大模型应用开发三个月学习规划

一、学习路径设计

本学习路径将分阶段进行，涵盖 PyTorch 基础与进阶、大模型微调方法以及智能体开发等核心内容，每个阶段都有明确的学习重点和目标。

1、PyTorch 基础学习阶段

学习重点

1. 深入理解张量的概念，掌握张量的创建、索引、切片、运算等基本操作。例如，学会使用不同的数据类型（如浮点型、整型）创建张量，以及如何进行张量的加法、乘法等运算。
2. 掌握自动求导机制，理解其在深度学习中的重要性。明白如何通过自动求导计算梯度，为后续的模型训练打下基础。
3. 学习 PyTorch 的基本数据加载和处理方法，如使用 `DataLoader` 加载数据集，对数据进行预处理（如归一化、裁剪等）。

学习目标

能够独立完成简单的模型训练，例如使用 PyTorch 构建一个简单的全连接神经网络，对 MNIST 手写数字数据集进行分类任务。在这个过程中，学会定义模型结构、选择合适的损失函数和优化器，以及进行模型的训练和评估。

2、PyTorch 进阶学习阶段

学习重点

1. 深入学习各种深度学习模型的结构，如卷积神经网络（CNN）、循环神经网络（RNN）及其变体（LSTM、GRU）等。理解它们的原理、应用场景和优缺点。

2. 掌握模型的训练流程，包括如何选择合适的超参数（如学习率、批次大小、训练轮数等），如何进行模型的验证和调优。
3. 学习使用 PyTorch 的高级特性，如自定义层、自定义损失函数等，以满足不同的应用需求。

学习目标

能够构建和训练更复杂的模型，例如使用 CNN 对图像进行分类，使用 RNN 进行序列数据的预测。在训练过程中，能够根据模型的性能表现调整超参数，提高模型的准确率和泛化能力。

二、时间规划

1、第一阶段：PyTorch 基础学习阶段（第 1 - 4 周）

第 1 周

星期	学习时长	具体任务
周一	理论学习 2 - 3 小时	深入理解张量的概念，学习使用不同数据类型创建张量
周二	理论学习 2 - 3 小时	掌握张量的索引、切片操作
周三	理论学习 2 - 3 小时	学习张量的加法、乘法等运算
周四	代码实践 3 - 4 小时	编写代码实现张量的基本操作
周五	代码实践 3 - 4 小时	完成简单的张量运算任务
周六	复习和总结 2 - 3 小时	检查本周学习成果，总结张量操作知识
周日	休息	

第 2 周

星期	学习时长	具体任务
周一	理论学习 2 - 3 小时	掌握自动求导机制的原理
周二	理论学习 2 - 3 小时	学习如何通过自动求导计算梯度
周三	理论学习 2 - 3 小时	理解自动求导在深度学习中的重要性
周四	代码实践 3 - 4 小时	编写代码实现自动求导计算梯度
周五	代码实践 3 - 4 小时	使用自动求导完成简单的模型训练任务
周六	复习和总结 2 - 3 小时	检查本周学习成果，总结自动求导知识
周日	休息	

第 3 周

星期	学习时长	具体任务
周一	理论学习 2 - 3 小时	学习 PyTorch 的基本数据加载方法，使用 `DataLoader` 加载数据集
周二	理论学习 2 - 3 小时	掌握数据预处理方法，如归一化、裁剪等
周三	理论学习 2 - 3 小时	理解数据加载和处理在模型训练中的作用
周四	代码实践 3 - 4 小时	编写代码实现数据加载和预处理
周五	代码实践 3 - 4 小时	使用处理后的数据进行简单的模型训练
周六	复习和总结 2 - 3 小时	检查本周学习成果，总结数据加载和处理知识
周日	休息	

第 4 周

星期	学习时长	具体任务
周一 - 周三	代码实践 3 - 4 小时	使用 PyTorch 构建一个简单的全连接神经网络，对 MNIST 手写数字数据集进行分类任务，定义模型结构
周四	代码实践 3 - 4 小时	选择合适的损失函数和优化器，进行模型的训练
周五	代码实践 3 - 4 小时	对训练好的模型进行评估
周六	复习和总结 2 - 3 小时	检查本周学习成果，总结简单模型训练的经验
周日	休息	

2、第二阶段：PyTorch 进阶学习阶段（第 5 - 8 周）

第 5 周

星期	学习时长	具体任务
周一	理论学习 2 - 3 小时	深入学习卷积神经网络（CNN）的结构和原理
周二	理论学习 2 - 3 小时	理解 CNN 的应用场景和优缺点
周三	理论学习 2 - 3 小时	学习 CNN 的基本组成部分，如卷积层、池化层等
周四	代码实践 3 - 4 小时	编写代码实现简单的 CNN 模型
周五	代码实践 3 - 4 小时	使用 CNN 对图像进行分类任务的初步尝试
周六	复习和总结 2 - 3 小时	检查本周学习成果，总结 CNN 知识
周日	休息	

第 6 周

星期	学习时长	具体任务
周一	理论学习 2 - 3 小时	深入学习循环神经网络（RNN）的结构和原理
周二	理论学习 2 - 3 小时	理解 RNN 的应用场景和优缺点
周三	理论学习 2 - 3 小时	学习 RNN 的变体（LSTM、GRU）的原理和区别
周四	代码实践 3 - 4 小时	编写代码实现简单的 RNN 模型
周五	代码实践 3 - 4 小时	使用 RNN 进行序列数据的预测任务的初步尝试
周六	复习和总结 2 - 3 小时	检查本周学习成果，总结 RNN 知识
周日	休息	

第 7 周

星期	学习时长	具体任务
周一 - 周二	理论学习 2 - 3 小时	掌握模型的训练流程，学习如何选择合适的超参数（如学习率、批次大小、训练轮数等）
周三	理论学习 2 - 3 小时	学习如何进行模型的验证和调优
周四 - 周五	代码实践 3 - 4 小时	使用 CNN 或 RNN 模型进行训练，根据模型性能调整超参数
周六	复习和总结 2 - 3 小时	检查本周学习成果，总结模型训练和调优知识
周日	休息	

第 8 周

星期	学习时长	具体任务
周一 - 周二	理论学习 2 - 3 小时	学习使用 PyTorch 的高级特性，如自定义层、自定义损失函数等
周三 - 周五	代码实践 3 - 4 小时	在 CNN 或 RNN 模型中应用自定义层和损失函数，进行模型训练和评估
周六	复习和总结 2 - 3 小时	检查本周学习成果，总结 PyTorch 高级特性的应用经验
周日	休息	

三、资源推荐

1、PyTorch 学习资源

书籍

1. **《深度学习框架 PyTorch：入门与实践》**
   - **作者**：陈云，Python 程序员、Linux 爱好者和 PyTorch 源码贡献者，在相关竞赛中取得优异成绩。
   - **内容**：结合基础知识和前沿研究，带领读者从零开始完成几个经典有趣的深度学习小项目，如 GAN 生成动漫头像、AI 滤镜、AI 写诗等。
   - **高效利用方法**：可先浏览目录，确定重点章节进行精读。书中案例代码可结合其 GitHub 仓库（**https://github.com/chenyuntc/pytorch-book** ）中的源码进行学习，加深理解。同时，做笔记记录关键知识点和编程技巧。
2. **《深度学习入门之 PyTorch》**
   - **作者**：廖星宇，就读于中国科学技术大学应用数学系，在个人博客、知乎等平台发布多篇深度学习文章。
   - **内容**：从机器学习和深度学习的基础理论入手，从零开始学习 PyTorch，了解基础及如何用其搭建模型，涵盖线性回归、Logistic 回归、优化方法、多种神经网络及生成对抗网络等内容，最后通过实战了解前沿研究成果及实际应用。

- **高效利用方法**：对于有一定深度学习了解但对 PyTorch 完全不懂的读者，可将其作为入门书。书中代码过时，但对应的 GitHub 代码（**https://github.com/L1aoXingyu/code-of-learn-deep-learning-with-pytorch** ）是 OK 的，可结合代码学习，并与 PyTorch 的官网教程一起参考。

3. **《PyTorch 深度学习实战》**
   - **内容**：介绍了 PyTorch 的核心概念和特性，如动态图、自动微分等，让读者快速掌握基础知识。详细讲解深度学习关键实践，包括使用 PyTorch 张量 API、Python 加载数据、监控训练以及结果可视化等，还包含真实、完整的案例项目，如肿瘤图像分类器的构建。
   - **高效利用方法**：通过阅读书籍掌握理论知识，然后结合案例项目进行实践操作，加深对深度学习流程的理解和动手能力。在实践过程中，遇到问题可查阅书中相关内容和代码示例。

在线课程

1. **《深入浅出 PyTorch》**
   - **平台**：DataWhale
   - **内容**：课程分为三个阶段，包括 PyTorch 深度学习基础知识、进阶操作、案例分析。内容涵盖 PyTorch 简介、安装、基础知识、主要组成模块、基础实战、模型定义、进阶训练技巧、可视化、生态简介、模型部署等方面。
   - **高效利用方法**：课程内容以 markdown 格式或 jupyter notebook 形式保存，除了多看加深理解，最重要的是动手练习。按照课程编排的学习周期，分阶段学习，如第一部分第一章到第四章学习周期为 10 天，第二部分第五章到第八章学习周期为 11 天。部分章节直播讲解可观看 B 站回放（**https://www.bilibili.com/video/BV1L44y1472Z** ）。
2. **PyTorch 官方入门教程**
   - **平台**：PyTorch 官网（**https://pytorch.org/get-started/locally/** ）
   - **内容**：提供了 PyTorch 本地安装及入门的详细指导，帮助学习者快速上手。
   - **高效利用方法**：在开始学习 PyTorch 时，首先按照该教程进行环境搭建和基础入门学习，为后续深入学习打下基础。学习过程中，可结合官网文档进一步了解 API 和使用方法。

GitHub 开源项目

1. **pytorch 官方仓库**（**https://github.com/pytorch** ）
   - **内容**：包含 PyTorch 核心代码，以及 Vision、Text、Reinforcement Learning 等方面的示例和教程，还有针对计算机视觉的 Datasets、Transforms 和 Models 等。
   - **高效利用方法**：可下载项目代码进行调试和修改，深入理解代码逻辑。对于其中的示例和教程，可按照步骤进行实践，学习 PyTorch 在不同领域的应用。同时，关注项目更新，了解 PyTorch 的最新发展。

文档或教程

1. **PyTorch 官方文档**（**https://pytorch.org/docs/stable/index.html** ）
    - **内容**：详细介绍了 PyTorch 的各种功能和 API，按发布状态分为稳定版、Beta 版和原型版。
    - **高效利用方法**：将其作为工具书，在学习和实践过程中遇到问题时及时查阅。对于稳定版功能，可放心使用并深入了解；对于 Beta 版和原型版功能，可关注其发展和变化。
2. **PyTorch 教程 | 菜鸟教程**（**https://www.runoob.com/pytorch/pytorch-tutorial.html** ）
    - **内容**：适合对深度学习和机器学习感兴趣的人学习，介绍了 PyTorch 的基本概念、适用领域，以及学习前需要具备的基础知识，还包含 PyTorch 中基本的张量操作示例。
    - **高效利用方法**：对于初学者，可先阅读该教程了解 PyTorch 的基本情况和学习要求。在学习过程中，可参考其中的示例代码进行实践，巩固所学知识。

2、大模型微调技术学习资源

书籍

1. **《从零开始大模型开发与微调》**
    - **作者**：王晓华
    - **内容**：系统介绍基于 PyTorch 2.0 和 ChatGLM 的大模型开发与微调技术，涵盖大模型基本理论、算法、程序实现、应用实战及微调技术，通过丰富示例和实战案例展示构建、训练、评估及微调大模型的方法。
    - **高效利用方法**：精读书籍内容，学习大模型开发与微调的理论和方法。结合案例进行实践操作，加深对知识的理解和掌握。在实践过程中，可参考书中的代码示例和思路，解决遇到的问题。
2. **《大规模语言模型》**
    - **作者**：张奇教授等
    - **内容**：介绍 LLM 的基本概念、发展历程以及构建流程，包括预训练、有监督微调、奖励建模到强化学习四个阶段，详细讨论每个阶段使用的算法、数据来源、面临的难题及实践经验，并提供大量工程实践方法和示例代码。
    - **高效利用方法**：按照书籍的章节顺序，系统学习大规模语言模型的构建和微调知识。对于每个阶段的内容，深入理解算法原理和实践方法，并结合示例代码进行实践验证。同时，关注书中提到的难题和解决方案，提高解决实际问题的能力。

在线课程

1. **大模型微调实战营 - 应用篇**（**https://www.greedyai.com/ai-courses/LLM_finetuning_application**）
    - **平台**：贪心科技
    - **内容**：包括大模型基础、大模型指令微调、常用的开源模型微调、大模型对齐、垂直领域大模型应用 5 个阶段，涵盖大模型微调应用场景、多种参数微调方法、训练框架解析、前沿技术应用等全方位知识讲解，并结合 8 个实战项目。
    - **高效利用方法**：按照课程大纲分阶段学习，在学习理论知识的同时，认真完成实战项目。对于课程中涉及的新技术和方法，及时查阅相关资料进行深入了解。在实践过程中，与其他学员交流经验，共同提高。

GitHub 开源项目

1. **llm - action**（**https://github.com/liguodongiot/llm-action**）
    - **内容**：分享大模型相关技术原理以及实战经验，包括大模型工程化、大模型应用落地等方面。
    - **高效利用方法**：查看项目中的文档和代码，学习大模型相关技术的实现原理和实践经验。可以参考项目中的代码结构和思路，应用到自己的项目中。同时，关注项目的更新和讨论，了解最新的技术动态。

文档或教程

1. **微调（Fine - tuning） | OpenAI 官方帮助文档中文版**（**https://openai.xiniushu.com/docs/guides/fine-tuning**）
    - **内容**：介绍了 OpenAI 模型微调的相关内容，包括微调的优势、适用模型、安装步骤、训练数据准备、创建微调模型和使用微调模型等方面。
    - **高效利用方法**：在进行 OpenAI 模型微调时，按照文档的步骤进行操作。详细了解训练数据的准备要求和格式，以及微调模型的创建和使用方法。对于文档中提到的注意事项和常见问题，要特别关注，避免在实践中出现错误。
2. **大模型微调实战指南：从零开始手把手教你微调大模型**（**https://blog.csdn.net/2401_84495872/article/details/143001564**）
    - **内容**：手把手教读者从零开始微调大模型，使用阿里魔塔社区提供的集成环境，以零一万物的 Yi 开源大语言模型为例，介绍了账号和环境准备、下载模型、微调实战等步骤。
    - **高效利用方法**：按照教程的步骤进行实践操作，在实践过程中理解大模型微调的流程和原理。对于教程中提到的依赖库安装、模型下载和配置文件修改等操作，要仔细执行，确保每个步骤都正确完成。

3、智能体开发学习资源

书籍

1. **《Agent 和多 Agent 系统的设计与应用》**
   - **内容**：开始介绍关于 Agent 的研究历程、理论等，然后提出基于 Agent 的分布式环境计算模型 MADOCE，接着讨论多 Agent 系统以及如何设计它。
   - **高效利用方法**：先了解 Agent 的研究历程和理论基础，再深入学习分布式环境计算模型 MADOCE。在学习多 Agent 系统设计时，结合实际案例进行分析，掌握设计方法和技巧。同时，做笔记总结关键知识点和设计思路。
2. **《多 Agent 系统引论》**
   - **内容**：结构严谨，列出了参考读者应具备的能力，包括了解高级编程语言、伪代码和分布式，熟悉人工智能基本概念和内容，有集合和逻辑符号的基础知识。书中对多 Agent 系统进行了详细介绍。
   - **高效利用方法**：在阅读书籍前，确保自己具备相应的基础知识。按照书籍的章节顺序，系统学习多 Agent 系统的相关知识。对于书中的概念和理论，结合实际例子进行理解，提高学习效果。

在线课程

1. **Hugging Face Agents Course**（**https://www.aibase.com/zh/tool/36115** ）
   - **平台**：Hugging Face
   - **内容**：免费在线课程，旨在帮助学习者从初学者成长为专家，掌握 AI 智能体的理论、设计和实践。课程内容丰富，涵盖从基础知识到实际应用的多个方面，通过理论学习、实践操作和挑战任务，帮助学习者深入理解 AI 智能体的工作原理，并学会使用最新的库和工具构建自己的智能体，还提供认证机会。
   - **高效利用方法**：按照课程的安排，系统学习 AI 智能体的理论知识。积极参与实践操作和挑战任务，通过实际项目提高自己的开发能力。完成特定任务后，争取获得证书，以证明自己的学习成果。

四、实践项目

1、PyTorch 实现模型训练项目：CIFAR - 10 图像分类

项目概述

本项目使用 PyTorch 构建一个卷积神经网络（CNN），对 CIFAR - 10 图像数据集进行图像分类任务。CIFAR - 10 数据集包含 10 个不同类别的 60000 张彩色图像，每个类别有 6000 张图像，其中训练集 50000 张，测试集 10000 张。

项目步骤

1. 数据加载与预处理

参考学习计划中第 3 周关于数据加载和处理的内容，使用 `torchvision` 库加载 CIFAR - 10 数据集，并进行数据预处理，如归一化、裁剪等。

```
import torch
import torchvision
import torchvision.transforms as transforms

# 定义数据预处理步骤
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

# 加载训练集
trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                        download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=4,
                                          shuffle=True, num_workers=2)

# 加载测试集
testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                       download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=4,
                                         shuffle=False, num_workers=2)

# 定义类别名称
classes = ('plane', 'car', 'bird', 'cat',
           'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
```

2. 模型定义

参考学习计划中第 5 周关于 CNN 模型的内容，定义一个简单的 CNN 模型。

```
import torch.nn as nn
import torch.nn.functional as F
```

```
3
4   class Net(nn.Module):
5       def __init__(self):
6           super().__init__()
7           self.conv1 = nn.Conv2d(3, 6, 5)
8           self.pool = nn.MaxPool2d(2, 2)
9           self.conv2 = nn.Conv2d(6, 16, 5)
10          self.fc1 = nn.Linear(16 * 5 * 5, 120)
11          self.fc2 = nn.Linear(120, 84)
12          self.fc3 = nn.Linear(84, 10)
13
14      def forward(self, x):
15          x = self.pool(F.relu(self.conv1(x)))
16          x = self.pool(F.relu(self.conv2(x)))
17          x = torch.flatten(x, 1) # flatten all dimensions except batch
18          x = F.relu(self.fc1(x))
19          x = F.relu(self.fc2(x))
20          x = self.fc3(x)
21          return x
22
23  net = Net()
```

3. 损失函数和优化器选择

参考学习计划中第 4 周关于模型训练的内容，选择合适的损失函数和优化器。

```
1   import torch.optim as optim
2
3   criterion = nn.CrossEntropyLoss()
4   optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
```

4. 模型训练

按照学习计划中关于模型训练的步骤，进行模型训练。

```
1   for epoch in range(2):  # 训练2个epoch
2       running_loss = 0.0
3       for i, data in enumerate(trainloader, 0):
4           # 获取输入数据
5           inputs, labels = data
6
7           # 梯度清零
8           optimizer.zero_grad()
9
10          # 前向传播 + 反向传播 + 优化
11          outputs = net(inputs)
12          loss = criterion(outputs, labels)
13          loss.backward()
14          optimizer.step()
15
```

```
            # 打印统计信息
            running_loss += loss.item()
            if i % 2000 == 1999:    # 每2000个小批量打印一次
                print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss / 2000:.3f}')
                running_loss = 0.0

print('Finished Training')
```

5. 模型评估

对训练好的模型进行评估。

```
correct = 0
 total = 0
 # 不需要计算梯度
 with torch.no_grad():
     for data in testloader:
         images, labels = data
         # 前向传播
         outputs = net(images)
         # 获取预测结果
         _, predicted = torch.max(outputs.data, 1)
         total += labels.size(0)
         correct += (predicted == labels).sum().item()

 print(f'Accuracy of the network on the 10000 test images: {100 * correct / total} %')
```

2、大模型微调实战项目：基于 GPT - Neo 的文本生成

项目概述

本项目选择开源的大语言模型 GPT - Neo，使用 LoRA 微调方法对其进行微调，以适应文本生成任务。

项目步骤

1. 环境准备

参考优质学习资源中关于大模型微调的文档和教程，安装必要的依赖库。

2. 数据准备

准备适合文本生成任务的数据集，如新闻文章、小说等。

3. 模型加载

加载 GPT - Neo 模型。

```
from transformers import GPTNeoForCausalLM, GPT2Tokenizer

model = GPTNeoForCausalLM.from_pretrained("EleutherAI/gpt - neo - 125M")
tokenizer = GPT2Tokenizer.from_pretrained("EleutherAI/gpt - neo - 125M")
```

4. LoRA 微调

使用 LoRA 方法对模型进行微调。参考大模型微调实战指南等学习资源，编写微调代码。

```
from peft import LoraConfig, get_peft_model

# 定义LoRA配置
config = LoraConfig(
    r=8,
    lora_alpha=16,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.1,
    bias="none",
    task_type="CAUSAL_LM"
)

# 获取LoRA模型
model = get_peft_model(model, config)

# 训练模型
model.train()
# 这里省略具体的训练代码，可参考学习资源中的示例
```

5. 模型评估

使用测试数据集对微调后的模型进行评估，评估指标可以包括困惑度等。

3、智能体开发场景应用项目：基于强化学习的游戏智能体

项目概述

本项目设计一个基于强化学习的游戏智能体，让智能体在游戏环境中学习如何做出最优决策。以 OpenAI Gym 中的 CartPole 环境为例。

项目步骤

1. 环境安装

安装 OpenAI Gym 库。

```
pip install gym
```

2. 智能体设计

使用强化学习算法，如 Q - learning，设计智能体。

```
import gym
import numpy as np

# 初始化环境
env = gym.make('CartPole - v1')

# 定义Q表
q_table = np.zeros([env.observation_space.n, env.action_space.n])

# 定义超参数
alpha = 0.1
gamma = 0.6
epsilon = 0.1

# 训练智能体
for i in range(1, 100001):
    state = env.reset()
    epochs, penalties, reward, = 0, 0, 0
    done = False

    while not done:
        if np.random.uniform(0, 1) < epsilon:
            action = env.action_space.sample() # 探索
        else:
            action = np.argmax(q_table[state]) # 利用

        next_state, reward, done, info = env.step(action)

        old_value = q_table[state, action]
        next_max = np.max(q_table[next_state])

        # 更新Q表
        new_value = (1 - alpha) * old_value + alpha * (reward + gamma *
next_max)
        q_table[state, action] = new_value

        if reward == -10:
            penalties += 1

        state = next_state
        epochs += 1

```

```
        if i % 100 == 0:
            print(f'Episode: {i}')

print('Training finished.')
```

3. 智能体评估

使用训练好的智能体在测试环境中进行评估，记录平均奖励等指标。

```
total_epochs, total_penalties = 0, 0
    episodes = 100

    for _ in range(episodes):
        state = env.reset()
        epochs, penalties, reward = 0, 0, 0
        done = False

        while not done:
            action = np.argmax(q_table[state])
            state, reward, done, info = env.step(action)

            if reward == -10:
                penalties += 1

            epochs += 1

        total_penalties += penalties
        total_epochs += epochs

    print(f'Results after {episodes} episodes:')
    print(f'Average timesteps per episode: {total_epochs / episodes}')
    print(f'Average penalties per episode: {total_penalties / episodes}')
```

五、评估与优化

1、小测验设计

（一）PyTorch 基础学习阶段

本阶段的学习重点在于张量操作、自动求导机制以及数据加载和处理。以下是根据这些重点设计的小测验题目：

1. **选择题**
   - 以下哪种数据类型不能用于创建 PyTorch 张量？（ ）
     A. 浮点型 B. 整型 C. 布尔型 D. 字符串型
   - 在 PyTorch 中，自动求导机制主要用于计算（ ）。
     A. 梯度 B. 损失函数 C. 模型参数 D. 学习率
   - 使用 `DataLoader` 加载数据集时，`shuffle` 参数的作用是（ ）。
     A. 打乱数据集顺序 B. 增加数据集大小 C. 减少数据集大小 D. 不改变数据集顺序
2. **填空题**
   - 创建一个形状为 (3, 4) 的全零张量的代码是 `torch.____((3, 4))`。
   - 在自动求导中，需要将张量的 `____` 属性设置为 `True` 才能计算梯度。
   - 数据预处理中，归一化操作通常使用 `transforms.____` 函数。
3. **简答题**
   - 请简要解释张量的概念，并举例说明如何进行张量的加法运算。
   - 自动求导机制在深度学习中有什么重要作用？请结合模型训练过程进行说明。
   - 简述使用 `DataLoader` 加载数据集的步骤和优势。

（二）PyTorch 进阶学习阶段

此阶段重点学习深度学习模型结构、训练流程和高级特性。小测验题目如下：

1. **选择题**
   - 卷积神经网络（CNN）中，卷积层的主要作用是（ ）。
     A. 提取特征 B. 降维 C. 分类 D. 池化
   - 在模型训练中，学习率是一个重要的超参数，以下关于学习率的说法正确的是（ ）。
     A. 学习率越大，模型收敛越快 B. 学习率越小，模型收敛越稳定 C. 学习率应该固定不变 D. 学习率与模型的泛化能力无关
   - 自定义 PyTorch 层时，需要继承（ ）类。
     A. `nn.Module` B. `nn.Linear` C. `nn.Conv2d` D. `nn.ReLU`
2. **填空题**
   - 循环神经网络（RNN）的主要问题是 ，其变体 LSTM 和 GRU 主要是为了解决这个问题。
   - 在模型验证和调优过程中，常用的评估指标有 （至少列举两个）。
   - 使用自定义损失函数时，需要在类中重写 `____` 方法。
3. **简答题**

- 请比较卷积神经网络（CNN）和循环神经网络（RNN）的应用场景和优缺点。
- 如何选择合适的超参数进行模型训练？请结合具体的超参数（如学习率、批次大小、训练轮数）进行说明。
- 简述自定义 PyTorch 层和损失函数的步骤和意义。

2、项目完成度评估

（一）PyTorch 实现模型训练项目：CIFAR - 10 图像分类

本项目的具体目标是使用 PyTorch 构建一个卷积神经网络（CNN），对 CIFAR - 10 图像数据集进行分类。评估按照以下步骤进行：

1. **数据加载与预处理（20 分）**
   - 成功使用 `torchvision` 库加载 CIFAR - 10 数据集，得 10 分。
   - 正确进行数据预处理，如归一化、裁剪等，得 10 分。
2. **模型定义（20 分）**
   - 参考学习计划中关于 CNN 模型的内容，定义一个简单的 CNN 模型，得 15 分。
   - 模型结构合理，参数设置正确，得 5 分。
3. **损失函数和优化器选择（20 分）**
   - 选择合适的损失函数（如交叉熵损失函数），得 10 分。
   - 选择合适的优化器（如随机梯度下降），并正确设置参数，得 10 分。
4. **模型训练（20 分）**
   - 按照学习计划中关于模型训练的步骤，进行模型训练，得 15 分。
   - 训练过程中，损失函数能够正常下降，得 5 分。
5. **模型评估（20 分）**
   - 对训练好的模型进行评估，计算准确率等指标，得 15 分。
   - 评估结果合理，与预期相符，得 5 分。

（二）大模型微调实战项目：基于 GPT - Neo 的文本生成

本项目目标是选择开源的大语言模型 GPT - Neo，使用 LoRA 微调方法对其进行微调，以适应文本生成任务。评估步骤如下：

1. **环境准备（10 分）**

- 参考优质学习资源中关于大模型微调的文档和教程，安装必要的依赖库，得 10 分。

2. **数据准备（10 分）**
   - 准备适合文本生成任务的数据集，如新闻文章、小说等，得 10 分。
3. **模型加载（10 分）**
   - 正确加载 GPT - Neo 模型，得 10 分。
4. **LoRA 微调（40 分）**
   - 使用 LoRA 方法对模型进行微调，参考大模型微调实战指南等学习资源，编写微调代码，得 30 分。
   - 微调过程中，模型能够正常训练，损失函数下降，得 10 分。
5. **模型评估（30 分）**
   - 使用测试数据集对微调后的模型进行评估，评估指标可以包括困惑度等，得 20 分。
   - 评估结果合理，微调后的模型性能有所提升，得 10 分。

（三）智能体开发场景应用项目：基于强化学习的游戏智能体

本项目设计一个基于强化学习的游戏智能体，让智能体在游戏环境中学习如何做出最优决策。评估步骤如下：

1. **环境安装（10 分）**
   - 安装 OpenAI Gym 库，得 10 分。
2. **智能体设计（30 分）**
   - 使用强化学习算法，如 Q - learning，设计智能体，得 20 分。
   - 智能体设计合理，能够在游戏环境中进行学习和决策，得 10 分。
3. **智能体训练（30 分）**
   - 对智能体进行训练，训练过程中智能体的性能逐渐提升，得 20 分。
   - 训练过程中，没有出现明显的错误或异常，得 10 分。
4. **智能体评估（30 分）**
   - 使用训练好的智能体在测试环境中进行评估，记录平均奖励等指标，得 20 分。
   - 评估结果合理，智能体在测试环境中表现良好，得 10 分。

3、代码质量检查

（一）代码可读性（30 分）

1. **变量命名（10 分）**
    - 变量命名具有明确的含义，能够清晰表达其用途，得 10 分。
    - 部分变量命名不规范，但不影响整体理解，得 5 分。
    - 变量命名混乱，难以理解，得 0 分。
2. **代码结构（10 分）**
    - 代码结构清晰，模块划分合理，易于阅读和维护，得 10 分。
    - 代码结构存在一些问题，但不影响整体功能，得 5 分。
    - 代码结构混乱，难以理解和修改，得 0 分。
3. **代码格式（10 分）**
    - 代码格式规范，符合 Python 编码规范，如缩进、空格等使用正确，得 10 分。
    - 代码格式存在一些小问题，但不影响代码的运行，得 5 分。
    - 代码格式混乱，影响代码的可读性，得 0 分。

（二）注释完整性（30 分）

1. **关键代码注释（15 分）**
    - 关键代码部分都有详细的注释，解释代码的功能和实现思路，得 15 分。
    - 部分关键代码有注释，但不够详细，得 8 分。
    - 关键代码几乎没有注释，得 0 分。
2. **函数和类注释（15 分）**
    - 每个函数和类都有规范的注释，说明其功能、输入参数和返回值，得 15 分。
    - 部分函数和类有注释，但不完整，得 8 分。
    - 函数和类几乎没有注释，得 0 分。

（三）算法复杂度（20 分）

1. **时间复杂度（10 分）**
    - 算法的时间复杂度合理，没有明显的性能瓶颈，得 10 分。
    - 算法的时间复杂度较高，但在可接受范围内，得 5 分。
    - 算法的时间复杂度过高，严重影响性能，得 0 分。
2. **空间复杂度（10 分）**

- 算法的空间复杂度合理，没有浪费过多的内存资源，得 10 分。
- 算法的空间复杂度较高，但在可接受范围内，得 5 分。
- 算法的空间复杂度过高，占用大量内存，得 0 分。

（四）错误处理（20 分）

1. **异常捕获（10 分）**
   - 代码中对可能出现的异常进行了合理的捕获和处理，避免程序崩溃，得 10 分。
   - 部分异常有处理，但不够完善，得 5 分。
   - 代码几乎没有异常处理，得 0 分。
2. **错误提示（10 分）**
   - 当出现错误时，能够给出明确的错误提示信息，帮助调试，得 10 分。
   - 错误提示信息不够明确，难以定位问题，得 5 分。
   - 没有错误提示信息，得 0 分。

4、根据评估结果调整计划

根据上述小测验、项目完成度评估和代码质量检查的结果，对学习计划进行调整。如果在某个阶段的小测验中成绩不理想，说明该阶段的知识掌握不够扎实，需要增加相应的理论学习时间，重新复习相关知识点，并通过更多的代码实践来巩固。如果项目完成度不高，分析是哪个环节出现问题，如数据处理、模型设计等，针对性地进行改进。对于代码质量方面的问题，加强代码规范的学习，多参考优秀的代码示例，提高代码的可读性、注释完整性、算法复杂度和错误处理能力。

六、工具与环境

1、Python 版本选择

为了确保与大多数深度学习库兼容，建议选择 Python 3.8 - 3.10 版本。这些版本在稳定性和广泛使用性方面表现出色，许多深度学习库都对其提供了良好的支持。你可以从 **Python 官方网站** 下载适合你操作系统的 Python 版本。

2、PyTorch 安装指南

（一）安装前准备

在安装 PyTorch 之前，需要确保已经安装了合适的 Python 版本。同时，根据你的硬件环境，确定是否需要使用 GPU 加速。如果需要使用 GPU，还需要安装相应的 CUDA 工具包和 cuDNN 库。

（二）不同操作系统和硬件环境的安装方法

1. Windows 系统

- **CPU 版本**：打开命令提示符或 PowerShell，运行以下命令安装 PyTorch CPU 版本：

```
pip install torch torchvision torchaudio
```

- **GPU 版本**：如果你有 NVIDIA GPU，并且已经安装了合适的 CUDA 工具包和 cuDNN 库，可以运行以下命令安装 PyTorch GPU 版本：

```
pip install torch torchvision torchaudio --extra-index-url
https://download.pytorch.org/whl/cuXX.X
```

其中 `cuXX.X` 是你安装的 CUDA 版本号，例如 `cu118`。

2. Linux 系统

- **CPU 版本**：打开终端，运行以下命令安装 PyTorch CPU 版本：

```
pip install torch torchvision torchaudio
```

- **GPU 版本**：如果你有 NVIDIA GPU，并且已经安装了合适的 CUDA 工具包和 cuDNN 库，可以运行以下命令安装 PyTorch GPU 版本：

```
pip install torch torchvision torchaudio --extra-index-url
https://download.pytorch.org/whl/cuXX.X
```

其中 `cuXX.X` 是你安装的 CUDA 版本号，例如 `cu118`。

3. macOS 系统

在 macOS 系统上，目前仅支持 CPU 版本的 PyTorch。打开终端，运行以下命令安装 PyTorch CPU 版本：

```
pip install torch torchvision torchaudio
```

（三）验证安装

安装完成后，可以在 Python 环境中验证 PyTorch 是否安装成功。打开 Python 解释器，运行以下代码：

```
import torch
print(torch.__version__)
```

如果没有报错，并且能够正确输出版本号，则说明 PyTorch 安装成功。

3、GPU 使用建议

（一）支持的 GPU 型号和 CUDA 版本

PyTorch 支持大多数 NVIDIA GPU 型号。为了获得最佳性能，建议使用较新的 GPU 型号，如 NVIDIA GeForce RTX 系列、NVIDIA Tesla 系列等。同时，需要安装与 GPU 型号和 PyTorch 版本兼容的 CUDA 工具包和 cuDNN 库。你可以在 **PyTorch 官方网站** 上查找具体的版本兼容性信息。

（二）GPU 加速的配置方法

在使用 PyTorch 进行模型训练时，可以通过以下代码将模型和数据移动到 GPU 上进行计算：

```
import torch

# 检查是否有可用的 GPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# 将模型移动到 GPU 上
model = YourModel().to(device)

# 将数据移动到 GPU 上
inputs, labels = inputs.to(device), labels.to(device)
```

（三）注意事项

- 在使用 GPU 进行计算时，需要确保 GPU 有足够的显存。如果显存不足，可能会导致程序崩溃或训练速度变慢。可以通过减少批量大小、释放不必要的变量等方法来减少显存占用。
- 不同的 GPU 型号和 CUDA 版本可能会对训练速度和性能产生影响。在选择 GPU 和 CUDA 版本时，需要根据实际情况进行权衡。

4、其他学习工具

（一）代码编辑器

- **PyCharm**：是一款功能强大的 Python 集成开发环境（IDE），提供了代码编辑、调试、代码分析等功能。你可以从 **JetBrains 官方网站** 下载适合你操作系统的 PyCharm 版本。安装完成后，打开 PyCharm，创建一个新的 Python 项目，即可开始编写代码。
- **VS Code**：是一款轻量级的代码编辑器，支持多种编程语言，并且有丰富的插件生态系统。你可以从 **VS Code 官方网站** 下载适合你操作系统的 VS Code 版本。安装完成后，安装 Python 插件，即可开始编写 Python 代码。

（二）版本控制工具

- **Git**：是一款广泛使用的版本控制工具，可以帮助你管理代码的版本和协作开发。你可以从 **Git 官方网站** 下载适合你操作系统的 Git 版本。安装完成后，打开命令提示符或终端，运行以下命令配置 Git：

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

然后，你可以使用 Git 来创建代码仓库、提交代码、拉取代码等操作。

七、最终目标

通过这三个月的学习，你将掌握大模型应用开发的核心技能，包括 PyTorch 框架、大模型微调技术以及智能体开发。能够独立完成一个综合性项目，如结合 PyTorch 模型训练、大模型微调以及智能体开发的项目，展示所学技能。同时，具备进一步深入研究大模型应用开发或从事相关工作的基础能力。