ChatGPT

DDA5001 Part II 项目规划与知识要点

项目任务分步规划

1. **环境与数据准备**：配置好所需的 Python 环境（安装 PyTorch、Huggingface Transformers、Datasets、PEFT 等库），并获取课程提供的代码仓库和数据集。在课程的官方仓库 **DDA5001-25Fall** 的 p2 目录下可以找到 Part II 所需的代码模板（如`prepare.py`，`finetune.py`，`rollout.py`）和 **Math500** 数据集 [1]。确保显卡驱动、CUDA 等正确安装，以利用 GPU 训练模型。

2. **数据集转换与聊天模板应用**：将 **Math500** 训练集的问题和解答转换为大模型聊天格式（即 prompt 格式）。具体来说：

3. **应用聊天模板**：将每个数学问题视作用户提问，将其正文作为 **User** 消息，并在结尾添加一句指令如："Please reason step by step, and put your final answer within \boxed{}."（请逐步推理，并把最终答案放在\boxed{}中）[2]。这相当于要求模型以链式思维（逐步推理）的方式作答。随后，将原始解题答案作为 **Assistant** 消息，用来模拟助手的回答 [3]。如果有系统消息位于对话开头，可在`prepare.py`中填入（如引导模型扮演数学导师等），但更重要的是在用户问题后附加上述逐步推理指令 [4] 以提高模型求解数学题的能力。

4. **Tokenization**：使用 Qwen-3 模型自带的 tokenizer 对应用模板后的训练文本进行标记，将字符转换为离散的 token ID 列表 [5]。这一步由`prepare.py`脚本完成：需要补全代码中关于 **chat 模板转换** 和 **tokenizer** 调用的`# TODO`部分 [6]。转换完成后，还需要对 **标签进行掩码**（mask），即对于每条样本，只计算回答部分的损失，不计算提示部分的损失——通常通过将输入 prompt 对应位置的标签设为 -100 来忽略这些 token 的损失 [7]。完成以上处理后，运行`prepare.py`脚本（或在提供的notebook中调用相应函数）生成**已编码的训练集和验证集**，以备模型微调使用 [8]。

5. **多优化器模型训练**：在完成数据准备后，加载预训练的 **Qwen3-0.6B-Base** 模型参数，准备对其进行有监督微调（SFT）。按照项目要求，需要使用**不同的优化策略**分别进行训练并比较：

6. **实现优化器调用**：在`finetune.py`中补全代码，初始化所需的优化器，包括 **AdamW**（Adam 的权重衰减变体）、**SGD**（随机梯度下降）和 **LoRA** 优化方法 [9]。其中 LoRA 属于参数高效微调方法，并非独立优化算法，而是通过冻结原模型的大部分权重，仅训练插入的低秩矩阵参数来节约显存 [10]。使用 Huggingface 的 PEFT 库可以方便地为模型的所有注意力和全连接层添加 LoRA 模块 [11]。注意为 LoRA 部分仍需指定一个内部优化器（默认用 AdamW）。
7. **设置超参数**：为不同优化器选择适当的超参数，包括**学习率**（例如 1e-3, 1e-4 等进行尝试）、**训练轮数**（建议不超过 3 个 epoch，以免过拟合 [12]）以及 **LoRA 的秩(r)** 等 [13]。在代码中填入这些超参数设置，确保每种优化策略都能以合理配置运行。

8. **启动微调训练**：运行`finetune.py`脚本（可通过项目提供的 Jupyter Notebook 调用）开始模型微调 [14]。训练过程中会输出每隔若干 step 的训练损失和验证损失，可将这些日志记录保存下来备用。由于不同优化策略对显存和收敛速度的影响不同，请密切关注训练过程中的显存占用（通过`nvidia-smi`或 PyTorch 提供的监控接口）以及每轮训练耗时，防止显存溢出并合理分配时间。LoRA 方法通过只训练新增参数，大幅降低了训练所需内存和计算量，使在相同 GPU 上微调大模型成为可能 [15] [16]。

9. **模型评估与测试**：训练完成后，利用最佳的超参数组合得到的微调模型，进行**数学问题求解能力**的评估。具体步骤：

10. **准备评估脚本**：在 `rollout.py` 中补全与训练阶段相同的聊天模板处理（与 `prepare.py` 中一致的逻辑），保证模型在推理时接收的问题格式与训练时一致 [17]。然后运行 `rollout.py` 生成微调模型对 **Math500** 测试集 500道题的回答，再用 `evaluate.py` 对这些回答进行评分 [18]。评分标准可能是准确匹配最终答案或公式（项目应已提供评测函数）。

11. **基线对比**：同时运行**基础模型（未微调的 Qwen3-0.6B-Base）**在相同测试集上的推理，并记录其得分作为对照 [19]。这样可以比较微调前后的模型在数学题上的表现提升幅度。

12. **结果分析与报告撰写**：根据不同优化方案的训练日志和评测结果，总结实验发现并撰写报告：

13. **绘制损失曲线**：整理各优化器在训练过程中记录的 **训练损失** 和 **验证损失**，绘制随时间（或 step）变化的曲线 [20]。可分别绘制两张图表（训练集和验证集），比较例如 AdamW 与 SGD 在收敛速度和稳定性上的差异，LoRA 方法相对于全参数微调的曲线走势等。曲线有助于分析哪种方法收敛更快、过拟合情况如何。
14. **比较资源消耗**：汇总不同优化器的**峰值 GPU 显存占用**以及**每轮训练耗时**，制作表格进行比较 [21]。一般而言，Adam(W)由于需维护一阶和二阶动量，单步可能稍慢且占用更多显存，而 SGD 较为轻量但收敛慢；LoRA 因冻结大部分参数，实际显存开销和计算量显著降低 [22]，预期会看到 **LoRA 占用显存最低**，而 **AdamW** 次之（实际实验可能表明动量开销相对总开销不算太大 [23]），**SGD**在小学习率下可能需要更多迭代才能达到同等收敛效果。表格量化这些差异，有助于权衡各种优化方案在资源受限场景下的适用性。
15. **评估指标对比**：将**基础模型**与**各微调模型**在 Math500 测试集上的总成绩整理成表格 [24]。这可以是每种优化器在最佳超参数下达到的正确率/得分，对比微调前（基础模型）的成绩。预期微调模型应优于未微调模型，在逐步推理指令的帮助下，数学题解答正确率有所提升。如果发现某优化方法效果特别突出或欠佳，需要在报告中分析可能原因。
16. **经验与心得**：在报告中记录项目实施过程中遇到的挑战（如环境安装、内存不足、损失发散等）以及解决办法，并总结观察到的现象 [25]。例如，可能描述LoRA如何解决显存不足问题、不同学习率对收敛的影响、链式思维提示对解题结果的作用等。这部分体现对项目的思考深度，也是报告的重要组成部分。

完成以上步骤后，整理撰写不超过4页的项目报告（不含附录和参考文献），附上必要的图表和表格 [26] [21]。报告需清晰说明方法、超参数和结果对比，并提交所有代码（无需提交模型文件） [25]。

完成项目所需背景知识

要顺利完成上述项目，你需要掌握以下背景知识：

- **大型语言模型原理**：理解 Transformer 架构及其自注意力机制是关键。典型的 Transformer 由**多头自注意力**和**前馈神经网络**等层堆叠而成，每个模块通过自注意力在序列内部不同标记间分配注意力权重，并通过前馈层进行非线性变换 [27]。这种架构使模型能够高效建模长距离的上下文依赖，从而在生成任务中取得出色表现。基于 Transformer 的大型语言模型通常采用**自回归语言模型**（因果语言模型）的训练范式：给定前文 x，模型预测下一个可能的词y，使得条件概率 $P(y|x)$ 最大 [28]。这种 next-token prediction 过程其实就是一个逐词的多分类问题 [29]。模型通过链式概率分解来产生序列，即 $P(y|x) = \prod_{j=1}^{m}P(y_j \mid x, y_{1:j-1})$ [30]，逐个生成词直到达到终止条件。模型训练则遵循**最大似然估计（MLE）**原则：在给定训练数据集 $D={(x_i, y_i)}$ 时，选择能够最大化数据概率的参数，即最小化负对数似然损失 [31]。公式上等价于最小化 $\sum_{(x,y)\in D}\sum_{j}-\log P_W(y_j|x, y_{<j})$ [32]。无论是大规模预训练还是下游任务微调，本质上都采用相同的目标函数，只是数据集不同 [33]。

- **Tokenization 与 Prompt Engineering**：熟悉 **分词(tokenization)** 过程以及提示设计对于处理文本输入非常重要。在输入送入模型之前，需要利用分词器将原始字符序列切分成 token 序列并映射为对应的 ID [34]。现代 LLM 常采用子词单元的分词算法（如 BPE、WordPiece 等），以兼顾词汇覆盖率与表示效

率：常见词保留为整词，不常见词拆解为有意义的子词 [34] 。高效的分词可以减少序列长度，从而提升推理和训练的速度，降低计算开销 [35] 。**Prompt Engineering** 则是构造有效提示的艺术。一方面，需要根据任务要求设置 User prompt 的内容和风格；另一方面，对于聊天模型，要按照其训练时约定的格式组织对话。例如OpenAI的ChatGPT使用特殊标记区分角色，而本项目中**Qwen**模型采用的是明文的「User: ... Assistant: ...」 格式作为聊天模板。设计提示时可以加入**系统指令**（system prompt）来指定 AI 的身份和语气，或者在用户请求中附加类似“逐步思考”的要求以引导模型展开推理 [36] 。例如，本项目就要求在用户问题后加入 “请逐步推理，并将最终答案用\boxed{}括起来” 的指令，属于**链式思维提示**的一种，能显著提高模型数学推理题目的作答准确性 [37] 。掌握不同提示的风格（如指令式、Few-shot 示例式、Chain-of-Thought 等 [38] ）及其对模型回答的影响，是提升模型性能的有效手段。

- **PyTorch 与 Huggingface 模型微调**：具备使用 **PyTorch** 搭建和训练模型的基本技能，以及熟悉 **Huggingface Transformers** 库来加载预训练模型、分词器和训练管线的经验。在本项目中，你需要读懂并修改提供的 PyTorch 代码。例如，理解 `prepare.py` 如何读取数据并调用 tokenizer，将文本转为模型可接受的张量格式；`finetune.py` 中如何构建 **优化器** 和设置 **训练循环**；以及利用 `torch.cuda` 模块监控显存。Huggingface 提供了高层次的API来加载预训练的 **Qwen-3** 模型和对应分词器，只需提供模型名称或路径即可实例化模型和 tokenizer。还需要掌握 **PEFT** (Parameter-Efficient Fine-Tuning) 工具包的用法，以应用 **LoRA** 等技术对大模型进行高效微调 [11] 。LoRA (低秩适应) 方法的原理是：冻结原模型的大部分权重，仅在某些层（如注意力的投影矩阵或FFN的权重矩阵）旁插入可训练的 **低秩矩阵**，通过训练这些小矩阵来近似更新原权重 [15] 。这样大幅减少了需要更新的参数量和内存占用，同时保持对模型性能的微调效果。实现上，可以使用 Huggingface PEFT 库方便地将 LoRA 模块附加到 Transformer 层中，并使用普通优化器只优化这些 LoRA 层的参数。了解 **AdamW** 和 **SGD** 等常用优化算法的区别也很重要：AdamW对不同参数自适应调整学习率，收敛较快但需要保存一阶/二阶矩；SGD实现简单、泛化性好但往往需要较小学习率和较长时间才能收敛。项目要求比较这些优化器的效果，因此需要懂得如何在代码中切换和配置不同优化器实例（如 `torch.optim.AdamW`、`torch.optim.SGD` 等) [9] 。

- **GPU 使用与性能评估**：由于大模型训练对算力和显存要求极高，必须掌握一些 **GPU 调优技巧**。首先，要能够监控 GPU 显存的使用，知道如何通过诸如 `torch.cuda.max_memory_allocated()` 或 `nvidia-smi` 来获取训练过程中显存占用的峰值，并学会根据显存余量调整 batch size 或采用梯度累积等方式防止 OOM。LoRA 技术可以缓解显存压力，其冻结大部分参数的策略使得微调时所需显存显著降低，这在资源有限的环境下非常实用 [22] 。其次，了解如何测量**训练时间**，可以使用 Python 的时间模块或日志记录每轮(epoch)的时长，以比较不同优化器的速度。项目报告要求提供不同优化器的 **最大显存占用** 与 **训练时间** 对比表 [21] ，因此需要编写小段代码来收集这些信息。在分析时，可以结合优化器原理解释差异来源。例如，AdamW由于保存梯度一阶/二阶矩可能略增内存，但相对于模型本身参数规模，这部分开销并不显著 [23] ；LoRA减少了需要训练的参数数量，因此显存和计算开销最低，可能稍微牺牲一些训练速度换取更低资源占用 [39] 。最后，懂得使用 GPU 的并行计算能力，如利用 **Pin Memory** 加速数据拷贝、适当调整 **num_workers** 提高数据加载效率等，也有助于缩短训练总时间。综合来说，熟练掌握以上技能和知识点，将为顺利完成本项目的模型微调、优化器对比、性能评估与结果分析打下坚实基础。

相似教程和项目资源推荐

为帮助你更直观地参考他人经验，以下是在 **CSDN 博客**和 **GitHub** 上找到的与本项目 Part II 类似的教程或项目资源。这些资源涵盖了 Qwen 模型微调、LoRA 全流程、分词器与聊天模板、以及多优化器训练比较等内容，均以图文形式呈现（无视频），大多为中文：

- **资源1：[(以Qwen为例)大模型训练和微调全流程实战指南：从预训练到对齐，小白也能掌握！]** – 数据准备 & 模型微调：博主详细介绍了大语言模型从预训练到指令微调再到对齐的完整流程，并以 **Qwen** 系列模型为例提供代码模板和配置建议 [40] 。文章特别强调了**数据格式处理**和 **chat模板**应用的关键点，例如如何使用 Huggingface 的 messages 格式构造对话样本，以及保持训练和推理时模板一致来避免标签错

位 [41]。此外，还讲解了 LoRA/QLoRA 等参数高效微调技术的设置（如 `rank` 值、目标模块选择）和**多种优化器**的实践心得。该指南适用于**数据预处理**（了解如何组织指令/回答对话）、**模型训练**（集成 LoRA 等技术）等阶段，对本项目非常有参考价值。

- **资源2：[LoRA 微调 Qwen 模型全流程]** – 完整微调流程：这是一篇**保姆级教程**，从零开始演示如何使用 LoRA 技术微调 Qwen 模型。内容包括：加载预训练的 Qwen 模型和 **Tokenizer** [42]；定义 LoRA 配置（目标模块、秩值、Alpha、Dropout 等）；准备和处理数据集（JSON 格式，含指令和期望输出的样本，并进行标记化和构造 **attention mask** 等 [43] [44]）；创建 PyTorch DataLoader 喂入模型；执行微调（使用 AdamW 优化器微调 LoRA 插入的参数 [45]）；以及保存微调后模型并进行推理测试的全流程。文章还总结了 LoRA 的技术优势，如显著降低训练计算量和显存占用但仍能保持性能 [46]。该资源非常贴合本项目的**模型训练（LoRA）**环节，尤其对如何编写代码实现 LoRA 微调提供了清晰的示例。

- **资源3：[使用LoRA微调LLM的实用技巧]** – 优化器选择 & 性能分析：这是一篇经验分享性质的博文，作者基于开源项目的实验总结了**微调大型模型的一些技巧**和洞见。其中有几条结论与本项目密切相关： [47] ① **优化器的选择对结果影响很小**：无论 AdamW 还是带调度器的 SGD，在LoRA微调下最终性能差异微乎其微（这意味着你在比较SGD/Adam效果时，应更多关注收敛速度等而非最终精度差别）。② **Adam 的显存开销不如想象中大**：虽然Adam会为每个参数存动量，但大型模型的大部分显存还是用于存储权重和前向计算，中小规模下Adam的额外开销对峰值显存影响不明显 [48]。③ **QLoRA 可权衡内存与速度**：比如将模型权重八位量化后再 LoRA，能以约33%显存节省换来39%训练时间增加 [39]。④ **LoRA 参数配置**：建议 LoRA 应用于**所有层**才能最大化效果，而不只是注意力层，并讨论了 LoRA 的秩值和 α 超参数的调优技巧 [49]。这篇文章适合在**训练调优和结果分析**阶段参考，帮助你更加理性地看待不同优化器的比较结果，并提供一些改进训练效率的思路。

- **资源4：[DDA5001-25Fall 项目官方仓库 – Part II 代码与数据]** – 代码实现 & 数据集：这是本课程官方提供的 GitHub 仓库（可能需要访问权限）。其中的 `p2` 文件夹正是 Part II 项目的完整代码和数据。 [50] 该仓库包括**Math500**数据集（训练/测试拆分）、以及项目所需的脚本源码，例如前处理（`prepare.py`）、训练（`finetune.py`）、推理和评估（`rollout.py`，`evaluate.py`）等。在阅读官方代码时，你可以直接查看如何调用 Qwen 模型和其 tokenizer，如何应用聊天模板格式，以及不同优化器（包括 LoRA）的实现细节。这对完成本项目的编码任务大有裨益，也是核对自己实现是否正确的重要参照。建议在着手实现各子任务前先通读相关源码，并结合上述教程加深理解。该资源贯穿**数据准备、模型训练、评估**等所有阶段，能够确保你的实现与项目要求一致。

以上资源在内容上相互补充：既有原理讲解和经验总结，也有具体的代码实例和踩坑提示。建议根据项目进度按需查阅：在数据处理阶段参考资源1关于聊天模板和数据格式的说明，在模型微调阶段参考资源2的代码流程，在对比结果和优化器影响时参考资源3的结论，并随时以资源4的官方代码为准绳进行核对。希望这些材料能帮助你更顺利地完成 DDA5001 第II部分的大模型微调项目！ [3] [9]

---

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [18] [19] [20] [21] [24] [25] [26] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [50] project manual-part II.pdf
file://file_0000000079e8722f8aed088be775b690

[15] 基于 Qwen2 大模型微调技术详细教程（LoRA 参数高效微调和 SwanLab 可视化监控） - 老牛啊 - 博客园
https://www.cnblogs.com/obullxl/p/18312594/NTopic2024071801

[16] [22] [42] [43] [44] [45] [46] lora微调Qwen模型全流程_qwen lora-CSDN博客
https://blog.csdn.net/qq_62231627/article/details/140756188

[17] [40] [41] (以Qwen为例)大模型训练和微调全流程实战指南：从预训练到对齐，小白也能掌握！_可以说一个模型有预训练和领域自适应两个阶段吗-CSDN博客
https://blog.csdn.net/2401_85343303/article/details/151356350

23 39 47 48 49 使用LoRA（低秩自适应）微调LLM的实用技巧 | 闲记算法
http://lonepatient.top/2023/11/30/practical-tips-for-finetuning-llms

27 什么是转换器？— 人工智能中的转换器详解 — AWS
https://aws.amazon.com/cn/what-is/transformers-in-artificial-intelligence/