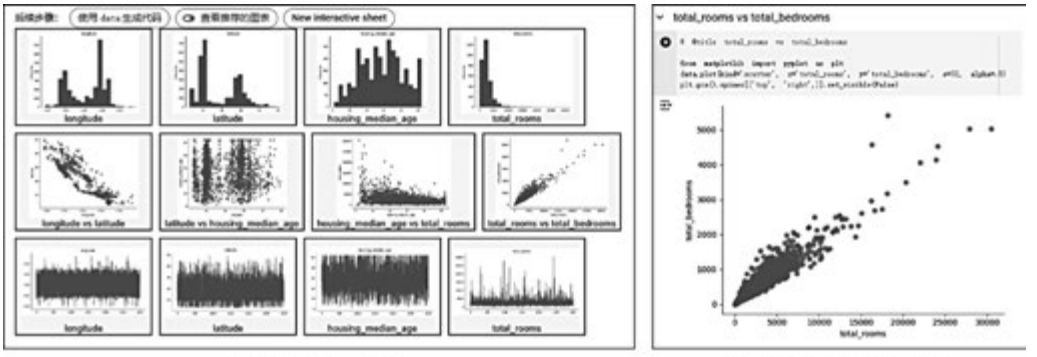
```
[1] import pandas as pd

[23] data = pd.read_csv("sample_data/california_housing_test.csv")

data
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-122.05	37.37	27.0	3885.0	661.0	1537.0	606.0	6.6085	344700.0
1	-118.30	34.26	43.0	1510.0	310.0	809.0	277.0	3.5990	176500.0
2	-117.81	33.78	27.0	3589.0	507.0	1484.0	495.0	5.7934	270500.0
3	-118.36	33.82	28.0	67.0	15.0	49.0	11.0	6.1359	330000.0
4	-119.67	36.33	19.0	1241.0	244.0	850.0	237.0	2.9375	81700.0
...	...	...	...	...	...	...	...	...	...
2995	-119.86	34.42	23.0	1450.0	642.0	1258.0	607.0	1.1790	225000.0
2996	-118.14	34.06	27.0	5257.0	1082.0	3496.0	1036.0	3.3906	237200.0
2997	-119.70	36.30	10.0	956.0	201.0	693.0	220.0	2.2895	62000.0
2998	-117.12	34.10	40.0	96.0	14.0	46.0	14.0	3.2708	162500.0
2999	-119.63	34.42	42.0	1765.0	263.0	753.0	260.0	8.5608	500001.0

3000 rows × 9 columns

图 3-30　Colab 中的数据可视化

数据表格下方的“查看推荐的图表”之后，Colab 首先将向用户展示一系列可能基于数据生成的图表，如图 3-31(a)所示。用户选择其中某一项可能的方案后，将在下方单元格中生成对应的绘图代码，运行即可得到图表，如图 3-31(b)所示。这里选择的是房产数据中房间总数量与卧室总数量之间关系的散点图。



(a) 可能的图表选项　　(b) 图表绘制代码和结果

图 3-31　Colab 利用 AI 生成数据图表

此外，在 Colab 中还可以完全使用 AI 来编写代码。在空白的代码单元格中单击“生成”后，将会弹出一个输入框，可以在此输入框中用自然语言描述自己的需求，如图 3-32 所示。

(a) 单击“生成”按钮

(b) 输入需求

图 3-32　Colab 利用 AI 根据需求生成代码

注意：在使用 AI 生成的代码之前务必对其进行检查并获得相关的许可。

3.1.3　在 Kaggle 中使用笔记本

首先打开 Kaggle 的官方网站，进行注册并登录，然后选择左侧菜单栏中的 Code 项，并在右侧单击 New Notebook 按钮。此时会跳转到新建的笔记本中，如图 3-33 所示。

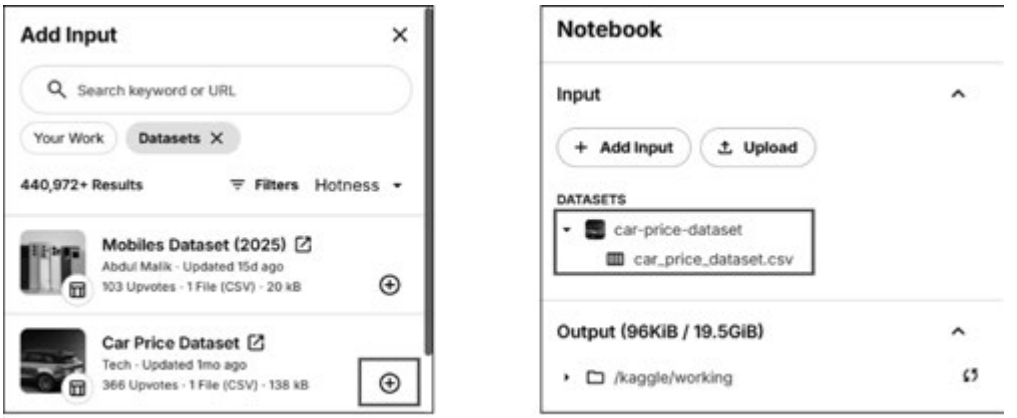
图 3-33　在 Kaggle 中新建笔记本

笔记本页面右侧是笔记本的输入(Input)和输出(Output)，其中输入中包含了笔记本代码可读取的文件，输出则包含了笔记本代码向文件系统中写入的文件。

可以直接添加 Kaggle 平台上的公开竞赛数据集或模型(单击 Add Input 按钮)，也可以由用户上传(单击 Upload 按钮)。

以添加 Kaggle 平台上的数据集为例，首先单击 Add Input 按钮，单击 Datasets 标签筛选出数据集，然后选择一个数据集，单击其右下角的加号按钮将其添加到当前笔记本的输入，如图 3-34(a)所示。

接下来，单击右上角的叉号关闭 Add Input 页面，可以看到，笔记本的输入中多出了刚才添加的数据集，如图 3-34(b)所示，其中的.csv 文件就是需要读取的数据文件。



(a) 添加数据集　　(b) 添加成功示意图

图 3-34　在 Kaggle 笔记本中添加输入

将鼠标置于.csv 文件上，文件名右侧会出现一个复制按钮。单击该按钮可以复制该文件的路径，随后粘贴到笔记本代码中即可完成对文件的读取，如图 3-35 所示。

```
[11]  df = pd.read_csv("/kaggle/input/car-price-dataset/car_price_dataset.csv")

[12]  df
```

[12]:

	Brand	Model	Year	Engine_Size	Fuel_Type	Transmission	Mileage	Doors	Owner_Count	Price
0	Kia	Rio	2020	4.2	Diesel	Manual	289944	3	5	8501
1	Chevrolet	Malibu	2012	2.0	Hybrid	Automatic	5356	2	3	12092
2	Mercedes	GLA	2020	4.2	Diesel	Automatic	231440	4	2	11171
3	Audi	Q5	2023	2.0	Electric	Manual	160971	2	1	11780
4	Volkswagen	Golf	2003	2.6	Hybrid	Semi-Automatic	286618	3	3	2867
...	...	...	...	...	...	...	...	...	...	...
9995	Kia	Optima	2004	3.7	Diesel	Semi-Automatic	5794	2	4	8884
9996	Chevrolet	Impala	2002	1.4	Electric	Automatic	168000	2	1	6240
9997	BMW	3 Series	2010	3.0	Petrol	Automatic	86664	5	1	9866
9998	Ford	Explorer	2002	1.4	Hybrid	Automatic	225772	4	1	4084
9999	Volkswagen	Tiguan	2001	2.1	Diesel	Manual	157882	3	3	3342

10000 rows × 10 columns

图 3-35　读取 Kaggle 输入中的数据文件

单击笔记本页面上方菜单栏中的 Settings 选项，可以选择 GPU 和 TPU 等计算资源对计算进行加速，如图 3-36 所示。

Kaggle 笔记本中代码的编写和运行及其他操作与 3.1.1 节中介绍的原生 Jupyter 笔记本及 3.1.2 节中介绍的 Colab 笔记本类似，在此不再赘述。

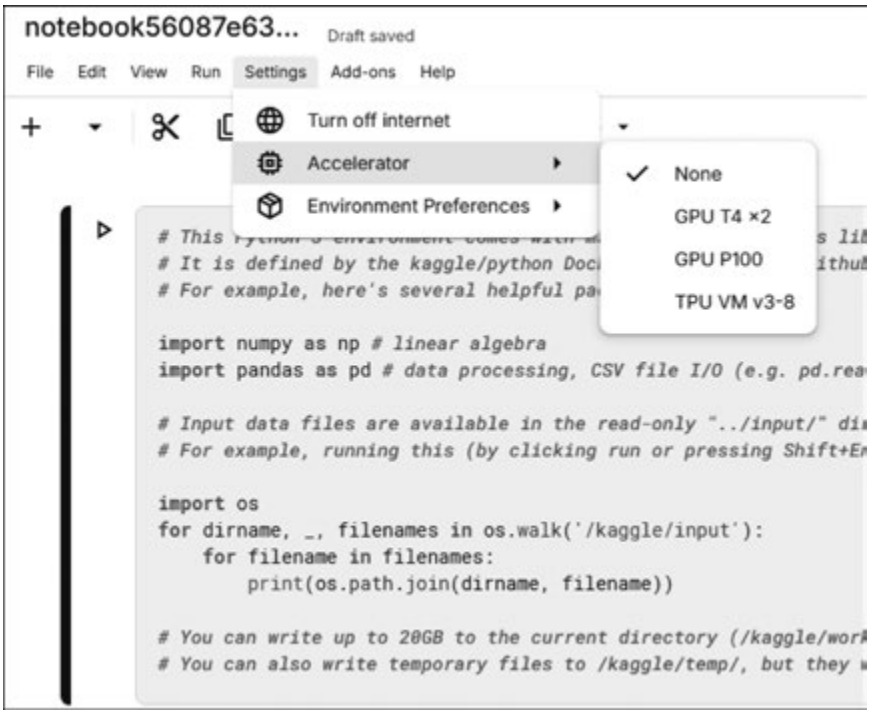


图 3-36　Kaggle 笔记本中的计算资源选项

3.2　在 Linux 计算机上搭建深度学习环境

云端交互式计算环境为用户提供了便捷的深度学习实践平台，适合初学者快速上手深度学习，以及进行简单的模型测试和验证，因为用户无须担心硬件资源的限制和环境搭建的复杂性。然而，在处理大规模数据和复杂模型时，可能会受到云端资源分配的限制。此外，对于一些对数据隐私和安全性要求较高的项目，将数据上传到云端可能存在风险。

在这些情况下，搭建自己的深度学习环境显得尤为重要。用户拥有自己的深度学习环境，可以根据项目的具体需求灵活配置硬件资源，如选择高性能的 GPU 以加速度计算，并且能够更好地控制数据的存储和处理，确保数据的安全性。同时，对于需要进行大量实验和调优的项目，自己的环境可以提供更稳定和更高效的运行条件。本节将以 Ubuntu 系统为例，详细介绍如何在 Linux 服务器上搭建基于 GPU 的深度学习环境。

注意：完成本节中的操作需要操作系统为 Linux 且具备 NVIDIA GPU 的计算机(PC 或服务器)，没有符合条件的计算机的读者可以阅读本节了解基本操作。另外，在 Windows 计算机中使用 Ubuntu 子系统的情况与直接使用 Ubuntu 计算机有所不同，将在 3.3 节中进行介绍。

3.2.1　安装 NVIDIA 显卡驱动

在 Linux 服务器上使用 GPU 进行深度学习，首先需要安装正确的 NVIDIA 显卡驱动。NVIDIA 显卡驱动是连接操作系统和 GPU 硬件的桥梁，它为 GPU 提供了必要的运行支持，使深度学习框架能够充分地利用 GPU 的强大计算能力。

为了安装 NVIDIA 显卡驱动，首先要添加官方驱动仓库，命令如下：

```
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt update
```

在上述命令中，sudo add-apt-repository ppa:graphics-drivers/ppa 用于添加官方的图形驱动软件源，sudo apt update 则用于更新软件源列表，确保系统能够获取最新的驱动软件包信息。

接下来，既可以选择自动安装推荐的驱动，也可以手动指定版本。自动安装推荐驱动的命令如下：

```
sudo ubuntu-drivers autoinstall
```

这条命令会根据系统的硬件配置，自动选择并安装最合适的 NVIDIA 显卡驱动版本。

安装完成后，需要重启计算机使驱动生效，命令如下：

```
sudo reboot
```

重启后，验证驱动是否安装成功，命令如下：

```
nvidia-smi
```

如果命令执行后输出了 GPU 的相关信息，如 GPU 型号、显存使用情况等，则说明驱动已经正确安装并可以正常工作了。

3.2.2　安装 CUDA

CUDA(Compute Unified Device Architecture)是 NVIDIA 推出的一种并行计算平台和编程模型，它允许开发者使用 C、C++等编程语言来编写在 GPU 上运行的程序，极大地提高了计算效率。在安装了 NVIDIA 显卡驱动后，需要安装 CUDA Toolkit 来为深度学习框架提供 GPU 计算支持。

CUDA Toolkit Archive 网站中提供了各种版本的 CUDA 安装包，如图 3-37 所示。选择安装包的版本时一般需要考虑 GPU 的兼容性和深度学习框架的兼容性。

GPU 的兼容性是指需要考虑特定型号的 GPU 所能兼容的最低版本的 CUDA。可以通过 nvidia-smi 命令查看 GPU 型号，然后在 https://developer.nvidia.com/cuda-gpus#compute 网站中查询该型号的 GPU 所支持的最低版本的 CUDA。以 GeForce RTX 4080

图 3-37 CUDA Toolkit Archive 网站

型号的 GPU 为例，首先在网页中找到 GeForce 系列产品的按钮，如图 3-38(a)所示。单击该按钮即可展开所有 GeForce 系列 GPU 的 CUDA 兼容性，如图 3-38(b)所示。可以看到，GeForce RTX 4080 型号的 GPU 所支持的最低 CUDA 版本为 8.9。

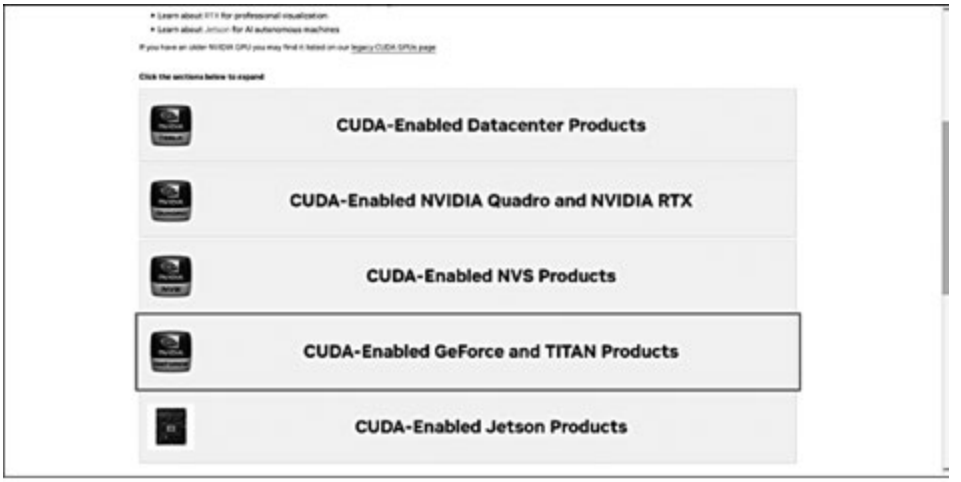
深度学习框架的兼容性是指 PyTorch 等深度学习框架所支持的 CUDA 版本。一般来讲，不同版本的深度学习框架需要使用不同版本的 CUDA。本书主要使用 PyTorch 框架，PyTorch 与 CUDA 的兼容性可参考文档 https://github.com/pytorch/pytorch/blob/main/RELEASE.md。

注意：PyTorch 与 CUDA 的兼容性并非向下兼容，而是只能兼容特定版本，例如，兼容性文档中显示，PyTorch 2.3 支持 CUDA 11.8，这说明 PyTorch 2.3 仅能正式支持 CUDA 11.8，对更低版本的 CUDA(如 11.7 或更早)不提供兼容性保证。

在 CUDA Toolkit Archive 网站中找到合适的 CUDA 版本后(以 CUDA 11.8 为例)，单击对应版本的链接后会跳转到下载页面，在这里需要按照网页上的指引选择正确的安装包，如图 3-39 所示。选项包括操作系统(Operating System)、系统架构(Architecture)、系统发行版(Distribution)、系统版本(Version)及安装器类型(Installer Type)。

由于本节的示例是在 Ubuntu 中进行安装，因此这里选择的是 Linux 操作系统、x86_64 架构、Ubuntu 20.04 发行版(根据实际情况调整系统版本)。安装器类型有 3 种选择，其中 deb (local) 和 deb (network) 是安装程序，runfile (local) 是安装脚本。选择不同的类型之后，页面下方会出现对应的安装命令。这里推荐使用 runfile (local) 进行安装。

选择 runfile (local) 后，可以在网站中看到以下命令。



(a) 根据GPU型号找到所属产品系列

GeForce and TITAN Products

GPU	Compute Capability
GeForce RTX 5090	10.0
GeForce RTX 5080	10.0
GeForce RTX 4090	8.9
GeForce RTX 4080	8.9
GeForce RTX 4070 Ti	8.9
GeForce RTX 4060 Ti	8.9
GeForce RTX 3090 Ti	8.6
GeForce RTX 3090	8.6
GeForce RTX 3080 Ti	8.6
GeForce RTX 3080	8.6
GeForce RTX 3070 Ti	8.6
GeForce RTX 3070	8.6
GeForce RTX 3060 Ti	8.6
Geforce RTX 3060	8.6
GeForce GTX 1650 Ti	7.5
NVIDIA TITAN RTX	7.5
Geforce RTX 2080 Ti	7.5
Geforce RTX 2080	7.5
Geforce RTX 2070	7.5
Geforce RTX 2060	7.5
NVIDIA TITAN V	7.0

GeForce Notebook Products

GPU	Compute Capability
GeForce RTX 5090	10.0
GeForce RTX 5080	10.0
GeForce RTX 4090	8.9
GeForce RTX 4080	8.9
GeForce RTX 4070	8.9
GeForce RTX 4060	8.9
GeForce RTX 4050	8.9
GeForce RTX 3080 Ti	8.6
GeForce RTX 3080	8.6
GeForce RTX 3070 Ti	8.6
GeForce RTX 3070	8.6
GeForce RTX 3060 Ti	8.6
GeForce RTX 3060	8.6
GeForce RTX 3050 Ti	8.6
GeForce RTX 3050	8.6
Geforce RTX 2080	7.5
Geforce RTX 2070	7.5
Geforce RTX 2060	7.5
GeForce GTX 1080	6.1
GeForce GTX 1070	6.1
GeForce GTX 1060	6.1

(b) 不同型号GPU的兼容性

图 3-38　查询 GPU 对 CUDA 的兼容性

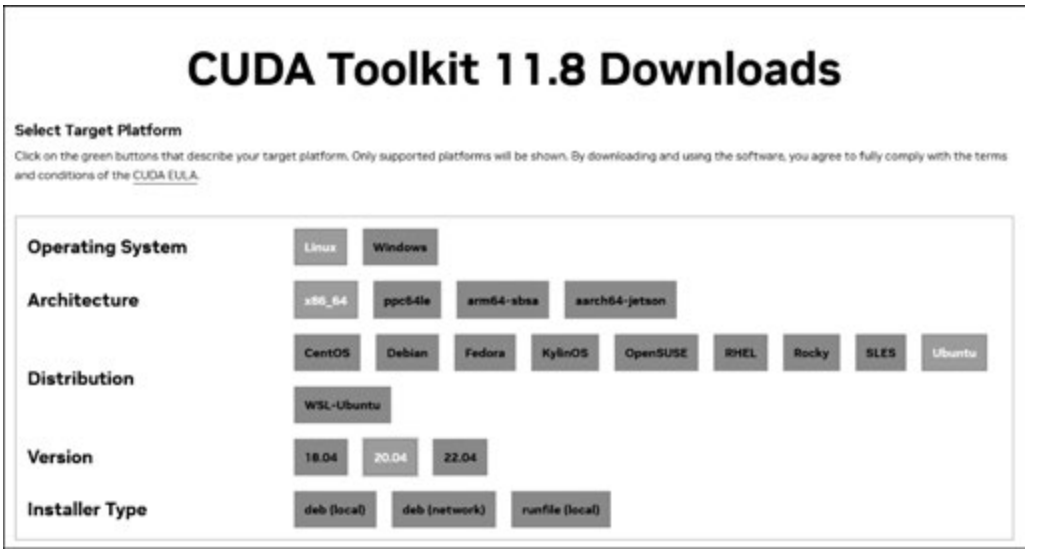


图 3-39 CUDA 安装包下载指引

```
wgethttps://developer.download.nvidia.com/compute/cuda/11.8.0/local_installers/
cuda_11.8.0_520.61.05_linux.run
sudosh cuda_11.8.0_520.61.05_linux.run
```

在 Ubuntu 系统终端(或远程连接的 SSH 终端)中运行上述命令即可下载并安装适用于 Ubuntu 20.04 操作系统的 CUDA 11.8。

在安装脚本运行的过程中,需要注意取消勾选驱动选项,因为在 3.2.1 节中已经单独安装了显卡驱动,避免重复安装导致冲突。另外,需要留意安装路径,以便后续配置环境变量。如果没有 sudo 权限,或者只想为特定的用户安装 CUDA,则可以不使用 sudo 操作而直接运行安装脚本,并将安装路径设置在用户目录下。

安装完成后,还需要配置环境变量,使系统能够找到 CUDA 的相关文件和库。将 CUDA 的路径添加到环境变量中,命令如下:

```
echo 'export PATH=/usr/local/cuda-11.8/bin:$ PATH' >> ~/.bashrc
echo 'export LD_LIBRARY_PATH=/usr/local/cuda-11.8/lib64:$ LD_LIBRARY_PATH' >>
~/.bashrc
source ~/.bashrc
```

在上述命令中,echo 命令用于将环境变量的设置写入~/.bashrc 文件中,source ~/.bashrc 则可以使环境变量的设置立即生效。

上述添加环境变量的命令仅适用于将 CUDA 安装到/usr/local/cuda-11.8/路径下的情况,如果 CUDA 被安装到了其他路径,则应将在上述命令中的/usr/local/cuda-11.8/替换为对应的路径。如果忘记 CUDA 被安装到了哪个路径,则可以使用命令进行查找,命令如下:

```
find / -type d -name cuda 2>/dev/null
```

3.2.3　安装 cuDNN

NVIDIA 开发的 cuDNN(CUDA Deep Neural Network Library)是一个针对深度神经网络的 GPU 加速库,它提供了高度优化的神经网络层实现,如卷积层、池化层等,能够显著地提高深度学习模型的训练和推理速度。

要安装 cuDNN,首先需要下载 cuDNN 的安装包。注意,cuDNN 的版本需要与 CUDA 的版本相匹配。建议前往 cuDNN Archive 网站 https://developer.nvidia.com/rdp/cudnn-archive 下载 cuDNN 的安装包,该网站中明确了每个 cuDNN 版本所支持的 CUDA 版本,如图 3-40 所示。通常建议下载支持对应 CUDA 的最新版本的 cuDNN。本节中安装的是 CUDA 11.8,因此应该选择图 3-40 中的第 2 个版本。

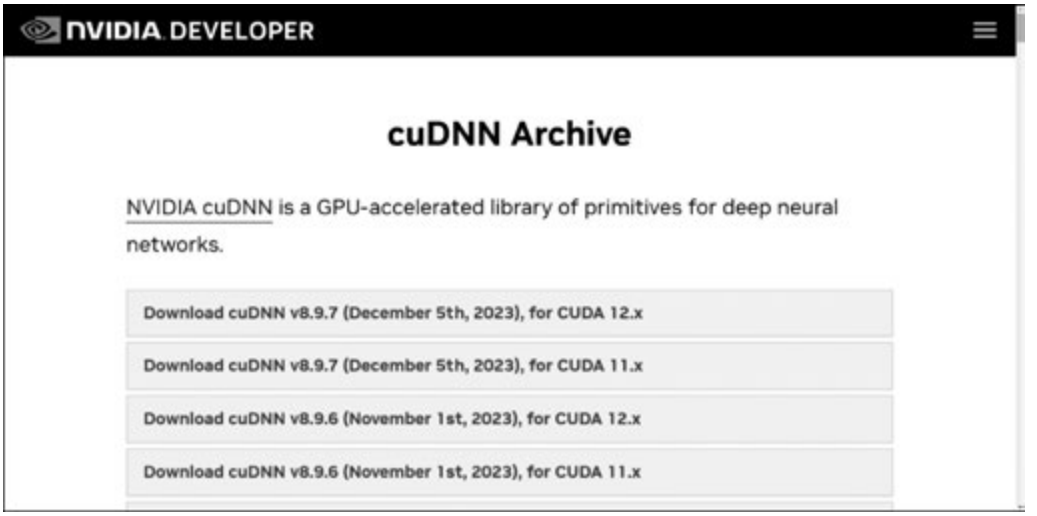


图 3-40　在 cuDNN Archive 中预览安装包

选择 cuDNN 版本后,将会弹出一系列适用于不同操作系统的安装包,如图 3-41 所示。对于 Ubuntu 系统而言,可以选择 Tar 或 Deb 安装包,这里以 Tar 安装包为例,即图 3-41 方框中的链接。

单击下载安装包后会弹出登录 NVIDIA 账号的提示,按照 NVIDIA 的流程进行注册和登录后方可完成下载。

下载完成后,将 Tar 安装包解压,命令如下:

```
tar -xvf cuDNN-linux-x86_64-8.9.3.28_cuda11-archive.tar.xz
```

然后将解压后的头文件和库文件复制到 CUDA 的安装目录中(注意根据实际情况替换 CUDA 的安装路径),命令如下:

图 3-41 适用于不同操作系统的安装包

```
sudo cp cuDNN-*-archive/include/cudnn*.h /usr/local/cuda-11.8/include/
sudo cp -P cuDNN-*-archive/lib/libcudnn*/usr/local/cuda-11.8/lib64/
```

最后为复制后的头文件和库文件赋予适当的权限：

```
sudochmoda+r /usr/local/cuda/include/cudnn*.h/usr/local/cuda/lib64/libcudnn*
```

至此，cuDNN 就安装完成了，它可以与 CUDA 和深度学习框架协同工作，加速深度学习任务的执行。

3.2.4 安装 Miniconda

3.1.1 节介绍了 Miniconda 并讲述了如何在 Windows 系统中安装 Miniconda。在 Ubuntu 中，只需在 Miniconda 官方网站下载适用于 Linux 系统的安装包并进行安装。与 3.1.1 节中相同，此处仍然安装 Miniconda3-py310_24.1.2-0 版本，在 Ubuntu 中可以直接使用的下载命令如下：

```
wgethttps://mirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/Miniconda3-py310_
24.1.2-0-Linux-x86_64.sh
```

下载完成后安装脚本，命令如下：

```
bash Miniconda3-py310_24.1.2-0-Linux-x86_64.sh
```

按照安装提示完成安装，安装完成后，初始化 Miniconda，命令如下：

```
source ~/.bashrc
```

接下来，创建一个新的 Python 虚拟环境，这里创建一个名为 dl 的环境，并安装 Python

3.10 版本，命令如下：

```
conda create -n dl python=3.10 -y
```

创建完成后，激活该环境，命令如下：

```
conda activate dl
```

至此已经进入名为 dl 的 Python 虚拟环境，接下来可以在这个环境中安装深度学习框架和相关的库。

3.2.5　安装 PyTorch

PyTorch 是一个基于 Python 的开源深度学习框架[22]，以其简洁且直观的 API 和强大的动态计算图特性而在学术界和工业界广泛地备受青睐。PyTorch 提供了灵活的张量操作和自动求导功能，使开发者能够方便快捷地构建和训练各种深度学习模型。

在安装 PyTorch 时，需要确保其版本与已安装的 CUDA 版本相匹配。这是因为不同版本的 PyTorch 对 CUD 的支持存在差异，如果版本不匹配，则可能会导致无法充分利用 GPU 的计算能力，甚至出现运行错误。本节中安装的 CUDA 11.8 通常可以较好地搭配 PyTorch 的一些较新稳定版本，如 PyTorch 2.0 及以上部分版本。具体版本匹配可参考 PyTorch 官方文档 https://github.com/pytorch/pytorch/blob/main/RELEASE.md。

PyTorch 官方网站首页提供了最新稳定版本的安装指引，如图 3-42 所示。这里依次选择 2.6.0 稳定版→Linux 系统→pip 命令安装→Python 语言和 CUDA 11.8，最终可以得到安装命令如下：

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

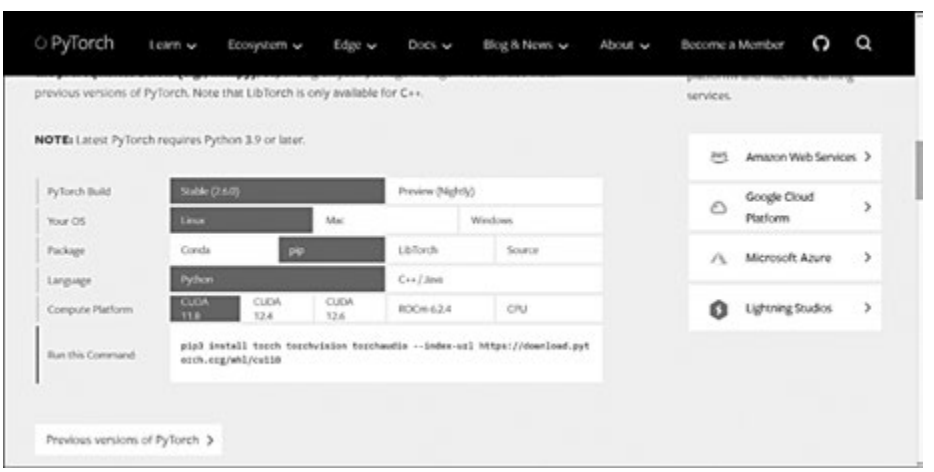


图 3-42　PyTorch 官网首页安装指引

在激活的 Python 虚拟环境中运行上述安装命令后，将使用 pip 安装相应版本的 PyTorch。安装完毕后，需验证是否能在 PyTorch 中使用 GPU。可以运行如下 Python 代码进行验证：

```
import torch
print(torch.cuda.is_available())   #输出 True 表示 GPU 可用，安装成功
```

至此便在 Linux 计算机上基本完成了深度学习环境的搭建。

3.3 在 Windows 计算机上搭建深度学习环境

直接在 Windows 系统下搭建深度学习环境，面临着复杂的依赖管理难题。深度学习涉及众多框架和库，如 PyTorch、TensorFlow 等，它们对不同版本的 Python、CUDA、cuDNN 等依赖关系复杂。Windows 系统下各软件包的版本适配较为烦琐，不同版本间的兼容性问题常常导致安装失败或运行出错。此外，Windows 上 GPU 驱动和相关加速库的安装配置也较为复杂，并且稳定性欠佳，容易出现各种不可预见的问题，给开发者带来诸多困扰。本节将介绍在 Windows 系统中安装 WSL2(Windows Subsystem for Linux 2)和 Ubuntu 子系统，并在子系统中配置深度学习环境。

注意：完成本节中的操作需要操作系统为 Windows 10 21H2 以上版本或 Windows 11 且具备 NVIDIA GPU 的计算机(PC 或服务器)。

3.3.1 安装 WSL2

相比 Windows 系统本身，WSL2 提供了一个更适合搭建深度学习环境的平台。WSL2 允许在 Windows 系统中运行一个完整的 Ubuntu 子系统，它继承了 Linux 系统在软件包管理和依赖处理方面的优势，能简化深度学习相关软件的安装和配置过程。同时，WSL2 能够高效地利用 Windows 的硬件资源，并且实现了对 GPU 的支持，使深度学习任务在该环境下能够顺利执行，因此在 Windows 系统中，先安装 Ubuntu 子系统再搭建深度学习环境成为许多开发者的选择。

在安装 Ubuntu 子系统之前需要启用一些 Windows 功能。在“控制面板→程序”中找到“启用或关闭 Windows 功能”，勾选“Hyper-V”“适用于 Linux 的 Windows 子系统”“虚拟机平台”三项。

接下来按快捷键 Win＋X，单击菜单中的 Windows PowerShell，以管理员权限运行 PowerShell。在 PowerShell 中运行以下命令对 WSL 进行版本更新：

```
#以管理员权限运行 PowerShell
wsl --update
```

接下来通过以下命令安装 Ubuntu-20.04 子系统：

```
wsl --set-default-version 2          #将 WSL2 设置为默认版本
wsl --install -d Ubuntu-20.04
```

安装完成后，在 PowerShell(或 CMD 等其他终端)中输入 wsl 并运行即可启动 Ubuntu 子系统。首次启动时，系统会提示设置用户名和密码，按照指引设置即可。

3.3.2　安装 NVIDIA 显卡驱动

在 Windows 的 Ubuntu 子系统中搭建 GPU 计算环境同样需要安装 NVIDIA 显卡驱动，可前往 NVIDIA GeForce 驱动程序网站 https://www.nvidia.cn/geforce/drivers/下载。在该网站中可以按照要求手动搜索驱动程序，如图 3-43 所示。需要明确自己的 PC 或服务器的显卡型号后搜索对应的驱动。在“产品系列”中，带 Notebooks 后缀的显卡为适用于笔记本电脑的显卡，需要注意区分。“下载类型”应选择“Game Ready 驱动程序”。

图 3-43　手动搜索显卡驱动程序

下载完成后，按照指引进行安装即可。安装完成后，可在终端中运行 nvidia-smi 命令查看显卡的基本信息，如图 3-44 所示。

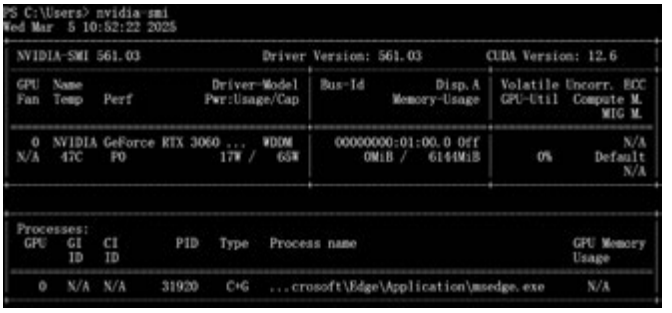


图 3-44　运行 nvidia-smi 命令查看显卡基本信息

3.3.3 安装 CUDA 和 cuDNN

安装 CUDA 和 cuDNN 的步骤与 3.2 节中在 Linux 计算机中的安装过程类似。安装 CUDA 时，首先仍然需要前往 CUDA Toolkit Archive 网站 https://developer.nvidia.com/cuda-toolkit-archive 选择正确的 CUDA 版本，这里仍然以 CUDA 11.8 为例，然后在安装包下载指引界面仍然选择 Linux 操作系统，而不是 Windows 操作系统，在选择发行版时选择 WSL-Ubuntu，而不是 Ubuntu，如图 3-45 所示。选择 runfile (local) 安装脚本后，网站给出的安装命令如下：

```
wgethttps://developer.download.nvidia.com/compute/cuda/11.8.0/local_installers/
cuda_11.8.0_520.61.05_linux.run
sudosh cuda_11.8.0_520.61.05_linux.run
```

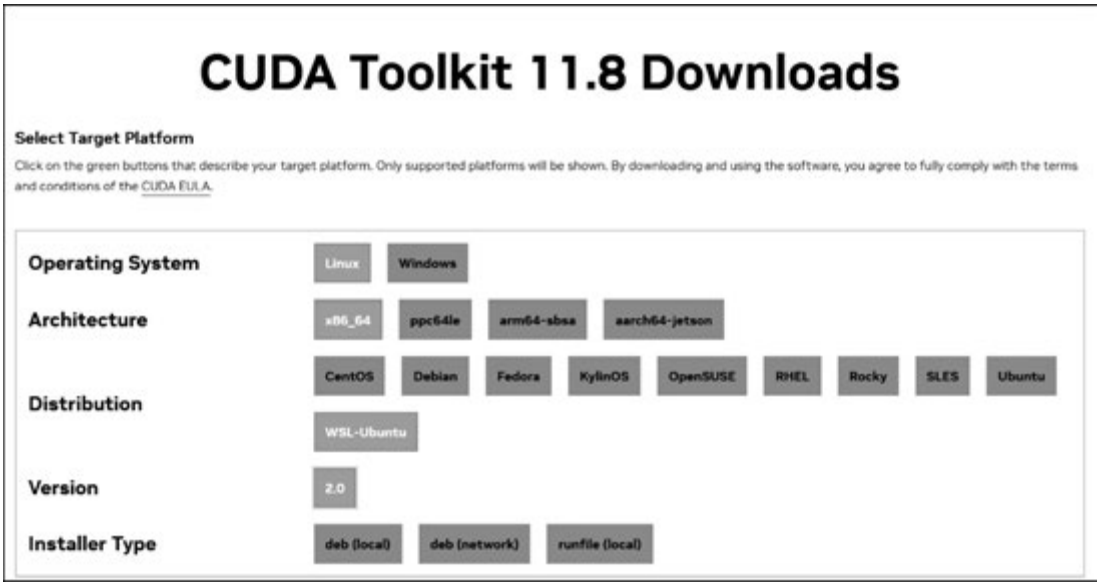


图 3-45 选择 WSL-Ubuntu

安装完成后，同样需要将 CUDA 环境变量添加到系统中，具体方法参见 3.2.2 节。

在 Ubuntu 子系统中 cuDNN 的安装与原生 Ubuntu 系统一致，只需下载对应 CUDA 版本下 cuDNN 的 Tar 安装包，解压并将头文件和库文件复制到 CUDA 安装目录，并不需要像 CUDA 一样区分原生 Ubuntu 和 WSL-Ubuntu。具体步骤不在此赘述，可参考 3.2.3 节。

在进行上述配置之后，在 Ubuntu 子系统中安装 Miniconda 和 PyTorch 的方法与常规 Linux 系统无异。可直接参考 3.2.4 节和 3.2.5 节中的做法。

3.4 本章小结

本章围绕深度学习环境搭建展开，全面地介绍了多种环境搭建方式，为深度学习的学习、研究和开发奠定基础。

在云端交互式计算环境方面，介绍了谷歌 Colab 和 Kaggle 这两个提供免费计算资源的平台。它们基于 Jupyter 搭建，用户能通过浏览器编写、运行代码，其中，详细地讲解了 Jupyter 在本地的部署过程，包括安装 Miniconda、启动 Jupyter 服务、新建笔记本、运行代码与命令、插入 Markdown、操作内核、选择运行模式、数据可视化及切换至 JupyterLab 等操作。同时，分别阐述了 Colab 和 Kaggle 的使用方法，如 Colab 中新建笔记本、插入代码、配置运行环境、访问文件系统和利用 AI 功能；Kaggle 中新建笔记本、添加输入数据、选择计算资源等，这些云端环境适合初学者快速上手和进行简单模型测试。

对于本地搭建深度学习环境，分别介绍了在 Linux 和 Windows 计算机上的搭建方法。在 Linux 计算机上，以 Ubuntu 系统为例，搭建基于 GPU 的深度学习环境需要依次安装 NVIDIA 显卡驱动、CUDA、cuDNN、Miniconda 和 PyTorch。安装过程中要注意各软件版本的兼容性，如 CUDA 与 GPU、深度学习框架的兼容，以及 cuDNN 与 CUDA 的匹配等。在 Windows 计算机上，由于直接搭建深度学习环境面临依赖管理难题，推荐安装 WSL2 和 Ubuntu 子系统。安装 WSL2 需先启用相关 Windows 功能，再更新 WSL 并安装 Ubuntu 20.04 子系统。之后，在子系统中安装 NVIDIA 显卡驱动、CUDA、cuDNN、Miniconda 和 PyTorch 的步骤与 Linux 系统类似，但安装 CUDA 时需注意选择 WSL-Ubuntu 发行版。

通过学习本章内容，读者能够根据自身需求和实际情况，选择合适的方式搭建深度学习环境，为后续深度学习的实践和探索做好充分准备。