

数学与深度学习基础



模块导读

在深度学习领域，模型训练的本质是参数优化的过程，而参数优化的核心依赖于梯度计算。**TensorFlow** 作为主流的深度学习框架，通过计算图和自动微分技术，极大地简化了梯度计算的复杂性。本模块将深入探讨 **TensorFlow** 的三种计算图构建方式（静态计算图、动态计算图、**Autograph**），并重点讲解如何利用 **tf.GradientTape** API 实现自动微分。通过本模块的学习，读者将掌握深度学习框架 **TensorFlow** 的核心计算逻辑，为后续模型开发与优化奠定坚实基础。



学习收获

- 了解 **TensorFlow** 计算图的构建方法，理解静态计算图、动态计算图与 **Autograph** 的区别与应用场景；
- 掌握自动微分技术，熟练使用 **tf.GradientTape** 对张量、变量及模型参数进行梯度计算；
- 理解梯度控制技巧，掌握梯度监视、中间结果梯度提取、高阶梯度计算等进阶操作。

任务 1.1 构建 TensorFlow 计算图

任务描述

本任务聚焦于 TensorFlow 中计算图的构建机制，重点介绍静态计算图、动态计算图与 Autograph 三种计算图构建方式的核心原理与使用方法。通过代码示例，读者将学习如何使用有向无环图（Directed Acyclic Graph, DAG）表达模型结构，理解图构建与执行的分离机制，并掌握 `@tf.function` 装饰器的使用技巧，实现动态代码的静态计算图转换，从而提升模型运行效率。



任务目标

- （1）理解三种计算图构建方式的核心原理与差异，能清晰区分静态计算图、动态计算图、Autograph 的构建逻辑与执行流程。
- （2）熟练使用 `@tf.function` 装饰器实现动态代码静态化，能将 Python 函数转化为 TensorFlow 静态计算图构建代码，并避免出现常见错误。
- （3）掌握计算图构建中的调试方法与错误处理技巧，能识别变量作用域冲突、图重复构建等问题并提出解决方案。
- （4）根据任务需求选择合适的计算图构建方式。



知识准备

1. 计算图的核心概念

一个机器学习任务的核心是模型的定义和参数求解，对这两者进行抽象后，可以确定一个唯一的计算逻辑。将这个逻辑用图表示，即计算图。计算图表现为有向无环图，它定义了数据的流转方式、计算方式，以及各种计算之间的相互依赖关系等。

计算图的类型一般包括以下三种。

（1）静态计算图：在计算之前需要先定义好计算图的结构，然后将数据输入计算图中进行计算。在静态计算图中，计算图的结构在运行前是不可变的，即使是数据的流动也需要通过特定的操作来实现。静态计算图的优点在于其高效性，这是因为计算图结构事先确定，可以进行一些静态优化（如剪枝和融合操作），从而提高计算效率。然而，静态计算图的缺点在于其灵活性较差，调试困难，且在模型理解和编程复杂性方面存在一定的挑战。

（2）动态计算图：在程序运行时动态构建计算图，其核心思想是“边执行



边运行”，即运算与计算图的构建同时进行。这种方式更加灵活，便于调试和调整模型结构。动态计算图的优点在于其灵活性和易用性，适合快速开发和调试，尤其是在处理动态神经网络架构，如循环神经网络（Recurrent Neural Network, RNN）时具有优势。然而，动态计算图的缺点在于其运行效率较低，因为每次执行都需要重新构建计算图。

（3）Autograph：兼顾易用性与性能，通过装饰器实现动态代码到静态计算图的转换。

2. Tensorflow 计算图

要实现自动微分，TensorFlow 需要记住在前向传递过程中哪些运算以何种顺序发生。然后，在后向传递期间，TensorFlow 以相反的顺序遍历此运算列表来计算梯度。TensorFlow 为自动微分提供了 `tf.GradientTape` API，即计算某个运算相对于某些输入（通常是 `tf.Variable`）的梯度。TensorFlow 会将在 `tf.GradientTape` 上下文管理器中执行的相关运算记录到 `tape` 上，然后使用该 `tape` 通过反向模式微分去计算一个被记录的计算过程的梯度。本任务将讲解 TensorFlow 的计算图和自动微分。

TensorFlow 有三种计算图的构建方式：静态计算图、动态计算图以及 Autograph。

（1）静态计算图：TensorFlow 1.x 采用的是静态计算图，需要先使用 TensorFlow 的各种算子创建计算图，然后开启一个会话（Session），显式执行计算图。

（2）动态计算图：TensorFlow 2.x 采用的是动态计算图，即每使用一个算子后，该算子会被动态加入隐含的默认计算图中，立即执行得到结果，而无须开启 Session。

（3）Autograph：如果需要在 TensorFlow 2.x 中使用静态计算图，可以使用 `@tf.function` 装饰器将普通 Python 函数转换成对应的 TensorFlow 计算图构建代码。运行该函数就相当于在 TensorFlow 1.x 中使用 Session 执行代码。使用 `@tf.function` 构建静态计算图的方式称作 Autograph。

3. 自动微分概念

自动微分（Automatic Differentiation, AD）是一种用于计算函数导数的数值方法，它结合了数值微分和符号微分的优点，能够高效、准确地计算函数的导数。而反向模式微分（Reverse Mode Differentiation）是自动微分的一种实现方式。梯度持久化（Gradient Persistence）是反向模式微分在实际应用中的一种优化策略。

（1）反向模式微分：TensorFlow 通过 `tf.GradientTape` 记录前向计算过程，反向遍历计算梯度。

（2）梯度持久化：在 TensorFlow 中，`persistent` 是一个布尔参数，用于控制梯



度计算的行为。使用 `persistent=True` 支持多次梯度计算，需手动释放资源。

4. Anaconda 简介

Anaconda 是一个流行的 **Python** 数据科学平台，集成了大量的科学计算和机器学习库，广泛应用于数据分析、机器学习和深度学习领域。它提供了便捷的环境管理功能，能够帮助开发者快速搭建和切换不同的开发环境。**Anaconda** 通过其包管理器 **Conda** 简化了包的安装和管理，并允许用户轻松创建隔离的 **Python** 环境，以避免依赖冲突。**Anaconda** 预装了大量常用的数据科学库，支持跨平台使用，并提供了图形界面和云服务，是数据科学家和研究人员进行科学计算和分析的强大工具。

1) Anaconda 的核心功能

(1) 集成环境：**Anaconda** 预装了数百个常用的数据科学库（如 **NumPy**、**Pandas**、**Matplotlib**、**Scikit-learn** 等），无须手动安装。

(2) 包管理：通过 **Conda** 工具，可以轻松安装、更新和管理 **Python** 包及其依赖。

(3) 环境管理：支持创建和管理多个独立的 **Python** 环境，避免不同项目之间的依赖冲突。

(4) 跨平台支持：支持 **Windows**、**macOS** 和 **Linux** 系统。

2) Anaconda 的组成部分

(1) **Anaconda Navigator**：一个图形化界面，方便用户管理环境、安装包和启动工具（如 **Jupyter Notebook**、**Spyder** 等）。

(2) **Conda**：一个命令行工具，用于包管理和环境管理。

(3) **Jupyter Notebook**：一个交互式编程环境，支持代码、文本、图像和可视化内容的混合展示。

(4) **Spyder**：一个专为科学计算设计的集成开发环境（**Integrated Development Environment, IDE**），类似于 **PyCharm**。

3) Conda 的概念

(1) **Conda** 是一个开源的软件包管理系统和环境管理系统，用于安装多个版本的软件包及其依赖关系，并在它们之间轻松切换。

(2) **Conda** 是为 **Python** 程序创建的，用于管理 **Python** 版本和 **Python** 环境。目前主流的 **Conda** 软件是 **Anaconda** 和 **Miniconda**。

5. Anaconda 与 Miniconda 区别

Anaconda 是一个开源的 **Python** 发行版本，它包含了 **Conda**、**Python**、**NumPy**、**Pandas** 等 180 多个科学包及其依赖项。**Miniconda** 是 **Anaconda** 的轻量级替代，默认只包含 **Python** 和 **Conda**，但是可以通过 **Pip** 和 **Conda** 指令来安装所需要的包。



任务实施



Anaconda 的安装

步骤 1 环境准备

- (1) 操作系统: Windows 10。
- (2) 已安装 Anaconda, 并创建了 Python 3.7.5 的虚拟环境, 虚拟环境名称 one_env。
- (3) 已安装 PyCharm, 并创建了项目, 项目配置了 one_env 虚拟环境中的 Python 解释器。
- (4) one_env 虚拟环境已安装 TensorFlow 2.x。

步骤 2 安装 Anaconda

访问 Anaconda 官网, 选择适合自己操作系统的安装包进行下载。双击下载的安装程序, 按照提示完成安装。在安装过程中, 建议将 Anaconda 添加到系统路径 (Path) 中, 以便在命令行中直接使用 Anaconda 命令。下面以安装 Anaconda3 (Python3.7 版本) 为例介绍安装流程。

- (1) 双击 Anaconda3-2022.10-Windows-x86_64.exe 安装文件, 弹出的安装界面如图 1.1 所示。

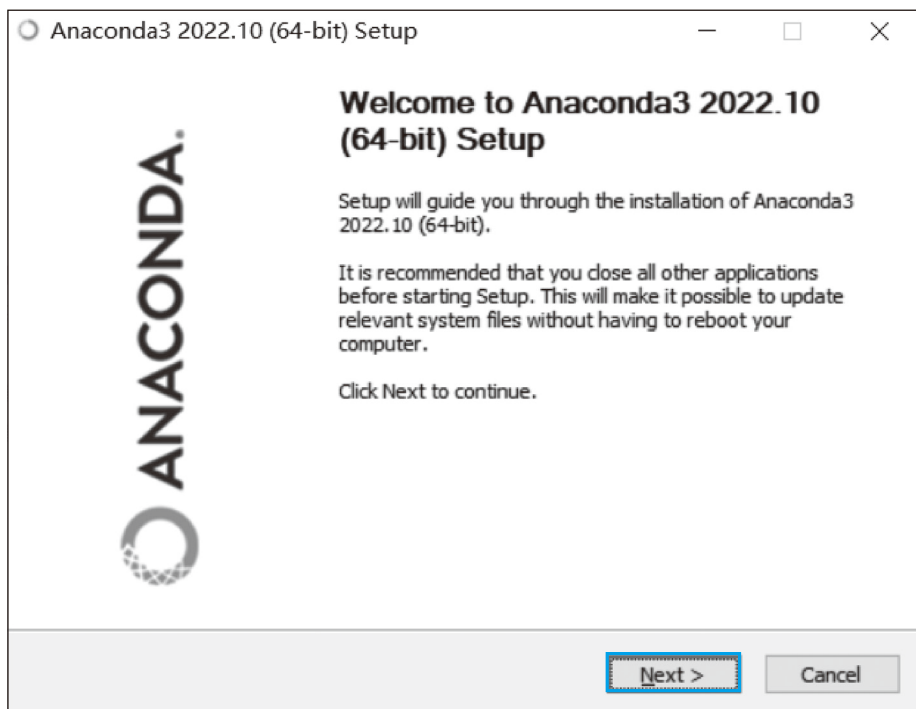


图 1.1 Anaconda3 安装界面

- (2) 单击 Next 按钮, 弹出 Anaconda3 安装许可界面, 如图 1.2 所示, 单击 I Agree 按钮。

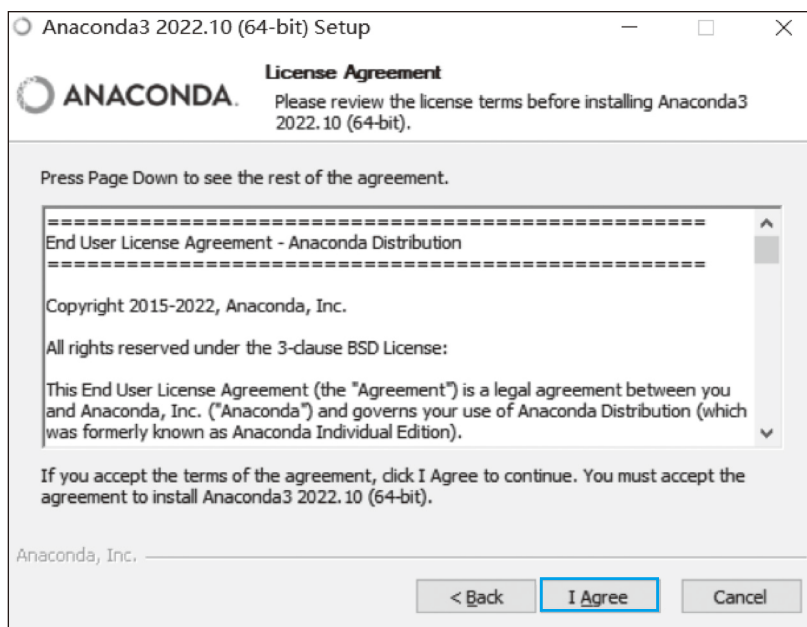


图 1.2 Anaconda3 安装许可界面

(3) 在弹出的“选择安装类型”的界面中，选择 All users 权限，单击 Next 按钮，如图 1.3 所示。

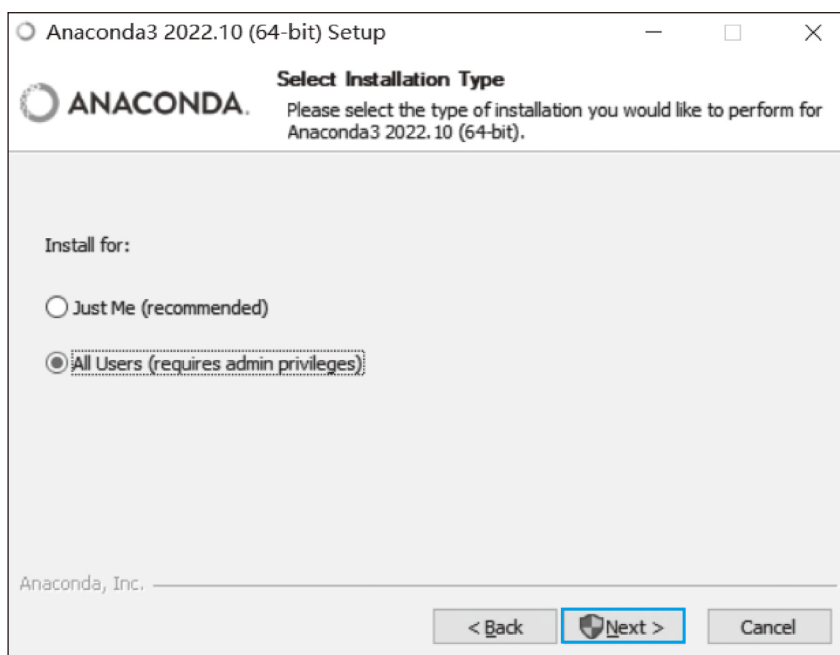


图 1.3 选择 All Users 权限

(4) 选择要安装的位置，如安装到 D:\ML_env\anaconda3 (图 1.4)，单击 Next 按钮。

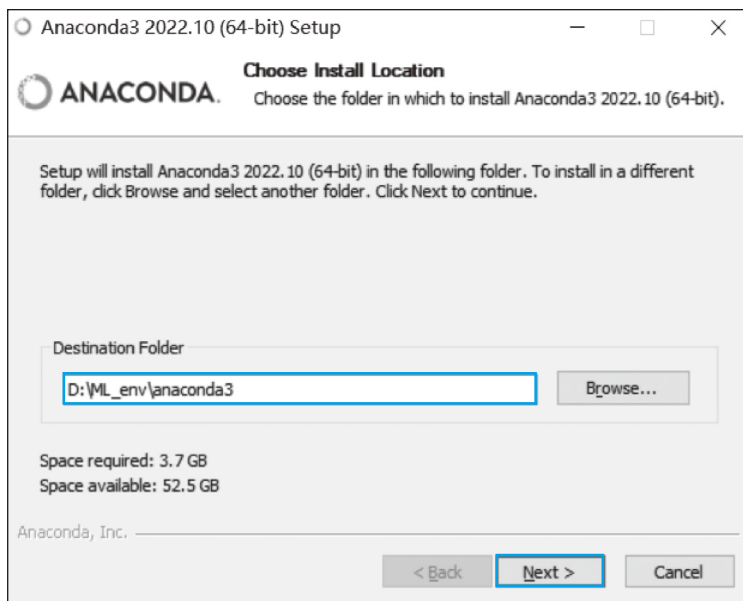


图 1.4 选择安装位置

⚠ 注意：安装位置尽量别选 C 盘，因为 C 盘通常是系统盘，安装了操作系统和其他关键系统文件。Anaconda 及其相关的数据科学工具（如 Jupyter Notebook、TensorFlow、PyTorch 等）会占用相当大的磁盘空间。选择 C 盘可能会导致磁盘空间不足，影响系统性能和稳定性。

（5）在弹出的界面中，勾选 Python 3.7 安装版本，然后单击 Install 按钮（图 1.5）。安装完成的界面如图 1.6 所示。

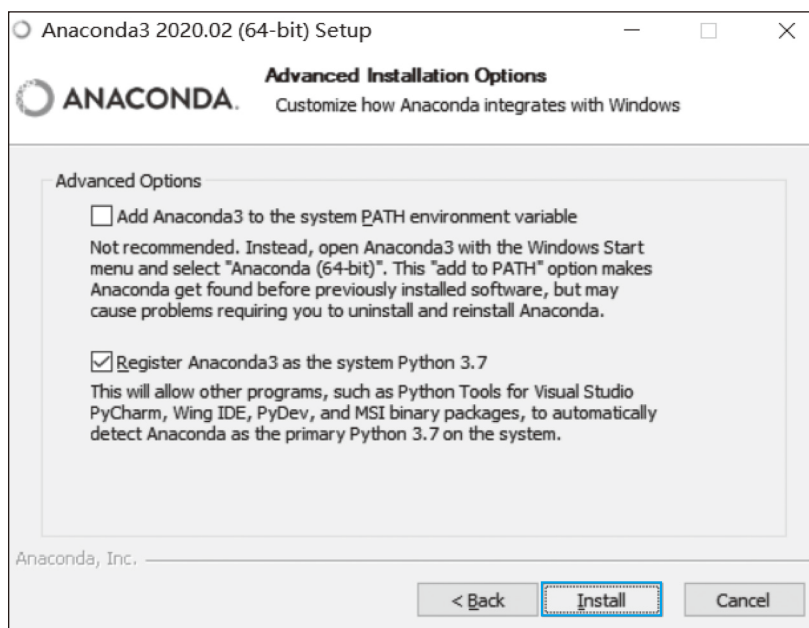


图 1.5 勾选 Python 3.7 安装版本

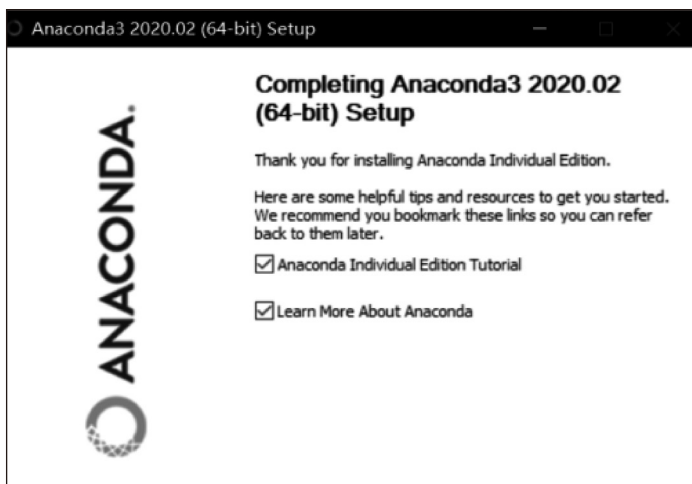


图 1.6 Anaconda 安装完成

步骤 3 Anaconda 环境变量配置

在安装 Anaconda 的过程中，建议将 Anaconda 添加到系统的环境变量中，否则需要手动配置环境变量。

(1) 确定 Anaconda 安装路径，如 Anaconda 的安装路径为 D:\DeepLearning\Anaconda3。读者也可根据自己需要对安装路径进行调整。

(2) 打开环境变量“设置”窗口。右击“此电脑”或“计算机”图标,单击“属性”标签。在弹出的“设置”窗口中，单击“高级系统设置”标签（图 1.7）。在弹出的“系统属性”窗口中，单击“环境变量”按钮（图 1.8）。



图 1.7 在“设置”窗口中单击“高级系统设置”标签



图 1.8 单击“环境变量”按钮

(3) 找到 Path 环境变量。在弹出的“环境变量”窗口中，找到“系统变量”区域中的 Path 变量，然后单击“编辑”按钮，如图 1.9 所示。

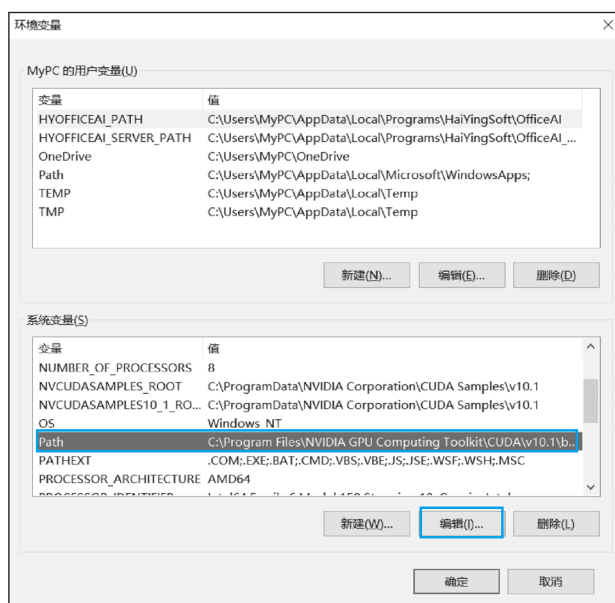


图 1.9 找到 Path 环境变量

(4) 在弹出的“编辑环境变量”窗口中，单击“新建”按钮，并依次添加以下路径（根据 Anaconda 安装路径调整）：

- D:\DeepLearning\Anaconda3 ;
- D:\DeepLearning\Anaconda3\Library\bin ;

- D:\DeepLearning\Anaconda3\Library\mingw-w64\bin ;
- D:\DeepLearning\Anaconda3\Scripts。

添加完成后，单击“确定”按钮保存设置，如图 1.10 所示。

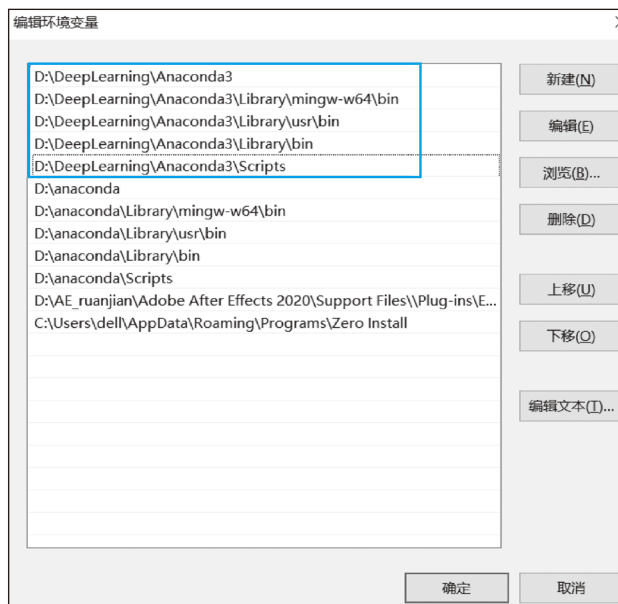


图 1.10 编辑和添加环境变量

配置完环境变量后，需要重新打开才能生效。

步骤 4 验证环境变量配置

打开一个新的命令提示符窗口或 PowerShell 窗口，输入以下命令验证环境变量是否配置成功：

```
conda --version
```

如果返回 Anaconda 的版本信息，则说明环境变量配置成功。此外，还可以通过以下命令验证 Python 是否可用：

```
python --version
```

步骤 5 执行 Tensorflow 计算图

1. 定义并执行静态计算图

通常通过两个主要步骤构建静态计算图：定义计算图和在执行会话中执行计算图。具体代码如下：

```
import tensorflow as tf
# 定义计算图
g = tf.Graph()
with g.as_default():
    # placeholder 为占位符，执行会话时指定填充对象
```