Jingfeng



现代小模型微调

从监督适配到 LoRA、偏好优化与可验证反馈

Jingfeng · 原稿：2026 年 9 月　修订：2026 年 10 月 22 日

一个已有的小模型无法稳定完成任务时，最有用的第一步往往不是选一个新的微调方法，而是找出失败发生在何处：它缺知识、误解输出格式、没有足够上下文，还是评测本身奖励了错误行为？本书围绕这个问题组织。每一种更新都同时改变模型函数、资源账本和证据要求；把这三者连起来，才能在有限预算下得到可解释的改进。

本书面向具备基础线性代数、概率和 Python 知识的读者。第 1 至 4 章建立任务、张量、数据与 SFT 闭环；第 5 至 6 章讨论低秩与低比特；第 7 至 9 章讨论偏好、在线反馈和不同知识迁移路径；第 10 至 13 章完成评测、方案、眞实微型实验与研究设计。每章的公式依赖显式约定，重要计算配有算例和可核查的练习提示。可以先完成最小实验，再返回机制；但不要跳过数据划分与监督掩码。

> 训练集上的进步是一项观测。只有在版本、监督对象、预算与独立评测都清楚时，它才可能支持一个有边界的结论。

这次修订以 2026 年 9 月原稿为基础，全文重组，技术材料核验截止 2026 年 10 月 22 日，微型训练记录来自 10 月 21 日。引用以原始论文、已有 PDF 与官方文档为主；滚动文档的快照版本与实验锁定依赖分别记录。论文中的性能不是本书实测；算例、资源下界和本次运行结果均有明确标识。实测是固定预训练小模型上的微型任务，旨在检验实现和比较机制，不代表通用能力评测；完整 NF4、在线 RL 和教师蒸馏未在本机运行。

“小模型”是相对可用资源和部署要求的称呼，本书不将某个参数量划为理论边界。微调只研究已有参数的适配；从零训练、架构规模和整体预训练预算由《现代小模型训练》展开。涉及一般后训练与 RL 时，本书给足独立阅读的解释，但不替代对应分册。

可复现入口。本主题的 `manuscript/main.tex` 是书稿入口，`build.sh` 输出独立新版 PDF。

`experiments/` 保存运行代码；实验配置、版本、种子、逐样本预测和结果保存在主题 `output/experiments/`。原始源码与 PDF 仍作为归档保存。版权与来源遵循原始项目及各字体许可；新增说明、算例和修订记录随本项目维护。

目录

第一章 已有模型的任务适配：先决定要改变什么 **1**

1.1 从序列概率到任务行为 1

1.2 先把失败拆成可区分的假设 1

1.3 三个相互独立的选择轴 2

1.4 容量、数据与预算共同限制适配 2

1.5 把基线写成能够重跑的对象 3

1.6 教学算例：工单字段抽取的决策 3

1.7 训练、评测与部署的边界 3

第二章 从聊天文本到梯度：张量、模板与监督对象 **5**

2.1 贯穿全书的形状约定 5

2.2 一个现代 decoder block 5

2.3 chat template 是模型输入协议 6

2.4 下一个 token、shift 与标签掩码 7

2.5 梯度如何经过被冻结的层 8

2.6 手算与边界检查 8

第三章 数据治理与配比：把监督变成可审计资产 **9**

3.1 从任务合同到数据记录 9

3.2 先定义分组边界，再切分数据 9

3.3 去重不是只算一次 hash 10

3.4 污染审计的对象与边界 10

3.5 合成数据：先验证，再扩大产量 10

3.6 难度要与学生能力和长度分开 11

3.7 样本比例、token 比例与损失权重 11

3.8 把课程变成可检查的配置 12

第四章 受控预算下的 SFT 工程：边界、归约与诊断 **13**

4.1 先做小到能够人工检查的闭环 13

4.2 区分序列条数与两本 token 账 13

4.3 变长微批的梯度累积不能盲目取平均 14

4.4 packing：节省空白，还是改变了上下文 15

4.5 截断与 padding 必须保护监督合同 15

4.6 学习率、正则与数值精度 16

4.7 诊断必须能够区分原因 17
4.8 停止、恢复与导出是训练的一部分 17

第五章 参数高效适配：从更新空间到 LoRA 的几何 18
5.1 不同 PEFT 方法限制了什么 18
5.2 固定矩阵约定，先核对前向 18
5.3 梯度、零初始化与缩放 19
5.4 一个同时核对前向、合并和字节的算例 20
5.5 变种一：把方向、幅值和初始化分开 21
5.6 变种二：把训练预算分配给哪里 21
5.7 注入位置、合并与公平消融 22

第六章 QLoRA 与低比特微调：建立可核查的显存账本 23
6.1 量化格式、计算精度与梯度是三个层次 23
6.2 先把所有内存项分开 24
6.3 双重量化节省的是元数据 25
6.4 分页优化器缓解驻留高峰，不能压缩状态 25
6.5 量化残差会占用多少适配能力 26
6.6 QLoRA、PTQ 与 QAT 的边界 26
6.7 从账本到一次可验收的试跑 26

第七章 从成对偏好到 DPO：把目标函数写清楚 28
7.1 偏好标签是一项有条件的观测 28
7.2 KL 正则目标如何给出最优策略 28
7.3 beta 的两种语境不能混为一句经验法则 29
7.4 序列概率、掩码和 reference 缓存 29
7.5 用三个数字理解目标函数 30
7.6 长度与标签噪声的受控诊断 30
7.7 其他偏好目标应按数据与约束选择 30

第八章 PPO、GRPO 与可验证反馈：策略更新的边界 32
8.1 三个策略与三种奖励来源 32
8.2 从对数导数到有用的基线 32
8.3 PPO clip 不是对全部概率变化的硬约束 33
8.4 GRPO 用组内比较替代显式 critic 33
8.5 KL 估计需要注明采样分布 33
8.6 验证器比格式奖励更接近目标，也仍会失效 34
8.7 小模型何时进入、何时退出在线训练 34

第九章 选择适配路径：教师蒸馏、领域知识与持续学习 **35**
9.1 先区分知识缺口、行为缺口与版本管理 35
9.2 蒸馏到底传递哪一种信号 35
9.3 答案、过程与可验证轨迹 36
9.4 On-policy 蒸馏：在学生会到达的位置提供反馈 37
9.5 领域继续预训练：改变分布，不能替代事实核查 37
9.6 持续学习：给新能力和遗忘各留一个位置 38
9.7 多 adapter 的路由、合并与低秩压缩 38
9.8 联邦更新与隐私边界 39

第十章 评测、导出与部署：从分数到可信决策 **40**
10.1 把评测拆成任务收益和代价 40
10.2 统计不确定性和搜索预算 40
10.3 人工与模型评判的偏置 41
10.4 把 adapter 导出成一个可追踪的对象 41
10.5 部署账本：权重小不等于响应快 42
10.6 推测解码与多 adapter 服务的条件 42

第十一章 五套有前提和退出条件的实验方案 **43**
11.1 所有方案共用的入口记录 43
11.2 方案 A：已有小模型的全参数 SFT 与 LoRA 对照 43
11.3 方案 B：NF4 QLoRA 与未量化 LoRA 的资源比较 43
11.4 方案 C：基于固定 SFT 起点的 DPO 44
11.5 方案 D：可验证任务的短期在线优化 44
11.6 方案 E：教师到小模型的分层蒸馏 45
11.7 统一完成契约与决定下一次实验 45

第十二章 算例与最小实验：把公式变成证据 **46**
12.1 实验边界与可复现入口 46
12.2 LoRA 前向、合并与梯度边界 47
12.3 DPO 损失的受控扰动 47
12.4 三臂小模型适配 48
12.5 结果解读与可否定的结论 48

第十三章 从复现走向研究：让下一次实验能够推翻假设 **50**
13.1 把观察变成有干预的机制问题 50
13.2 每个对照只回答一个清楚的问题 50
13.3 研究资产与负结果同样值得保存 50

13.4 本书的边界与下一步 51

附录 A 符号、公式与单位的核查表 **52**

附录 B 十二周阅读与实验路线 **53**

附录 C 复现与交付检查清单 **54**

参考文献与原始资料 **55**

第一章　已有模型的任务适配：先决定要改变什么

拿到一个已有小模型，第一项工作是把目标写成可以检验的输入输出约定。模型要把客服记录转换成固定字段，还是回答某个领域的问题？资料会每天更新，还是主要缺少稳定的表达方式？一条回答失败，是因为不知道事实、没有遵循格式，还是无法完成必要的计算？这些问题对应不同干预。直接开始训练，容易把资料缺失、提示错误和模型容量不足都归结为“微调不够”。

本书把**适配**定义为：在给定基础模型、数据权限与资源限制下，改变模型在目标输入分布上的行为，并验证这种改变是否值得保留。已有模型可以是仅经过预训练的基座，也可以是已经指令微调的模型；两者在对话模板、指令能力和可容忍的学习率上都可能不同。模型参数量只是条件之一。“小”在这里指训练和部署必须认真考虑有限资源，不划定一个适用于所有硬件与任务的参数门槛。

1.1　从序列概率到任务行为

自回归语言模型给 token 序列 $u_{1:L}$ 赋予概率

$$p_\theta(u_{1:L}) = \prod_{t=1}^{L} p_\theta(u_t \mid u_{<t}). \tag{1.1}$$

训练把一些上下文中的目标 token 概率提高；推理还要用解码规则从分布中取出回答。概率分布、解码输出、任务成功率因此是三个不同对象。训练交叉熵下降，可能来自标点与惯用语更熟练，也可能来自真正学会任务；要靠独立评测区分。Transformer 提供了这一概率模型的常见实现，后续架构又改变了归一化、位置编码与注意力组织，但这里的条件概率视角保持适用 [1, 2, 3, 4]。

预训练在大量文本上拟合续写分布；监督微调（supervised fine-tuning，SFT）则通过示范强调某种输入到回答的映射。领域继续预训练使用领域文本继续语言建模；偏好优化使用回答之间的相对判断；可验证反馈用程序或环境定义成功。它们可以串联，也可以省略，并不存在所有任务必须走完的唯一流水线。对已经能够正确完成任务、只缺少可靠解析的系统，约束生成与输出校验可能已经够用。

1.2　先把失败拆成可区分的假设

假设一个模型回答不了某项产品政策。给它准确的政策片段后仍然答错，与不给片段时答错，含义不同。前者可能涉及读懂材料、定位条件或组织答案的能力，后者还包括资料无法从参数中提取。若政策频繁更新，检索与来源展示通常比反复把事实写入参数更容易维护；若它知道答案却总输出错误字段，SFT 更接近问题本身。

这张表是诊断顺序，不是互斥分类。例如工具任务既可能需要格式 SFT，也需要检索资料，还可能使用 LoRA 更新参数。关键是每次干预都能回答一个具体假设，而不是一次替换提示、数据、参数化与解码后，只留下无法解释的总分变化。

表 1.1 从一个可观察失败选择最小干预

观察	优先验证的干预	进入训练前的判据
答案缺少时效事实	提供可靠上下文或检索结果	给定证据时能否正确回答
格式、工具参数不稳定	明确模板、约束解码、少量 SFT	解析失败是否下降，内容是否正确
术语与领域表达陌生	领域语料适配，再做任务示范	领域文本与标注覆盖是否足够
多个可行回答质量不同	人工规则、偏好数据与偏好优化	偏好标准是否一致且可复查
需要搜索、计算或执行	工具调用、蒸馏或可验证反馈	验证器是否可靠，初始成功率是否非零

1.3 三个相互独立的选择轴

训练信号决定为什么更新。SFT 的信号来自目标 token，偏好目标来自回答间比较，强化学习的信号来自采样轨迹及奖励。示范正确不等于覆盖充分，奖励可计算也不等于目标完整。一个只奖励 JSON 可解析的函数，会容忍内容全错的 JSON。

更新空间决定什么可以更新。全参数微调允许较大范围的参数变化；参数高效微调（parameter-efficient fine-tuning，PEFT）只训练适配器、提示向量或低秩因子。LoRA 是一种更新参数化，既能用于 SFT，也能用于偏好训练或强化学习 [5]。它减少可训练状态，并不凭此保证更少遗忘或更高质量。

数值与系统实现决定如何存储和计算。QLoRA 在低比特冻结基座上训练适配器，量化主要改变底座的存储与相应计算路径 [6]。它仍然需要选择训练目标和更新位置。训练时冻结的四比特底座、推理时四比特权重、以及会更新量化权重的训练方法，不能只用“量化微调”四字混为一谈。

把三个轴分开，才会得到明确的实验名称，例如“固定聊天模板，在 BF16 基座上进行回答 token SFT，只训练注意力与前馈层的 LoRA”。单写“用了 LoRA”无法推断它学了什么、用了多少监督或能否在目标机器部署。

1.4 容量、数据与预算共同限制适配

参数规模较小并不自动推出“更容易遗忘”或“评测方差更大”。遗忘取决于更新范围、学习率、任务分布与训练时长；评测方差还取决于样本量、任务难度和解码随机性。合理做法是把这些影响写成待验证的假设，并为它们设计比较。

小容量尤其值得检查两类失配。第一类是任务所需的表示变化超出当前更新空间：训练损失与验证表现都停滞，扩大合适位置的更新容量可能有帮助。第二类是监督复杂度超过模型在预算内能够利用的范围：教师给出很长的推理，学生却只模仿连接词和最终答案位置。此时增加困难样本不一定有效，可能需要更短且正确的示范、外部工具或更合适的底座。数据设计本身就是小模型研究的重要变量，SmolLM2 的工作可作为数据与模型联合考虑的实例，而非任何微调任务的保证 [7]。

可以把一次适配的选择写为约束问题：

$$\begin{aligned} \max_{\mathcal{D},\phi,\mathcal{C}} \quad & M_{\text{task}}(\theta_0,\phi), \\ \text{s.t.} \quad & U_{\text{sup}} \le U_{\max}, \quad M_{\text{peak}} \le M_{\max}, \quad C_{\text{train}} \le C_{\max}, \\ & \Delta_{\text{forget}} \le \delta, \quad C_{\text{serve}} \le C_{\text{serve,max}}. \end{aligned} \tag{1.2}$$

这里 $\mathcal{D}$ 是训练数据，ϕ 是可训练参数，$\mathcal{C}$ 是训练配置；U_{sup} 是实际参与损失的监督 token 次数，重复训练也重复计数。这不是一个已经求解的算法，而是提醒我们：任务分数只是目标的一部分。资

源约束不同，会得到不同的可取方案。

1.5 把基线写成能够重跑的对象

适配前保留一个**冻结基座基线**：权重不更新，在相同评测集上使用固定提示与解码运行。记录模型和 tokenizer 的精确版本、聊天模板、最大输入输出长度、停止条件和评测脚本。若选择检索，还要固定检索库版本及给定上下文。这样才能判断后续变化来自训练还是输入条件改变。

第二个基线是最简单且可负担的训练方案。对于已经具备任务能力但格式不足的模型，可以先在少量经审计样本上做短 SFT；对于基座模型，先检查它能否在少量指令数据上建立基本对话行为。全参数训练可作为一个参照，但只有实际运行并调参充分时才有资格承担“较宽更新空间的对照”；它不是天然上界，更不是无条件的最优解。

训练集、验证集和最终测试集承担不同责任。训练集提供梯度，验证集用于选择数据、超参数与 checkpoint，测试集用于最终报告。持续根据测试错误修改训练集，会把测试变成开发数据；应另建未参与选择的测试版本。第三章会把这一区分落实到去重簇和来源切分。

1.6 教学算例：工单字段抽取的决策

设目标是在本地把设备工单转换为三个字段：设备编号、故障类型、是否需要升级处理。一个已有指令模型在给定字段定义后，能读懂工单，却经常附加解释。这里最先需要验证的是约束输出和格式 SFT，而不是通过继续预训练记住设备知识。

建立三个评测切片：常见故障、罕见表达、缺失信息。前两个切片检查泛化，第三个切片要求输出约定的空值而不是猜测。若罕见型号的升级规则依赖外部手册，就把同一版本手册片段提供给所有比较方法，另测资料定位错误；不要只给训练后的模型更多资料。

假设仅用于教学的预算是每个方案处理 20 000 个监督 token。这不能解释为 20 000 条工单，因为每条回答长度不同。比较冻结基座、LoRA SFT 和低比特基座上的 LoRA 时，保持保留数据、监督 token 预算和评测提示相同，并记录各自的输入 token、峰值内存和耗时。若某个方法需要额外教师生成或量化校准，还要单列这部分成本。这些数字是实验设计示例，不是本次实测结果。

退出条件也应提前规定。例如校验器仍然发现缺失字段、出现把工单文本当系统指令执行的错误，或关键故障的漏判超出容忍范围，就不能因为平均准确率上升而交付。若所有方法都卡在同一种语义错误，应回到任务定义、数据覆盖与模型能力诊断，而不是盲目扩大 rank。

1.7 训练、评测与部署的边界

训练一般以整段序列前向与反向为主；评测生成还涉及解码与停止；部署包含预填充（prefill）、逐 token 生成（decode）、并发及缓存管理。训练吞吐高不能推出首 token 延迟低，可训练参数少也不能推出推理内存低。原始权重、适配器、KV 缓存和临时工作区都可能影响实际成本。

本书后续先解释训练对象，再讨论 PEFT 与量化；之后把偏好、可验证反馈、蒸馏和领域适配放回同一决策框架。部署与评测章节负责检验收益是否保留下来。本书将在小模型适配语境下讲清必要的概率、梯度与策略更新。

练习与提示

1. 为“每天更新政策的问答”和“固定字段的离线抽取”分别写出一种无需参数更新的基线，以及进入 SFT 的判据。**提示：**先区分资料缺失与行为不稳定，再确定如何单独检查两者。
2. 某方法使用两倍训练 token 后分数更高，另一个方法只有一半可训练参数。能否据此称前者更高效？**提示：**分别固定 token、时间或峰值内存，效率结论依赖哪一项资源被固定。
3. 把自己准备进行的一次微调写成“信号、更新空间、数值实现”三项配置，再列出一个可能迫使你放弃训练的观察。**提示：**“使用 LoRA”尚未说明损失函数、数据和退出条件。

第二章　从聊天文本到梯度：张量、模板与监督对象

一条聊天记录并不会直接进入损失函数。它先经过角色序列化、tokenization、截断或拼接，再生成输入张量、注意力约束和监督标签；模型输出 logits 后，训练器还要完成预测位置与目标位置的对齐。任何一步改变了边界，实际学习的任务都可能改变。本章把这条链完整写出，使后面每一次 LoRA 注入和低比特训练都有明确对象。

2.1　贯穿全书的形状约定

用 B 表示一个微批的序列条数，L 表示补齐后的序列长度，d 表示隐藏维度，V 表示词表大小，N 表示 Transformer 层数。B 不是整个训练集大小，L 也不是每条样本实际有效长度。LoRA 章节中的因子矩阵 A, B 有明确矩阵形状，那个 B 与本章批量大小只共享字母，并非同一对象。

表 2.1　一次因果语言模型前向的主要张量

对象	形状	含义
`input_ids`	$B \times L$	token 的离散索引
隐藏状态 H	$B \times L \times d$	各位置的上下文表示
词表投影 W_{vocab}	$V \times d$	从隐藏维度映射到词表
logits Z	$B \times L \times V$	归一化前的预测分数
标签与 loss mask	$B \times L$	目标索引与是否监督
有效目标数 S	标量	shift 后参与损失的位置数

本书在线性层中统一把权重写成“输出维度乘输入维度”：$W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$，行向量输入按 $Y = XW^{\top}$ 计算。这与许多深度学习库的权重存储一致。若阅读另一篇文献时看到 $Y = XW$，先转换约定，再比较梯度；转置不同不等于方法不同。

2.2　一个现代 decoder block

将上一层输出 $H^{(\ell)}$ 先归一化，再做注意力与残差，最后进入前馈层：

$$\widetilde{H}^{(\ell)} = H^{(\ell)} + \mathrm{Attn}\big(\mathrm{Norm}(H^{(\ell)})\big), \tag{2.1}$$

$$H^{(\ell+1)} = \widetilde{H}^{(\ell)} + \mathrm{FFN}\big(\mathrm{Norm}(\widetilde{H}^{(\ell)})\big). \tag{2.2}$$

这是一种常见的 pre-norm 结构，不意味着每个模型都完全相同；实际微调应读取模型配置中的层类型、维度、偏置与权重绑定关系 [1, 2, 3]。

设 query 头数为 h_q，key/value 头数为 h_{kv}，每头维度为 d_h。常见实现有 $d = h_q d_h$，投影权重为

$$W_Q \in \mathbb{R}^{h_q d_h \times d}, \qquad W_K, W_V \in \mathbb{R}^{h_{kv} d_h \times d}. \tag{2.3}$$

这里 W_V 的下标表示 value，与词表大小 V 不同。投影并重排后，Q 的形状是 $B \times h_q \times L \times d_h$，$K, V$

的形状是 $B \times h_{kv} \times L \times d_h$。多头注意力（MHA）有 $h_q = h_{kv}$；多查询注意力（MQA）取 $h_{kv} = 1$；分组查询注意力（GQA）让若干 query 头共享一组 key/value，通常要求 h_q 能被 h_{kv} 整除。

对一个 query 头及其关联的 key/value 头，有

$$P_{b,h} = \mathrm{softmax}\left(\frac{Q_{b,h}K_{b,g(h)}^{\top}}{\sqrt{d_h}} + M_b\right), \qquad O_{b,h} = P_{b,h}V_{b,g(h)}. \tag{2.4}$$

softmax 沿 key 位置进行，$P_{b,h} \in \mathbb{R}^{L\times L}$。将各头输出拼接后乘 $W_O^{\top}$，得到 $B \times L \times d$。因果 mask 在 $j > i$ 时禁止 query 位置 i 读取 key 位置 j；padding key 也必须被排除。后面独立样本的 packing 还会增加样本边界约束。它们共同决定“允许看见什么”，不决定“哪些位置计算损失”。

注意力实现未必显式存储整个 $L \times L$ 概率矩阵，FlashAttention 等方法通过分块和重计算降低访存及中间存储，而不是取消因果关系 [8]。因此理论张量形状用于理解计算，并不等于训练时每一项都作为完整数组常驻内存。

2.2.1 归一化、门控与位置

RMSNorm 对每个位置的 d 维向量做

$$\mathrm{RMSNorm}(x) = \frac{x}{\sqrt{d^{-1}\sum_{j=1}^{d} x_j^2 + \epsilon}} \odot g, \qquad g \in \mathbb{R}^d. \tag{2.5}$$

门控前馈层的一种常见形式是 SwiGLU：

$$\mathrm{FFN}(x) = \left[\mathrm{SiLU}(xW_g^{\top}) \odot (xW_u^{\top})\right]W_d^{\top}, \tag{2.6}$$

其中 $W_g, W_u \in \mathbb{R}^{d_{\mathrm{ff}}\times d}$，$W_d \in \mathbb{R}^{d\times d_{\mathrm{ff}}}$。两个升维分支与一个降维分支都会影响适配器参数量，不能把“全线性层”理解为只包括四个注意力投影。

旋转位置编码 RoPE 对 query、key 的二维子空间做随位置变化的旋转，使点积包含相对位置关系。它没有把任意长输入都变得可靠；位置缩放方案、训练长度与注意力窗口仍会限制长上下文行为。训练 packing 时是否重置位置索引，也必须与注意力隔离和部署使用方式一起检查。

2.3 chat template 是模型输入协议

聊天模板（chat template）把角色与文本转换成模型训练时使用的控制 token 序列。例如抽象的单轮消息可能写成

```
BOS USER question END ASSISTANT answer EOS。
```

这里的单词仅是教学占位，不是应复制给任何模型的真实 token。模型可能使用不同的角色终止符、轮次终止符、工具消息格式，以及不同的 system 处理方式。先读取该模型 tokenizer 的实际模板，并保存模板文本与版本 [9]。

训练通常提供完整的 assistant 回答，不添加一个让模型继续生成的新 assistant 起始提示；生成评测则通常需要把 prompt 截止在待回答的位置。对支持相应接口的 tokenizer，训练序列化可显式设定 `add_generation_prompt=False`。如果先生成字符串、再单独 tokenization，要确认模板已经插入的 BOS/EOS 不会被 `add_special_tokens` 再添加一次。不同模型的模板约定不能凭经验互换。

多轮训练还要明确监督范围：可以监督每一轮 assistant 消息，也可以只监督最后一轮，但这两者是不同目标。工具输出通常是上下文而非 assistant 的可生成目标；若要训练工具调用，调用语法

和参数属于哪个角色、是否监督，都应从 schema 明确派生。不要用搜索某个角色字符串的方式定位标签边界，用户内容本身可能包含该字符串。

支持角色级 assistant mask 的模板可以减少手工标注，但仍需打印实际 token 检查。不支持时，应在结构化序列化阶段记录片段与 token 的对应关系；分段单独 tokenize 后再拼接可能受子词边界影响，不能假定必然等于对完整字符串 tokenize 的结果。

2.4 下一个 token、shift 与标签掩码

设完整输入为 $u_{b,0}, \dots, u_{b,L-1}$。位置 t 的隐藏状态已经读过 $u_{b,t}$，其 logits $z_{b,t}$ 预测的是 $u_{b,t+1}$。定义 $m_{b,j}$ 表示输入位置 j 上的 token 是否属于目标回答，右移后的 token 平均交叉熵为

$$S = \sum_{b=1}^{B} \sum_{t=0}^{L-2} m_{b,t+1}, \tag{2.7}$$

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{S} \sum_{b=1}^{B} \sum_{t=0}^{L-2} m_{b,t+1} \log p_\theta(u_{b,t+1} \mid u_{b,0:t}). \tag{2.8}$$

只对 $S > 0$ 的批量定义这项平均。用 -100 作为忽略标签，是许多训练器与交叉熵实现的约定；它不是词表中的一个 token[10]。输入中的实际 token 仍是正常索引，不能把 `input_ids` 改成 -100。

表 2.2 教学序列中，回答与 EOS 是目标，提示与 PAD 不监督

输入位置	0	1	2	3	4	5	6	7
token	BOS	USR	Q	END	AST	A	EOS	PAD
m_j	0	0	0	0	0	1	1	0

表 2.2 中，logits 位置 4 预测回答 A，位置 5 预测 EOS；不是位置 5 的 logits 预测它已经读过的 A。将 labels 对提示和 PAD 设为忽略值后，代码必须同步使用 `logits[:, :-1]` 与 `labels[:, 1:]`。许多模型在接收 `labels` 时已经内部 shift，使用这种接口就不能再预先 shift 一次。

Listing 2.1 显式 shift 的 token 平均损失

```
import torch.nn.functional as F

targets = labels[:, 1:]
pred = logits[:, :-1, :].float()
n_supervised = targets.ne(-100).sum()
assert n_supervised.item() > 0
loss_sum = F.cross_entropy(
    pred.transpose(1, 2), targets,
    ignore_index=-100, reduction="sum",
)
loss = loss_sum / n_supervised
```

损失掩码与注意力掩码有不同作用。user token 不计损失，不表示模型不能读取它；PAD 不计损失，也不表示它自动从注意力中消失。若 EOS 与 PAD 共用 token ID，不能通过 `input_ids == pad_id` 一律删掉监督，否则真实 EOS 也会被屏蔽。应依靠真实长度或位置级 padding mask 保留结束信号。

2.5 梯度如何经过被冻结的层

若单个受监督位置的预测分布是 p，真实索引是 y，则

$$\frac{\partial \mathcal{L}}{\partial z_{b,t,k}} = \frac{m_{b,t+1}}{S}\big(p_{b,t,k} - \mathbf{1}[k = u_{b,t+1}]\big). \tag{2.9}$$

这条式子既包含熟悉的 $p - y_{\text{onehot}}$，也包含掩码与归一化分母。一个困难 token 的正确类概率较低时，其正确类梯度幅度较大；但它是否对共享参数产生更大有效更新，还要看中间层的雅可比和与其他 token 的梯度抵消，不能只凭预测错误率决定重要性。

把一个线性层的前两维合并，写成 $X \in \mathbb{R}^{BL\times d_{\text{in}}}$、$Y = XW^{\top}$。若 $G = \partial \mathcal{L}/\partial Y$，则

$$\frac{\partial \mathcal{L}}{\partial W} = G^{\top}X, \qquad \frac{\partial \mathcal{L}}{\partial X} = GW. \tag{2.10}$$

冻结 W 只是不积累或更新第一项，不会自动取消第二项。上游仍可能有适配器需要梯度，所以不能把包含冻结层的整个前向随意包在 `no_grad` 中。PEFT 节省的是大量权重梯度和优化器状态，反向路径及所需激活仍可能存在。

同样，提示位置的 loss mask 为零，也不意味着提示 embedding 对损失没有影响。回答通过注意力读取提示，反向梯度会沿读取路径回到相关表示。区分“该位置没有直接预测目标”与“该位置不参与计算图”，可以解释为什么错误的注意力边界会改变训练结果。

2.6 手算与边界检查

沿用表 2.2，若模型对 A 的概率为 0.8，对 EOS 的概率为 0.6，两项 token 平均损失是

$$-\frac{\log 0.8 + \log 0.6}{2} \approx 0.366985. \tag{2.11}$$

两个正确类 logits 的导数分别是 $(0.8-1)/2 = -0.1$ 和 $(0.6-1)/2 = -0.2$。这是合成教学算例，可直接作为实现的独立参考值。若误把 PAD 加入分母，损失会被稀释；若遗漏 EOS，模型就少了一项明确停止的监督。

GQA 的形状也可以手算。令 $B = 2, L = 8, d = 64, h_q = 8, h_{kv} = 2, d_h = 8$，则 Q 有 $2\times 8\times 8\times 8$ 个元素，K 与 value 各有 $2\times 2\times 8\times 8$ 个元素，W_K 是 16×64。若共有 N 层，推理时每层缓存 K 与 value，长度为 L_{ctx} 的 B_{serve} 个请求需要的缓存元素数近似为

$$N_{\text{KV}} = 2NB_{\text{serve}}L_{\text{ctx}}h_{kv}d_h. \tag{2.12}$$

再乘每元素字节数才得到内存；服务实现的分页、对齐及元数据另算。训练时的激活不能直接用这条推理缓存式代替。

练习与提示

1. 给表 2.2 再添加一轮 user 与 assistant，分别写出“监督所有回答”和“只监督最后回答”的 labels。**提示：**先按角色定位输入位置，再做一次 shift。
2. 在 EOS 与 PAD 共用 ID 时，如何证明没有屏蔽真实 EOS？**提示：**构造一个有真实 EOS、右侧还有两个 PAD 的序列，比较按 ID 屏蔽与按实际长度屏蔽的结果。
3. 某层 X 为 6×4、W 为 5×4。写出 Y、G、$G^{\top}X$、GW 的形状，并解释冻结 W 后为什么后两项的处理不同。**提示：**参数是否需要梯度与输入是否需要梯度是两个条件。

第三章　数据治理与配比：把监督变成可审计资产

训练数据决定模型在哪些条件下被要求预测哪些 token。所谓“高质量”，必须落实为正确性、任务覆盖、标签一致性和来源可追踪；所谓“有难度”，也要区分真正需要推理的题目与错误、歧义或缺失条件的样本。本章把数据从一个下载地址变成一组能够复查、重建和比较的对象。

3.1　从任务合同到数据记录

先写任务合同：允许的输入是什么，回答应包含什么，缺失信息如何处理，哪些错误代价最高，评测如何确定成功。工单抽取中的“未知”与“否”并不是同一个标签；把两者混用，训练会产生无法靠学习率解决的冲突。

原始记录至少包含稳定 ID、来源、采集或生成时间、许可与使用范围、原始内容、角色化消息、任务标签，以及校验结果。合成数据还要记录教师、提示模板、生成参数与过滤过程。清洗不应覆盖原始记录：保留原始层、结构化层、可训练层与剔除原因，才有可能定位某个训练 token 来自哪里。

Listing 3.1　一条可训练记录的元数据骨架

```
record = {
    "id": "sample-00017",
    "source_id": "source-v1",
    "group_id": "document-or-template-family-9",
    "messages": [],
    "task": "structured_extraction",
    "license": "recorded-in-source-manifest",
    "teacher_revision": None,
    "verification": {"schema": True, "answer": True},
    "split": "train",
    "template_sha256": "...",
    "input_tokens": 0,
    "supervised_tokens": 0,
    "filter_reasons": [],
}
```

字段里的 token 数应在最终 tokenizer、模板和截断规则下重新计算。字符长度、教师 token 数和学生 token 数都不能互相替代。若涉及工具调用，还应校验调用标识与返回消息能对应，工具参数满足 schema，失败返回与成功返回都得到恰当处理；仅检查角色属于一个合法集合还不够。

3.2　先定义分组边界，再切分数据

随机把所有行分成训练、验证和测试，很容易把同一文档的不同片段、同一道题的改写、同一模板的数字替换分进不同集合。这时测试看似没有完全相同的字符串，仍可能只是在考模型记住同一种结构。

较可靠的顺序是：确定部署所需的泛化边界，识别文档、实体、时间、来源和近重复簇，再按这些组分配 split。测试新文档就按文档切分，测试未来数据就按时间切分，测试新表达就保留模板族。分组能同时发生：一份文档及其所有人工改写、摘要与合成问答应共享足够严格的来源组。若

任务必须在相同实体上预测未来事件，也不必机械地把实体完全隔离，但应明确它测的是时间泛化而非新实体泛化。

训练集负责提供梯度，验证集负责选择配比与阈值，最终测试集不参与筛选决策。数据生成前就冻结测试来源名单，生成后仍检查近重复；只要求教师“不要生成测试题”不构成污染控制。

3.3 去重不是只算一次 hash

精确去重可对规范化文本计算哈希。规范化应服务于任务：统一换行与无意义空白通常合理，删除代码缩进、单位、否定词或设备编号却可能改变答案。保存原文哈希和规范化版本，避免用过度清洗制造伪重复。

近重复检测可以分级进行：字符或 token 的 *n*-gram 重合、MinHash 候选召回，再用文本相似度与来源信息复核。embedding 能帮助发现语义改写，但“同类任务”不一定是重复；两个不同病症的处理步骤可能很像，直接按相似度阈值删除会丢掉关键差异。阈值要通过抽查候选对来校准。

冲突检查与重复检查同样重要。同一个提示的两个回答可能有等价表达，也可能是真正矛盾。对可验证任务先执行答案校验；不可自动判断时保留冲突标记与人工裁决。把互相矛盾的样本留在集合中，模型可能只学会输出频率更高的版本。

去重之后还应按簇计数。一个来源虽然有一万行，却可能只有几十个独立模板；它对覆盖范围的贡献不能用行数表达。记录每个来源的原始条数、精确去重后条数、近重复簇数、剔除原因与最终监督 token，是判断“更多数据”是否真正增加信息的基础。

3.4 污染审计的对象与边界

污染是训练或数据选择过程接触了本应保留用于独立评测的信息。精确包含整道测试题只是最容易发现的一种；题干改写、公开答案解析、同一代码题的替换变量版本，以及教师对测试题的回答，也可能破坏独立性。

对有权限使用的评测材料，建立题干、答案、长片段与来源 URL 的审计索引，检查所有训练来源及合成候选。长文本可分窗口搜索局部重合；短题和常识题重合率天然更高，需要人工判断，不能把一个相似度数字当作污染事实。记录候选、裁决依据和处理方式，比宣称“已经去污染”更有价值。

对不能公开题面的私有测试，可以由受控脚本返回重合标记，或在有权限的环境中做全文检查；哈希匹配只能发现完全一致的输入，不能证明不存在语义重合。即使自己的微调集没有发现泄漏，也通常无法证明基础模型预训练集绝无该题。因此应分别报告**本次适配数据的审计结论**与**基座预训练信息未知的限制**。

如果发现污染，优先在来源组层面剔除或隔离，并重新生成版本与预算统计。不能只删掉测试题的最后答案、保留整段解题过程后仍视为独立。最终报告还要标明是否曾利用该测试集选择数据；这类信息暴露不是后来删几行训练样本就能撤销的。

3.5 合成数据：先验证，再扩大产量

Self-Instruct 展示了用模型生成候选指令、扩展任务覆盖并进行筛选的路线；Magpie 利用对齐模型的模板机制合成指令数据 [11, 12]。这些工作支持“可以构造候选监督”，并不说明任意教师生成的回答都正确。LIMA 的研究也不意味着越少数据越好，其结论应放在具体基础模型、数据构成与评测条件下理解 [13]。

一条合成记录可以经过四层过滤。第一层检查可解析性、角色、引用与工具格式；第二层检查答案或程序执行结果；第三层检查指令与回答是否一致、条件是否完整；第四层检查与已有数据的重复和来源覆盖。对无法自动验证的事实，应使用可靠资料或人工抽查，避免让生成答案的同一个模型仅凭自身偏好再给自己确认。

表 3.1 不同监督信号的可验证部分与盲点

样本类型	可执行检查	仍需留意
结构化抽取	schema、字段值与原文对齐	合法 JSON 仍可能填错事实
算术与规则推导	独立计算器、约束检查	仅答案正确不能证明过程正确
代码	单元测试、超时、资源限制	测试覆盖不足、硬编码答案
领域问答	来源定位、关键条件核对	引用存在不等于引用支持答案
开放写作	指令约束、人工量表	文风偏好不能代替事实判断

保存每一层的通过率，按来源、难度与长度抽查保留集和剔除集。只抽查保留数据，无法估计过滤器错误删掉多少有用样本。教师、过滤 judge 与最终评测 judge 若共享偏好，学生可能只学得更像同一种评判器；至少保留一部分人工可核查的独立标准。

教师调用次数、生成输入输出 token、候选数、过滤率与最终保留 token 都属于数据成本。一个方法如果生成十个候选选一个，就不能与只调用一次教师的方法仅按学生训练量比较。

3.6 难度要与学生能力和长度分开

可用基础学生的解题成功率、每个监督 token 的负对数似然、必要推理步骤或人工分级描述难度，但这些量不等价。低概率可能来自陌生的名字或 tokenizer 的罕见切分；长回答可能只是更啰嗦；教师写了许多步骤，也不说明学生必须或能够学习它们。

先用答案验证把“正确但难”与“错误或歧义”分开，再观察学生在各难度桶的训练与验证曲线。如果困难桶训练损失下降，但未见模板的成功率不变，可能只学到了语言表面。如果训练损失始终不降，先检查 mask、标签和可用上下文，再判断是更新容量或模型能力不足。不能把最高损失样本一律赋予最高权重。

长度需要独立统计：输入长度、回答长度、监督 token 数、模板开销、各分位数、超窗比例，以及被删掉的是提示、过程、答案还是 EOS。对学生容量较小的推理任务，可以比较正确短解、正确长解与仅答案三个版本；固定监督预算并对照最终成功率，才能判断过程信息是否值得成本。

3.7 样本比例、token 比例与损失权重

设每次先以概率 p_k 选数据源 k，该源样本经过最终预处理后，平均含有 $\bar{s}_k$ 个监督 token。若对所有监督 token 等权，源 k 贡献的监督比例近似为

$$q_k = \frac{p_k \bar{s}_k}{\sum_j p_j \bar{s}_j}. \tag{3.1}$$

要达到目标 token 比例 q_k，可以以

$$p_k = \frac{q_k / \bar{s}_k}{\sum_j q_j / \bar{s}_j} \tag{3.2}$$

作为采样起点，再用训练中真正出现的监督 token 校验。截断、拒绝无效样本、packing 和变长采样都会使实际比例偏移。较稳妥的实现是对各源维护 token 计数器，在接近预算时按剩余额度调度，而

不是只在启动配置里写一串概率。

教学算例：两个源各抽一半样本，平均监督长度分别为 20 和 100，则第二个源贡献 100/(20 + 100) = 5/6 的监督 token。若希望二者各贡献一半，短回答源应按 5/6 的样本概率抽取，长回答源按 1/6 抽取。数值是合成示例，不是对任何数据集长度的估计。

另一种做法是逐源计算平均损失再组合：

$$\mathcal{L} = \sum_{k=1}^{K} \lambda_k \mathcal{L}_k, \qquad \lambda_k \geq 0, \quad \sum_k \lambda_k = 1. \tag{3.3}$$

这时 λ_k 是目标函数权重，不等于抽样概率，也不必等于实际 token 比例。逐样本平均还会使短回答的单个 token 权重更高。可以使用这些目标，但必须命名、记录和验证，不能在不同实验中悄悄切换归约方式。

3.8 把课程变成可检查的配置

“由易到难”只有写出难度定义、时间轴、token 配额与保留数据才是实验方案。FLAN 与开放后训练配方展示了任务混合的重要性，但具体比例仍需由目标任务验证 [14, 15]。一种**待验证的教学方案**可把总监督预算分成 20%、60%、20% 三段：前段以明确格式和中等任务为主，中段加入困难样本，末段继续保持简单与保留样本，而不是完全替换成最难题。

表 3.2 按监督 token 定义的示例课程，各行比例和为 100%

预算阶段	简单任务	中等任务	困难任务	通用保留
前 20%	50%	20%	0%	30%
中 60%	25%	35%	15%	25%
末 20%	20%	35%	20%	25%

表 3.2 不是推荐的通用最优比例。检验课程效果时，应让固定混合基线看到相同来源、相同总监督 token 和相同重复次数分布，尽量只改变顺序。若课程方案同时多看了困难题，就无法把收益归因于顺序。每个阶段的退出条件可以是格式通过率达到预设要求，或者验证指标停滞且通用能力保持在容忍范围；阈值应提前约定。

最后冻结一份数据清单：原始来源版本、样本 ID 与 split、过滤器版本、模板哈希、tokenizer 版本、预处理参数、各源统计和随机种子。生成训练文件的脚本应能从这份清单重建同一数据；重新生成教师回答通常不保证逐字相同，所以应保存已采用的候选输出。

练习与提示

1. 同一篇文档生成了十个问答，随机划分后只有一个问答进入测试。为什么精确去重不能证明测试独立？**提示：**区分字符串重复与来源信息共享。
2. 三个源的目标监督比例为 0.5, 0.3, 0.2，平均监督长度为 25, 50, 100。计算样本采样概率。**提示：**先算 $q_k/\bar{s}_k$ 再归一化，结果约为 0.7143, 0.2143, 0.0714。
3. 为一个合成代码集设计过滤与抽查方案，至少包含保留集和剔除集。**提示：**测试通过率、错误保留率与错误剔除率回答不同问题；同时报告教师生成成本。

第四章　受控预算下的 SFT 工程：边界、归约与诊断

可靠的训练闭环不是“损失在下降”，而是能够说明数据经过怎样的处理、每次更新使用了多少监督，以及生成行为为什么发生改变。先把最小闭环跑通，再提高吞吐；否则 packing、混合精度与梯度累积会同时掩盖错误。本章给出一个可以迁移到全参数、LoRA 与低比特训练的共同工程基线。

4.1 先做小到能够人工检查的闭环

从已经审计的训练集取少量样本，保留每条消息、最终 token、labels 与实际监督片段。第一步只做前向：检查维度、有限数值、监督位置与分母。第二步做一次反向：检查应训练参数有梯度、冻结参数没有被加入优化器，并确认梯度不是全部为零。第三步在这批数据上短暂过拟合，验证模型确实能够提高目标回答的概率并生成预期格式。

小样本过拟合是流水线检查，不是泛化实验。若连几个正确样本都拟合不了，应优先排查模板、shift、mask、优化器参数列表和学习率，而不是立刻增加数据。若能拟合，再切换到独立验证集；训练与验证都应使用同一明确定义的 masked loss，同时保留任务级的解析或执行指标。

配置至少固定模型与 tokenizer 版本、模板哈希、数据版本、种子、精度、序列长度、监督规则、更新参数名、优化器、学习率调度及保存间隔。为了复查最初的行为，还要保存训练前基座的生成结果。更换模板后重新评测的分数不能直接归因于权重更新。

4.2 区分序列条数与两本 token 账

在每设备微批均为 B_{device}、设备数为 D、每次更新累积 G 个微批且没有残缺微批时，序列条数为

$$B_{\text{eff}} = B_{\text{device}} D G. \tag{4.1}$$

packing 后一条序列又可能包含多个原始样本，因此 B_{eff} 也不能解释为独立示范条数。训练记录应同时维护两本账：

$$U_{\text{in}} = \sum \text{非 padding 输入 token 次数}, \tag{4.2}$$

$$U_{\text{sup}} = \sum \text{实际参与损失的目标 token 次数}. \tag{4.3}$$

前者更接近序列处理成本，后者描述监督曝光量；两者都将重复 epoch 重复计数。词表、序列长度、注意力实现和反向路径仍影响成本，所以相同 token 数也不保证相同时间。

在比较两个训练方案时，固定相同的保留数据、监督规则与 U_{sup}，并另外报告 U_{in}、更新次数、墙钟时间和峰值内存。若一个方案在更长提示上训练，同样的监督量也可能需要更多计算。训练预算控制与系统效率测量承担不同责任，应同时保留。

4.3 变长微批的梯度累积不能盲目取平均

第 g 个微批有监督 token 数 S_g、损失和 ℓ_g。目标 token 平均损失是

$$\mathcal{L}_{\text{update}} = \frac{\sum_{g=1}^{G} \ell_g}{\sum_{g=1}^{G} S_g}. \tag{4.4}$$

如果先计算各微批平均 ℓ_g/S_g 再除以 G，得到的是微批等权目标。只有各 S_g 相同时，两者才相等。**教学算例：**两个微批分别有 2 与 6 个监督 token，平均损失为 1 与 3；正确 token 平均为 $(2+18)/8 = 2.5$，微批平均却是 $(1+3)/2 = 2$。数值差异会直接改变梯度方向与各数据源的权重。

下面的骨架在**单进程、没有 loss scaling** 的前提下，先累积损失和的梯度，再按本次更新的真实监督数归一化。`update_windows` 负责把数据组织成每次更新的微批列表；模型、优化器和调度器在外部按配置创建。代码展示归约规则，不包含分布式包装或底座加载。

Listing 4.1 按真实监督 token 归一化的累积骨架

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

trainable = [p for p in model.parameters() if p.requires_grad]
for window in update_windows:
    optimizer.zero_grad(set_to_none=True)
    supervised = 0
    for batch in window:
        labels = batch["labels"][:, 1:]
        count = int(labels.ne(-100).sum())
        assert count > 0
        with torch.autocast(
            device_type=device_type,
            dtype=torch.bfloat16,
            enabled=use_bf16,
        ):
            logits = model(
                input_ids=batch["input_ids"],
                attention_mask=batch["attention_mask"],
            ).logits
        loss_sum = F.cross_entropy(
            logits[:, :-1, :].float().transpose(1, 2),
            labels, ignore_index=-100, reduction="sum",
        )
        loss_sum.backward()
        supervised += count
    assert supervised > 0
    for p in trainable:
        if p.grad is not None:
            p.grad.div_(supervised)
    torch.nn.utils.clip_grad_norm_(trainable, max_norm=1.0)
    optimizer.step()
    scheduler.step()
```

最后一个不足 G 的窗口仍按自身真实监督数归一化。对较大的损失和，应注意梯度累计范围；也可以先知道整个窗口的 $S = \sum_g S_g$，每次反传 ℓ_g/S，得到相同目标。二者的浮点舍入次序可能不同，应以适当容差检查，而非要求逐 bit 相等。

分布式训练还要核对梯度归约。若 DDP 将各设备梯度取平均，要得到全局 token 平均，应结合全局监督数和设备数处理缩放；不能在每张卡独立按本地 S 平均后就认为全局正确。最小检查是关

闭 dropout，将同一批数据一次完整计算，再分成不同长度微批或不同设备计算，比较可训练参数梯度。训练框架若已经实现 token 归约，应先确认其版本行为，避免再归一化一次。

4.4 packing：节省空白，还是改变了上下文

把短样本拼接到长序列可以减少 padding，但拼接存在两种不同语义。**连续文本训练**可以有意让后一个片段读取前文；**独立指令样本 packing** 则通常希望保持逐样本训练的条件概率，必须阻止跨样本读取。二者都可以定义为合法目标，但不能当作严格等价的加速开关。

在标准因果注意力下，后面的样本能看到前面的样本，前面的样本不能看到未来。EOS 是一个输入 token，**不会自动隔离注意力**；loss mask 也不会。对独立 packing，query 位置 i 能读取 key 位置 j 的条件应至少包括

$$j \le i, \qquad \text{segment}(i) = \text{segment}(j), \qquad i, j \text{ 均为真实位置.} \tag{4.5}$$

显式 block mask、支持序列边界的变长注意力接口等都能表达这种限制；仅提供一个二维“非 PAD 为 1”的 mask，通常表达不了样本间隔离。具体 kernel 是否支持所需 mask，必须由真实前向验证。

Listing 4.2 无 padding 的最小 packing 边界断言

```
import torch

segment = torch.tensor([0, 0, 0, 1, 1, 1])
pos = torch.arange(len(segment))
causal = pos[None, :] <= pos[:, None]
same = segment[:, None].eq(segment[None, :])
allowed = causal & same
assert allowed[2, 0]
assert not allowed[0, 2]
assert not allowed[3, 2]
assert not allowed[2, 3]
assert allowed.any(dim=-1).all()
```

这段代码只验证逻辑，不证明模型实际采用了该 mask。进一步应选两个短样本，在关闭 dropout、重置逐样本位置索引、相同监督规则下，比较分别前向与 packing 前向的逐 token logits 或损失和。再交换样本顺序：若独立样本的输出明显改变，应检查跨边界注意力、位置编码或 kernel 接口。

边界第一个 token 也不能从上一个样本的末位 logits 获得监督。常见做法是保留各样本的起始控制 token，不监督其第一个位置；assistant 标签依据原始片段映射，不能在拼接之后用一个大区间覆盖所有内容。位置索引重置本身不能替代注意力隔离。含 padding 的实现还应避免给 padded query 生成全负无穷 softmax 行导致 NaN，可使用框架支持的安全处理或排除这些 query，且始终忽略其损失。

4.5 截断与 padding 必须保护监督合同

截断不是简单地把长度限制为一个数。右截断可能删去最终答案或 EOS；左截断可能删掉 system 约束、用户问题与必要证据；多轮截断还可能留下没有对应调用的工具返回。应在消息与任务字段层面制定优先级：保留哪些约束、允许移除哪些历史、超长回答是剔除还是另作长序列任务。所有方案都要统计被截字段与剩余监督量 [16]。

一个基本边界是：预处理后的每条可训练序列至少有一个可预测的监督 token，且至少保留完成目标所需的输入。对于推理示范，保留开头过程却截掉答案，会让训练目标偏向“开始解释”而不

是完成任务。可以专门研究部分轨迹训练，但必须另行定义目标，不应将其悄悄算作完整示范。

padding 用来凑齐张量形状，不增加数据。训练中应同时设置 padding 注意力约束和 `labels=-100`；只做其中一项仍可能出错。右 padding 常便于训练检查，生成批处理则需遵循模型及生成器的 padding 约定，不能从训练惯例推断。左 padding 时也要检查 position IDs、真实起点与 shift 后的第一个监督位置。EOS 与 PAD 共用 ID 的情况按第二章的位置级规则处理。

预处理验收可以保存每种边界的一个小样本：空回答、只有 EOS 的回答、长提示、长回答、两轮对话、工具失败、EOS 与 PAD 共用 ID、跨样本 packing。它们比随机抽到的一段流畅文本更容易暴露错误。

4.6 学习率、正则与数值精度

先从模型或所用实现明确记录的可行配置出发，再围绕学习率做少量对数间隔试验，例如参考值的约三分之一、一倍和三倍。这个搜索设计不承诺哪个点最优。全参数、LoRA、不同初始化和量化底座的合适范围可能不同；LoRA 的 rank 与缩放还会改变实际更新尺度，因此不能只比较名义学习率。

warmup 让早期更新逐渐增大，调度器通常按优化器更新步推进；如果每步监督量变化很大，还要记录对应 token 进度。梯度裁剪放在归一化之后；若使用 FP16 loss scaling，应先正确 unscale 再裁剪，且仅在优化器真正更新时推进调度器。BF16 具有不同的指数范围，通常不依赖 FP16 式动态缩放，但仍可能出现非法数值、坏数据与不稳定更新。不能看到 NaN 就统一归因为“BF16 溢出”。

AdamW 的 weight decay、dropout 和早停都可以控制过拟合，但需要明确作用参数和验证依据。冻结 embedding 或词表输出层能减少可训练状态，却可能妨碍新控制 token 的学习；新增 token 时必须处理 embedding 扩展、输出权重以及权重绑定关系，不能只把 tokenizer 文件加进导出目录。

4.6.1 NEFTune 与其他目标改动

NEFTune 在训练 embedding 上添加噪声，常见参考形式为

$$\tilde{e} = e + \frac{\alpha}{\sqrt{L_{\text{noise}} d}} \epsilon, \qquad \epsilon_j \sim \mathcal{U}(-1, 1). \tag{4.6}$$

其中 L_{noise} 反映实现采用的序列长度，原参考实现由 embedding 张量的序列维与隐藏维确定缩放 [17]。变长、padding 与 packing 会改变这个长度的含义，所以应记录实际实现、噪声强度与采样位置；只写 $\alpha/\sqrt{d}$ 不是同一个缩放。推理应关闭噪声。论文中的收益不能保证迁移到任意小模型和数据，需与无噪声基线比较。

混入通用文本可以作为保留能力的干预，但原始文本的全 token 语言建模与聊天数据的 assistant-only SFT 有不同监督范围。第三章的配比账本应分别记录，再明确组合权重；领域继续预训练与任务 SFT 的不同作用见领域适配章节 [18]。

标签平滑将 one-hot 目标改为 $q = (1-\varepsilon)y + \varepsilon/V$，改变了每个位置的目标分布。逐样本长度归一化改变样本权重，偏好优化中把序列 log-prob 除以长度则改变偏好目标本身。这些都不能当作无条件的“消除长度偏置”按钮。任何改动都要固定基线目标、单独消融，并在长度桶与正确性切片上检查。

表 4.1 出现症状后先做一个能区分解释的小实验

症状	候选原因	最小检查
损失 NaN 或突变	无监督批、非法 mask、数值范围、坏数据	保存原批；检查 $S > 0$、有限 logits；FP32 重放
训练 loss 不下降	参数未更新、shift 错、学习率或容量不合适	检查更新前后参数差与单批过拟合
训练好、验证差	重复、来源差异、过拟合或标签噪声	按来源与近重复簇分桶，不只缩短 epoch
loss 降、任务不变	模板词占比高、答案被截、生成协议不一致	检查逐字段监督与固定解码输出
任务升、通用能力降	分布集中或更新过大	固定预算比较保留数据、更新范围和较小步长
packing 后质量变	跨样本读取、位置或边界监督改变	比较独立前向、交换顺序与边界 logits

4.7 诊断必须能够区分原因

日志除训练和验证损失外，还应保存监督 token 进度、输入 token、每源实际比例、梯度范数、学习率、任务指标、输出长度与格式通过率。显存报告区分已分配与保留内存，时间报告是否包含数据预处理、编译与评测。对异常保存最小可重放批量、模型 checkpoint 与运行配置；不能只留下一个平均曲线。

4.8 停止、恢复与导出是训练的一部分

在验证曲线达到预定目标或不再改善时停止；当通用能力下降超过容忍范围、错误类型恶化、出现数值异常或预算耗尽时，也应停止或回滚。选 checkpoint 的规则应预先明确，不能在测试集上选择表现最好的一次。

恢复训练需要权重、适配器、优化器、调度器、随机数状态、采样器进度和累计 token 计数，仅保存推理权重无法保证接续同一个实验。导出则记录基础模型精确版本、适配器结构与权重、量化配置、tokenizer、模板、特殊 token 与生成配置。若还训练了 embedding 或输出层，必须明确它们如何随适配器保存。

导出之后，用固定提示从新目录重新加载并生成，比较内存中模型与重载模型的 logits 和任务输出；允许适当浮点容差，记录差异。适配器合并或重新量化还需要单独回归，不能以“文件成功写出”代替有效产物检查。

练习与提示

1. 让一个长度为 8 的回答与一个长度为 80 的回答分属两个微批。如何构造一次更新，使每个监督 token 权重相同？**提示：**反传损失和，并按总监督数归一化；与一次完整批量梯度比较。
2. 拼接 A、B 两个样本，仅在中间放 EOS。在哪一个方向存在跨样本读取？如何证明实际实现已隔离？**提示：**看 causal mask 的行列方向，并对比独立前向与交换顺序后的结果。
3. 验证损失下降而生成总不停止，列出三个可排除的原因。**提示：**检查 EOS 标签、截断规则、生成停止配置，并保持模板不变。

第五章　参数高效适配：从更新空间到 LoRA 的几何

全参数微调允许所有权重移动，参数高效微调（parameter-efficient fine-tuning，PEFT）则把可学习的变化限制在较小的参数集合里。它首先是一种模型约束，其次才是一种省内存的工程手段。若任务所需的变化落在这个集合能够表达的范围内，少量参数也可以有效适配；若限制排除了关键方向，继续训练未必能够弥补容量不足。本章先明确这种约束，再逐步推导 LoRA，最后把变种放回各自要解决的问题中。

5.1　不同 PEFT 方法限制了什么

对权重重参数化，可以写成 $\theta(\phi) = \theta_0 + f_\phi$，其中 θ_0 冻结、ϕ 可训练。这个表达适合 LoRA 和部分稀疏更新，但不能把所有 PEFT 都当成同一个权重差：连续提示改变输入，prefix 改变注意力中的状态，插入式 adapter 则改变计算图。更一般的共同对象是函数 $f(x; \theta_0, \phi)$。

表 5.1　四类 PEFT 基线的约束与部署代价

方法	可学习的变化	比较时要核查的条件
瓶颈 adapter	插入 $h + U\sigma(Dh)$，瓶颈宽度限制新分支	非线性分支通常不能直接并入原线性层；测额外激活与延迟
Prompt / prefix	输入端连续向量，或各层注意力的前缀状态	两者的注入位置不同；前缀占用上下文或 KV 存储
BitFit	更新已有偏置参数	无偏置架构可用参数很少；结果依赖模型与任务
LoRA	原线性层增加低秩矩阵 sBA	可合并，但训练时有额外分支与保存状态

这些方法各有明确的原始问题与实验范围 [19, 20, 21, 22, 5]。可训练参数少不意味着训练时间按相同比例下降：冻结层仍须执行前向，若前面存在可训练模块，反向还需要穿过冻结层传递输入梯度。它也不自动意味着更少遗忘；约束更新可能降低破坏，也可能迫使任务变化集中在少数敏感方向。

5.2　固定矩阵约定，先核对前向

本书把线性层权重存为

$$W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}, \qquad x \in \mathbb{R}^{d_{\text{in}}}, \qquad y = W_0 x. \tag{5.1}$$

LoRA 学习两个因子

$$A \in \mathbb{R}^{r \times d_{\text{in}}}, \qquad B \in \mathbb{R}^{d_{\text{out}} \times r}, \qquad W = W_0 + sBA, \tag{5.2}$$

其中 $s > 0$ 是缩放系数，标准选择为 $s = \alpha / r$。偏置是否训练须另行声明；下文参数量不含偏置。将微批的 B_μ 条长度为 L 的序列展平为 $n_{\text{pos}} = B_\mu L$ 个位置，令 $X \in \mathbb{R}^{n_{\text{pos}} \times d_{\text{in}}}$，则

$$Y = XW_0^\top + s(XA^\top)B^\top \in \mathbb{R}^{n_{\text{pos}} \times d_{\text{out}}}. \tag{5.3}$$

实现中的线性层通常接收任意前导维度，实际只变换最后一维；展平仅用于说明矩阵形状，不是要求改变数据组织。

更新的秩至多为 r，新增参数量为

$$N_{\text{LoRA}} = r(d_{\text{in}} + d_{\text{out}}), \qquad \rho = \frac{r(d_{\text{in}} + d_{\text{out}})}{d_{\text{in}} d_{\text{out}}}. \tag{5.4}$$

这不等于独立自由度：对可逆 $C \in \mathbb{R}^{r\times r}$，$(BC)(C^{-1}A) = BA$，因子的坐标并不唯一。相同的更新矩阵可以对应不同因子范数，因而因子梯度、权重衰减和优化轨迹也可能不同。仅比较 $\|A\|$、$\|B\|$ 不能直接解释模型变化，应同时记录 $\|sBA\|_F$ 和更新的奇异值谱。

LoRA 的低秩假设针对相对于既有底座的任务更新，并不声称底座权重或知识本身低秩。若某个理想全参数更新的奇异值快速衰减，截断 SVD 能给出 Frobenius 范数下的最佳秩约束近似；但实际没有已知的理想更新，网络也非线性，这只是理解约束的工具，不能据此保证某个 rank 足够。

5.3 梯度、零初始化与缩放

令 $G = \partial \mathcal{L}/\partial Y \in \mathbb{R}^{n_{\text{pos}}\times d_{\text{out}}}$。由式 (5.3) 可得

$$\frac{\partial \mathcal{L}}{\partial B} = sG^{\top}XA^{\top}, \tag{5.5}$$

$$\frac{\partial \mathcal{L}}{\partial A} = sB^{\top}G^{\top}X, \tag{5.6}$$

$$\frac{\partial \mathcal{L}}{\partial X} = GW_0 + sGBA. \tag{5.7}$$

三式的形状分别与 B, A, X 一致。冻结 W_0 表示不累积它的参数梯度、不给它分配优化器状态；式 (5.7) 中的 GW_0 仍然存在，不能因冻结而把这段反向全部关闭。

常见初始化令 A 随机、$B = 0$，从而初始更新为零，确定性前向与底座一致。此时损失对 A 的梯度为零，而对 B 的梯度可能非零；若 $XA^{\top} = 0$ 或上游梯度恰好正交，B 的梯度也可以为零。B 移动后，A 才获得这条损失路径的信号。若两因子都置零，两者的损失梯度都会为零。这里说的是损失梯度；优化器的解耦权重衰减仍可能改变非零的 A。

直觉与机制

零初始化保留初始函数，却没有让优化从一个对称的零点开始。随机的 A 先提供输入方向，B 先学习这些方向如何影响输出；后续两因子共同调整方向。初始化、缩放和学习率因此应作为一个整体检查。

标准缩放 α/r 在保持 α 不变时随 rank 增大而减小。rsLoRA 使用 $\alpha/\sqrt{r}$，其动机是在特定随机初始化与宽度分析下，缓解 rank 增加造成的更新和梯度尺度衰减 [23]。这不是对任意初始化和优化器都严格保持范数的恒等式：训练首步 $BA = 0$，其范数本来就为零；后续尺度还依赖数据、因子分布、优化器与学习率。

因此，比较缩放规则时应固定数据、初始化分布和搜索预算，报告 r, α, s、两因子学习率及梯度范数。可以先观测少量更新的实际矩阵变化，再设计学习率网格；不能先强行令所有方法的更新范数相同，再宣称测到了缩放规则的全部效果，因为范数变化本身正是被研究的机制。

5.3.1 从因子步长看实际权重步长

令同一位置的全矩阵梯度为 $D = G^{\top}X$。只考虑首步普通 SGD、不含衰减，$B_0 = 0$ 时有

$$\delta B = -\eta_B s D A_0^{\top}, \qquad \delta W = s\delta B A_0 = -\eta_B s^2 D A_0^{\top} A_0. \tag{5.8}$$

这一步没有取 D 的截断 SVD；它先在随机 A_0 的行空间中组合输入方向，再由 B 形成输出响应。除非有特定的正交归一化，$A_0^\top A_0$ 也不是正交投影矩阵。实际首步同时受 s^2 与初始化协方差影响，因而只报告名义学习率会遗漏关键尺度。对 Adam，这个普通 SGD 等式不能原封不动地当作更新公式，因为按坐标的矩估计还会改变步长。

后续改变两因子时，局部变化为

$$\delta W = s\big(\delta B A + B\delta A + \delta B\delta A\big). \tag{5.9}$$

两条一阶路径共同决定运动，不能把训练理解为一直在一个固定随机子空间里做回归。与此同时，任意时刻相对于冻结底座的 sBA 仍满足秩约束。若同时对两因子做解耦衰减，一次无梯度步骤会把产品乘以两个衰减系数；这与直接衰减完整更新矩阵也不相同。因此因子衰减是需要记录的优化选择，不是全参数微调中同名系数的自动等价物。

5.4 一个同时核对前向、合并和字节的算例

取

$$W_0 = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & -1 & 2 \end{pmatrix}, \quad B = \begin{pmatrix} 2 \\ -1 \end{pmatrix}, \quad x = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}. \tag{5.10}$$

令 $r = 1, \alpha = 1/2$，故 $s = 1/2$。这里给出的非零 B 表示某个已训练状态，不是上一节的初始状态。逐步计算得到

$$Ax = -3, \qquad W_0 x = \begin{pmatrix} 2 \\ 4 \end{pmatrix}, \tag{5.11}$$

$$sBA = \begin{pmatrix} 1 & -1 & 2 \\ -1/2 & 1/2 & -1 \end{pmatrix}, \qquad sBAx = \begin{pmatrix} -3 \\ 3/2 \end{pmatrix}. \tag{5.12}$$

于是未合并前向为 $y = (-1, 11/2)^\top$。合并后的权重是

$$W_{\text{merged}} = \begin{pmatrix} 2 & -1 & 1 \\ 3/2 & 3/2 & -1 \end{pmatrix}, \qquad W_{\text{merged}} x = \begin{pmatrix} -1 \\ 11/2 \end{pmatrix}. \tag{5.13}$$

实数运算中，两种前向完全相等；有限精度下，运算顺序和存储舍入可造成差异，应按 dtype 设置容差。本例所有数都是小的整数或二进制可精确表示的分数，适合作为独立参考结果。

新增参数为 $1(3+2) = 5$ 个。BF16 adapter 张量的有效载荷下界为 $5 \times 2 = 10$ 字节，BF16 梯度再需 10 字节；若 Adam 两个状态均为 FP32，再需 $5 \times 8 = 40$ 字节，合计 60 字节。若实现额外保留 FP32 主副本，再加 20 字节。冻结的 BF16 基座另占 $6 \times 2 = 12$ 字节。这里没有计入张量对象、内存对齐、激活和临时缓冲，文件也会有格式头，因此这些数是张量有效载荷下界。这个小矩阵中 LoRA 只比全矩阵少一个可训练参数；真正的比例优势依赖矩阵尺寸和 rank，不能由方法名字推出。

把同样的计数方法用于较常见的矩阵尺寸：一个 4096×4096 投影在 $r = 8$ 时有 65,536 个 adapter 参数，BF16 有效载荷为 128 KiB。假设一个模型恰有 32 层、每层只适配同尺寸的 Q/V 两个投影，总计 4,194,304 个参数，导出载荷为 8 MiB；将 rank 改为 64，分别变成 33,554,432 个参数与 64 MiB。这里的 KiB、MiB 分别表示 2^{10}、2^{20} 字节。前一种配置若采用上述 BF16 参数、BF16 梯度和 FP32 双状态，常驻训练张量为 48 MiB；加主副本则为 64 MiB。它仍没有计入基座与激活。实际模型的 K/V 尺寸、层数和模块集合不同，不能把这组假设作为所有 7B 模型的显存报价。

5.5 变种一：把方向、幅值和初始化分开

5.5.1 DoRA 的范数沿哪一个轴

DoRA 将权重的幅值与方向分开学习 [24]。原文用列向量说明范数；与本书的存储约定对应时，应明确所讨论的矩阵。令 $V = W_0 + sBA$，把 $V^\top \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ 的每一列归一化，等价于对存储矩阵 V 的每一行归一化：

$$W_{j,:} = m_j \frac{V_{j,:}}{\|V_{j,:}\|_2}, \qquad m_j^{(0)} = \|(W_0)_{j,:}\|_2. \tag{5.14}$$

这里 $m \in \mathbb{R}^{d_{\text{out}}}$，非零行的初始化复原原权重；零范数行需要实现定义的稳定处理。直接在 $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ 上改为列归一化则是另一种分组，不能把两种公式和代码混用。

一个输出通道的方向只有很小变化、幅值却需要明显调整时，独立的 m_j 可提供直接路径。DoRA 的动机是改善这种耦合的优化几何，不是证明标准 LoRA 完全不能改变范数。归一化后整体权重差也不必仍为秩 r。训练增加幅值参数和范数运算；部署时可以先生成完整的合并矩阵，消除独立分支。是否增加推理开销取决于是否合并与具体内核，不能笼统声称必然增加延迟。

5.5.2 PiSSA 必须保留残差恒等式

设 $W_0 = U\Sigma V^\top$，主秩 r 成分为 $M_r = U_r\Sigma_r V_r^\top$。PiSSA 将这部分放进可训练分支，而冻结

$$W_{\text{res}} = W_0 - M_r, \qquad W = W_{\text{res}} + sBA. \tag{5.15}$$

当 $s > 0$ 时，可取

$$B_0 = s^{-1/2} U_r \Sigma_r^{1/2}, \qquad A_0 = s^{-1/2} \Sigma_r^{1/2} V_r^\top, \tag{5.16}$$

保证 $sB_0A_0 = M_r$，因此初始 $W = W_0$[25]。关键在于同时更换冻结残差，不能保留完整 W_0 又加一次主成分。近似 SVD 时也应以实际得到的乘积构造残差；数学恒等式仍可成立，数值误差另行检查。非零分支若启用训练态 dropout，单次训练前向还可能偏离这个确定性恒等式，应记录相应配置。

PiSSA 给出了与底座谱方向相联系的初始可训练子空间，但主权重方向不必就是下游任务最重要的方向。它的收益必须由目标任务与遗忘评测检验；初始化分解的时间、内存和保存方式也属于成本。

5.5.3 LoftQ 联合处理量化与低秩残差

设 $\mathcal{Q}$ 是指定量化格式可表示的矩阵集合。LoftQ 的核心可抽象为

$$\min_{\widehat{W} \in \mathcal{Q},\, \operatorname{rank}(L) \le r} \|W_0 - \widehat{W} - L\|_F^2, \qquad L = sBA. \tag{5.17}$$

交替固定 L 量化 $W_0 - L$，再固定量化矩阵，对 $W_0 - \widehat{W}$ 求低秩近似，使冻结量化基座与 adapter 共同接近原权重 [26]。原始权重重建过程不要求校准文本；任务感知的后续扩展可以使用数据，但不应把它写成基本算法的必要条件。与 PiSSA 的残差恒等式不同，量化约束下通常只近似恢复原函数。更小的权重重建误差也不必带来相同比例的任务收益，尤其当输入协方差集中于少数方向时。

5.6 变种二：把训练预算分配给哪里

AdaLoRA 根据重要性估计调整不同矩阵的有效秩，并通过预算调度与分解结构实现裁剪 [27]。用于规划时，可写一个简化的资源问题

$$\max_{r_1,\dots,r_J} \sum_{j=1}^{J} I_j(r_j), \qquad \sum_{j=1}^{J} r_j(d_{\text{in},j} + d_{\text{out},j}) \le C. \tag{5.18}$$

I_j 表示待估计的收益，不是已知真值；这个式子也不是原算法的逐行实现。不同矩阵形状不同，只限制 $\sum_j r_j$ 不能保证相同参数预算。有效秩被掩码裁剪后，若仍保存完整张量与优化器状态，实际显存未必下降，须分别报告活跃参数、物理分配和最终导出大小。

其他方向可用来构造有针对性的基线：LoRA-FA 固定一侧因子以减少相关激活与反向需求，VeRA 用共享随机矩阵配合少量缩放向量，LoRA+ 为两因子分配不同学习率 [28, 29, 30]；MoRA 探索相同量级参数下的结构化高秩更新，LoRA-XS 则进一步压缩可训练核心 [31, 32]。它们改变的约束并不相同。比较时应写明冻结哪些张量、真实可训练参数以及是否产生额外算子，避免仅把名字列成从弱到强的次序。

5.7 注入位置、合并与公平消融

注意力的 Q、K、V、O 投影影响寻址、内容与输出组合，FFN 的 gate、up、down 投影影响非线性特征变换；embedding 和输出头则直接关联词汇空间。Q/V 是可解释的起点，不是所有任务的固定答案。在 GQA 中 K/V 投影尺寸不同，按相同 rank 注入时参数成本也不同。新增 token 若只给 embedding 增加行，还需核查输出头是否绑定，以及新行到底有没有梯度。

模块名不能代替架构语义。融合 Q/K/V 投影的单个 rank-r 分支，与三个子矩阵各自的 rank-r 分支，参数量和输入低秩基的共享方式都不同。融合 FFN、转置存储和共享输出头也须核查。注入前打印模块全名、形状与共享存储关系，注入后核对可训练参数；先用小 batch 确认 adapter 有梯度、冻结参数不在优化器中，再开展搜索。

公平比较包括两组：容量曲线固定位置、扫描 rank，报告质量与资源；预算分配固定可训练参数总数，比较更多矩阵的小 rank 与少数矩阵的大 rank。均须固定数据版本、监督 token、验证提示、解码预算和搜索次数。扫描 rank 时参数量通常随之改变；全参数或高 rank 基线只是给定预算下的参照，也不是已知最优解。

对确定的 A, B，关闭 dropout 后可用 $W_{\text{merged}} = W_0 + sBA$ 合并，推理不再需要低秩分支。验收应同时比较层输出、最终 logits 与任务结果；在量化权重上直接原地加浮点增量并非同一个运算，第六章进一步讨论重新量化。多 adapter 的加权、压缩与路由将在第九章展开。实验记录至少保留底座标识、模块全名、每层 rank、缩放、初始化、dropout、学习率、权重衰减及导出方式。

一致性检查应覆盖零分支、训练后的非零分支及多种输入长度。零分支已偏离基座，先查 dropout、其他可训练参数、模板和 dtype；仅非零分支不一致，先查缩放、转置与重复合并。同时报告绝对、相对误差，避免近零输出放大相对误差。贪心解码可能因极小 logit 差异跨过并列边界而分叉，应先比较相同前缀的 logits，再解释文本差异。

练习与提示

把手算例的 B 改为零，任选非零的上游梯度 G，代入式 (5.5) 与式 (5.6)，判断哪些梯度能够非零。再令两个任务使用同一底座和 rank，但一个注入 Q/V，另一个注入 Q/K/V/O，说明为什么这不是等参数比较。解题时先列每个投影的实际形状，再按式 (5.4) 求和。

本章要点

LoRA 的合并来自 $W_0 + sBA$，优化取决于因子坐标、初始化与缩放。DoRA 分离幅值和方向，PiSSA 改变初始分解，LoftQ 联合处理量化残差，动态 rank 分配容量。比较方法前先核对形状、初始函数与真实资源。

第六章　QLoRA 与低比特微调：建立可核查的显存账本

第五章减少了需要更新的参数，但冻结基座仍要存储。QLoRA 将基座以低比特格式保存，在计算时恢复到适当精度，并让梯度经过这些冻结线性变换传给 adapter[6]。理解它的关键是区分三个问题：哪些信息被压缩，哪些运算仍用较高精度，以及训练高峰时所有存活张量加起来有多少。一个 checkpoint 的文件大小不能回答后两个问题。

6.1　量化格式、计算精度与梯度是三个层次

先用均匀仿射量化固定零点约定。令步长 $a > 0$、整数零点 z，则

$$q = \mathrm{clip}\left(\mathrm{round}(w/a) + z, q_{\min}, q_{\max}\right), \tag{6.1}$$

$$\widehat{w} = a(q - z). \tag{6.2}$$

量化与反量化的正负号必须配套。若没有截断且忽略舍入边界，重建误差绝对值至多为 $a/2$；发生饱和时没有这个误差上界。对称格式通常取零点为零，非对称格式需要存储或隐含零点；逐张量、逐通道和逐组粒度则决定哪些数共享同一个步长。

NF4（4-bit NormalFloat）使用非均匀的 16 值码本，按近似正态分布构造代表值，并结合块级缩放 [6]。概念上可写为

$$\widehat{w}_i = c_g \, \upsilon_{q_i}, \qquad q_i \in \{0, \ldots, 15\}, \quad i \text{ 属于块 } g, \tag{6.3}$$

其中 υ 是码本，c_g 是块尺度。它不是等间隔 INT4，也不是把 IEEE 浮点数简单截成四位。NF4 针对某类权重分布的设计动机不意味着所有层、所有离群值与所有任务都最优。块大小改变重建误差和元数据成本，码本的排列、打包顺序也必须与加载内核一致。

至少应分别记录存储格式、矩阵乘输入精度、激活精度与累加精度。典型 QLoRA 路径是低比特存储基座，使用 BF16 等计算精度执行线性运算；内核可能按块反量化，也可能生成较大的临时张量，不能仅凭计算图假设整模型同时存在一个 BF16 副本。反过来，也不能假设反量化完全不占工作空间。BF16 比 FP16 有更大的指数范围，但并非不会产生 NaN，归约、优化器状态和部分敏感算子还可能使用 FP32。

对冻结的量化基座，训练不更新离散码和量化常数；仍需利用重建的权重计算输入梯度。若 $Y = X\widehat{W}^{\top}$，则 $\partial \mathcal{L}/\partial X = G\widehat{W}$。普通 QLoRA 因此不要求对四位整数码本身使用直通估计器。训练量化器、裁剪阈值或权重的量化感知训练，是另一个需要明确说明的方案。

6.2 先把所有内存项分开

设量化参数数为 P_q、保留高精度的冻结参数数为 P_f、可训练参数数为 P_{train}。一个可审计的常驻张量账本为

$$M_{\text{base}} = \lceil P_q b_q / 8 \rceil + M_{\text{meta}} + P_f c_f, \tag{6.4}$$

$$M_{\text{trainable}} = P_{\text{train}}(c_p + c_g + c_m + c_v) + M_{\text{master}}, \tag{6.5}$$

$$M_{\text{peak}} = \max_t \{M_{\text{base}}(t) + M_{\text{trainable}}(t) + M_{\text{act}}(t) + M_{\text{temp}}(t) + M_{\text{other}}(t)\}. \tag{6.6}$$

这里所有 c 都以字节为单位；c_p, c_g 分别是可训练参数和梯度每元素字节数，c_m, c_v 是优化器一阶、二阶状态。M_{master} 仅在实际保留主副本时计入。M_{meta} 包括尺度、零点、二级尺度、偏移及格式所需的其他元数据。M_{other} 包括运行时、通信、输入批次等独立开销；具体工具的 reserved memory 还可能包含未用于活跃张量的缓存，报告时应注明口径。

式 (6.6) 对同一时刻的存活分配求和，再对时间取最大；把各项在不同时刻的独立最大值相加通常只是保守估算。对没有分页与分片的稳定训练，基座和大部分可训练状态近似常驻；首次分配、梯度释放、检查点重计算与分页会改变随时间的驻留。显存曲线若在 optimizer step 形成高峰，应先检查状态和临时更新张量；若随序列长度快速增长，应检查激活与注意力实现。这两种高峰不能靠同一个“减少 rank”建议解释。

表 6.1 训练账本中容易混淆的对象

对象	随什么增长	不能遗漏的条件
冻结基座	量化或高精度参数数	部分 norm、embedding、输出头可能未量化
量化元数据	量化块数与二级块数	四位权重的载荷没有包含尺度
参数与梯度	真正可训练参数数	adapter 常被框架保留为 FP32，须检查实际 dtype
优化器与主副本	状态格式、更新实现	状态可能首个 step 才分配；主副本不是必选项
保存的激活	微批次、长度、层数和算子	冻结权重仍可能需要保存输入与中间状态
临时工作空间	内核、反量化、损失和导出	一次加载或合并的峰值可能高于稳定训练步

例如 BF16 参数与 BF16 梯度各 2 字节，FP32 Adam 双状态共 8 字节，合计每个可训练参数 12 字节；若另有 FP32 主副本，则为 16 字节。另一种实现直接把可训练参数和梯度都保留为 FP32，此时仅参数、梯度和双状态就是 16 字节，不能再机械地加一次相同参数副本。八位优化器、分片和 offload 会改变账本；应读取实际张量和实现文档，而不是把“混合精度”固定翻译成某一个倍数。

激活往往是最难估计、也最值得测量的一项。未做检查点时，许多隐藏状态随 $B_\mu L d N$ 增长；朴素注意力矩阵还随 $B_\mu N h_q L^2$ 增长，其中 B_μ 为设备微批序列数，L 为序列长度，N 为层数，h_q 为 query 头数。融合注意力可改变需要物化的中间张量，梯度检查点用重计算换内存；常数项和实现差异很大，所以这些量纲关系不是精确报价。最后一层完整 logits 的大小还与词表 V 有关。只对 assistant 位置计算损失，不意味着其他上下文位置的激活可以全部删除。

梯度累积在保持每次更新监督 token 大致相同的情况下，可用更小微批次降低部分激活峰值，但不能消除参数与优化器常驻成本。各微批次的长度和监督比例不同时，还需使用正确的 token 权重归约。为了追求显存而改变归约方式，会把系统比较变成另一个优化目标。

6.3 双重量化节省的是元数据

若每 64 个四位权重存一个 FP32 尺度，权重载荷为每参数 4 bit，尺度另需 32/64 = 0.5 bit。双重量化再把这些尺度用八位值表示，并为每 256 个一级尺度保存一个 FP32 二级尺度，忽略边界与少量偏移时，平均成本变为

$$4 + \frac{8}{64} + \frac{32}{64 \times 256} = 4.126953125 \quad \text{bit/参数}. \tag{6.7}$$

所以该配置相对于四位权重加 FP32 块尺度，平均减少约 0.373 bit/参数 [6]。它没有把四位码进一步压成三位，也没有压缩激活。有限大小张量需要向上取整，真实格式还包含偏移和头部；这一理论平均数必须与实际序列化和内存分配区分。

给出一个假想账本，用于复核量纲而非预测某块显卡能否训练：假设恰好有 10^9 个冻结参数，全部量化，没有高精度例外；权重块为 64，二级尺度块为 256；另有 1.6×10^7 个 BF16 可训练参数及 BF16 梯度，采用 FP32 Adam 双状态。按上述打包假设计数，四位载荷为 500,000,000 字节，15,625,000 个八位一级尺度与 61,036 个 FP32 二级尺度合计 15,869,144 字节。少量偏移和格式头部在此未计入。

表 6.2 假想十亿参数基座的常驻有效载荷，不含激活与临时缓冲

配置与计数范围	十进制 GB	二进制 GiB
冻结 BF16 基座本身	2.000	1.863
四位载荷加 FP32 一级尺度	0.563	0.524
四位载荷加上述双重量化元数据	0.516	0.480
BF16 基座加 adapter、梯度及双状态	2.192	2.041
双重量化基座加相同训练张量	0.708	0.659
上一行再加 adapter 的 FP32 主副本	0.772	0.719

表中 GB 为 10^9 字节，GiB 为 2^{30} 字节。adapter 训练张量共 $16 \times 10^6 \times 12 = 192 \times 10^6$ 字节，主副本另为 64,000,000 字节。这些数既不是峰值显存，也不是一份真实模型的完整配置；未量化层、激活、长序列、框架和导出缓冲均可能改变结论。对同一个基座做全参数 BF16 Adam，若参数、梯度和双状态分别为 2、2、8 字节，仅这些张量就是 12 GB，若保留主副本则为 16 GB。冻结参数带来的节省与低比特存储带来的节省应分别归因。

6.4 分页优化器缓解驻留高峰，不能压缩状态

QLoRA 的分页优化器利用受支持系统中的统一内存机制，让优化器状态页在 CPU 与 GPU 之间迁移，缓解某些瞬时高峰 [6]。若双状态逻辑上共有 128 MB，把其中一些页迁到主存并没有让这些状态消失。物理驻留显存可能下降，主存和传输需求却增加，频繁换页还会拖慢更新。是否可用取决于硬件、驱动、运行时和优化器实现；不能把 NVIDIA 统一内存方案直接当成 CPU 或其他后端的通用能力。

分页也不能保证不会 OOM：当前算子必须同时访问的激活与工作区可能已经超过设备容量，主存也有上限。一次偶然可完成的步骤不能证明长时间训练稳定，尤其当长度分布存在长尾、后台任务共享设备或保存检查点时。验收应说明最大已测长度与微批次，并保留分配失败发生在哪个阶段的日志。

减少优化器位宽、状态分片、offload 与分页是不同手段：第一种改变存储字节，第二种改变设备间分布，后两者改变驻留位置与传输行为。报告“用了 paged Adam”还不够，应说明状态 dtype、

主存开销、是否发生明显传输，以及峰值测量是否覆盖优化器更新。首次 step 的状态延迟分配尤其容易让只测前向的 smoke test 给出过分乐观的估算。

6.5 量化残差会占用多少适配能力

令量化误差 $E_q = W_0 - \widehat{W}$。若某个目标任务理想上需要更新 ΔW^*，量化基座上的适配分支要满足

$$\widehat{W} + sBA \approx W_0 + \Delta W^*, \qquad sBA \approx E_q + \Delta W^*. \tag{6.8}$$

这是对两个表示之间差异的解释，不是声称训练明确知道 E_q 或理想更新。两种误差可能同向，也可能相互抵消，不能断言量化一定需要更高 rank。若量化误差主要落在任务很少访问的方向，权重 MSE 较大也未必明显影响任务；对实际输入 X，线性层输出误差为

$$\|X(W_0 - \widehat{W} - sBA)^\top\|_F^2, \tag{6.9}$$

它受到输入协方差加权，与不带数据的权重 Frobenius 误差不同。LoftQ 改进初始化，增加 rank 改变补偿容量，调整块大小或保留敏感层高精度改变量化误差，三者应分开消融。

逐层比较高精度与量化路径的输出、最终 logits 和验证损失，可以定位误差累积；但较小的 logits 均方差不能代替任务评测。结构化输出中的一个分隔符、代码中的一个运算符，可能决定完整执行是否成功。应在同一输入和生成预算下记录任务成功、格式、长度与通用能力变化，并保留具体失败样本。

为避免把不同失败混在一起，训练前先比较高精度冻结基座与量化冻结基座，测量量化本身的损失；随后比较同一监督预算下的高精度 LoRA 与量化基座 LoRA。若两种训练都没有收益，应先查数据、mask 与任务可学性；若仅量化冻结基座明显退化，优先查格式、敏感层与重建；若量化训练可以恢复任务却耗时异常，则查反量化、换页和内核路径。所有判断都应在相同提示、长度与解码条件下进行，避免把较短输出的速度优势归因于四位算子。

6.6 QLoRA、PTQ 与 QAT 的边界

训练后量化（PTQ）通常指把已训练权重转换为部署格式，某些方法利用校准样本，简单块量化则可直接由权重计算尺度；不能把“需要校准文本”写成一切量化的必要条件。量化感知训练（QAT）在训练前向中显式考虑量化重建，可能通过代理梯度更新浮点潜在权重或量化参数。冻结量化底座、只训练 adapter 的 QLoRA，与同时更新潜在全权重的 QAT 有不同的状态与梯度路径。

如果用高精度模型约束量化模型的分布，可以加入蒸馏项，但要满足第九章讨论的词表、前缀与温度条件；同时必须计入 teacher 的前向与缓存。把低比特训练产物用于推理也需要单独的选择：保留量化基座与浮点 adapter，或者先在足够精度中合并，再量化成服务内核支持的格式。一般有

$$Q(W_0 + sBA) \neq Q(W_0) + sBA, \tag{6.10}$$

而对右侧再次取整还会引入新的舍入误差。NF4 训练格式不自动等于手机或服务端最优的 INT4 推理格式，也不能由权重位数直接推出 decode 延迟。

6.7 从账本到一次可验收的试跑

准备阶段先固定模型与 tokenizer revision，列出量化层、例外层、组大小、量化常数格式、compute dtype、adapter dtype、优化器状态 dtype 和可训练参数。加载阶段记录是否先创建完整高精度模型

再量化；这个瞬时峰值可能不同于直接加载预量化权重。随后用代表性的短序列与高分位长度各运行完整前向、反向、优化器更新和梯度清空，记录设备分配、缓存预留、主存及步时。若后端异步，测时与峰值统计应在适当的同步点读取。

训练时使用动态 padding，实际最长微批次会影响峰值；固定短样本试跑通过，只能支持短样本配置可运行。开启检查点、改变注意力内核或降低微批次以后，应重新核对损失归约和有效监督 token。评测生成还可能创建 KV cache；训练常关闭 cache 的配置不应未经检查地沿用到所有部署测量。最后实际执行一次保存、重新加载、合并或转换流程，比较固定提示下的 logits 和任务输出，确认配置与张量一起导出了。

本书第十二章把小矩阵验证、可负担的训练对照和尚未运行的硬件方案分开记录。教学实现中的均匀四位重建可验证梯度与误差机制，却不能称为对 NF4 内核、分页优化器或 CUDA 显存峰值的验证。只有真正运行的路径才能报告实测结果，其余应保留账本与可执行条件。

练习与提示

把表 6.2 的一级块大小从 64 改为 128，重算不带双重量化与带双重量化的元数据，保留向上取整。然后假设输出头有两千万个参数保留 BF16，说明该从四位载荷中减去什么、再加回什么。最后解释为什么这些计算仍无法决定长度 8,192 的微批次能否运行。提示：分开讨论常驻载荷、保存激活和内核临时空间。

本章要点

QLoRA 同时利用冻结参数与低比特存储，但计算路径仍需足够精度。NF4 改变码本，双重量化压缩尺度，分页优化器改变状态驻留；三者没有互相替代。显存结论必须从冻结载荷、元数据、可训练状态、激活和临时空间逐项建立，再由完整训练与导出流程的实测收尾。

第七章　从成对偏好到 DPO：把目标函数写清楚

监督微调告诉模型“在这个上下文里应该输出什么”，但同一问题可能有多个正确答案，差别在是否遵守约束、是否有帮助、是否安全以及是否冗长。偏好数据把这个差别表达成比较。它适合改变已有行为的相对概率，并不自动提供模型缺失的知识。若 chosen 与 rejected 都计算错误，优化二者的相对顺序仍可能得到一个更自信的错误模型。本章先推导最常用的直接偏好优化（direct preference optimization，DPO），再说明代码中的概率究竟对应哪段 token。

7.1　偏好标签是一项有条件的观测

样本记为 (x, y^+, y^-)，其中 x 是包含完整对话历史的提示，y^+ 是标注者较偏好的回答。标签不是“真实奖励”的直接读数：比较依赖评分准则、候选生成模型、显示顺序和标注者。应保留候选原文、来源模型、生成配置、准则版本、标注分歧与两项长度。人工反馈与 AI 反馈是不同的数据来源；InstructGPT 与 Constitutional AI 分别提供了相关流程的实例 [33, 34]，不能据此假定任何 AI judge 等价于人工事实验证。

Bradley–Terry 模型以标量奖励解释比较概率：

$$P(y^+ \succ y^- \mid x) = \sigma\big(r(x, y^+) - r(x, y^-)\big), \qquad \sigma(a) = \frac{1}{1 + e^{-a}}. \tag{7.1}$$

这里把标注噪声的温度吸收进 r 的尺度。若数据含稳定的偏好循环、不同任务相互矛盾的尺度或事实性与风格性混标，单一标量模型未必拟合得好。保留维度标签和“平局/不可判断”比强行制造一个 chosen 更有价值。

7.2　KL 正则目标如何给出最优策略

固定提示 x，先考虑一个理想化分布优化问题。设 reference $\pi_0(y \mid x)$ 固定，$\beta > 0$，并限制策略支持集包含在 reference 的支持集内：

$$\max_{\pi} \sum_y \pi(y \mid x) r(x, y) - \beta \sum_y \pi(y \mid x) \log \frac{\pi(y \mid x)}{\pi_0(y \mid x)}, \qquad \sum_y \pi(y \mid x) = 1. \tag{7.2}$$

这里的 KL 是对整个回答分布的散度，不是每条回答上再算一个相同常数。假定奖励和配分函数有限。加入乘子 λ，对每个 $\pi(y \mid x)$ 求导：

$$r(x, y) - \beta \left(\log \frac{\pi(y \mid x)}{\pi_0(y \mid x)} + 1 \right) + \lambda = 0. \tag{7.3}$$

整理并归一化得到

$$\pi^*(y \mid x) = \frac{\pi_0(y \mid x) e^{r(x,y)/\beta}}{Z(x)}, \qquad Z(x) = \sum_y \pi_0(y \mid x) e^{r(x,y)/\beta}. \tag{7.4}$$

因此

$$r(x, y) = \beta \log \frac{\pi^*(y \mid x)}{\pi_0(y \mid x)} + \beta \log Z(x). \tag{7.5}$$

把它代入式 (7.1)，同一提示下的 $\log Z(x)$ 消去。这是 DPO 可以绕开显式训练奖励模型的关键 [35]。然而，实际神经网络并非任意分布，离线数据也并非覆盖所有回答；这个代换不会给有限数据下的训练自动附带“全局最优”保证。

定义 reference-relative margin

$$\Delta_\theta(x) = \log \pi_\theta(y^+ \mid x) - \log \pi_\theta(y^- \mid x), \tag{7.6}$$

$$\Delta_0(x) = \log \pi_0(y^+ \mid x) - \log \pi_0(y^- \mid x), \tag{7.7}$$

$$u_\theta(x) = \Delta_\theta(x) - \Delta_0(x). \tag{7.8}$$

DPO 的负对数似然为

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{\mathcal{D}} \log \sigma(\beta u_\theta). \tag{7.9}$$

对一个样本求梯度：

$$\nabla_\theta \mathcal{L} = -\beta \sigma(-\beta u_\theta) \left[\nabla_\theta \log \pi_\theta(y^+ \mid x) - \nabla_\theta \log \pi_\theta(y^- \mid x) \right]. \tag{7.10}$$

reference 固定，不参与反向传播。在初始化 $\pi_\theta = \pi_0$ 时，$u = 0$，损失为 $\log 2$，但梯度通常非零；若程序得到零梯度，应检查是否错误地把 policy 一起 detach。

7.3 beta 的两种语境不能混为一句经验法则

在分布问题 (7.2) 中，固定真实奖励时，较大 β 对偏离 reference 的惩罚更强。但在实际 DPO 的局部梯度 (7.10) 中，β 又直接乘在 margin 前；初始化附近的梯度幅度正比于 β。因此“beta 越小训练越保守”没有普遍依据。优化器、数据噪声、训练步数与 margin 饱和程度共同决定最终偏移。

比较 $\beta \in \{0.05, 0.1, 0.2\}$ 可以作为一个明确的实验网格，不能当作跨任务推荐值。保持学习率搜索次数和训练 token 一致，记录验证集上的 chosen/rejected log-prob、margin、回答长度、任务正确率和对 reference 的偏离估计。验证 margin 很大但正确率下降时，应回看标签而不是继续延长训练。

7.4 序列概率、掩码和 reference 缓存

标准自回归回答概率是

$$\log \pi(y \mid x) = \sum_{t=1}^{|y|} \log p(y_t \mid x, y_{<t}). \tag{7.11}$$

这里的回答包含哪些终止标记由数据契约规定。prompt 参与上下文，但不计入这段和；padding 只用于张量对齐，其位置既不计入 loss，也应在注意力中屏蔽。chosen 与 rejected 的 prompt 应完全相同，模板、特殊 token、截断上限亦须一致。两个回答分别被 token 化后，边界 token 的合并可能不同；优先使用模板提供的 assistant mask，并检查解码后的监督段。不能用“prompt 字符串的 token 数”盲切一整段对话。

Listing 7.1 DPO 的核心计算；输入是回答 token 的 log-prob 和

```
# All four inputs are one scalar per preference pair.
policy_margin = policy_chosen - policy_rejected
reference_margin = ref_chosen - ref_rejected
u = policy_margin - reference_margin
loss = softplus(-beta * u).mean()
```

使用 `softplus` 避免直接计算 $\log \sigma$ 的下溢。reference 缓存键至少含模型 commit、adapter 状态、tokenizer、模板哈希、样本 token 序列、掩码和截断规则。使用同一基座上两个 adapter 时，应确认切

换到 reference 分支确实关闭了可训练更新；dropout 也要按定义处理，不能把随机 reference 分数当固定标签。

7.5 用三个数字理解目标函数

考虑一个人工构造的比较：policy 对好、坏回答的序列 log-prob 为 $-2,-4$，reference 为 $-3,-4$。于是 $u=(2)-(1)=1$。当 $\beta=0.1$ 时，损失为 $\log(1+e^{-0.1})\approx 0.644397$；交换标签后变为 $\log(1+e^{0.1})\approx 0.744397$。这些是给定对数概率下的教学数值，不是真实语言模型训练结论。

如果 policy 和 reference 都对某个回答增加相同的对数概率量，reference-relative margin 不变。若只让 policy 对 rejected 的概率下降，则偏好损失可以下降，即使 chosen 的绝对概率也下降。这解释了为什么需要同时监控两者的 NLL，而不能把 DPO loss 下降直接称作“学会正确答案”。第十二章给出独立可重跑的数值扰动。

7.6 长度与标签噪声的受控诊断

长回答含更多负的 token log-prob，但 DPO 用的是 policy 与 reference 的差。若新增 token 在两模型中有相同条件 log-prob，增加长度不会改变 u；若其平均 log-ratio 不为零，margin 会随长度累积。长度偏置的方向依赖这条比率，不能仅凭“序列概率是乘积”判断。

标准 DPO 的和若改为平均，就改了目标。SimPO 明确使用长度归一化奖励和 margin[36]；这可以作为另一方法进行比较，不能仍标作完全相同的 DPO。对同义短答/长答、正确短答/错误长答、错误短答/正确长答分别做分桶，并交换 judge 位置，才能分清事实偏好与形式偏好。

若标签以概率 η 独立翻转，某一真实比较的平均二元损失为

$$\ell(z)=-(1-\eta)\log\sigma(z)-\eta\log\sigma(-z),\qquad z=\beta u. \tag{7.12}$$

当 $0<\eta<1$，其无约束最优点满足

$$\sigma(z^*)=1-\eta,\qquad z^*=\log\frac{1-\eta}{\eta}. \tag{7.13}$$

在无噪声且可分的理想数据上，逻辑损失倾向于把 margin 推向无穷；含噪数据不支持这种无限自信。这个结论只适用于该独立翻转模型，真实标注噪声通常依赖题目、长度与来源。对争议样本重复标注，或设置软标签，必须说明新目标和其假设 [37, 38]。

7.7 其他偏好目标应按数据与约束选择

原始方法定义分别见 [37, 39, 40, 36, 41, 42]，目标与代价的差异汇总于表 7.1。不要把“无 reference model”翻译为“零额外成本”，也不要把多个方法的训练 loss 数值横向比较。公平的共同输出是同一评测集上的行为与资源。

表 7.1 几种偏好方法改变的对象，而非普遍性能排名

方法	数据和主要目标	需要核验的代价
DPO	成对比较，reference-relative logistic	固定 reference、覆盖与标签一致性
IPO	成对比较，采用不同偏好目标	尺度和理论假设，不等于自动抗所有噪声
KTO	期望/不期望的单项反馈	正负比例与参考点估计
ORPO	NLL 与 odds-ratio 偏好项	无显式 reference，但有两项权重与数据约束
SimPO	长度归一化隐式奖励、目标 margin	长度、采样与评测需单独控制
RRHF/SLiC-HF	候选排序/序列似然校准	生成候选的质量与成本

练习与提示

设标签翻转率 $\eta = 0.2$。求 z^*，并比较 $\beta = 0.1$ 和 0.2 时对应的 u^*。提示：目标的最优分类 logit 相同，reference-relative margin 为 z^*/β。再给 chosen 添加两枚 token，使 policy/reference 各枚 log-prob 分别为 $-1.0/-1.2$，计算 u 的变化。答案应为 $z^* = \log 4$，新增 $u = 0.4$；这不代表真实模型可以不改变其他概率而实现这种局部干预。

偏好优化的可交付结果应是一套有版本的比较准则、数据与诊断，而不是一个最小训练 loss。下一章进入需要在线采样的反馈优化：它更接近策略实际访问的状态，也更依赖采样预算和验证器的可靠性。

第八章　PPO、GRPO 与可验证反馈：策略更新的边界

DPO 主要复用离线候选，强化学习则让当前策略产生回答，并用反馈决定哪些行为应当更常出现。在线并不天然更好：若初始模型几乎没有正确轨迹、验证器只覆盖表面格式或采样代价超出预算，系统可能很快优化了错误目标。本章保留足以阅读实现的基本公式；通用策略优化理论由《强化学习》分册展开，通用后训练范式由《大语言模型后训练》比较。本书关心已有小模型是否值得付出这项成本。

8.1　三个策略与三种奖励来源

区分三个对象：π_θ 是正在更新的策略，π_{old} 是生成当前一批轨迹的行为策略，π_0 是约束行为漂移的 reference。第一轮后二者可能不同；把它们共用一个会持续更新的模型对象，会改变算法。

反馈可以是人类比较训练出的奖励模型、AI 评判，也可以是程序/环境验证。RLVR（reinforcement learning with verifiable rewards，可验证奖励强化学习）描述奖励来源；PPO 和 GRPO 描述优化与优势估计。因此 RLVR 可以用 PPO 或 GRPO，GRPO 也可以使用不完全可验证的奖励，三者不能列作互斥算法。

对固定提示分布 D_x，理想目标可写成

$$J(\theta) = \mathbb{E}_{x\sim D_x, y\sim\pi_\theta}[R(x,y)] - \beta\mathbb{E}_{x\sim D_x} D_{\text{KL}}(\pi_\theta(\cdot \mid x)\|\pi_0(\cdot \mid x)). \tag{8.1}$$

R 是环境或奖励模型给出的任务分数，KL 是正则项。实现可把 KL 分配到 token 奖励并进入优势，也可把它单独加在 loss 中；必须标明是哪一种，避免把同一系数计入两次。

8.2　从对数导数到有用的基线

生成第 t 枚 token 时的状态为 $s_t = (x, y_{<t})$，动作是 y_t。令 r_t 为该步奖励，T 为轨迹终点，$G_t = \sum_{k=t}^{T} r_k$ 为从该步起的累计回报。忽略折扣并假定可交换微分与期望，任务奖励梯度为

$$\nabla J_R = \mathbb{E}\left[\sum_t \nabla \log \pi_\theta(y_t \mid s_t)(G_t - b(s_t))\right]. \tag{8.2}$$

$b(s_t)$ 若不依赖当前采样动作，且在策略梯度项中 stop-gradient，则条件期望为零，可以降低方差。任意用同组包含自身的奖励平均值不再完全符合这个动作独立条件；组相对估计是一种有具体偏差/尺度的估计器，不能直接套用“所有基线完全不改变期望”的口号。

PPO 常用价值函数 V_ϕ 和广义优势估计（GAE）。引入折扣因子 $\gamma \in [0,1]$ 与控制估计偏差和方差的 $\lambda \in [0,1]$，令

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t), \tag{8.3}$$

$$\widehat{A}_t = \sum_{k\geq 0} (\gamma\lambda)^k \delta_{t+k}. \tag{8.4}$$

终止与截断的 bootstrap 规则必须明确：回答正常 EOS 终止和因最大长度被切断，未必是相同的环境终点。λ 在方差与偏差间折中；价值网络的额外存储、前向、反向和训练稳定性都应列进预算。

8.3 PPO clip 不是对全部概率变化的硬约束

在一批 old-policy 轨迹上，token 重要性比率为

$$\rho_t(\theta) = \frac{\pi_\theta(y_t \mid s_t)}{\pi_{\text{old}}(y_t \mid s_t)}. \tag{8.5}$$

PPO 的常用代理目标 [43] 为

$$J_{\text{clip}} = \mathbb{E}_t \min\left(\rho_t \widehat{A}_t, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\widehat{A}_t\right). \tag{8.6}$$

正优势动作在 $\rho_t > 1+\epsilon$ 后不再由这一项获益；负优势动作在 $\rho_t < 1-\epsilon$ 后同样被截平。但参数共享、其他样本与其他 loss 仍可改变该动作概率，clip 不保证 KL 或所有比率留在区间内。应记录 clip fraction、old-policy 近似 KL、reference 偏离与更新次数；过多复用同一 rollout 会使训练越来越偏离采样分布。

对优势、回报、奖励和 token loss 使用何种标准化，会改变任务的权重。例如按回答平均与按 token 平均，对短答和长答的有效权重不同；以“同一 PPO”命名不能替代完整配置。

8.4 GRPO 用组内比较替代显式 critic

对同一提示，π_{old} 独立生成 G 个候选。若各回答只有终点奖励 R_i，常见组相对优势是

$$\bar{R} = G^{-1}\sum_i R_i, \qquad \widehat{A}_i = \frac{R_i - \bar{R}}{\sigma_R + \varepsilon}. \tag{8.7}$$

标准差使用总体还是样本定义、ε 放置位置、是否 stop-gradient 都应记录。将同一 $\widehat{A}_i$ 赋给该回答中的 token，一个常见的结果奖励版本为

$$J_{\text{group}} = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i}\min(\rho_{i,t}\widehat{A}_i, \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\widehat{A}_i) - \beta\widehat{D}_{\text{KL}}. \tag{8.8}$$

这是指定归约方式的一种 GRPO 形式，不能把所有实现都归结成这一行。DeepSeekMath 引入相应组相对路线，DeepSeek-R1 展示了更大规模的推理训练流程 [44, 45]。取消 critic 节省其内存，但不取消策略、reference、rollout KV、奖励执行与组采样成本。

给定 $R = [0, 0, 1, 1]$，总体均值为 0.5，标准差为 0.5，忽略数值稳定项后优势为 $[-1, -1, 1, 1]$。若全为零，分子全为零，不会因为除以一个很小的数就出现有用学习信号。$G = 1$ 也没有组内比较。全对组同样可能没有优势；动态筛选会改变实际训练提示分布，所以必须记录抽样和丢弃比例。

组标准差还会按题目重新缩放梯度。二值奖励的通过概率接近边界时，尺度与难度相关；长回答按 $1/T_i$ 归约也会改变权重。DAPO 和 Dr. GRPO 等工作讨论了这些设计选择 [46, 47]。这里引用它们解释归约和筛选不能省略，而不宣称一个变体对所有小模型更优。复现时记录 token/sequence normalization、组方差归一化、动态采样、最大回答长度和截断奖励处理。

8.5 KL 估计需要注明采样分布

在当前策略下采样动作时，$\log \pi_\theta(a \mid s) - \log \pi_0(a \mid s)$ 的条件期望为 forward KL；单个样本可以为负。另一常用非负样本形式令 $q = \pi_0(a \mid s)/\pi_\theta(a \mid s)$：

$$k(q) = q - 1 - \log q \geq 0. \tag{8.9}$$

若动作来自当前策略且支持集相容，$\mathbb{E}[q] = 1$，其期望同样为该状态上的 forward KL。但实际 rollout 来自 old-policy；更新后的比率与状态分布都有偏移。把它作为代理正则项时，不应把观测均值直接称为对最终策略的无偏全轨迹 KL。

同时区分任务奖励与 KL 罚项后的总回报。若只画总回报曲线，模型可能因更像 reference 获得更高分，却没有提高正确率。验证集应以固定解码或固定采样预算单独测任务成功率。

8.6 验证器比格式奖励更接近目标，也仍会失效

一个答案验证器至少分解析、语义/执行检查和资源约束三层。对代码任务，不可信生成代码只能在隔离环境中执行，固定工具权限、时间和内存上限；不能为运行一个教程而在宿主上直接执行任意模型输出。对数学任务，字符串匹配需明确数值容差、单位、等价表达式与多答案；对工具任务应检查最终环境状态，避免只奖“看起来像调用”的文本。

表 8.1 奖励测试要覆盖的反例

情形	错误验证器可能奖励什么	独立检查
多个最终答案	把正确数字藏在许多错误答案中	仅接受规定位置的单一合法答案
格式正确但事实错误	合法 JSON 或完整标签	分离格式通过率与语义正确率
超长与超时	靠重试穷举或卡住执行器	总 token、工具次数和墙钟限制
测试泄漏	复述隐藏答案或测试内容	来源隔离、未见模板与私有测试
解析器差异	Unicode/空白/单位边界漏洞	规范化测试、人工抽查和变形测试

结果奖励稀疏但定义直接；过程奖励可能改善信用分配，也可能把有说服力的错误推理评为高分。引入过程模型要另外记录标注或教师成本、误报与漏报，并对正确答案但不同有效步骤的轨迹做检查。可验证任务和通用人类价值不等价，不能把数学通过率外推为普遍安全性。

8.7 小模型何时进入、何时退出在线训练

进入条件是：固定 SFT 基线已能产生一定比例的有效候选；采样和奖励代码可重放；人工可核查的小集合上验证器一致；预留了 OOD 与通用回归集。没有区分度时，应先比较增加可验证示范、降低任务难度或修正 parser，而不是只增大 RL 训练步数。

每轮保存提示 ID、old checkpoint、采样种子、候选 token、原始奖励分项、解析失败原因、优势、KL 和裁剪统计。若 OOD 成功率下降、长度持续冲顶、正确率与奖励分离或 verifier 新漏洞出现，停止并回滚。容忍阈值由应用事先规定；在没有需求依据时不编一个普适百分比。

练习与提示

令 $G = 4$，奖励依次为 $[0, 0, 0, 1]$，采用总体标准差。求四个标准化优势，说明与 $[0, 0, 1, 1]$ 相比唯一成功轨迹的权重怎样改变。提示：均值 $1/4$，标准差 $\sqrt{3}/4$，前三项为 $-1/\sqrt{3}$，最后为 $\sqrt{3}$。这说明标准差归一化会改变不同通过率提示的梯度尺度，并非仅仅“减去均值”。

本书本次执行的实验没有运行 PPO/GRPO，也没有获得 RLVR 性能结论。第十一章给出进入和退出门槛，第十二章把实际运行范围与未运行方案列开；读者可以在验证器和预算满足前提后扩展，而无需把未执行的计划误认为复现结果。

第九章　选择适配路径：教师蒸馏、领域知识与持续学习

SFT、偏好优化和可验证反馈都需要训练信号，但信号来自哪里、模型缺少什么能力，是另一个决策问题。教师蒸馏借助其他模型的输出，领域继续预训练改变语言建模的数据分布，持续学习关注按时间到来的新任务，多 adapter 则管理多套更新的隔离与组合。它们可以组合，却不是必须依次经历的阶段。本章将这些路径放在同一张决策地图中，并说明每条路径能够证明什么、还需要验证什么。

9.1　先区分知识缺口、行为缺口与版本管理

表 9.1　不同适配路径对应的问题

路径	输入与目标	应设置的独立参照
教师蒸馏	教师答案、概率或表示，学习其可迁移行为	原数据 SFT、不同教师成本与过滤规则
领域继续预训练	领域原始文本，适应词汇与语言统计	不做该阶段直接 SFT；可检索知识的方案
持续学习	随时间到来的任务，兼顾新旧能力	独立任务模型、联合训练与历史检查点
多 adapter	多个领域或任务的增量与路由	给定正确路由的上限、实际路由及静态合并

例如，模型会提取信息却不了解某行业术语，可以先检查领域语料或检索是否有帮助；模型已有知识却总输出错误 JSON，则更直接的干预是格式示范与解析器反馈。若多个客户各自需要不同风格，隔离 adapter 可能比反复覆盖一个模型更易维护。这个判断仍需要小规模对照，不能由任务名称直接推导出固定配方。

9.2　蒸馏到底传递哪一种信号

答案蒸馏把教师生成的文本作为监督标签，本质上仍可使用 SFT 的 token 交叉熵。它不需要教师公开 logits，易于离线缓存，也可跨架构与 tokenizer 使用；代价是只观察被采样的答案，不能知道教师如何给其他候选分配概率。缓存必须保存教师版本、提示、采样参数、原始输出、过滤原因和许可条件。验证正确性后保留的数据，与未过滤的教师文本是不同实验条件。

Logit 蒸馏则传递某个共同输出空间上的分布。设教师与学生在同一个前缀 s_t 上，使用相同词表与 token 含义，温度为 $\tau > 0$：

$$p_T^{(\tau)} = \mathrm{softmax}(z_T/\tau), \qquad p_S^{(\tau)} = \mathrm{softmax}(z_S/\tau). \tag{9.1}$$

一种常见目标是前向 KL

$$\mathcal{L}_{\mathrm{KD}} = \tau^2 D_{\mathrm{KL}}\big(p_T^{(\tau)} \,\|\, p_S^{(\tau)}\big), \tag{9.2}$$

再与真实标签或经过验证的硬标签损失组合。DistilBERT 与 MiniLM 展示了 logits、表示或注意力关系等不同知识迁移信号 [48, 49]；它们的特定架构与任务结果，不能直接作为任意 decoder-only 模型的效果保证。

9.2.1 为什么常见公式前面有温度平方

把教师分布视为固定目标，不乘温度平方时，对学生原始 logit 的梯度是

$$\frac{\partial D_{\mathrm{KL}}(p_T^{(\tau)} \| p_S^{(\tau)})}{\partial z_{S,k}} = \frac{p_{S,k}^{(\tau)} - p_{T,k}^{(\tau)}}{\tau}. \tag{9.3}$$

在两边使用同一温度、词表固定，并且温度相对于居中 logits 足够大的近似下，概率差本身还约按 $1/\tau$ 缩小，故乘 τ^2 可补偿这一量级的梯度衰减。它不是任意温度下的精确梯度不变定律。温度会改变分布中哪些差异更重要，KL 方向和混合系数也会改变优化行为。因此，应报告温度、KL 方向、硬软损失系数以及各项实际梯度尺度，而不是把温度平方当作保证所有配置公平的开关。

温度较高会使分布更平滑，既可能传递相似候选之间的信息，也可能稀释学生最需要学习的尖锐决策；容量受限时，完整复现教师的细粒度概率并非总是最有价值。推理通常仍使用原模型温度定义及独立解码配置，训练蒸馏温度不能直接当作推理温度。

9.2.2 Tokenizer 不同，不能逐下标做 KL

两个模型的第 100 个词表位置未必表示同一个文本片段，同一个字符串的分词长度也可能不同。即使词表大小相等，原始 logits 也不能自动对齐；随意截取共同长度、补零或按词表序号配对都没有概率意义。逐 token KL 需要共同支持集与相同前缀语义，tokenizer revision、特殊 token、chat template 与截断方式都属于这一前提。

跨 tokenizer 最稳妥的起点是教师输出文本，验证后按学生 tokenizer 重新编码，进行序列级硬标签学习。若要传递软概率，需要定义共同的字符串、字节或片段事件并处理多种分词路径的概率与位置对齐，这是一项独立方法，不能由一次张量 reshape 完成。只拿到教师 top-k logits 时，未返回 token 的总概率质量也不能忽略；把 top-k 重归一化后得到的是截断分布目标，应报告截断规则与丢失质量。

中间层蒸馏同样需要对齐假设。学生隐藏维度较小时，可以学习投影 P，比较 $H_S P$ 与教师表示；但前提是比较位置对应同一文本跨度，层的语义和归一化尺度合理。不同注意力头数下直接逐元素匹配注意力图并非天然成立。增加投影、对齐或教师缓存，还会增加训练参数和计算，不能只报告学生主网络成本。

9.3 答案、过程与可验证轨迹

过程蒸馏试图传递中间分解、工具操作或解题步骤，其价值在于为学生提供比最终答案更密集的监督。但教师的长文本解释可能冗余、前后不一致，或依靠学生尚不具备的知识。最终答案正确也不能证明每个中间步骤正确；把所有 token 一律当作真值，可能放大教师错误与无效风格。

可执行代码、经检查的证明步骤、带输入输出的工具轨迹和简洁的解题依据，提供了不同强度的可验证性。应区分轨迹解析通过、每步执行通过与最终任务通过，保留失败轨迹及拒收原因。比较 answer-only 与过程数据时同时报告监督 token 和教师成本：更长的轨迹意味着更多训练计算，只固定样本条数不足以归因。教师不公开中间过程时，应使用能够合法获得和验证的答案或操作序列，不能把不可访问的内部状态当作数据来源。

一个实用的对照是对相同问题分别保留简洁正确答案、验证过的过程、未经过程检查但最终正确的长回答。若第三组训练损失下降得最快、任务验证却较差，解释应落在信号质量和容量分配上，而不是把“更长过程”自动等同于更强推理。

9.4 On-policy 蒸馏：在学生会到达的位置提供反馈

离线教师答案主要覆盖教师自身访问的前缀。学生推理早期发生偏差后，会进入另外的状态分布，后面的模仿信号便可能不够。On-policy 蒸馏让学生先生成轨迹，再请教师对这些前缀提供目标。Agarwal 等的工作以广义蒸馏目标研究这种分布错位 [50]。用前向 KL 表示其中一种可理解的实例，可写为

$$\mathcal{L}_{\mathrm{on}}(\theta;\bar{\theta}) = \mathbb{E}_{x,\, y\sim\pi_{\bar{\theta}}(\cdot|x)}\left[\frac{1}{|y|}\sum_{t} D_{\mathrm{KL}}\big(p_T(\cdot \mid s_t)\,\|\,p_\theta(\cdot \mid s_t)\big)\right], \tag{9.4}$$

其中 $s_t = (x, y_{<t})$，$\bar{\theta}$ 是本轮采样用的学生快照。优化这批已生成前缀时，采样路径通常停止梯度；不能把式子误读为已经包含对离散采样分布求导的策略梯度。使用其他 KL 方向或广义散度会得到不同目标，不能将其混称为同一公式。

这里的教师调用成本需要细拆。学生生成是自回归的；若能本地读取完整教师 logits，可把学生的完整轨迹交给教师做一次因果前向，得到各位置的条件分布，不必每个 token 发一次独立请求。教师的评分 token、学生生成 token、教师额外生成 token 和候选筛选次数仍都要计入。远程接口若只返回采样文本而不提供所需分布，就无法直接实现式 (9.4)，需要改成回答纠正、候选选择等另一种方案，并明确其训练目标。

On-policy 不等于教师总能修正错误前缀。一个已不合法的工具调用、互相矛盾的历史或截断到一半的表达，可能使教师条件分布也没有好的恢复策略。可以将教师示范前缀与学生前缀混合，限制生成长度，或对可恢复状态筛选；这些选择都会改变训练分布。学生持续更新后旧轨迹逐渐变成离策略数据，应记录采样模型版本、轨迹年龄和重复使用次数。

蒸馏公平比较至少控制两种预算：学生监督 token 与教师资源。相同学生 token 下，更高成本的教师可能带来更好数据；相同教师预算下，不同方法能获得的数据量又不同。宜分别画两条质量曲线，列出每提示候选数、教师生成与评分 token、过滤保留率、学生 rollout 长度和实际耗时。只以最终保留样本量作分母，会把被拒绝候选的成本隐藏掉。

9.5 领域继续预训练：改变分布，不能替代事实核查

领域继续预训练（domain-adaptive pretraining，DAPT）在原始领域文本上继续优化语言建模目标；任务自适应预训练还可使用更接近具体任务的无标签语料。相关经典研究说明领域与任务语料可以提供不同适配信号 [18]。对自回归模型，一个两阶段方案为

$$\theta_1 = \arg\min_{\theta} \mathbb{E}_{u\sim D_{\mathrm{domain}}}\left[-\sum_{t}\log p_\theta(u_t \mid u_{<t})\right], \tag{9.5}$$

$$\theta_2 = \arg\min_{\theta} \mathcal{L}_{\mathrm{SFT}}(\theta; D_{\mathrm{task}}), \qquad \theta \text{ 从 } \theta_1 \text{ 开始.} \tag{9.6}$$

这是目标的说明，不要求实际达到全局极小值。DAPT 的语料分布、分词效率和文档边界，与 SFT 的回答掩码不同；可按预算采用全参数或 adapter，但约束后的可学习容量也不同。

领域文本较多不意味着知识必然更新准确。语料可能包含过时规则、互相矛盾的版本与重复网页，语言建模损失还会奖励复现这些内容。对于经常更新、需要可追溯引用的事实，检索或外部工具应作为独立基线；微调可能负责理解格式与使用证据，而不必把所有事实写入权重。DAPT 若改善领域困惑度却没有改善目标问答，需要区分领域语言熟悉程度与实际可用知识。

最小对照应包括原基座、直接 SFT、仅 DAPT、DAPT 后 SFT；按总 token 预算报告各阶段支出，避免把多训练一个阶段的收益全部归为阶段顺序。领域文档应按来源或实体分组划分，近重复版本不能跨训练与验证；评测题、答案解释和检索知识库之间也要检查污染。版权、隐私与许可记录沿用第三章的数据流程。

9.6 持续学习：给新能力和遗忘各留一个位置

设按顺序学习 K 个任务，$R_{i,j}$ 表示学完第 i 个任务后在任务 j 上的分数，$R_{0,j}$ 为基座分数。新任务收益可以看 $R_{j,j} - R_{0,j}$；学到最后再报告 $R_{K,j}$，才知道收益是否保留。对于分数越高越好的指标，历史回落的一种描述为

$$F_j = \max_{j \le i < K} R_{i,j} - R_{K,j}, \qquad j < K. \tag{9.7}$$

它衡量相对于历史最好检查点的下降，可能与相对基座的下降不同；困惑度等越低越好的量应改符号或单独报告。历史最大值还受评测噪声影响，因此完整任务矩阵、固定提示及配对样本差值比一个遗忘平均数更可解释。

Replay 混入旧任务或通用样本，可写成每个来源先按监督 token 归一化的损失混合

$$\mathcal{L} = \lambda \mathcal{L}_{\text{new}} + (1 - \lambda)\mathcal{L}_{\text{replay}}. \tag{9.8}$$

若数据采样比例已经实现了所需权重，再额外乘同样系数会重复加权。应记录实际见到的各来源 token、旧样本重复次数和难度变化。Replay 会增加训练成本，也可能放大旧数据中的污染或偏差，不能只记录其名义占比。

另一种做法是在历史提示上约束新旧模型输出分布，或使用近似 Fisher 权重的局部参数正则。前者依赖旧模型及其前向成本，后者依赖重要性估计和局部近似；它们都不能在没有覆盖的能力上保证零遗忘。Adapter 隔离保留历史参数，降低相互覆盖的直接风险，但任务共享底座、路由和生成环境变化后仍可能发生行为回归。应把参数可回滚与能力已验证区分开。

9.7 多 adapter 的路由、合并与低秩压缩

在同一底座上，任务 k 有 $\Delta W_k = s_k B_k A_k$。给定输入的混合可以写成

$$\Delta W(x) = \sum_{k=1}^{K} g_k(x) \Delta W_k, \qquad g_k(x) \ge 0, \quad \sum_k g_k(x) = 1. \tag{9.9}$$

固定路由可由已知任务标识选择，动态路由则根据输入预测。路由器训练集、拒识策略和额外延迟都属于系统；应分别测已知正确路由下的 adapter 能力与真实路由后的端到端结果。若门控随 token 或隐藏状态变化，通常不能预先合成一个固定权重矩阵。仅对每个请求固定系数时，才可为该请求讨论静态加权合并。

即使系数固定，权重相加也不保证行为叠加。两个任务可能争用同一特征，或在某类提示上产生相反的输出偏好。除任务平均分外，应检查各任务、交叉任务提示、通用能力、格式与安全边界；权重余弦相似度只是诊断量，不能代替行为测量。新增任务多时，还要测 adapter 加载、缓存命中与切换成本。

对非负权重 w_k，加权更新可通过堆叠精确表示：

$$B_* = [w_1 s_1 B_1 \ \cdots \ w_K s_K B_K], \qquad A_* = \begin{bmatrix} A_1 \\ \vdots \\ A_K \end{bmatrix}, \qquad B_* A_* = \sum_k w_k s_k B_k A_k. \tag{9.10}$$

其 rank 上限随各 adapter rank 之和增长；若再压成较小 rank，就引入了新的近似误差。SVD 给出权重 Frobenius 意义下的最佳截断，但任务更关心激活加权后的误差，应重新评测。合并前还必须确认底座权重哈希、模块映射、tokenizer 与特殊 token 一致；来自不同底座的同名矩阵不能直接视作共同坐标系。

9.8 联邦更新与隐私边界

在客户端上传 LoRA 因子可以减少通信，但因子的坐标不唯一。一个最小反例是标量 $A_1 = B_1 = 1$，$A_2 = B_2 = -1$，两个客户端的产品都为 1；逐因子平均却得到 $\bar{A} = \bar{B} = 0$，其产品为 0。因此通常有

$$\left(\sum_i w_i B_i\right)\left(\sum_i w_i A_i\right) \neq \sum_i w_i B_i A_i. \tag{9.11}$$

即使所有 rank 相同，也不能忽略这个双线性问题。应在共同基座的更新产品空间定义聚合，再决定是否堆叠、对齐或压缩。不同客户端的数据量、任务分布和局部步数还会改变聚合含义，参数平均不能自动等价于联合训练。

小文件也不提供隐私保证。更新可能保留训练样本信息，分发的 adapter 还可能含有非预期行为。若需要差分隐私，应明确裁剪单位、噪声机制、采样方式、隐私会计及累计预算；安全聚合保护传输中的单个更新可见性，也不等于最终模型满足同一种隐私定义。这些机制都有质量和系统成本，应按实际部署要求评测，不能用“只训练少量参数”替代证明。

练习与提示

设计一个教师与学生 tokenizer 不同、只能访问教师文本接口的实验。说明哪些目标仍能实现，哪些逐 token 软蒸馏公式不能直接使用。然后为三个按顺序到来的领域画出 $R_{i,j}$ 的空白矩阵，写出必须保留的检查点与 replay token 记录。提示：先确定可观测信号，再选择损失；先保留完整矩阵，再压缩为摘要指标。

本章要点

蒸馏需要信号与输出空间对齐，on-policy 方法还需要计入学生轨迹和教师评分成本。DAPT 与 SFT 解决不同分布问题，持续学习以完整新旧任务矩阵衡量保留能力。多 adapter 的存储隔离、产品聚合、动态路由与隐私保护各有独立假设，不能由低秩形式自动推出。

第十章　评测、导出与部署：从分数到可信决策

一个 adapter 可以在训练提示上更准确，同时在未见表达、通用指令或长输入上变差；合并后的模型也可能因错误的底座版本而丢失全部收益。因此，本章把评测和导出视为同一条证据链：先固定输入与能力定义，再验证权重转换，最后用真实部署负载衡量成本。训练 loss 是过程诊断，正式结论应回答“对谁、在哪种输入、用多少资源、改善了什么”。

10.1　把评测拆成任务收益和代价

至少保留三份相互独立的数据：训练集、用于选配置与早停的验证集、最终只报告一次的测试集。合成任务按规则族或模板族切分，业务数据按实体、用户、时间或来源簇切分；相同答案不等于泄漏，但相同推理实例的改写往往会泄漏。训练中反复查看测试样本会使它成为事实上的验证集。

表 10.1　同一 checkpoint 应回答的评测问题

维度	可操作指标	解释时需要的条件
目标任务	exact match、字段 F1、执行成功率	parser、大小写/空白规则与失败分母
通用保持	固定通用集的得分和 NLL 变化	模板、shot 数、答案长度和污染范围
未见输入	来源/模板/实体分组后的成功率	明确何种 OOD，不能声称覆盖所有域外
偏好与安全	分维度胜率、误拒率、攻击成功率	judge 版本、位置交换、人工核查
行为长度	输出 token 分位数、截断/空答率	max new tokens、终止规则与重试数
部署成本	延迟、吞吐、峰值内存、存储	硬件、精度、批量、并发和上下文

MMLU、GSM8K 与 IFEval 各有具体数据设计 [51, 52, 53]，可以提供通用知识、数学与指令遵循的部分坐标，却不能替代目标场景的测试。微型任务上成功也不等于通过这些基准。本书的实测指标和运行范围见第十二章。

对高分更好的指标，定义遗忘差 $F = M_{\text{base}} - M_{\text{adapt}}$；对 NLL 这类低分更好的指标，应报告 $\Delta\text{NLL} = \text{NLL}_{\text{adapt}} - \text{NLL}_{\text{base}}$。逐个切片报告，避免把方向不同的量直接平均。基座原本不会的任务不能仅用“遗忘率”描述。

10.2　统计不确定性和搜索预算

同一输入在两个模型上评测，优先分析配对差 $d_i = m_i^{\text{adapt}} - m_i^{\text{base}}$。bootstrap 每次有放回抽取样本索引 I_b，计算

$$\delta_b = \frac{1}{n} \sum_{i \in I_b} d_i. \tag{10.1}$$

若输入按同一来源或模板高度相关，应重采样这些组，不能把近重复句子当独立样本。区间反映指定样本总体和抽样假设下的不确定性，不覆盖未尝试随机种子、数据构造偏差或模型选择偏差。

两个模型在 100 题上分别对 70、74 题，不足以单凭 4 个百分点判定可靠提升：还要知道哪些题改变、错误是否相关以及测试集是否参与选择。报告每条预测、配对变更、样本数和区间。单种子

试验可报告固定运行结果，但不能用 bootstrap 伪装成跨训练种子稳定性。若方法甲搜索 20 组超参，方法乙只跑默认一次，比较还混入了搜索成本；预算表中记录失败与 OOM 运行，不能只保留胜出的配置。

10.3 人工与模型评判的偏置

Judge 应同时看到同一问题和明确准则，将正确性、完整性、风格、格式与安全分开。MT-Bench 研究讨论了位置与冗长度等评判偏置 [54]。对每一对回答交换显示顺序；同一个回答和自己的重复副本也可做一致性检查。若交换后结论翻转，保留为不确定项，而不是强行计胜。

长度匹配可以帮助排查“更长所以更好”的竞争解释，但长度本身可能是目标能力的一部分。例如有严格 20 字限制的任务与需要完整证明的任务不能用同一种长度控制。先定义任务，再按长度分桶，人工抽查事实错误与约束违例。训练使用过的教师充当唯一评判者，可能奖赏自身风格；至少补一个独立验证器或盲评子集。

10.4 把 adapter 导出成一个可追踪的对象

独立 adapter checkpoint 不含完整基座。最小产物契约应记录：基座仓库与不可变 commit、tokenizer 及模板哈希、目标模块的完整名字、rank/alpha/缩放、dtype、adapter 权重哈希、可训练非 adapter 参数、训练数据版本和训练代码 commit。如果新增 token 或解冻 lm head，仅保存低秩矩阵会丢掉这部分更新。

Listing 10.1 实际导出记录应填具体值；字段名不替代记录本身

```
artifact = {
  "base_revision": base_commit,
  "tokenizer_sha256": tokenizer_hash,
  "chat_template_sha256": template_hash,
  "target_modules": resolved_module_names,
  "rank": rank,
  "scaling": {"alpha": alpha, "kind": scaling},
  "adapter_sha256": adapter_hash,
  "extra_trainable_tensors": extra_tensor_names,
  "data_manifest_sha256": data_hash,
  "training_code_revision": code_revision,
}
```

在推理态关闭 dropout，对固定输入比较未合并分支与高精度合并权重的 logits。FP32 下应接近浮点误差；BF16/FP16 矩阵运算顺序改变会有误差，阈值应由精度和任务测试确定，而不是盲用一个绝对常数。比较最大绝对误差、相对误差、首个分歧 token 以及完整行为；greedy 解码在近似并列 logits 下可能放大小误差。

量化基座的反量化值为 $\hat{W}_0$，在同一值上合并得到 $\hat{W}_0 + sBA$；直接把同一个 adapter 加到原始 W_0 是另一个模型。若训练时 adapter 补偿了量化残差，这一替换可能破坏行为。重新量化又加入一轮舍入误差。应保存原始底座引用、独立 adapter、合并高精度权重和部署量化配置四者关系；任何分支都在同一提示集上回归。

10.5 部署账本：权重小不等于响应快

设模型有 N 层，KV head 数为 n_{kv}，head 维度为 d_h，并发请求的有效上下文长度为 L_b，KV 每元素占 s_{kv} 字节，则不含分页/对齐开销的缓存下界为

$$M_{KV} = 2Nn_{kv}d_h s_{kv} \sum_b L_b. \tag{10.2}$$

因子 2 来自 K 和 V。GQA/MQA 使 KV head 数减少；LoRA 本身不自动减少 KV。训练时通常不保留自回归推理用的 KV cache，不能把训练激活和服务 KV 账本混在一起。

Prefill 并行处理整个提示，decode 逐步追加 token；瓶颈随批量、长度、硬件、精度和 kernel 改变。FlashAttention 减少特定注意力实现的 IO 代价 [8]，并不把所有长上下文成本变成常数。NF4 训练存储也不保证部署引擎有快速 NF4 kernel；有时导出到另一种量化格式更合适，但必须重测质量。

TTFT 为请求进入约定计时边界后到首 token 的时间，需说明是否含排队、tokenization 与网络；TPOT 是后续生成 token 的平均间隔，短回答和 EOS 处理会影响分母。吞吐报告完成请求/s 和生成 token/s，并同时报告尾延迟；取消、重试和超时也是用户成本。建议固定短/长提示与短/长回答四类负载，预热后多次测量 p50/p95，给出批量和并发。GPU 计时需同步，否则测到的可能只有异步提交时间。

10.6 推测解码与多 adapter 服务的条件

小模型作为 draft model 生成候选，再由 target 验证，只有当接受率、draft 代价、并行验证效率共同有利时才加速。经过激进偏好微调或量化，draft 分布可能偏离 target 而降低接受率；参数更少不保证总延迟更低。在目标硬件上记录接受率、draft 耗时与总延迟，再报告加速比。

多 adapter 服务还需测切换与加载延迟、缓存命中、并发隔离和路由错误。每个请求绑定正确版本，防止请求中途更换 adapter。业务上线门槛可以是“任务质量下界满足需求且总成本不超过预算”，其中需求阈值由应用预先规定；所有门槛都需带回滚版本。

练习与提示

假设 $N = 24, n_{kv} = 8, d_h = 64$，4 个请求均为 2048 个上下文 token，KV 使用 2 字节。计算理论下界。提示：$2 \times 24 \times 8 \times 64 \times 2 \times 4 \times 2048 = 402653184$ 字节，即 384 MiB。再把请求长度翻倍；即使权重不变，KV 也翻倍。此计算没有包含预分配、碎片和工作空间。

评测的最终交付应让另一个人能定位每一个输入、每一次转换和每一项资源。下一章将这些要求写成五个可以进入、可以暂停、也可以明确退出的实验方案。

第十一章　五套有前提和退出条件的实验方案

本章将原稿的五套配方改为实验协议。它们不共享一个“最佳学习率”，而共享一份数据与证据契约：训练前先说明要改变的行为，固定版本与预算，保留一个能否定方法的基线，训练后同时评测任务收益、遗忘、长度与部署成本。第十二章列出的本次实测才是已经执行的结果；本章较大模型、偏好训练、RL 与蒸馏方案均是供后续运行的计划。

11.1 所有方案共用的入口记录

先给输入生成 manifest：每条样本含稳定 ID、来源/许可、原始文本哈希、切分组、tokenizer commit、模板哈希、输入 token 数、监督 token 数、截断原因。验证与测试在训练前锁定，所有方法读同一分片。padding 不计有效监督 token；packing 效率另报为输入占用率，不能把填满 padding 说成看了更多数据。

预算用三张表表达。第一张是训练预算：总输入 token、监督 token、更新数、微批与累积、梯度精度、seed 和早停规则。第二张是搜索预算：每个方法允许的学习率/位置/rank 配置数以及失败运行。第三张是资源预算：权重、梯度、优化器和激活估算，硬件、软件、墙钟与磁盘上限。固定 token 和固定墙钟是不同的公平问题；不能一边宣称固定全部资源，一边让慢方法多跑几倍时间。

完成一次单 batch 前向与反向后，检查监督位置、每个可训练模块的梯度、冻结参数未被更新，以及优化器 step 后参数哈希是否改变。先能拟合几条明确样本，再扩大；若验证器、mask 或数据版本不一致，停止全部方法比较。

11.2 方案 A：已有小模型的全参数 SFT 与 LoRA 对照

适用前提。基座在原始提示下至少能表达任务格式，标签可核查，数据量足以保留独立验证集，且全参数优化的状态账本在硬件内。这里“全参数”是对已有模型继续训练，不是从零预训练。若只能训练小子集，应把结论限制在该规模。

基线与干预。先测冻结基座，使用完全相同的 tokenizer 和提示；再比较全参数 SFT 与 LoRA。对 LoRA 首先解析模块名字，按架构选 attention 投影，随后才扩展 FFN。固定每一组对照的监督 token、数据顺序和评测提示；学习率各自做同样次数的粗网格搜索，不要求两种优化几何强行共用名义学习率。rank 扫描和固定参数量的位置扫描分开进行。

可检查配置。第一轮不做 packing、不加 NEFTune、不混合额外教师，减少竞争解释。记录 token 加权 loss、验证任务指标、通用回归和参数变化范数。第二轮只改变一个变量，例如在相同监督 token 下加入回放；回放占用了目标任务预算，应在总量中扣除。

退出条件。若训练 loss 不动，先排 mask、注入位置与梯度，而非先加 rank；若训练集进步、分组验证退步，暂停扩大数据重复次数；若目标任务收益建立在明显通用能力损失上，按预先阈值拒绝该配置。导出 adapter 及额外解冻参数，保存同一组回归预测。

11.3 方案 B：NF4 QLoRA 与未量化 LoRA 的资源比较

适用前提。有已验证支持的硬件/backend 和量化库，基座权重允许相应加载，准备了能容纳未量化对照的设备或较小规模。没有兼容 kernel 时，“模型能反量化前向”不代表运行了高效 QLoRA。

基线与干预。最低对照是冻结高精度基座、冻结量化基座、高精度 LoRA 和同位置/rank 的量化 LoRA。第二个冻结基线识别单纯量化损失，第三个训练基线识别适配方法差异。如果显存只容得下量化模型，明确缺少高精度同规模对照，并另用小模型验证机制；不能把不同模型的分数当作纯量化消融。

Listing 11.1 对应 CUDA NF4 方案的关键配置；完整脚本另附

```
bnb = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=torch.bfloat16,
)
# Check BF16 support first; otherwise select a supported dtype.
# Resolve target module names against the actual model.
peft = LoraConfig(r=4, lora_alpha=8,
                  target_modules=["q_proj", "v_proj"],
                  lora_dropout=0.0, bias="none")
```

上述数值是一个可被替换的起始配置，不是本书实证得到的最优值。实际脚本位于 `experiments/`；本机执行状态见第十二章。QLoRA 的 NF4、双重量化和分页机制各自作用不同 [6, 55]，需要分别记录；“load in 4bit”不足以说明格式和计算精度。

退出条件。启动后核对可训练参数数、实际存储 dtype 和 NF4 状态；模型加载后/反向后/优化器 step 后分别测峰值。若 fallback 造成浮点副本或实际峰值超预算，停止，不把未达到的内存节约写进报告。若量化前后任务变化显著，做层级替换和 logits 诊断再调 adapter；最终重新量化的部署权重要独立回归。

11.4 方案 C：基于固定 SFT 起点的 DPO

适用前提。SFT 模型能完成基本任务，偏好对的两个回答都可解析，chosen 确有可解释的优势。对“两项都错”的数据单独标记，优先补示范或事实纠正。reference 通常取该 SFT checkpoint 并冻结，不能随着 policy 一起更新。

数据与预算。按 prompt/来源组切分，记录两项回答长度、标注分歧、采样 policy 和评分标准。先在少量手工对上复现 $\pi_\theta = \pi_0$ 时的 $\log 2$，再训练。固定比较所用的成对监督 token；若用 cached reference，缓存计算时间和磁盘也进资源表。

干预。对 reference、β 和长度处理分开做实验。先固定 reference 扫少量 beta；再固定搜索预算比较另一个可解释 reference。不要随意把标准序列和改成均值后仍称相同 DPO。ORPO 或 SimPO 作为独立方法加入时，保存各自目标和超参；训练 loss 之间不作直接性能排名。

退出条件。chosen/rejected 都越来越不可能但相对 margin 上升、判分只在长答上增加、人工正确率不升或通用指令退化时，暂停并检查标签与长度。部署判定用盲测正确率、位置交换后的偏好结果和资源，不能只看 reward margin。第七章的噪声模型用于检查逻辑，不是一个已知真实噪声率的假设。

11.5 方案 D：可验证任务的短期在线优化

适用前提。独立验证器通过边界、反例与超时测试；SFT rollout 具有非零正确率和组内差异；可支付候选生成、奖励执行、reference 计算和更新成本。安全地执行代码需要隔离环境，不在宿主上

直接运行模型生成程序。

对照。冻结 SFT、等候选/等教师或验证调用预算的拒绝采样加 SFT、短 GRPO 或 PPO。拒绝采样也是强基线：如果过滤正确轨迹后监督训练已经达到目标，在线策略优化未必值得增加复杂度。固定 prompt 分布、最大回答长度和总采样 token，不只固定训练 step。

观测。按任务记录组大小、奖励均值/方差、全对/全错/解析失败组比例、舍弃 prompt 数、实际采样 token、优势归约、clip fraction、old-policy KL 和 reference 正则。PPO 另记 critic 配置和 value loss，GRPO 记组标准差与 token/sequence 归一化。KL 的来源与加入位置只能有一个清楚的定义。

退出条件。验证器出现新漏洞、奖励上升但固定测试通过率不升、输出不断冲到长度上限、熵或通用能力异常时回滚。全错组占主导应先降低难度或增加示范，不能把无学习信号归咎于“训练还不够久”。本书没有执行此在线方案，因而不声称其在选定小模型上优于 SFT。

11.6 方案 E：教师到小模型的分层蒸馏

适用前提。教师使用与数据许可允许蒸馏，答案可独立验证；先明确能取得的是完整 logits、top-k、隐藏态还是只有文本。学生和教师 tokenizer 不一致时，不直接逐 token 做 KL；文本答案可重新用学生 tokenizer 编码，代价是丢失教师完整分布信息。

对照。冻结学生、同预算人工/原数据 SFT、答案蒸馏、过滤的过程蒸馏。若支持对齐词表与前缀，再增加 logit 蒸馏；最后才比较学生自生成前缀上的 on-policy 蒸馏 [50]。不要把不同信息访问权限的方案伪装成仅换了一个 loss。

预算。同时记录教师输入/输出 token、候选数、批次数、过滤率、保留监督 token、学生训练 token 和端到端墙钟。on-policy 每轮需要教师评价学生访问的前缀，无法直接复用所有离线教师缓存；教师可并行计算一个完整生成序列上的分布，但查询上下文仍来自学生。

退出条件。教师步骤经验证含错误、轨迹远超学生上下文、只提高模仿风格而无任务收益，或调用成本超出应用预算时，停止增加软目标；比较较短可验证答案和不同难度配比。学生较小不意味着必须复制教师的全部长推理，也不能用相同训练 epoch 掩盖不同 token 和教师成本。

11.7 统一完成契约与决定下一次实验

每套方案最终保存同样的四组对象：输入 manifest 和配置；全部运行及失败日志；冻结/适配/导出状态的逐样本预测；有版本和哈希的可回滚产物。图表标题写明实测、估算或待运行。若没有充分预算，可交付一个边界明确的否定结论，例如“该设置在未见模板上没有改善”，而不是继续搜索到偶然出现一个更高分。

练习与提示

方案 B 中，量化 LoRA 优于冻结高精度基座，但弱于同 token 的高精度 LoRA。可以得出什么结论？提示：说明适配可能有效、量化可能引入代价；没有冻结量化基线时，还不能分离量化对初始行为的影响。再问：若量化方法用了两倍搜索配置，如何重新设计比较？

第十二章　算例与最小实验：把公式变成证据

本章把前面约定的张量、监督对象、适配器和偏好目标放进几个可以独立核查的实验。它们回答的是“实现是否符合公式、在一组固定的小任务上发生了什么”。所有数字均来自主题目录的保存记录；手算、合成数据、论文报告和本机实测分别标注。

12.1　实验边界与可复现入口

2026 年 10 月 21 日的适配实验使用 Hugging Face 的 SmolLM2-135M 固定快照 `93efa2f0...75a2`（完整 revision 和文件哈希见 `manifest.json`），三臂共用 tokenizer、提示格式、随机种子（20261021）和数据顺序。任务是根据提示中的映射表，将键 0–3 查成字母 A–D。例如下列提示的目标续写为一个带前导空格的 `A`，随后是 EOS：

Listing 12.1　可人工核查的合成查表任务

```
Table: 0=A, 1=D, 2=C, 3=B
Input: 0
Return the table value.
Answer:
```

数据生成器把 A–D 的 24 种排列按种子洗牌，取其中 12 种。训练集用 8 种规则、2 种措辞模板和 4 个键，共 $8 \times 2 \times 4 = 64$ 条；留出集用另外 4 种规则和另外 2 种模板，共 $4 \times 2 \times 4 = 32$ 条。规则与模板同时留出，因而该设计不能分别估计“新规则”和“新措辞”的影响。它是可追踪的合成任务，不是公开基准，也不能证明预训练语料绝对去污染。

这里使用 base 模型的普通续写，未套 chat template，不添加 BOS、不 packing、不截断。prompt 的 labels 为 −100，答案及 EOS 受监督；padding 按位置同时屏蔽注意力和 loss，不能因为 PAD 与 EOS 共用 ID 而屏蔽真实 EOS。每条样本恰有 2 个监督 token。两条训练臂的微批为 4 条，每步 8 个监督 token；60 步共 480 个监督 token。Frozen 只评测，不更新参数。独立记录审计会复核标签、规则/模板隔离、掩码、预算和逐样本 EM；受控的错标签、漏掩码与错计数副本均被拒绝。

运行记录、配置、逐样本预测、训练日志、模型快照文件哈希和环境说明保存在 `output/experiments/`。摘要由 `summarize.py` 从三臂 JSON 重新计算，避免手工抄写。重跑命令是：

Listing 12.2　重建算例与最小实验记录

```
cd 现代小模型微调
bash manuscript/experiments/run.sh
python3 manuscript/experiments/summarize.py
```

原运行在 Apple Silicon 的 MPS 上完成，保存了模型 revision、软件版本、种子、配置和资源记录。原运行未在执行开始时捕获源码快照；`manifest.json` 中的当前源码哈希只标识交付版本，不是历史执行哈希。新的运行入口会先保存 `run_provenance.json`。MPS 不保证跨版本逐位一致，不能宣称任意机器会得到相同的最后一位。`nf4_cuda_unverified.py` 保存 CUDA NF4 的独立运行方案；本机没有 CUDA，NF4、双重量化和分页优化器均未实测。

已运行项目包括独立标量算例、受控 DPO 扰动与三臂小模型适配。各 LoRA 变体尚未全部复现；7B 级 NF4、DPO 完整微调、PPO/GRPO 与教师蒸馏均未运行，相应公式和方案用于检查实现。

12.2 LoRA 前向、合并与梯度边界

使用第 5.4 节同一组 2×3 权重和 rank $r = 1$ 因子，以 $s = 0.5$ 核对维度与实现。独立结果为

$$Ax = -3, \qquad W_0 x = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \qquad sBAx = \begin{bmatrix} -3 \\ 1.5 \end{bmatrix},$$

从而 $y = W_0 x + sBAx = (-1, 5.5)^\top$。合并后的矩阵为

$$W_{\text{merged}} = W_0 + sBA = \begin{bmatrix} 2 & -1 & 1 \\ 1.5 & 1.5 & -1 \end{bmatrix},$$

再次乘以 x 得到同一个 y，与式 (5.13) 对应。

脚本 `arithmetic_checks.py` 用独立标量循环、向量化计算、合并计算和中心差分相互对照。最大梯度差低于 10^{-8}；$B = 0$ 的初始化边界也通过：首步 $\nabla_A = 0$ 而 ∇_B 通常非零，$\alpha = 0$ 时回到冻结前向。这些检查验证 LoRA 算式，不验证语言模型的 causal shift；后者应另查模型接口与监督位置，不能由矩阵算例代替。

本例有 $r(d_i + d_o) = 1(3 + 2) = 5$ 个适配器参数。若适配器和梯度用 BF16，Adam 两个 FP32 状态各占 20 字节，则不含主权重的下界为 $10 + 10 + 40 = 60$ 字节；若另存 FP32 master copy，再加 20 字节。冻结 W_0 的 6 个 BF16 元素另占 12 字节。这是状态下界，不包括激活、对齐、分配器和框架缓存；训练峰值不能从新增参数量直接推出。

12.3 DPO 损失的受控扰动

第二个算例给定三组人工可读的 token 级对数概率数组，直接检查

$$\ell_{\text{DPO}} = -\log \sigma\big(\beta\left[(\log \pi_\theta^+ - \log \pi_\theta^-) - (\log \pi_0^+ - \log \pi_0^-)\right]\big).$$

这些数组是教学输入，不是预训练模型的输出，也没有隐含的 EOS 项。以第一组“2 + 2 = ?”为例，policy 的 chosen/rejected 分数为 $-0.4/-1.2$，reference 为 $-0.5/-0.7$，因此 reference-relative gap 为 0.6。取 $\beta = 0.5$ 得 margin 0.3，损失为 0.554355。

表 12.1 第一组偏好对的 DPO 扰动（教学算例）

条件	relative gap	margin ($\beta = 0.5$)	loss
标准 reference-relative	0.6	0.3	0.554355
移除 reference（对照）	0.8	0.4	0.513015
翻转 chosen/rejected	-0.6	-0.3	0.854355

脚本还检查 $\beta = 0$ 时损失为 $\log 2$，policy 与 reference 完全相同时 margin 为零。第二组提示要求只输出 `OK`，chosen 的 policy/reference 分数均为 -0.2；rejected 为 `OK and more`，两组 token 分数分别是 $(-0.1, -0.3, -0.3)$ 和 $(-0.1, -0.2, -0.2)$。其标准 gap 为 0.2；$\beta = 0.1, 0.5, 1$ 时损失分别为 $0.683197, 0.644397, 0.598139$。

在第二组 rejected 末尾追加一个 policy 分数 -1.2、reference 分数 -0.1 的 token，gap 变为 1.3，$\beta = 0.5$ 时损失降为 0.420055。变化取决于追加 token 的相对概率，不能概括成“长回答总被惩罚”。若改用各自的 token 均值，原第二组 gap 变为 1/15，损失为 0.676619；这是非标准目标。reference、β、标签噪声和长度处理因此需要分开控制，不能把 loss 下降直接解释成答案更正确。

12.4 三臂小模型适配

Frozen 不更新参数；**LoRA** 在全部 30 层的 `q_proj/v_proj` 注入 $r = 4, \alpha = 8, s = 2$、dropout 为零的适配器，共 230,400 个训练参数。**教学 INT4+LoRA** 使用相同适配器，把全部冻结 Linear（含 `lm_head`）按输出行用对称 scale 量化到 $[-7, 7]$，每字节打包两个四比特值，每次调用时密集反量化；embedding 和 norm 仍为 FP32。它没有 NF4、bitsandbytes 或双重量化，目的是观察低比特误差与适配容量的相互作用。

未量化权重、激活、适配器、梯度与 Adam 状态均为 FP32。两条训练臂都使用 AdamW，学习率 0.002、weight decay 为零、梯度范数裁剪阈值为 1，微批 4、60 步。保持集另有 8 条可人工核查的常识续写探针，只能提示局部回归，不能代表完整通用能力或遗忘评测。

表 12.2 质量指标：训练臂各用 480 个监督 token，Frozen 不更新；均为单次运行

方法	留出 EM	留出 response NLL	留出 answer NLL	保持 answer NLL
Frozen	0/32	4.4373	2.9075	1.7426
LoRA	23/32	1.0093	2.0148	1.3583
教学 INT4+LoRA	17/32	0.7699	1.5327	4.9626

表 12.2 的 EM 使用贪心生成，上限为 4 个新 token，去掉 EOS 并去除首尾空白后比较完整字母。三臂 EOS 前平均生成长度分别为 4、1、1。response NLL 同时计入答案和 EOS，answer NLL 只计答案；LoRA 与 INT4 臂的 EOS NLL 分别降到 0.0037、0.0071，故整体 NLL 很容易被“学会及时停止”改善。NLL 是 teacher-forcing 下的平均对数损失，EM 则是完整生成行为，两者不保证同序。

表 12.3 资源记录（MiB 为 2^{20} 字节；分配器统计不能相加）

方法	冻结参数及 buffers	训练时间（秒）	MPS tensor 采样峰值	进程 RSS 峰值
Frozen	513.2	0	574.6	602.3
LoRA	513.2	18.85	903.9	921.5
教学 INT4+LoRA	173.1	30.08	1171.9	1028.0

两条训练臂另有 230,400 个 FP32 适配器参数，其载荷为 921,600 字节，未混入表 12.3 的冻结存储列。MPS driver 采样峰值为 1157.1、1530.2、1626.2 MiB；它与 tensor 分配、RSS 可能重叠，不能相加，轮询也可能漏掉短暂峰值。冻结存储下降而 INT4 运行峰值更高，与该教学实现需要密集反量化临时空间的机制一致；这不是 CUDA QLoRA 的显存报价。训练时间只覆盖训练循环，不含下载、加载、初末评测与导出。

12.5 结果解读与可否定的结论

这组实验支持三个有限结论。第一，LoRA 的矩阵维度、合并和首步梯度可以不依赖大型模型而独立验证。第二，在固定合成任务、revision 和 480 个有效监督 token 下，两条训练臂都改变了 Frozen 的行为，变化的方向和局部保持代价不同。第三，DPO 的 reference-relative margin 对 beta、标签翻转和长度处理敏感，因而需要保存逐项概率，而不只保存平均 loss。

它们不支持以下更强断言：LoRA 在任意小模型上都优于全参数微调；教学 INT4 等价于 NF4

QLoRA；较低的 response NLL 必然带来更高正确率；或一次种子运行可以估计泛化置信区间。若要推进到产品决策，应增加真实标注集，分开规则与模板的留出因素，重复种子，加入冻结高精度与冻结低比特基线，并在目标能力、通用保持、输出长度和部署端到端成本之间预先设定门槛。

本章要点

证据链是：独立标量算例检查公式，受控 DPO 扰动检查目标，三臂实验与记录审计检查局部训练行为。保存条件和输入后可以重跑这些检查。

练习与提示

如果只看到教学 INT4+LoRA 的留出 NLL 低于 LoRA，下一步应先做什么？提示：固定数据与 token 预算，加入冻结 INT4 与冻结高精度基线，分开报告答案、EOS、生成长度和保持探针。若冻结 INT4 已经改变初始输出，就不能把后续变化全部归因于适配器。

第十三章　从复现走向研究：让下一次实验能够推翻假设

一本微调讲义的终点不应是记住更多方法缩写，而是能提出一个有限、可测量、可被反例推翻的问题。第十二章的微型任务若出现提升，只说明那个模型、数据、种子和预算下发生了变化；走向研究，需要设计竞争解释，让数据告诉我们收益来自哪里。

13.1　把观察变成有干预的机制问题

一个可执行问题包含观察、机制假设、干预和预测。例如，在同一监督 token 下，QLoRA 在长回答上的下降大于短回答；一个假设是早期 token 的小误差改变了后续状态，而非所有线性层的权重 MSE 同等重要。可比较固定 prefix 下逐 token 的分布偏离与自由生成轨迹的偏离，再对少数层恢复高精度。若固定 prefix 差异很小但自由生成差异大，支持的是误差累积解释；若两者都大，局部表征损失可能更重要。两种结果都需要额外实验，不能凭一条最终正确率曲线归因。

另一个问题是“更多 rank 是否有用”。先固定注入模块、数据和 token，扫描 rank，并记录参数量；这是容量曲线。随后固定总参数量，在更多模块低 rank 与更少模块高 rank 之间比较；这是预算分配曲线。rsLoRA 改变缩放，PiSSA 改变初始化与冻结分解，不能把二者混放在“初始化方法”一列。名义学习率相同不等于更新范数相同；同时记录初期梯度、$\|sBA\|$ 和有效函数变化。

13.2　每个对照只回答一个清楚的问题

表 13.1　可以区分竞争解释的实验设计

假设	受控干预	能推翻它的观测
收益来自更合适缩放	固定 rank/位置，比较两种缩放并等额调 LR	调好 LR 后收益消失，说明不能归因于缩放本身
收益来自数据质量	固定监督 token 与来源配比，改变过滤规则	增益只在重复模板，未见来源不变或变差
量化误差可低秩修复	同量化状态扫 rank 并测残差谱/任务	参数增加只降 MSE 不改善任务
长轨迹利于学生	同教师调用与学生 token 预算，比较轨迹长度	答案正确率不升且截断和成本增加
在线反馈优于筛选	固定生成/验证预算比较 RL 和拒绝采样 SFT	筛选 SFT 达到相同质量且成本更低

表中是研究方案，不是本书完成的实验结果。没有观察到收益时，不应自动把预算放大直到出现一个正结果；先检查测量是否有辨别力、干预是否足够、置信区间是否过宽，再决定继续或报告负结果。

13.3　研究资产与负结果同样值得保存

保存失败样本、OOM 配置、未能通过的合并验证和未核验的来源。一个正确复现的负结果比省略限制的排行榜分数更容易被复用。将运行记录分为“论文报告”“本次实测”“纯计算下界”“待运

行设计”；同一张图若混用这些对象，必须逐项注明。

性能结论必须带模型 revision、数据与 token 预算、软件/硬件、推理配置、搜索成本和日期。论文中的大模型规模、分布式并行或专门 kernel 不是实现细节，而是结果成立的条件。不要把可训练参数比例直接称为训练速度比例，也不要把文件大小直接称为峰值内存。

13.4 本书的边界与下一步

本书提供从监督对象到部署回归的方法链及微型教学实验；各方法的执行状态与可支持的结论见第十二章。微型数据避免复用公开评测题，但无法证明预训练语料从未包含类似任务；未见模板和置换划分是本次控制手段，不是预训练去污染保证。

下一步优先把真实需求中最常见的失败样本变成一份有来源和可验证标签的小验证集，固定基座与模板，复跑冻结和 LoRA 基线；确认 mask、budget 和导出无误，再引入低比特或偏好。若问题来自外部知识更新，可先比较检索；若来自低质量标签，先修数据。方法应由失败机制和部署约束选择。

练习与提示

写一页研究协议：一句观察、一句机制假设、一项干预、一个最小基线、一项主要指标和一条退出条件。将“提高性能”改写成给定分组测试与预算下可测量的变化；将“使用最新方法”改写成它改变的张量、目标或采样分布。最后列出一个会让你放弃当前解释的结果。

附录 A　符号、公式与单位的核查表

全书用 B 表示微批样本数，L 表示序列长度，N 表示层数；LoRA 的矩阵 B_A 或上下文中明确的 B 是输出侧因子，其语义不等于批量。不同章在局部简写时均说明维度。列向量线性层采用 $y = Wx$，行批输入采用 $Y = XW^\top$；切换时必须连梯度一起转置。

表 A.1　用于查错的公式与前提

对象	定义或核查点
因果预测	位置 t 的 logits 预测下一个 token；shift 只执行一次，真实上下文可参与 attention 但不必计 loss。
监督 token 平均	$\mathcal{L} = -\sum_{b,t} m_{b,t} \log p(z_{b,t} \mid z_{b,<t}) / \sum_{b,t} m_{b,t}$；全零分母拒绝该 batch。
线性层梯度	对 $Y = XW^\top$，$\nabla_W \mathcal{L} = G^\top X$，$G = \nabla_Y \mathcal{L}$。
LoRA 更新	$W = W_0 + sBA$；$W_0 \in \mathbb{R}^{d_o \times d_i}$，$A \in \mathbb{R}^{r \times d_i}$，$B \in \mathbb{R}^{d_o \times r}$。
LoRA 参数	$r(d_i + d_o)$；标准 $s = \alpha/r$，rsLoRA $s = \alpha/\sqrt{r}$。rank 改变时记完整组合。
仿射量化	$q = \mathrm{clip}(\mathrm{round}(w/s) + z, q_{min}, q_{max})$，$\hat{w} = s(q - z)$；$z$ 为整数零点。NF4 不是此均匀格点。
常驻训练状态	基座存储与元数据 + adapter 权重 + 梯度 + optimizer 状态 + 可选高精度主副本；激活/缓冲/分配器另计。
DPO	$-\log \sigma(\beta u)$，$u = (\log \pi_\theta^+ - \log \pi_\theta^-) - (\log \pi_0^+ - \log \pi_0^-)$。回答概率按 token 求和。
PPO 比率	$\rho_t = \pi_\theta(y_t \mid s_t) / \pi_{old}(y_t \mid s_t)$，old 不是固定 reference 的同义词。
组相对优势	$(R_i - \bar{R})/(\sigma_R + \varepsilon)$；组标准差定义和零方差处理是算法配置的一部分。
温度 KD	$\tau^2 D_{\mathrm{KL}}(p_T^{(\tau)} \| p_S^{(\tau)})$；逐 tokenKL 需对齐词表与前缀，$\tau^2$ 缩放有指定参数化前提。
KV 存储下界	$2N n_{kv} d_h s_{kv} \sum_b L_b$ 字节；不含分配、对齐、分页和其他缓存。

GB 按 10^9 字节，GiB 按 2^{30} 字节，MiB 按 2^{20} 字节。FP32、BF16 和 FP16 的单元素存储分别是 4、2、2 字节；“4bit”权重还要统计 scale、zero point/码本、未量化层及格式头。峰值 RSS、统一内存与 CUDA 显存是不同资源指标，不能写入同一列而不标明。

附录 B　十二周阅读与实验路线

这是一条可以压缩或拉长的教学路线，不要求具备所有硬件才能开始。每一步以一个可以检查的产物结束；某项未运行就标为方案，不用预计结果填表。

表 B.1　从概念到独立适配项目

周	阅读主题	核查产物
1	任务与基座行为	失败分类、冻结基线、版本和需求门槛
2	张量与梯度	手算 attention 形状和 shift/CE, 用小例子验证 mask
3	模板与数据	来源簇划分、污染范围说明、监督段解码
4	SFT 工程	单 batch 拟合、token 加权累积、完整训练日志
5	PEFT 与 LoRA	矩阵算例、首步梯度、合并一致性
6	rank 与模块	容量扫描和等参数位置对照分别报告
7	量化账本	理论字节与实测峰值对照，不能运行 NF4 则留方案
8	适配与遗忘	未见来源、通用保持、长度分桶和失败样本
9	偏好	手工 DPO 损失、reference 缓存、标签翻转扰动
10	在线反馈	verifier 反例；预算足够时再运行短 RL
11	蒸馏/持续适配	教师访问权限、tokenizer 限制、查询预算与路由测试
12	导出与研究	可回滚产物、部署回归、一页可证伪研究协议

附录 C　复现与交付检查清单

1. 基座、tokenizer、adapter、模板均有不可变版本或哈希；模型许可和数据来源已记录。
2. 训练、验证、测试按适当来源/模板/实体组隔离；去重和污染检查的覆盖边界明确。
3. 前向、loss 与 shift 索引一致；prompt、EOS、padding、工具和多轮 assistant 的监督规则可解码核查。
4. 有效输入 token、监督 token、微批、累积、设备数、更新数与归约方式一致；packing 是否跨样本 attention 明确。
5. 学习率、rank、alpha、缩放、初始化、目标模块、dtype 和量化状态全部记录。
6. 冻结基线、适配、量化冻结及导出后状态按同一评测提示测试；未运行项不冒充基线。
7. 任务、保持、未见输入、长度、评判偏置和资源分别记录；样本数、种子与区间的含义明确。
8. 偏好 reference 缓存可追踪；在线 RL 的 old/reference、奖励分项、KL 位置及采样预算可追踪。
9. 导出包含额外解冻参数；merged/unmerged 及量化前后经过数值和行为回归。
10. 逐样本预测、失败运行、实验命令、环境版本、硬件和墙钟保留；读者可以区分实测、计算与计划。

本书独立构建

在本主题目录运行：

```
bash manuscript/build.sh
```

需要 TeX 发行版中的 LuaLaTeX、Biber、latexmk 以及本项目使用的常规宏包。字体、Logo 和许可证均在 `manuscript/` 内，以相对路径加载，不需要修改公共模板。完整 PDF 在主题 `output/pdf/`，构建缓存位于 `output/build/modern_small_model_finetuning-jingfeng/`，页面渲染与排版检查保存在 `output/qa/`。实验依赖和命令见 `manuscript/experiments/README.md`；重新训练与重新排版是两个独立步骤。

参考文献与原始资料

[1] A. Vaswani et al. *Attention Is All You Need*. June 12, 2017. arXiv: 1706.03762. URL: https://arxiv.org/abs/1706.03762v7 (visited on 10/21/2026).

[2] H. Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. Feb. 27, 2023. arXiv: 2302.13971. URL: https://arxiv.org/abs/2302.13971v1 (visited on 10/21/2026).

[3] A. Q. Jiang et al. *Mistral 7B*. Oct. 10, 2023. arXiv: 2310.06825. URL: https://arxiv.org/abs/2310.06825v1 (visited on 10/21/2026).

[4] Qwen et al. *Qwen2.5 Technical Report*. Dec. 19, 2024. arXiv: 2412.15115. URL: https://arxiv.org/abs/2412.15115v2 (visited on 10/21/2026).

[5] E. J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. June 17, 2021. arXiv: 2106.09685. URL: https://arxiv.org/abs/2106.09685v2 (visited on 10/21/2026).

[6] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. *QLoRA: Efficient Finetuning of Quantized LLMs*. May 23, 2023. arXiv: 2305.14314. URL: https://arxiv.org/abs/2305.14314v1 (visited on 10/21/2026).

[7] L. B. Allal et al. *SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model*. Feb. 4, 2025. arXiv: 2502.02737. URL: https://arxiv.org/abs/2502.02737v1 (visited on 10/21/2026).

[8] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. May 27, 2022. arXiv: 2205.14135. URL: https://arxiv.org/abs/2205.14135v2 (visited on 10/21/2026).

[9] Hugging Face. *Chat templates*. Rolling documentation; snapshot checked 2026-10-21. URL: https://huggingface.co/docs/transformers/chat_templating (visited on 10/21/2026).

[10] PyTorch Contributors. *CrossEntropyLoss*. Official v2.8.0 source and docstring; ignore-index behavior checked. URL: https://github.com/pytorch/pytorch/blob/v2.8.0/torch/nn/modules/loss.py (visited on 10/22/2026).

[11] Y. Wang et al. *Self-Instruct: Aligning Language Models with Self-Generated Instructions*. Dec. 20, 2022. arXiv: 2212.10560. URL: https://arxiv.org/abs/2212.10560v2 (visited on 10/21/2026).

[12] Z. Xu et al. *Magpie: Alignment Data Synthesis from Scratch by Prompting Aligned LLMs with Nothing*. June 12, 2024. arXiv: 2406.08464. URL: https://arxiv.org/abs/2406.08464v2 (visited on 10/21/2026).

[13] C. Zhou et al. *LIMA: Less Is More for Alignment*. May 18, 2023. arXiv: 2305.11206. URL: https://arxiv.org/abs/2305.11206v1 (visited on 10/21/2026).

[14] H. W. Chung et al. *Scaling Instruction-Finetuned Language Models*. Oct. 20, 2022. arXiv: 2210.11416. URL: https://arxiv.org/abs/2210.11416v5 (visited on 10/21/2026).

[15] N. Lambert et al. *Tulu 3: Pushing Frontiers in Open Language Model Post-Training*. Nov. 22, 2024. arXiv: 2411.15124. URL: https://arxiv.org/abs/2411.15124v5 (visited on 10/21/2026).

[16] Hugging Face. *Padding and truncation*. Official documentation source, Transformers v4.56.2. URL: https://github.com/huggingface/transformers/blob/v4.56.2/docs/source/en/pad_truncation.md (visited on 10/22/2026).

[17] N. Jain et al. *NEFTune: Noisy Embeddings Improve Instruction Finetuning*. Oct. 9, 2023. arXiv: 2310.05914. URL: https://arxiv.org/abs/2310.05914v2 (visited on 10/21/2026).

[18] S. Gururangan et al. *Don't Stop Pretraining: Adapt Language Models to Domains and Tasks*. Apr. 23, 2020. arXiv: 2004.10964. URL: https://arxiv.org/abs/2004.10964v3 (visited on 10/21/2026).

[19] N. Houlsby et al. *Parameter-Efficient Transfer Learning for NLP*. Feb. 2, 2019. arXiv: 1902.00751. URL: https://arxiv.org/abs/1902.00751v2 (visited on 10/21/2026).

[20] X. L. Li and P. Liang. *Prefix-Tuning: Optimizing Continuous Prompts for Generation*. Jan. 1, 2021. arXiv: 2101.00190. URL: https://arxiv.org/abs/2101.00190v1 (visited on 10/21/2026).

[21] B. Lester, R. Al-Rfou, and N. Constant. *The Power of Scale for Parameter-Efficient Prompt Tuning*. Apr. 18, 2021. arXiv: 2104.08691. URL: https://arxiv.org/abs/2104.08691v2 (visited on 10/21/2026).

[22] E. Ben-Zaken, S. Ravfogel, and Y. Goldberg. *BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models*. June 18, 2021. arXiv: 2106.10199. URL: https://arxiv.org/abs/2106.10199v5 (visited on 10/21/2026).

[23] D. Kalajdzievski. *A Rank Stabilization Scaling Factor for Fine-Tuning with LoRA*. Nov. 28, 2023. arXiv: 2312.03732. URL: https://arxiv.org/abs/2312.03732v1 (visited on 10/21/2026).

[24] S.-Y. Liu et al. *DoRA: Weight-Decomposed Low-Rank Adaptation*. Feb. 14, 2024. arXiv: 2402.09353. URL: https://arxiv.org/abs/2402.09353v6 (visited on 10/21/2026).

[25] F. Meng, Z. Wang, and M. Zhang. *PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models*. Apr. 3, 2024. arXiv: 2404.02948. URL: https://arxiv.org/abs/2404.02948v4 (visited on 10/21/2026).

[26] Y. Li et al. *LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models*. Oct. 12, 2023. arXiv: 2310.08659. URL: https://arxiv.org/abs/2310.08659v4 (visited on 10/21/2026).

[27] Q. Zhang et al. *AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning*. Mar. 18, 2023. arXiv: 2303.10512. URL: https://arxiv.org/abs/2303.10512v2 (visited on 10/21/2026).

[28] L. Zhang, L. Zhang, S. Shi, X. Chu, and B. Li. *LoRA-FA: Efficient and Effective Low Rank Representation Fine-tuning*. Aug. 7, 2023. arXiv: 2308.03303. URL: https://arxiv.org/abs/2308.03303v3 (visited on 10/21/2026).

[29] D. J. Kopiczko, T. Blankevoort, and Y. M. Asano. *VeRA: Vector-based Random Matrix Adaptation*. Oct. 17, 2023. arXiv: 2310.11454. URL: https://arxiv.org/abs/2310.11454v2 (visited on 10/21/2026).

[30] S. Hayou, N. Ghosh, and B. Yu. *LoRA+: Efficient Low Rank Adaptation of Large Models*. Feb. 19, 2024. arXiv: 2402.12354. URL: https://arxiv.org/abs/2402.12354v2 (visited on 10/21/2026).

[31] T. Jiang et al. *MoRA: High-Rank Updating for Parameter-Efficient Fine-Tuning*. May 20, 2024. arXiv: 2405.12130. URL: https://arxiv.org/abs/2405.12130v1 (visited on 10/21/2026).

[32] K. Bałazy, M. Banaei, K. Aberer, and J. Tabor. *LoRA-XS: Low-Rank Adaptation with Extremely Small Number of Parameters*. May 27, 2024. arXiv: 2405.17604. URL: https://arxiv.org/abs/2405.17604v3 (visited on 10/21/2026).

[33] L. Ouyang et al. *Training language models to follow instructions with human feedback*. Mar. 4, 2022. arXiv: 2203.02155. URL: https://arxiv.org/abs/2203.02155v1 (visited on 10/21/2026).

[34] Y. Bai et al. *Constitutional AI: Harmlessness from AI Feedback*. Dec. 15, 2022. arXiv: 2212.08073. URL: https://arxiv.org/abs/2212.08073v1 (visited on 10/21/2026).

[35] R. Rafailov et al. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. May 29, 2023. arXiv: 2305.18290. URL: https://arxiv.org/abs/2305.18290v3 (visited on 10/21/2026).

[36] Y. Meng, M. Xia, and D. Chen. *SimPO: Simple Preference Optimization with a Reference-Free Reward*. May 23, 2024. arXiv: 2405.14734. URL: https://arxiv.org/abs/2405.14734v3 (visited on 10/21/2026).

[37] M. G. Azar et al. *A General Theoretical Paradigm to Understand Learning from Human Preferences*. Oct. 18, 2023. arXiv: 2310.12036. URL: https://arxiv.org/abs/2310.12036v2 (visited on 10/21/2026).

[38] W. Xiao et al. *A Comprehensive Survey of Direct Preference Optimization: Datasets, Theories, Variants, and Applications*. Oct. 21, 2024. arXiv: 2410.15595. URL: https://arxiv.org/abs/2410.15595v4 (visited on 10/21/2026).

[39] K. Ethayarajh, W. Xu, N. Muennighoff, D. Jurafsky, and D. Kiela. *KTO: Model Alignment as Prospect Theoretic Optimization*. Feb. 2, 2024. arXiv: 2402.01306. URL: https://arxiv.org/abs/2402.01306v5 (visited on 10/21/2026).

[40] J. Hong, N. Lee, and J. Thorne. *ORPO: Monolithic Preference Optimization without Reference Model*. Mar. 12, 2024. arXiv: 2403.07691. URL: https://arxiv.org/abs/2403.07691v2 (visited on 10/21/2026).

[41] Z. Yuan et al. *RRHF: Rank Responses to Align Language Models with Human Feedback without tears*. Apr. 11, 2023. arXiv: 2304.05302. URL: https://arxiv.org/abs/2304.05302v3 (visited on 10/21/2026).

[42] Y. Zhao et al. *SLiC-HF: Sequence Likelihood Calibration with Human Feedback*. May 17, 2023. arXiv: 2305.10425. URL: https://arxiv.org/abs/2305.10425v1 (visited on 10/21/2026).

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. *Proximal Policy Optimization Algorithms*. July 20, 2017. arXiv: 1707.06347. URL: https://arxiv.org/abs/1707.06347v2 (visited on 10/21/2026).

[44] Z. Shao et al. *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*. Feb. 5, 2024. arXiv: 2402.03300. URL: https://arxiv.org/abs/2402.03300v3 (visited on 10/21/2026).

[45] DeepSeek-AI et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. Jan. 22, 2025. arXiv: 2501.12948. URL: https://arxiv.org/abs/2501.12948v2 (visited on 10/21/2026).

[46] Q. Yu et al. *DAPO: An Open-Source LLM Reinforcement Learning System at Scale*. Mar. 18, 2025. arXiv: 2503.14476. URL: https://arxiv.org/abs/2503.14476v2 (visited on 10/21/2026).

[47] Z. Liu et al. *Understanding R1-Zero-Like Training: A Critical Perspective*. Mar. 26, 2025. arXiv: 2503.20783. URL: https://arxiv.org/abs/2503.20783v2 (visited on 10/21/2026).

[48] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. Oct. 2, 2019. arXiv: 1910.01108. URL: https://arxiv.org/abs/1910.01108v4 (visited on 10/21/2026).

[49] W. Wang et al. *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*. Feb. 25, 2020. arXiv: 2002.10957. URL: https://arxiv.org/abs/2002.10957v2 (visited on 10/21/2026).

[50] R. Agarwal et al. *On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes*. June 23, 2023. arXiv: 2306.13649. URL: https://arxiv.org/abs/2306.13649v3 (visited on 10/21/2026).

[51] D. Hendrycks et al. *Measuring Massive Multitask Language Understanding*. Sept. 7, 2020. arXiv: 2009.03300. URL: https://arxiv.org/abs/2009.03300v3 (visited on 10/21/2026).

[52] K. Cobbe et al. *Training Verifiers to Solve Math Word Problems*. Oct. 27, 2021. arXiv: 2110.14168. URL: https://arxiv.org/abs/2110.14168v2 (visited on 10/21/2026).

[53] J. Zhou et al. *Instruction-Following Evaluation for Large Language Models*. Nov. 14, 2023. arXiv: 2311.07911. URL: https://arxiv.org/abs/2311.07911v1 (visited on 10/21/2026).

[54] L. Zheng et al. *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. June 9, 2023. arXiv: 2306.05685. URL: https://arxiv.org/abs/2306.05685v4 (visited on 10/21/2026).

[55] Hugging Face. *Bitsandbytes quantization*. Rolling documentation; snapshot checked 2026-10-21. URL: https://huggingface.co/docs/transformers/quantization/bitsandbytes (visited on 10/21/2026).