**实验 3: 从0训练大模型**

**一、实验目的：**

（1）从0开始训练模型MiniMind2-small

（2）调整预训练参数

（3）绘制训练过程中的Loss曲线

（4）设计测试数据评估模型

**二、实验环境：**

（1）Anaconda3 开发环境

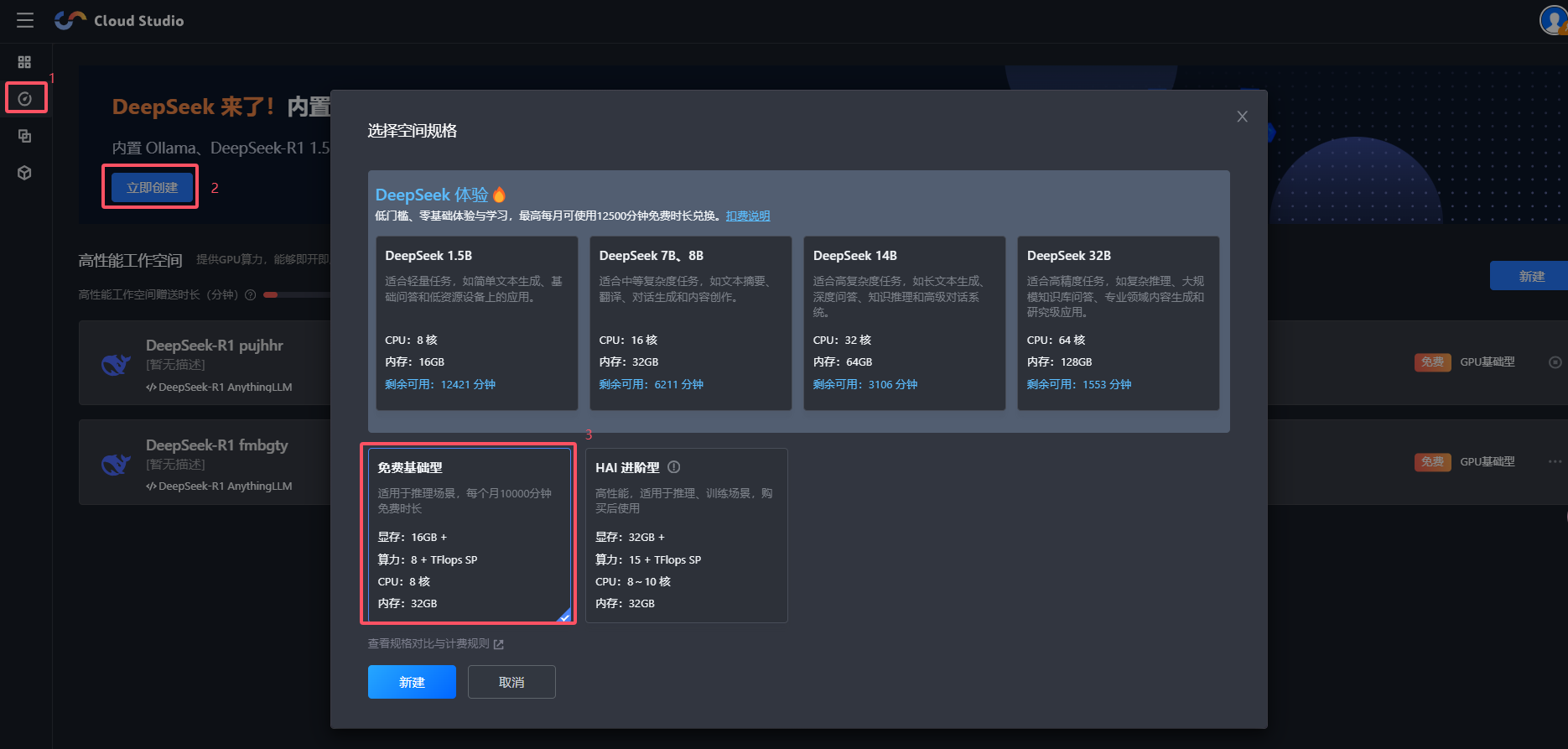
（2）IDE是Cloud Studio(云端IDE)

（3）大模型常用的库：PyTorch,Transformers等

**三、实验内容：**

**1.从0开始训练模型MiniMind2-small**

1.创建一个“高性能工作空间”



2.创建项目目录

|  |
| --- |
| #打开终端，输入下面命令  git clone https://github.com/jingyaogong/minimind.git |

3.配置环境

|  |
| --- |
| #创建环境  conda create -n minimind python=3.10  #激活  conda activate minimind  pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple |

4.下载模型

|  |
| --- |
| git clone https://huggingface.co/jingyaogong/MiniMind2 |

5.体验已有模型（MiniMind2）效果（命令行问答）

|  |
| --- |
| # load=0: load from pytorch model, load=1: load from transformers-hf model  python eval\_model.py --load 1 --model\_mode 2 |

6.从0开始自己训练MiniMind2-small 模型

|  |
| --- |
| # 环境准备  pip install -r requirements.txt -i <https://pypi.tuna.tsinghua.edu.cn/simple>  #数据集下载  在https://www.modelscope.cn/datasets/gongjy/minimind\_dataset/files中下载数据数据文件（默认推荐下载pretrain\_hq.jsonl + sft\_mini\_512.jsonl），并放到./dataset下（需自己在项目目录下创建./dataset文件夹）  #预训练（学知识），执行预训练，得到 pretrain\_\*.pth 作为预训练的输出权重（其中\*为模型的dimension，默认为512）  python train\_pretrain.py  #监督微调（学对话方式），执行监督微调，得到 full\_sft\_\*.pth 作为指令微调的输出权重（其中full即为全参数微调）  python train\_full\_sft.py  #测试模型效果  python eval\_model.py --model\_mode 1 # 默认为0：测试pretrain模型效果，设置为1：测试full\_sft模型效果 |

**2.调整预训练参数**

|  |
| --- |
| #以下参数调整仅供参考  #学习率 --learning\_rate,**过大**的学习率会导致 Loss 波动大，难以收敛甚至发散；**过小**的学习率训练速度太慢，容易陷入局部最优解。**基础调整**: 1e-3, 5e-4, 1e-4, 5e-5  #命令行输入下列代码进行训练  python train\_pretrain.py --learning\_rate 1e-4  #批次大小 --batch\_size，**大Batch** 提升训练速度，但会消耗更多显存，还可能导致 Loss 振荡大，泛化能力下降；**小Batch** 训练稳定，但收敛速度慢，可能陷入局部最优。推荐尝试:16，32，64  #命令行输入下列代码进行训练  python train\_pretrain.py --batch\_size 64 --accumulation\_steps 4  #更换优化器  #SGD (带动量Momentum)，经典随机梯度下降，带动量缓冲防止振荡  optimizer\_sgd = optim.SGD(model.parameters(), lr=learning\_rate, momentum=0.9)  #RMSprop，自适应学习率，每层按梯度平方的均值调整  optimizer\_rmsprop = optim.RMSprop(model.parameters(), lr=learning\_rate, alpha=0.9)  #Adagrad学习率自适应，稀疏参数调整大，频繁参数调整小  optimizer\_adagrad = optim.Adagrad(model.parameters(), lr=learning\_rate) |

**3.绘制Loss损失曲线**

参考代码示例：

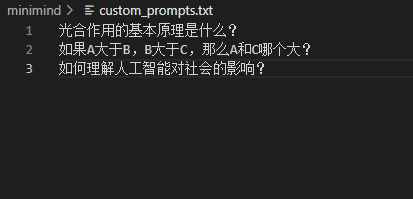
在train\_pretrain.py中，可以在train\_epoch函数中增加如下代码记录loss：

|  |
| --- |
| def train\_epoch(epoch, wandb):  loss\_fct = nn.CrossEntropyLoss(reduction='none')  start\_time = time.time()  epoch\_losses = [] # 新增：存储该轮训练的 loss 值  ...  # 记录loss  epoch\_losses.append(loss.item() \* args.accumulation\_steps)  if step % args.log\_interval == 0:  spend\_time = time.time() - start\_time  ...  torch.save(state\_dict, ckp)  model.train()  # 保存Loss数据到CSV文件  if not ddp or dist.get\_rank() == 0:  loss\_file = os.path.join(args.save\_dir, f"loss\_epoch\_{epoch+1}.csv")  pd.DataFrame(epoch\_losses, columns=["Loss"]).to\_csv(loss\_file, index=False)  Logger(f"Loss data saved to {loss\_file}")  #训练结束后用额外的脚本来读取保存的 .csv 文件，绘制 Loss 曲线：  import pandas as pd  import matplotlib.pyplot as plt  def plot\_loss\_curve(file\_path):  loss\_data = pd.read\_csv(file\_path)  plt.plot(loss\_data["Loss"], label="Training Loss")  plt.xlabel("Iteration")  plt.ylabel("Loss")  plt.title("Training Loss Curve")  plt.legend()  plt.grid(True)  plt.show()  # 示例使用  plot\_loss\_curve("out/loss\_epoch\_1.csv") |

1. **设计测试数据并评估模型**

4.1根据目标任务设计一组问答。

知识问答："光合作用的基本原理是什么？"；推理类："如果A大于B，B大于C，那么A和C哪个大？"；开放类："如何理解人工智能对社会的影响？"等。将这些问题保存在custom\_prompts.txt中，格式为一行一个问题。



4.2改动eval\_model.py代码加载自定义问答数据和系统地保存结果。

在 main() 里加一个读取自定义prompt的逻辑，支持自定义的问答数据：

|  |
| --- |
| # 从文本文件读取自定义prompt，每行一个问题  def get\_custom\_prompts(file\_path):  try:  with open(file\_path, 'r', encoding='utf-8') as f:  prompts = [line.strip() for line in f.readlines() if line.strip()]  return prompts  except FileNotFoundError:  print(f"❌ 文件 {file\_path} 未找到！")  return []  def main():  ...  # 新增参数：读取自定义prompt路径  parser.add\_argument('--custom\_prompt\_path', default=None, type=str, help="自定义prompt文件路径")  args = parser.parse\_args()  model, tokenizer = init\_model(args)  # 判断是否使用自定义prompt  if args.custom\_prompt\_path:  prompts = get\_custom\_prompts(args.custom\_prompt\_path)  if not prompts:  print("⚠️ 自定义Prompt文件为空，回退到默认Prompt！")  prompts = get\_prompt\_datas(args)  else:  prompts = get\_prompt\_datas(args)  # 选择自动 or 手动模式  test\_mode = int(input('[0] 自动测试\n[1] 手动输入\n'))  ...  **#同时在参数解析部分加上新选项：**  parser.add\_argument('--custom\_prompt\_path', default='', type=str,  help="指定自定义的问答数据路径") |

在 main() 里的 for 循环加上保存评估逻辑

|  |
| --- |
| results = []  ...  messages.append({"role": "assistant", "content": answer})  results.append({"prompt": prompt, "answer": answer})  # 保存评估结果  with open('eval\_results.json', 'w', encoding='utf-8') as f:  import json  json.dump(results, f, ensure\_ascii=False, indent=4) |

4.3命令行运行代码

|  |
| --- |
| python eval\_model.py --custom\_prompt\_path custom\_prompts.txt --model\_mode 1 |

**5.实验要求**

**5.1 分析不同参数设置对训练minimind模型loss的影响**

1. 根据2中提到的参数设置，任选两组参数进行搭配训练，参数搭配如下表仅供参考。

|  |  |  |  |
| --- | --- | --- | --- |
| 实验编号 | 学习率 | 批次大小 | 优化器 |
| A | 5e-4 | 64 | AdamW |
| B | 5e-5 | 32 | AdamW |
| C | 5e-4 | 32 | SGD |
| D | 5e-4 | 32 | RMSprop |
| E | 1e-3 | 32 | AdamW |
| F | 1e-4 | 32 | AdamW |
| G | 5e-4 | 64 | SGD |
| H | 5e-5 | 32 | SGD |

1. 每次更改参数配置后，记录实验环境、参数配置、最终loss值及使用Matplotlib或其他可视化工具，将loss曲线绘制出来。
2. 讨论与分析：分析不同参数设置对loss的收敛趋势的影响。
3. 总结最优参数。

**5.2 设计测试数据并评估模型**

1. 根据模型的训练任务，设计测试输入保存在custom\_prompts.txt中，涵盖不同类型和难度的样本
2. 将设计的测试数据输入训练好的模型，生成相应的输出结果。
3. 根据生成结果的准确性、流畅性和多样性，对模型的生成能力进行评估，并总结模型的优缺点。