Jingfeng

现代小模型训练

从数据与缩放规律到蒸馏和可部署模型

Jingfeng · 2026 年 10 月 23 日修订



在预算约束下，获得一个眞正可用的模型。

从数据和目标函数出发，分清训练、蒸馏、适配与压缩改变了什么，再用可复现的评测检验它们是否满足部署约束。

本书的三条证据线

公式推导给出假设和单位；原始论文说明方法与适用条件；本次本地实验证明一段可执行流程能跑通。三者分别标明，不把教学算例当作硬件测量，也不把分类玩具实验推广为语言模型能力结论。

原稿：2026 年 9 月　本版核验截至：2026 年 10 月 23 日

前言：从预算开始提问

小模型没有固定参数区间。对离线文档抽取器，几百毫秒内稳定输出可验证字段可能比开放问答重要；对手机应用，模型能否与系统、相机和长上下文共存，也比参数排名更有意义。本书追问：预算有限时，如何获得满足质量与可靠性要求的模型？

回答要连接三个层面：数据和监督提供什么信号，参数与量化怎样改变模型，运行时能否兑现节省。可训练参数、部署存储和实际速度因此要分别检查。

本版逐章修订原有十二章，补齐缩放假设、数据闭环、蒸馏方向、显存账本与可证伪实验。原版源码和 PDF 保留归档，新版使用独立入口。

阅读路线与前置知识

读者需熟悉矩阵乘法、导数、概率、交叉熵与基本 Python。先读第 1–4 章，理解三条路线及预算比较；领域适配接第 5、8、10、11 章，部署接第 6、7、9、10 章。第 12 章讨论研究设计，附录提供公式、实测与阅读清单。

本书解释 LoRA、QLoRA 和偏好优化的必要原理，具体适配流程可继续阅读同系列《现代小模型微调》。量化算子与打包格式、GPU 内核、线上服务系统分别由《智能体模型量化压缩与部署》《Kernel 与 CUDA 编程》《大模型推理加速与 AI 基础设施》展开。本书不依赖这些分册的具体页码，独立给出理解后文所需的定义。

怎样判断这里的证据

公式中的算例均为明确假设下的手算；论文引用只支撑对应的方法或报告，不能替代本机复现；附录 B 中只有带配置和原始日志的数值才是本次实测。该实验使用合成分类数据和小型网络，用于检查蒸馏与压缩的实验流程，不代表大语言模型上的能力增长。修订核验截至 2026 年 10 月 23 日；该日期是本版核验边界，不表示穷尽此前所有工作。

先约定几个基础概念

token（词元）是 tokenizer（分词器）把文本编码后交给模型的离散单位，可能是字节、字、词的一部分或特殊符号，不能一概换算成汉字数。模型输出的 logits 是尚未归一化的实数分数；softmax 将这些分数转成和为 1 的概率。交叉熵衡量模型给目标事件分配的负对数概率，后文以自然对数计。

BF16 和 FP16 是两种 16 位浮点格式，每元素通常占 2 字节；BF16 的指数范围较宽、尾数较短，两者不能仅因位数相同而互换。FP32 是 32 位浮点，通常占 4 字节。INT8 是 8 位整数；量化还需要比例因子等元数据，不能只按一字节估计全部存储。浮点乘法、累加、权重存储与优化器状态也可以使用不同精度。

预训练学习一般性的序列分布；监督微调（supervised fine-tuning, SFT）用任务示范塑造回答；知识蒸馏（knowledge distillation，KD）利用教师信号；参数高效微调（parameter-efficient fine-tuning, PEFT）限制更新的参数集合。KV cache 是自回归生成时保存各层历史 key 和 value 的缓存，与训练反向传播保存的激活分开记账。这些术语将在对应章节展开。

符号、单位与约定

符号	本书约定
N, N_{train}	总参数量与可训练参数量；冻结参数仍占存储和前向计算。
D, C	实际训练 token 次数与训练浮点运算量；重复样本重复计入 D。
L, T, b	层数、上下文 token 长度、批内序列数；动态批可逐请求求和。
d, d_h, H_{kv}	隐藏宽度、每头宽度、每层 KV 头数。
s_W, s_G, s_O	每个权重、梯度、优化器状态元素的字节数。
τ	蒸馏温度；不会与序列长度 T 混用。
p_T, p_S	教师、学生的概率分布；下标 T 在此为 teacher 的缩写。
$\mathrm{KL}(p\|q)$	正向比较的方向由参数顺序决定：$\sum_i p_i \log(p_i/q_i)$。
FLOP / FLOPs	浮点运算 / 浮点运算次数；FLOP/s 是速率。一次乘加按 2 FLOPs 计。
GB / GiB	10^9 字节 / 2^{30} 字节；B 表示十亿参数时另加“参数”说明。
TTFT / TPOT	首 token 延迟 / 后续输出 token 的平均间隔；均注明是否含排队。

未特别说明时，对数为自然对数，损失的单位为 nat；分类率写百分比，准确率之差写百分点。模型尺寸、磁盘文件、进程 RSS 和设备已分配内存分别报告，不能互相代称。

目录

前言：从预算开始提问 **i**

符号、单位与约定 **ii**

第 1 章 预算、能力与小模型路线选择 **1**
1.1 先定义工作负载，再定义“小” . . . 1
1.2 可行域与 Pareto 前沿 . . . 2
1.3 能力从哪里来：三条可组合的路线 . . . 2
1.4 一个可以手算的路线决策 . . . 3
1.5 把一次训练成本扩展为全生命周期成本 . . . 4
1.6 建立一个能被推翻的基线 . . . 4
1.7 练习与解题提示 . . . 5

第 2 章 语言建模、缩放规律与训练显存 **6**
2.1 下一 token 预测究竟预测什么 . . . 6
2.2 从参数和 token 推到训练计算 . . . 7
2.3 计算最优比例是怎样推出来的 . . . 8
2.4 为什么计算最优不等于部署最优 . . . 9
2.5 训练显存：从状态而不是口号开始 . . . 9
2.6 让预算估计接受实测修正 . . . 10
2.7 练习与解题提示 . . . 10

第 3 章 数据、tokenizer 与可控的学习信号 **11**
3.1 从来源清单开始，而不是从大文件开始 . . . 11
3.2 去重与划分：保持独立性比减少文件数更重要 . . . 12
3.3 污染检查不能只搜索题目全文 . . . 13
3.4 质量不是一个可以直接相乘的系数 . . . 13
3.5 tokenizer 是数据与模型之间的成本接口 . . . 14
3.6 配比决定每个领域实际得到多少梯度 . . . 14
3.7 怎样区分质量收益与更多计算的收益 . . . 15
3.8 教材式合成与继续预训练的收益边界 . . . 15
3.9 练习与解题提示 . . . 16

第 4 章 知识蒸馏：把教师能力变成学生能力 **17**
4.1 软标签多提供了什么 . . . 17
4.2 把词元监督写成可实现的目标 . . . 18
4.3 特征、关系与不同 tokenizer 的限制 . . . 19
4.4 序列级蒸馏与前缀分布 . . . 20
4.5 从答案到可验证的过程蒸馏 . . . 21
4.6 怎样证明蒸馏值得付出成本 . . . 21
4.7 练习与提示 . . . 22

第 5 章 先决定更新什么：参数高效与低显存训练 **23**
5.1 四种节省，对应四本账 . . . 23
5.2 LoRA 的低秩增量限制了什么 . . . 24

5.3 QLoRA、LoftQ 与部署量化的关系 . . . 25
5.4 前缀、梯度投影与状态分片 . . . 26
5.5 按瓶颈选择，并留下反例 . . . 26

第 6 章 压缩改变什么：量化、剪枝与低秩分解 27
6.1 从一个量化器开始 . . . 27
6.2 PTQ 与 QAT：何时引入量化误差 . . . 27
6.3 低秩补偿：应该保留哪些误差方向 . . . 29
6.4 剪枝：零值不等于结构消失 . . . 30
6.5 低秩分解与层裁剪 . . . 30
6.6 把压缩顺序当作实验变量 . . . 31

第 7 章 架构与系统：把复杂度还原成成本 32
7.1 注意力的计算与存储不是同一件事 . . . 32
7.2 SSM 与混合架构改变了状态的形状 . . . 32
7.3 KV cache 的完整账本 . . . 33
7.4 预填充与解码为何有不同瓶颈 . . . 33
7.5 怎样选择架构或运行时 . . . 34

第 8 章 从语言建模到可靠行为 35
8.1 继续预训练、SFT 与偏好不是同一目标 . . . 35
8.2 DPO：从偏好对优化概率比 . . . 35
8.3 推理过程监督与测试时计算 . . . 36
8.4 停止、拒答与工具边界 . . . 36

第 9 章 小型多模态与端侧工作负载 37
9.1 编码器、桥接器与语言骨干 . . . 37
9.2 视觉 token 为什么会扩大文本成本 . . . 37
9.3 端侧训练必须考虑更新后的生命周期 . . . 38
9.4 热、能耗与异构硬件 . . . 38

第 10 章 评测：把分数变成可复现证据 39
10.1 四层评测框架 . . . 39
10.2 先逐任务报告，再考虑汇总 . . . 39
10.3 校准、失败样例和统计不确定性 . . . 40
10.4 遗忘与鲁棒性需要各自的对照 . . . 40
10.5 真实延迟的测量协议 . . . 41
10.6 练习与提示 . . . 41

第 11 章 统一训练配方：从预算到可部署模型 42
11.1 先写约束，再选算法 . . . 42
11.2 一条可执行的基线流水线 . . . 42
11.3 把一次运行写成可追踪记录 . . . 43
11.4 按失败位置回退 . . . 43
11.5 训练与压缩的联合目标 . . . 44
11.6 消融要回答一个有限的问题 . . . 44
11.7 练习与提示 . . . 44

第 12 章 从研究问题到可证伪实验 45
12.1 一条弹性的十二周学习路线 . . . 45
12.2 先定义失败，再投入计算 . . . 45
12.3 让失败成为下一次实验的输入 . . . 47
12.4 练习与提示 . . . 47

附录 A 公式速查表 **48**

附录 B 最小实验与复现方案 **50**
B.1 目录、命令与复核边界 50
B.2 固定预算下的蒸馏对照 50
B.3 压缩文件与部署代价的实测区别 51
B.4 独立数值参考与边界核验 52
B.5 生成模型与端侧的未运行方案 52

附录 C 术语表与阅读清单 **53**
C.1 术语表 53
C.2 常见误区 54
C.3 主干阅读顺序 54

参考文献 **55**

第 1 章

预算、能力与小模型路线选择

一台设备能载入模型，与它能可靠地完成工作，是两件需要分别证明的事。假设要在离线电脑上抽取维修记录中的设备编号、故障和处理措施。一个模型也许只占几百兆字节，却频繁把缺失字段补成看似合理的内容；另一个模型准确率较高，却在长记录上超过可用内存。把前者称为“小模型成功”，或仅凭参数量淘汰后者，都没有回答实际问题。

本书把小模型训练看作一项预算约束下的设计任务：先定义要完成的工作、可以接受的错误和资源上限，再决定从哪里获得模型能力。读者需要的前置知识是概率分布、向量和矩阵的基本运算，以及使用程序划分训练集与测试集的经验。本章先建立全书共用的账本；下一章再说明这些账本为什么会随目标函数、模型规模和序列长度变化。

1.1 先定义工作负载，再定义“小”

参数规模通常记为 N，例如 $N = 10^9$ 表示十亿个参数。这里的 N 是模型的**总参数量**。另外两个常用数字必须单独写：N_{train} 是一次训练中允许更新的参数量，N_{active} 是一次前向计算实际使用的参数量。稠密模型通常使用几乎全部主干参数；稀疏专家模型的活跃参数可能少于总参数；参数高效微调则可能只更新很少的参数，却仍需载入整个基座。一个仅训练一千万参数的适配器，不能因此被当作一千万参数的独立模型。

权重字节数的第一近似是 Ns_W，其中 s_W 为每个权重的存储字节数。例如十亿参数的 BF16 权重约占 2×10^9 字节，即 2 GB，约 1.86 GiB。GB 使用十进制，GiB 使用二进制，本书不混用二者。嵌入与输出投影的权重已计入 N；分组量化的比例因子、tokenizer 词表文件和配置等元数据，以及运行时、激活和生成缓存，还需另计。反过来，磁盘上的压缩文件也可能在载入时展开。因此“文件小于内存”只是必要性检查的一部分。

工作负载 s 至少包含输入分布、任务协议、设备和服务方式。输入分布决定短句、长文、罕见符号和目标语言的比例；任务协议规定输出格式、是否允许拒答以及能否调用检索；设备规定可用内存与算子；服务方式规定批大小、并发和等待时间。相同权重在单请求分类与多用户长文本生成中，表现为不同的资源对象。

把形容词改写成可检验的规格

“要小、快、准确”可以改写为：在指定设备和运行时、单请求、输入不超过给定 token 数时，峰值进程内存不超过预算，响应延迟的第 95 百分位不超过上限；在锁定的测试集上，字段级 F1、无依据字段生成率和拒答率满足要求。规格中的每一个数字都应来自用户需求或测量，不能由算法名称代替。

“小”因而是一个相对工作负载的描述。小参数模型可能拥有很大的词表；短上下文可用的模型可能在长上下文中被 KV cache 挤出内存；窄领域学生可能在目标格式上可靠，却不具备通用助手能

力。这些边界应写入目标工作负载的设计规格。

1.2 可行域与 Pareto 前沿

设一个候选方案为 a。它不仅包含权重，还包含数据、训练过程、精度配置、运行时和解码策略。在工作负载 s 上记录质量向量 $\mathbf{q}(a;s)$、资源向量 $\mathbf{c}(a;s)$ 和风险向量 $\mathbf{r}(a;s)$。质量可以包括准确率与覆盖率，资源可以包括峰值内存、延迟与能耗，风险可以包括事实错误、隐私泄漏与越权行为。它们的单位不同，通常不能直接相加。

先用硬约束定义可行域：

$$\begin{aligned}\mathcal{F}_s = \{a :\ & q_k(a;s) \geq q_k^{\min} \text{ 对所有必要任务 } k,\\ & c_j(a;s) \leq B_j, \quad r_m(a;s) \leq R_m,\\ & \text{数据、模型与部署许可满足本项目用途}\}.\end{aligned} \tag{1.1}$$

如果一个候选的平均准确率很高，但不能在设备上运行，它不在可行域内；如果它满足内存预算，却在关键少数类别上失败，也可能不在可行域内。约束应尽量使用置信区间或留出余量，而不只用一次测量的点估计。小样本测试中“没有观察到失败”也不等于失败概率为零。

对两个可行候选 a 和 b，如果 a 在所有指定质量指标上不差于 b，在所有成本和风险指标上不高于 b，并且至少有一项严格更好，就说 a **支配** b。没有被其他已知候选支配的集合是已测方案的 Pareto 前沿。它不是宇宙中所有模型的全局最优证明，也不是把一个综合分数排到第一的别名。

考虑三个纯教学候选：甲的任务得分为 0.90、内存为 1.2 GB；乙为 0.88、1.5 GB；丙为 0.92、2.0 GB。若其他指标相同，甲支配乙，甲与丙则各有优势。若新发现甲的尾延迟明显较差，它们的支配关系就要重新计算。必须把参与比较的指标和工作负载写在表头，不能事后挑有利维度。

项目最终仍需选一个方案。可在硬约束之后使用货币成本、优先级或加权效用排序，但权重应来自应用取舍。例如错误一次需要人工复核多少钱，可以帮助换算质量的经济意义；涉及安全底线时，用一项平均收益抵消严重失败则可能不合适。Pareto 分析的作用，是让这些取舍显露出来。

1.3 能力从哪里来：三条可组合的路线

1.3.1 从零预训练：控制底座，也承担底座成本

从零预训练从随机初始化开始，通过语言建模等目标学习表示。它允许控制词表、架构、语料和训练历史，适合已有模型的语言覆盖、许可或设备结构确实不适用的情形。代价是必须支付建立通用表示的成本。只有少量领域文档时，从零训练通常不能仅靠“更贴近领域”弥补数据与计算不足。

TinyLlama 提供了小规模模型长时间预训练的一个公开实例 [1]。它的意义是具体训练配方与工程经验，而不是证明任何项目都应从零开始。复制参数规模而没有复制数据覆盖、训练 token、优化和评测，不是在复现同一条路线。

1.3.2 教师迁移：购买监督，形成较小的学生

蒸馏利用教师的概率分布、生成序列或中间表示训练学生。学生既可以从随机初始化开始，也可以从已有基座开始。教师生成可解释的答案、程序或过程轨迹，有时能把昂贵的在线推理转为离线训练资料，但教师调用、验证和筛选都需要预算。模型学到的是有限数据与目标下的行为近似，教师更强并不保证学生就能吸收更多能力。

“教师迁移”不等于“参数压缩”。一个有十亿参数的学生可能原本就有十亿参数，蒸馏改善的是它的监督信号。反之，先剪枝得到小骨干，再用教师恢复，也属于蒸馏与结构压缩的组合。区分

能力来源与表示压缩，才能解释结果。

1.3.3 已有模型适配与压缩：复用能力，再调整约束

已有模型适配包括继续预训练、监督微调和参数高效更新。它首先复用基座已经获得的语言与知识表示，再改变领域分布或输出行为。压缩包括量化、剪枝、低秩分解和层裁剪，主要改变存储或计算结构。适配与压缩可以前后组合，但它们解决不同问题：减少 N_{train} 不一定减少 N，减少权重字节不一定减少延迟，减少算术量也不一定使目标硬件更快。

表 1.1 路线选择先回答的三个问题，而非固定的优劣排名。

路线	优先核实的条件	早期可以否决它的证据
从零预训练	数据权利和覆盖、训练计算、词表或架构的必要性	目标分布的数据不足，或小规模预实验无法稳定下降
教师迁移	教师访问条件、输出权利、可验证性、学生容量	教师在目标任务上不可靠，或筛选后的成本超过预算
已有模型适配与压缩	基座许可、零样本能力、目标硬件支持	必要能力缺失，或质量与资源约束无法同时满足

这些路线没有严格互斥的边界。可先选择小基座，继续预训练后蒸馏任务轨迹，再做量化。项目应说明每一步预计改变哪个量、用什么对照确认，而不能把所有处理捆在一起后，把收益归给最后一个方法。

1.4 一个可以手算的路线决策

以下数字完全是**教学假设**，用于演示如何排除候选，不是本次实验或任何硬件的性能报告。某离线抽取服务运行在 4 GB 设备上，其中给模型进程的峰值预算为 2.5 GB。输入上限已固定，粗估非权重内存为 0.8 GB；学生训练计算代理预算为 10^{18} FLOPs；教师接口预算为 1000 元。三个候选暂用 BF16 权重：

表 1.2 教学候选：D 表示学生训练实际读入的累计 token，允许包含混合和重放。

候选	从零预训练甲	蒸馏学生乙	适配基座丙
总参数 N	0.30×10^9	0.60×10^9	1.00×10^9
学生 token D	8×10^9	2×10^8	2×10^7
权重估算 $2N$	0.60 GB	1.20 GB	2.00 GB
含假设开销的内存	1.40 GB	2.00 GB	2.80 GB
稠密训练代理 $6ND$	1.44×10^{19}	7.2×10^{17}	1.2×10^{17}

甲在这个训练代理预算下超额约 14.4 倍；丙虽然训练代理便宜，却在当前精度与假设运行时下超过内存预算；乙通过这两项粗筛。这里不能把乙写成“获胜”：它的质量尚未知，教师也还没有记账。表中为丙计算的是稠密训练代理；若改用参数高效更新，实际训练算量与显存布局另测，不能直接把 N 换成 N_{train}。下一章会解释 $6ND$ 的适用边界。

再假设教师为 10 万个问题生成资料，每题计费输入 500 token、输出 300 token，统一假设价格

为每百万计费 token 10 元。则计费量与费用为

$$D_T = 10^5(500 + 300) = 8 \times 10^7, \tag{1.2}$$

$$C_T^{\text{money}} = \frac{8 \times 10^7}{10^6} \times 10 = 800 \text{ 元}. \tag{1.3}$$

这只是自定费率下的算例，不代表任何眞实服务报价。教师输出为 3000 万 token，与学生累计训练的两亿 token 不相同；后者可以包含眞实数据混合与重复训练，必须在数据清单中解释。若过滤后只保留一半样本，费用没有自动减半；若为了补足通过筛选的样本又生成一轮，教师预算可能超额。

下一步应运行小规模试验：乙是否达到质量下限？丙能否在目标运行时用压缩配置进入内存预算且保持质量？甲是否存在必须自建词表或架构的理由？若压缩后的丙通过所有硬约束，并以更低全生命周期成本达到同样质量，它可能比乙更合适。决策的正确产物是一组可验证的下一步，而不是在看见蒸馏算式后立即训练全部数据。

1.5 把一次训练成本扩展为全生命周期成本

训练项目常忽略三笔费用：淘汰实验、教师与验证器，以及部署后的重复调用。对一个计划服务 H 次请求的候选，可先写货币口径的账本：

$$\begin{aligned} C_{\text{life}}(H) = {} & C_{\text{data}} + C_{\text{teacher}} + C_{\text{search}} \\ & + C_{\text{train}} + C_{\text{compress}} + C_{\text{integration}} \\ & + H\overline{C}_{\text{request}} + C_{\text{maintenance}}. \end{aligned} \tag{1.4}$$

每项必须在同一货币计价基础上才可相加；FLOPs、元、秒和焦耳应分别记录，再按明确费率换算。价格会变化，因而还要保留原始量，例如教师输入输出 token、训练卡时和设备上的实测能耗。维护项包括数据更新、回归测试、版本回滚和坏例人工处理。

若候选甲比乙多花一次性成本 $\Delta C_0 > 0$，但每次请求节省 $\Delta c > 0$，不计其他变化时，摊销临界调用量为

$$H_\star = \frac{\Delta C_0}{\Delta c}. \tag{1.5}$$

例如额外投入 2000 元，每次省 0.002 元，需要约一百万次请求才能在这个简单账本中收回。若设备数、输入长度或人工复核率变化，临界点也会变化。因此短期原型可选易实现的适配路线，长期大规模部署可能值得训练更充分的小骨干，但不能仅凭预计请求量保证收益。

1.6 建立一个能被推翻的基线

最先建立的基线应能揭示项目是否需要训练。字段抽取可以先比较规则、检索辅助和已有小模型；只改变输出格式时，提示或受约束解码也可能足够。基线不是为了让新方法容易胜出，而是为了测出“训练到底补了什么”。若失败来自文档中根本没有信息，增加参数或蒸馏轨迹都不能产生可靠证据。

训练前锁定一个最小评测集，覆盖常见样本、长输入、缺失字段和异常格式；另保留从不用于调参的测试集。对每个候选同时保存零样本、训练后和部署转换后的结果。相同模型在训练框架里通过，不代表导出或量化后的版本通过。质量与成本的变化应对应同一组样本、同一输入输出长度与同一运行条件。

本书的组织沿这条证据链展开：目标与缩放规律约束“训练多少”，数据工程约束“学到什么”，蒸馏约束“向谁学习”，适配与压缩约束“改变哪些状态”，架构与系统约束“如何实际运行”，评测与研究设计检验这些选择是否成立。与《现代小模型微调》的分工，是在本书说明何时适配、如何

比较，而将 LoRA、QLoRA 与偏好适配的更细操作留给相应分册；量化内核与服务设施也分别有专门分册。跨书指引不替代本书自己的概念和必要算例。

1.7 练习与解题提示

1. 一个三十亿总参数模型只训练两千万适配参数。它的训练参数减少了多少？BF16 基座权重是否因此减少？**提示：**前一问计算比例；后一问检查冻结与删除的区别，还要考虑适配器能否合并。
2. 设两个候选在准确率、峰值内存、尾延迟上分别为 $(0.91, 1.5, 80)$ 与 $(0.90, 1.3, 60)$，后两项单位分别为 GB 与毫秒。谁支配谁？若最低准确率为 0.905，结论怎样变化？**提示：**先分别检验可行域，再比较各维；不要先加权平均。
3. 将本章教学案例中的教师筛选通过率改为 0.4。若仍需要同样数量的通过样本，独立同分布且费用不变的粗估下需花多少钱？**提示：**所需候选量除以通过率，800 元也按相同比例放大；这仍不包括筛选器成本，且通过率可能随新增数据变化。
4. 选一个真实工作负载，写出一个可以否决自己首选路线的实验。**提示：**写“若固定预算下压缩基座已达到所有门槛，就不做从零预训练”，比写“试试最新模型”更能支持决策。

第 2 章

语言建模、缩放规律与训练显存

预算不是训练程序之外的行政数字。预测哪些 token、怎样平均损失、用多长的序列，都会改变一次更新所代表的统计问题与系统成本。本章从下一 token 预测建立目标函数，再推导计算最优的含义，最后把参数量拆成可以检查的显存账本。需要区分三类数字：公式中的理论代理、按明确假设手算的估计，以及在某个框架与设备上测得的峰值。只有第三类能直接用于设备验收。

2.1 下一 token 预测究竟预测什么

给定文本经 tokenizer 转换的序列 $x_{1:T}$，自回归语言模型按链式法则定义

$$p_\theta(x_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{<t}). \tag{2.1}$$

序列起点可用专门的起始符号表示。训练时最大化似然等价于最小化负对数似然。它直接教模型在给定前缀下分配下一个 token 的概率，不直接保证事实正确、指令服从或长程规划。错误资料中的下一词仍可能很可预测，真实但罕见的信息则可能难以预测。

对含 b 条序列的一个批次，令 $m_{it} \in \{0, 1\}$ 表示第 i 条序列的第 t 个目标是否参与损失。padding、只作为提示的用户输入和某些特殊位置可取零。按有效目标 token 平均的目标为

$$\mathcal{L}_{\text{token}}(\theta) = -\frac{\sum_{i=1}^{b}\sum_{t} m_{it} \log p_\theta(x_{it} \mid x_{i,<t})}{\sum_{i=1}^{b}\sum_{t} m_{it}}. \tag{2.2}$$

分母应是参与监督的 token 数，而不一定是 bT。注意力掩码决定模型能看什么，损失掩码决定哪些预测受监督。把某个用户 token 的损失设为零，不会自动让模型看不到它；在指令微调中，用户输入通常正是应当保留的条件。

另一种目标先求每条序列的平均，再对序列平均：

$$\mathcal{L}_{\text{sequence}}(\theta) = -\frac{1}{b}\sum_{i=1}^{b}\frac{\sum_{t} m_{it} \log p_\theta(x_{it} \mid x_{i,<t})}{\sum_{t} m_{it}}. \tag{2.3}$$

这里假设每条序列至少有一个监督 token。两个目标都可以合理，但它们赋予样本的权重不同。假设两条序列分别含 1 个和 9 个目标 token，平均损失分别为 1 和 3，则序列平均为 2，token 平均为 $(1 + 27)/10 = 2.8$。前者让两条任务贡献相同权重，后者让每个目标 token 贡献相同权重。长推理轨迹若直接采用 token 平均，会获得更多总权重；这可能是有意设计，也可能是未意识到的偏置。

梯度累积时还要检查分母。如果各微批有效 token 数不同，简单平均每个微批的平均损失，不等于整个大批次的 token 平均。应按有效 token 数加权，分布式训练则需与框架的梯度归约方式一起核对。学习率与损失缩放也须保持一致，不能只修打印出的 loss。

2.1.1 移位、因果掩码与文档边界

decoder-only Transformer 在输入位置 t 产生用于预测下一位置的 logits。常见实现把输入与标签错开一个位置：

Listing 2.1 标签移位与监督掩码的最小示意；注意具体模型接口可能已在内部移位。

```
inputs = tokens[:, :-1]
targets = tokens[:, 1:]
mask = target_mask[:, 1:]
logits = model(inputs, attention_mask=input_mask[:, :-1])
per_token = cross_entropy(
    logits.transpose(1, 2), targets, reduction="none"
)
count = mask.sum()
if count == 0:
    raise ValueError("batch has no supervised token")
loss = (per_token * mask).sum() / count
```

若模型接口接受 `labels` 并自行移位，再手动移位一次会错过一个 token。最小边界检查是构造只有几个已知符号的序列，逐位置检查输入、目标与 mask；不能只看训练损失下降。没有监督 token 的批次应跳过或报错，不能默默以零损失进行优化器更新，因为动量、权重衰减和步数仍可能变化。

Transformer 的注意力可写为 [2]

$$\mathrm{Attn}(Q,K,V)=\mathrm{softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}+M\right)V. \tag{2.4}$$

在按输入位置编号的因果注意力中，$M_{ij}=0$ 对应允许读取的位置，未来位置取 $-\infty$。softmax 逐行归一化。当前位置可以读取当前输入 token，因为标签是下一 token；允许看当前标签才会造成泄漏。

为提高利用率，训练程序可能把多个短文档拼成定长块。加入文档结束符但允许跨文档注意力，与在每个文档边界重置注意力，是不同的训练分布。二者均有实际用途，必须记录策略。检验训练与验证时还应防止把同一文档切成邻近片段后分别放入两侧。

2.1.2 困惑度的可比条件

使用自然对数时，困惑度定义为 $\mathrm{PPL}=\exp(\mathcal{L}_{\mathrm{token}})$。它可以理解为几何平均意义上的有效候选数，但不意味着模型每一步真的面对同样多的选项。不同 tokenizer 会把同一文本切成不同数量和难度的 token，因而它们的 per-token loss 与 PPL 不能直接横比。对跨 tokenizer 的语言建模比较，可以统一到同一原始文本的每字节负对数似然：

$$\mathrm{bits/byte}=\frac{-\log p_\theta(\text{文本})}{n_{\mathrm{bytes}}\log 2}. \tag{2.5}$$

这仍要求无损文本处理、相同字节编码、边界规则和可计算的文本概率，且它并不替代下游任务评测。减少 PPL 是改善目标分布预测的一项证据，不是“通用能力提高”的完整证明。

2.2 从参数和 token 推到训练计算

对常见稠密 Transformer，若主要成本来自参数矩阵乘，一次前向在每个 token 上约需 $2N$ FLOPs，反向常按前向的约两倍估计。于是累计训练 D 个 token 的常用代理为

$$C_{\mathrm{train}}\approx \kappa ND, \qquad \kappa\approx 6. \tag{2.6}$$

这里把一次乘法和一次加法计为两个浮点操作。D 是累计处理的训练 *token*，重复读同一语料也重复计入 D；它不是唯一文本量，也不一定等于损失 mask 中取一的 token 数。一个仅监督短答案、却前向处理很长提示的训练，不能只用答案 token 估算全部计算。

这个式子适合预算量级检查，不是账单或计时器。自注意力的序列二次项、较大的输出词表、激活重计算、路由与负载均衡、优化器操作、通信、输入管线及教师调用都会造成偏离。对专家模型，使用总参数或活跃参数都需要说明被忽略的公共模块和通信。参数高效微调即使 $N_{\text{train}} \ll N$，仍需主干前向，通常也需为适配器计算传播经过冻结层的梯度，故不能直接写成 $6N_{\text{train}}D$。

设累计处理的训练 token 为 $D = 2 \times 10^{10}$，总参数 $N = 5 \times 10^{8}$，则代理计算为 6×10^{19} FLOPs。若实际持续吞吐折算为 P_{eff} FLOPs/s，时间粗估为 C/P_{eff}；不能代入芯片峰值后把它当训练工期。更稳妥的做法是在目标长度、精度和 batch 下测几个完整步骤的平均时间，再按步数外推，并为评测、存盘和失败恢复留预算。小规模试跑还需确认吞吐是否随数据加载、热状态或分布式通信而改变。

2.3 计算最优比例是怎样推出来的

Kaplan 等人与后来的 Chinchilla 研究在各自实验条件下观察并拟合了语言模型损失的缩放关系[3, 4]。它们不是同一组指数和同一训练策略的重复报告。为理解预算分配，可使用一个常见二项经验形式：

$$L(N, D) = E + AN^{-\alpha} + BD^{-\beta}, \qquad A, B, \alpha, \beta > 0. \tag{2.7}$$

E 表示该拟合中的不可约或渐近部分，参数项与数据项表达边际收益递减。它假设我们在指定数据分布、架构族、优化和训练范围内得到了有意义的拟合。换 tokenizer、改变数据质量、反复训练有限语料或进入新的规模区间，都可能需要重新拟合。没有这些条件，套入论文指数计算出的“最优模型”只是形式上的数字。

在固定计算代理 $C = \kappa ND$ 下，消去 D：

$$L(N; C) - E = AN^{-\alpha} + B\left(\frac{\kappa N}{C}\right)^{\beta}. \tag{2.8}$$

第一项随 N 增大而降低，第二项因为留给数据的预算变少而升高。对 N 求导并令其为零，得到

$$\alpha AN^{-\alpha} = \beta BD^{-\beta}. \tag{2.9}$$

这不是“两个损失项必相等”，而是它们对对数规模的边际收益相等。继续求解可得

$$N_\star = \left(\frac{\alpha A}{\beta B}\right)^{1/(\alpha+\beta)} \left(\frac{C}{\kappa}\right)^{\beta/(\alpha+\beta)}, \tag{2.10}$$

$$D_\star = \left(\frac{\beta B}{\alpha A}\right)^{1/(\alpha+\beta)} \left(\frac{C}{\kappa}\right)^{\alpha/(\alpha+\beta)}. \tag{2.11}$$

当拟合出的 α 与 β 接近时，最优参数和 token 随预算以接近的指数增长；其常数比例仍由数据、模型和拟合决定。“约多少 token 对应一个参数”不是对所有小模型、所有数据和所有部署需求都有效的自然常数。

一个常见错误是同时写 $AN^{-\alpha} + BD^{-\beta} + GC^{-\gamma}$，再在 $C = 6ND$ 下把三项看作独立旋钮。若 C 已由 N, D 约束，新增一项需要独立的统计建模理由；不能把相互依赖的预算重复计作额外的信息来源。另一错误是把累计 D 当无限独立数据。训练十遍同一小语料会增加计算，却未必产生与十倍唯一语料相同的收益。

缩放规律支持什么决策

它适合在一组可比训练中插值、分配下一轮实验的参数与数据预算，并检查“只增参数”的代价。它不能代替数据许可、能力门槛与部署实测，也不能在有限语料、极端长上下文和不同架构之间无条件外推。先用小规模实验检查趋势和残差，再使用拟合做较大投入。

2.4 为什么计算最优不等于部署最优

计算最优通常问：“给定一次训练预算，怎样获得最低验证损失？”部署项目还问：“这个模型在未来要被调用多少次，能否装入设备，并按时返回？”若训练后处理 H 个累计推理 token，一种极简总计算代理为

$$C_{\text{total}} \approx \kappa ND + \kappa_{\text{inf}} NH. \tag{2.12}$$

第二项只用于说明重复使用权重的影响，忽略了注意力、缓存与设备差异；实际部署必须换成工作负载对应的测量。要求损失不高于 L_0 时，由式 (2.7) 可形式上解出

$$D(N) \geq \left[\frac{B}{L_0 - E - AN^{-\alpha}}\right]^{1/\beta}, \quad L_0 - E - AN^{-\alpha} > 0. \tag{2.13}$$

在可行范围内，降低 N 往往需要更多 D；若分母非正，无论增加多少数据，这个拟合都不允许该规模达到目标。无限多数据不能在此模型中补偿无限小容量。

可用一个**无量纲教学模型**看出差异。假设 $L = 1/N + 1/D$，训练成本为 ND，目标为 $L = 0.2$。候选 $(N, D) = (10, 10)$ 的训练成本为 100；候选 $(7.5, 15)$ 的损失同为 0.2，训练成本为 112.5。若归一化部署成本为 hN，后者多花 12.5 的训练成本，却省 $2.5h$ 的部署成本，在 $h > 5$ 时总成本更低。这里的 N, D, h 都是教学单位，指数 1 不是论文拟合值，也不代表某个真实模型。它只证明一次训练最优与重复部署最优的目标函数不同。

实际项目更复杂：小模型可能需要更长的答案或更多重试；大模型也可能依靠更高质量减少工具调用与人工复核。部署总量因此不能只按每 token 的权重乘法估计。真正的优化对象是“完成一次合格任务的成本”，并满足延迟、风险与内存约束，而非强制让 N 越小越好。

2.5 训练显存：从状态而不是口号开始

全参数训练常用的账本为

$$M_{\text{peak}} \approx M_W + M_G + M_O + M_{\text{master}} + M_A + M_{\text{tmp}} + M_{\text{runtime}}. \tag{2.14}$$

各项分别是工作权重、梯度、优化器状态、可选的高精度主权重、保存的激活、临时工作区以及框架和分配器开销。峰值发生在某个具体时刻，部分状态可复用，另一些实现会产生额外副本；按项求和是容量规划的起点，而不是精确预测。

前三项合计 12 字节/参数；若另有 FP32 主权重则为 16 字节/参数。若梯度精度、优化器状态或参数分片方式改变，应重新计算。因此“全参训练固定需要 16 字节”与“BF16 训练只要 2 字节”都不完整。优化器状态还可能到第一次更新才分配，只测首次前向会严重低估峰值。

设一个教学模型有 $N = 5\times10^8$、$L = 12$ 层、隐藏维度 $d = 1024$，微批 $b = 4$、序列长度 $T = 1024$。采用上述 BF16 权重与梯度、FP32 Adam 矩及主权重，则参数相关部分为

$$M_{\text{states}} = 16N = 8 \times 10^9 \text{ 字节} \approx 7.45\,\text{GiB}. \tag{2.15}$$

保存一份形状为 $L \times b \times T \times d$ 的 BF16 隐藏张量需要

$$2LbTd = 100{,}663{,}296 \text{ 字节} = 96\,\text{MiB}. \tag{2.16}$$

表 2.1 一个明确假设的混合精度 AdamW 账本；不同实现不必采用相同布局。

状态	每参数字节	说明
BF16 工作权重	2	用于主要前向与反向计算
BF16 梯度	2	某些实现使用 FP32 梯度，此项改为 4
FP32 Adam 一阶、二阶矩	8	两个状态，各 4 字节
FP32 主权重（若有）	4	不能假定所有混合精度实现都另存一份
激活、工作区与运行时	不按 N 固定	随长度、批量、算子、重计算与分配策略变化

但训练每层不只保存这一份：前馈中间量、归一化输入、投影和其他反向所需数据均可能存在。若仅为演示假定需要 12 份等量张量，便为 1152 MiB = 1.125 GiB；系数 12 是教学假设，必须用框架测量替换。某些注意力实现还显式保存 bH_qT^2 级别的分数或概率矩阵，采用内存高效注意力后布局又不同。

由这个账本可以逐项判断措施的作用。微批从 4 降为 2，使式 (2.16) 的激活单位约减半，参数状态不变；通过两次梯度累积保持每次更新 token 数，仍会影响步时。T 从 1024 降为 512，会减少线性激活与注意力成本，却也改变模型实际见到的长依赖。只把工作权重从 16 bit 降至 4 bit，理想存储从 1 GB 降为 0.25 GB，其他状态未必减少；低比特训练还需要相应算法，不能把这个理想节省当作可直接获得的全参训练配置。

激活检查点以重新前向换取少保存中间量；分片把状态分给不同设备；卸载把部分状态移至主机内存；低精度优化器改变 M_O；参数高效方法主要改变梯度与优化器所覆盖的参数集合。它们可以组合，但相应的重算、通信、带宽和吞吐代价也会叠加。第 1 章的资源约束不能只检查显存而忽略完成训练所需的总时间。

2.6 让预算估计接受实测修正

最小训练记录须包含模型、tokenizer、监督分母、累计输入 token、精度与状态布局，并测初始化、完整前向、反向及首次更新后的峰值。GPU 同时报分配与保留内存；CPU 或统一内存报进程峰值和系统可用量，注明工具口径。释放缓存不等于清空全部硬件占用。

吞吐比较需固定任务与累计训练量。梯度累积不保证优化轨迹相同；packing 可能改变边界分布；缩短上下文虽提高吞吐，却可能删去长依赖。若预算改动改变了学习问题，须同时报告质量。

附录 B 给出本次最小实验与设备条件，用于检查受控比较流程。手算帮助定位待测项，实测决定配置是否可行；小网络的结果不能验证这里任意十亿参数的估算。

2.7 练习与解题提示

1. 两个微批的有效监督 token 分别为 100 和 300，平均损失分别为 2 和 4。整个批次的 token 平均损失是多少？**提示：**答案为 $(100 \times 2 + 300 \times 4)/400 = 3.5$，不是 3。
2. 式 (2.7) 中若 $\alpha = \beta$，预算扩大 16 倍，$N_\star, D_\star$ 各增几倍？**提示：**指数均为 1/2，各增 4 倍，常数比仍待拟合。
3. 八亿参数，BF16 权重、FP32 梯度与两个 Adam 矩，无主副本，状态占多少？**提示：**每参数 $2 + 4 + 8 = 14$ 字节，共 11.2 GB，未计激活和临时区。
4. 为何不能用 LoRA 参数量替换 $6ND$ 中的 N？**提示：**冻结主干仍参与前向及通往适配器的梯度传播。

第 3 章

数据、tokenizer 与可控的学习信号

在预算固定时，训练程序反复回答同一个问题：下一步把计算花在哪些文本上？数据工程正是对这个问题作出可追踪的选择。它既包括删除无效重复，也包括保留少见却重要的表达；既包括寻找高质量资料，也包括确认资料的使用条件和来历。一个只有“清洗后共多少 token”的数据集，无法解释模型为什么变好，更无法定位它后来为什么遗忘。

本章按资料进入项目、形成版本、转换为 token、混合采样、继续训练的顺序展开。数据处理的结果必须能回溯到源文档；训练结果则应能回溯到某个固定的数据版本。公开语料工程如 FineWeb 提供了流程与消融的参考 [5]，但没有一个现成过滤器能自动适应所有语言、领域与设备任务。

3.1 从来源清单开始，而不是从大文件开始

3.1.1 使用权利与内容质量是不同维度

网上能访问一份文本，不等于任意训练、发布和商业部署用途都已经获得许可。数据集的许可证也不一定消除了原始来源的权利问题。项目应在获取阶段记录来源、版本、时间、授权条件与预期用途；拿不到明确条件的资料应标记为待定，避免在训练结束后才发现权重不能按计划发布。

来源记录同时影响科学解释。包含大量教育材料的数据桶与普通网页桶有不同语言风格和任务分布；后来增加的代码仓库也许含有基准答案。没有版本和日期，就无法判断是过滤器起效，还是新增来源改变了覆盖。来源清单因此同时服务于可用性、可追踪性和复现。

最小记录可以采用一行一条的结构化清单，字段既不过度简化，也不要求暴露敏感原文：

Listing 3.1 来源清单示意；哈希标识固定内容，不能代替授权信息。

```
record = {
    "doc_id": "sourceA/2026-10/doc001",
    "source_url": "https://example.org/document",
    "retrieved_at": "2026-10-21",
    "source_version": "snapshot-001",
    "license": "record-exact-license-or-pending",
    "allowed_use": ["record-reviewed-project-uses"],
    "content_sha256": "...",
    "language": "zh",
    "domain": "maintenance",
    "parent_id": None,
    "is_synthetic": False
}
```

合成资料还要记录教师版本、提示模板、采样设置、生成时间和筛选器版本；切片、翻译和改写保留父文档标识。若不能再分发原文，仍可保存允许公开的来源列表、内容指纹、处理代码与统计，明确复现需要哪些访问条件。内容指纹可以核验候选内容，但不保证匿名化：低熵字段或已知文档仍

可能被字典匹配识别；公开前也须审查敏感性和许可。

3.1.2 把隐私处理放在生成派生数据之前

文档中的邮箱、手机号、用户标识和内部记录可能在复制、切片、合成问答时扩散。适合脱敏的字段应先处理，保留变换策略与审计计数。简单正则只能覆盖明确格式；过度删除又可能破坏任务，例如把设备编号和电话号码一并删除。抽样人工检查与任务敏感的规则比“过滤已完成”这个布尔值更有信息。

数据管线还应区分训练用文本和运行日志。模型不把用户原文发送到云端，不代表本地缓存、训练样本和适配器天然没有隐私风险。这个问题与小模型的部署位置有关，但不是靠减少参数量就能消失。

3.2 去重与划分：保持独立性比减少文件数更重要

3.2.1 精确去重先确定“相同”的含义

原始字节相同可以用密码学哈希识别；去除稳定的网页导航后再相同，则是另一种等价关系。文本规范化可以统一换行、Unicode 表示和多余空白，却不能任意改变有意义的大小写、缩进或数值。代码、数学和表格的规范化策略尤其要保守。好的实践是分别保留原文哈希与规范化哈希，而不是永久覆盖原始资料。

重复来源本身也有信息。相同文档从多个镜像进入时，可保留一个训练副本，并保存所有来源关系。对同一模板中的不同病例、同一公式的多种推导，不能因为局部相似就全部删除。去重的目标是减少不必要的重复暴露与切分泄漏，不是使所有句子都互不相似。去重能改善语言模型训练和减少记忆风险的证据需要绑定具体方法与语料 [6]。

3.2.2 近重复是一种带阈值的估计

MinHash 常用于近似估计文本片段集合的 Jaccard 相似度。将一篇文档表示为 n-gram 集合 S_i，两篇文本的相似度为

$$J(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}. \tag{3.1}$$

局部敏感哈希可减少候选对数量，之后再用更准确的相似度复核。n、分词方式、阈值和短文处理都会改变结果。语义向量相似适合发现改写，却也可能把同主题但不同事实的文章误当重复。没有一个阈值同时保证零漏检和零误删。

可先在抽样文档对上标注“应合并、应保留、待判断”，检查各类来源和语言的误差，再选择阈值。对于特别长的文档，段落或子串去重可能比整文相似更合适；对于短公式、公共协议和常见法律用语，公共片段又不能成为直接删除的唯一证据。去重簇内选哪个代表，也要考虑来源权利、格式完整性和质量。

3.2.3 在文档族而不是随机片段上划分

如果同一篇文章前半放训练集、后半放验证集，验证 loss 往往偏乐观；如果一道练习题及其改写分到两侧，下游分数也可能被高估。一个稳妥流程是先识别文档、近重复簇、改写族和来源族，再按组划分训练与验证。对已存在的外部测试集，还要在训练资料中单独检查重叠，并把处理记录保存下来。

流程顺序可调整，但应记录原因。清除网页模板可能提升去重精度；先划分再各自去重可能留下跨切分重复。若用验证指标调整过滤器，验证集已参与方法选择，最终测试须独立保留。

Listing 3.2 保留处理关系的数据版本流程示意。

```
docs = load_sources(manifest)
docs = normalize_with_raw_hash(docs, policy="v1")
docs = redact_sensitive_fields(docs, policy="v1")
docs = exact_dedup_keep_provenance(docs)
clusters = near_duplicate_clusters(docs, policy="v2")
train, valid = split_by_group(clusters, seed=17)
train, overlap_report = audit_against_eval_registry(
    train, frozen_eval_registry
)
train = apply_versioned_filters(train, policy="v3")
save_dataset_version(train, valid, overlap_report)
```

3.3 污染检查不能只搜索题目全文

评测污染是训练或开发过程使用了足以影响目标测试结果的信息。精确题面重合最容易检测，但答案页、翻译、变换数字、选项顺序变化和教师记忆中的题目也可能产生泄漏。反过来，模型见过牛顿定律不等于某个物理测试被污染：通用知识重叠与具体题目泄漏需要区别。

建立评测登记表时，固定题目版本、来源、获取日期和允许的开发用途。对训练语料做逐字匹配、规范化匹配和片段匹配；对高风险重叠再人工复核。把保留、删除与不确定的规则写清。若教师来源未知，不能声称已证明没有污染；可增加截止日期之后独立编写的题目、任务变体和跨来源测试，报告仍无法排除的风险。

用测试成绩反复挑数据过滤器，也是一种开发信息使用，即使没有把题面放进训练。最终报告应区分训练污染检查与开发集使用记录。最有价值的不是写“零污染”承诺，而是给出检测覆盖、阈值、命中数量、复核结论及未覆盖的泄漏类型。

3.4 质量不是一个可以直接相乘的系数

“有效 token”经常被用来表达优质且有信息量的数据。然而质量、重复、覆盖和污染相互相关，没有一套通用仪器能为每个桶测出四个独立的折扣系数。将它们相乘得到 D_{eff}，容易把启发式包装成可预测的物理量。本书改为记录可审计的计数：原始字节、清洗后文档与 token、近重复簇数、每个来源实际采样的 token、重复暴露次数，以及各类评测的覆盖。

质量过滤器可以基于规则、分类器或教师评分。它们实际优化的是某种代理：语法完整、教育性、可执行性或与目标领域的相关性。代理分数高，不保证资料真实或任务相关。代码可通过解析却没有正确功能；文本行文流畅却含有错误；教育性分类器可能偏好某一种文体，误删少数语言、口语和困难讨论。

表 3.1 过滤决策应同时保留收益假设和可能的覆盖损失。

处理	预期改善的量	应单独检查的损失
去导航、乱码与广告	降低无任务意义的 token	代码、表格或专业符号被误删
教育性或可读性筛选	增加解释与结构清晰的内容	真实口语、罕见表达和少数语言覆盖降低
真实性或执行校验	减少可检测错误	校验器只覆盖容易验证的题型
领域重加权	改善目标任务分布匹配	通用、跨域与安全能力遗忘
合成生成与拒绝采样	补充稀缺监督与难例	教师偏差、筛选偏差和生成成本增加

DataComp-LM 将数据筛选放到受控训练与评测中研究 [7]。对自己的项目，关键也是把一个过滤器当作可证伪假设：在相同学生、tokenizer、训练预算与数据切分下，它是否改善目标任务？它删除了哪些来源？若少数类别变差，总体平均值会不会把损失遮住？过滤后的 loss 较低还可能只是文本更容易预测，并不保证能力提高。

3.5 tokenizer 是数据与模型之间的成本接口

tokenizer 将字符或字节串映射为 token 序列。BPE 逐步合并常见片段，unigram 方法从候选词表中优化概率分词；SentencePiece 提供直接从原始文本学习子词模型的常用工具 [8]。这些是不同层次：算法、工具、规范化与特殊符号配置应分别记录。“使用 BPE”不足以唯一确定 tokenizer。

词表大小为 V、隐藏维度为 d 时，一份词嵌入矩阵含 Vd 个参数。若输入嵌入与输出投影不共享，通常还要另一份同量级参数；共享权重减少参数，却不消除对词表计算 logits 的成本。取 $d = 768$，把 V 从 32000 增至 64000，共享嵌入的参数从 2457.6 万增加到 4915.2 万，差额为 2457.6 万。对十亿规模底座，这是一项可管理的成本；对一亿规模底座，它可能改变主体结构的预算。

大词表可能让同样文本占用更少 token，也可能把稀有片段分配到训练不足的向量；小词表减少词表参数，却拉长序列并消耗上下文。应在目标语言、代码、数字、专业符号和噪声输入上测量压缩率，同时检查可逆性、未知 token 和规范化副作用。把中文标点、大小写或数学符号静默归一化，可能损失任务信息。

假设**同一批教学文本**在两个 tokenizer 下分别需要 1200 和 900 token，那么后者的长度减少 25%。若其他结构与精度相同，线性随长度的缓存项约变为 0.75 倍，显式注意力分数项约变为 $0.75^2 = 0.5625$ 倍。这不是吞吐预测，因为词表投影、实际内核、packing 与模型质量也可能变化。不同 tokenizer 的“训练十亿 token”包含不同量的原始文本，不能把相同 token 数当作相同数据量或相同计算量。

已有模型的 tokenizer 通常与嵌入参数绑定。新增领域 token 或更换词表，需要初始化和学习新的向量，常还需适配输出投影，不能无代价地替换。对蒸馏而言，不同 tokenizer 还会造成逐 token 概率空间不一致；下一章将解释为什么文本级迁移通常更容易执行。

3.6 配比决定每个领域实际得到多少梯度

把数据分成 K 个桶，目标 token 份额为 π_k，满足 $\pi_k \geq 0$、$\sum_k \pi_k = 1$。训练累计 token 为 D 时，第 k 桶预期得到 $D\pi_k$ 次 token 暴露。令 U_k 为按既定清洗和去重规则保留的语料 token 位置总数；相同 token ID 出现在不同位置时仍分别计数，它不是词表中不同 token ID 的类别数。粗略重复倍数为 $D\pi_k/U_k$。这个值不是独立信息量，却能提醒我们某个小桶是否已经被反复训练。

采样单位必须说清。如果按文档选择领域，概率为 w_k，且该桶平均文档长度为 ℓ_k，没有截断等因素时，token 份额近似为

$$\pi_k \approx \frac{w_k \ell_k}{\sum_j w_j \ell_j}. \tag{3.2}$$

例如两个桶的平均长度为 200 和 1000 token，等概率抽文档会产生约 1/6 与 5/6 的 token，而不是各一半。希望两桶 token 各占一半，可将文档采样概率设为 5/6 与 1/6，再检查实际截断与 packing 后的份额。固定长度块采样则更接近直接控制 token 份额。

温度采样常将桶大小 U_k 转为 $\pi_k \propto U_k^{\rho}$。$\rho = 1$ 近似按规模分配，$\rho = 0$ 近似各桶等份；中间值可以上采样小桶。它不是天然公平或最优的策略：小桶可能迅速过拟合，大桶也可能包含关键多样性。配比需结合去重语料量、目标任务和遗忘指标做消融，不能认定“代码桶总是越多越好”。

3.7 怎样区分质量收益与更多计算的收益

比较原始语料与筛选语料时，至少要明确要回答哪一个问题。固定**累计 token**、同 tokenizer、同结构与优化步数，可以较好地考察每次 token 暴露的收益，但不同长度分布仍可能改变注意力成本。固定**实际计算或设备时间**，可以考察给定资源下谁更划算，但此时每组读到的 token 可能不同。固定**唯一原始文本量**，则更接近比较处理策略如何利用同一份资料。三种实验不能合并成一句“质量更好”。

一个最小消融可从同一原始池构建四组：随机抽样基线、规则筛选、质量分类器筛选、真实与合成的混合。锁定学生初始化、tokenizer 与独立测试集，先按同 token 训练；另在同一设备时间预算下重复可行比较。每组记录去重语料 token 数、累计暴露、平均长度、重复簇和成本。若过滤器本身调用大教师，把其成本也记入资源表。若筛选后资料不足，不应偷偷重复更多轮后仍称“同等数据”。

超参数搜索预算也要控制。对新方法试几十种阈值，只给随机基线一个学习率，比较的已不只是数据质量。小样本预实验可以筛选趋势，最终比较则需说明每组搜索次数和保留规则。在附录 B 的最小实验中，数据切分和学生预算正是用来防止监督信号对比被其他变量混淆。

3.8 教材式合成与继续预训练的收益边界

3.8.1 教材式数据改变的是学习分布

教材式数据把概念、示例、解答和难度组织成易学习的结构。phi-1 的研究展示了这类数据与合成练习在特定代码训练设置中的价值 [9]。它支持的结论是：数据组织与任务覆盖能明显影响有限容量模型的学习；它不证明合成文本普遍优于真实文本，更不证明教师未见过的知识可以凭空产生。

合成管线可先覆盖任务模板，再变换条件、难度与表达，并用执行器或规则校验可验证部分。对错误样本保留原因，比只保存通过样本更有助于改进。教师风格单一、拒答倾向或常见事实错误会被复制；只接受教师自评高分样本，还可能收窄到它最擅长的题型。外部验证与独立人工抽样能减少风险，却不能证明未检查的维度都正确。

合成数据的独立性也有边界。同一模板改变姓名生成一万题，未必覆盖更多推理结构；教师在自己输出上继续生成，可能强化同一种模式。应跟踪题型、约束组合、来源父节点和失败类型，并在真实输入分布上验证。样本数增加与新信息增加不是同一事实。

3.8.2 继续预训练不等于可靠知识更新

继续预训练（continued pretraining，CPT）从已有权重出发，在新文本上继续语言建模。它适合改善领域术语、文体和局部分布，但“见过某事实”不保证模型在提问时能稳定调用它。变化频繁、需要明确出处的知识，检索可能更容易更新和审计；输出格式与交互行为，则通常需要任务监督。三者可以组合，但应分别衡量贡献。

领域 CPT 的主要风险之一是遗忘。只优化新领域的平均 loss，可能使通用语言、安全行为或原有任务退化。保留通用重放数据是一个候选措施，不是保证。可以固定 CPT 总 token，比较领域比例 π_{domain} 的几个值，并在训练前后同时评测领域集、通用集和风险集。若重放比例提高后领域收益减少、通用损失也减小，应报告这条权衡曲线，而不是只挑一个平均分。

实践上先用小学习率和短运行检查训练稳定性，保存多个可回滚检查点；用锁定的验证规则决定何时停止。若为 CPT 换了 tokenizer 或扩展上下文，新增参数学习和位置分布变化又成为独立变量。做出有益更新的证据，应是目标任务、遗忘与资源账本同时改善到可接受范围，而不是领域 loss 持续下降。

3.9 练习与解题提示

1. 一个桶去重后的语料含 5000 万个 token 位置，计划在两亿累计 token 中占 40%。粗略重复倍数是多少？**提示：**为 $2\times10^8\times0.4/(5\times10^7)=1.6$；近重复和不同切分会使真实信息重复比更复杂。
2. 两个文档桶平均长度分别为 100 和 400 token，怎样按文档采样使 token 份额相同？**提示：**概率与平均长度成反比，因此为 0.8 与 0.2；训练后仍需统计实际 token。
3. 为什么“筛选组的验证 PPL 更低”还不能证明过滤器改善了真实任务？**提示：**检查验证分布是否也变容易、tokenizer 是否相同、困难类别是否被删，以及测试是否独立。
4. 设计一个检验“合成教材缓解领域 CPT 遗忘”的对照。**提示：**固定总 token 和初始化，比较真实领域、真实领域加通用重放、真实领域加合成教材；分别报告领域、通用与风险指标，并将教师生成费用单列。

第 4 章

知识蒸馏：把教师能力变成学生能力

蒸馏把教师提供的监督转化为学生能够表达、检验和部署的函数。教师可以提供答案、词元概率、隐藏表示、关系矩阵或可验证的过程；它们带来的信息、成本与偏差并不相同。读本章只需熟悉交叉熵与链式法则。我们先固定监督事件和概率方向，再把词元、特征、序列与过程目标逐一写清楚，最后讨论学生自己生成前缀时怎样查询教师。

4.1 软标签多提供了什么

硬标签只告诉学生哪个答案被选中；软标签还给出其他答案的相对概率。例如，教师把两个易混类别分得接近，可能比单个类别编号更能反映输入结构。但这份不确定性也可能来自教师失误或校准不足，不能把“更软”直接解释为“更正确”。经典蒸馏把教师分布作为学习目标 [10]。

设教师和学生在同一条件 h 下、对同一大小为 V 的词表给出 logits $z_T, z_S \in \mathbb{R}^V$。温度 $\tau > 0$ 下，

$$
\begin{aligned}
p_T^{(\tau)}(v \mid h) &= \frac{\exp(z_{T,v}/\tau)}{\sum_{u=1}^{V} \exp(z_{T,u}/\tau)}, \\
p_S^{(\tau)}(v \mid h) &= \frac{\exp(z_{S,v}/\tau)}{\sum_{u=1}^{V} \exp(z_{S,u}/\tau)}.
\end{aligned}
\tag{4.1}
$$

温度升高会让分布趋于均匀，但不改变 logits 的大小顺序。这里的温度用于训练目标，部署时仍通常用未缩放的 logits；生成时的采样温度是另一项配置。

教师到学生的正向 KL 为

$$
\begin{aligned}
D_{\mathrm{F}}(h) &= \mathrm{KL}\left(p_T^{(\tau)} \middle\| p_S^{(\tau)}\right) \\
&= \sum_{v} p_T^{(\tau)}(v \mid h) \log \frac{p_T^{(\tau)}(v \mid h)}{p_S^{(\tau)}(v \mid h)}.
\end{aligned}
\tag{4.2}
$$

它等于交叉熵 $H(p_T^{(\tau)}, p_S^{(\tau)})$ 减去教师熵；教师固定时，减去的项不影响学生梯度。反向 KL 则是 $\mathrm{KL}(p_S^{(\tau)} \| p_T^{(\tau)})$，期望由学生概率加权。正向方向会惩罚学生遗漏教师仍赋予概率的事件；反向方向会更直接惩罚学生在教师低概率区域放置概率。有限容量下，两者可能得到不同折中，不能在代码中默默交换。

例如，取教师概率 $(0.8, 0.1, 0.1)$、学生概率 $(0.6, 0.3, 0.1)$，直接代入定义，两个方向分别约为 0.1203 与 0.1570，单位为 nat。这是概率分布的手算，未涉及模型训练。完整 softmax 在有限 logits 下各项为正；若缓存时裁掉小概率或只保存 top-k，就必须额外定义剩余概率质量和零概率处理，此时目标已发生变化。

4.1.1 为什么通常乘温度的平方

保持教师固定，利用

$$\frac{\partial \log p_S^{(\tau)}(v)}{\partial z_{S,j}} = \frac{\mathbf{1}\{v=j\} - p_S^{(\tau)}(j)}{\tau}, \tag{4.3}$$

可得

$$\begin{aligned} \frac{\partial D_{\mathrm{F}}}{\partial z_{S,j}} &= \frac{p_S^{(\tau)}(j) - p_T^{(\tau)}(j)}{\tau}, \\ \frac{\partial(\tau^2 D_{\mathrm{F}})}{\partial z_{S,j}} &= \tau\big(p_S^{(\tau)}(j) - p_T^{(\tau)}(j)\big). \end{aligned} \tag{4.4}$$

令 $\tilde{z}_j = z_j - V^{-1}\sum_v z_v$。当 τ 相对 logits 的间隔足够大时，softmax 的一阶展开为

$$p^{(\tau)}(j) = \frac{1}{V} + \frac{\tilde{z}_j}{V\tau} + O(\tau^{-2}), \tag{4.5}$$

$$\tau\big(p_S^{(\tau)} - p_T^{(\tau)}\big) = \frac{\tilde{z}_S - \tilde{z}_T}{V} + O(\tau^{-1}). \tag{4.6}$$

因此，未缩放的梯度在这一高温近似中按 τ^{-2} 缩小，乘 τ^2 可补偿主要量级。它不是任意温度下梯度恒定的定理，也不保证在有限学生容量、混合硬标签等条件下，不同温度仍给出相同的最优解。

用两类算例检查温度缩放

取教师 logits $(2,0)$、学生 logits $(0,0)$。当 $\tau=1$ 时，教师第一类概率约为 0.8808，缩放损失对学生第一项 logit 的导数约为 -0.3808；当 $\tau=2$ 时，这两个量分别约为 0.7311 与 -0.4621。高温极限的导数为 -0.5。数值来自 softmax 与式 (4.4) 的手算，说明乘平方保留的是近似量级。

4.2 把词元监督写成可实现的目标

对样本 (x_n, y_n)，令 $h_{nt} = (x_n, y_{n,<t})$，$m_{nt} \in \{0,1\}$ 表示该位置是否参与监督，$M = \sum_{n,t} m_{nt} > 0$。硬标签与词元蒸馏分别定义为

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{M}\sum_{n,t} m_{nt} \log p_S^{(1)}(y_{n,t} \mid h_{nt}), \tag{4.7}$$

$$\mathcal{L}_{\mathrm{KD,tok}} = \frac{\tau^2}{M}\sum_{n,t} m_{nt}\,\mathrm{KL}\Big(p_T^{(\tau)}(\cdot \mid h_{nt}) \Big\| p_S^{(\tau)}(\cdot \mid h_{nt})\Big), \tag{4.8}$$

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{\mathrm{CE}} + \alpha\mathcal{L}_{\mathrm{KD,tok}}, \qquad 0 \le \alpha \le 1. \tag{4.9}$$

这里对有效词元平均，长样本因位置较多而获得更大总权重。若先对每条序列平均再对样本平均，短样本的单词元权重会更大，两种目标都可使用，但必须报告选择。只监督回答时，提示与 padding 的掩码为零；EOS 是否监督、packing 边界怎样处理，也应固定。分类任务可视为每个样本只有一个有效位置。

同一次学生前向的未缩放 logits 用于硬标签交叉熵。实现时可以缓存教师 logits 或对数概率，但要同步保存教师版本、温度、词表、有效位置和数值精度。先做稳定的 `log_softmax`，通常比先 softmax 再取对数稳健；低精度 logits 在 FP32 中计算概率不能挽回此前已经丢失的信息。常用框架的 KL 接口可能要求第一个参数为学生对数概率、第二个参数为教师概率，参数顺序不等于数学 KL 的书写顺序。

Listing 4.1 同词表、相同前缀下的词元蒸馏；省略优化器细节

```
# logits: [batch, time, vocab]; targets, mask: [batch, time]
# All positions are aligned to next-token labels.
with no_grad():
    teacher_logits = teacher(prefixes)
    log_p_t = log_softmax(teacher_logits.float() / tau, dim=-1)
student_logits = student(prefixes)
log_p_s = log_softmax(student_logits.float() / tau, dim=-1)
kl_per_token = (exp(log_p_t) * (log_p_t - log_p_s)).sum(-1)
ce_per_token = cross_entropy_unreduced(student_logits, targets)
M = mask.sum()
assert M > 0
kd = tau**2 * (mask * kl_per_token).sum() / M
ce = (mask * ce_per_token).sum() / M
loss = (1 - alpha) * ce + alpha * kd
loss.backward()
```

先验证事件对齐，再检查损失下降

对一个短样本打印输入、下一词元标签与掩码，确认两个模型预测的是相同位置。再用标量求和核对 KL，检查教师无梯度、padding 不改变结果、学生与教师完全相同时 KL 近于零，并用有限差分核对式 (4.4)。一个形状能广播的实现也可能在语义上错位。

4.3 特征、关系与不同 tokenizer 的限制

表 4.1 蒸馏信号对应不同的监督对象。

信号	传递的内容	主要条件与风险
硬标签	真实答案或教师选出的答案	两者来源要区分；教师答案可能错误
词元分布	候选词元的相对偏好与不确定性	需要相同事件空间或严格的概率映射
隐藏表示	位置上的中间特征	要定义层映射、宽度投影和归一化
关系矩阵	词元间相似度或注意力关系	不要求相同宽度，但关系未必与任务一致
序列与过程	文本、程序、方程或工具轨迹	生成和筛选会改变数据分布；过程需验证

例如，将学生第 ℓ 层与教师第 $g(\ell)$ 层对齐。令位置特征分别为 $h^S_{nt\ell} \in \mathbb{R}^{d_S}$、$h^T_{ntg(\ell)} \in \mathbb{R}^{d_T}$，可训练投影为 $P_\ell \in \mathbb{R}^{d_T \times d_S}$。一种特征损失是

$$\mathcal{L}_{\text{feat}} = \frac{1}{M|\mathcal{J}|d_T} \sum_{n,t} \sum_{\ell \in \mathcal{J}} m_{nt} \left\| P_\ell h^S_{nt\ell} - h^T_{ntg(\ell)} \right\|_2^2. \tag{4.10}$$

$\mathcal{J}$ 是参与蒸馏的学生层集合。投影会引入训练参数；若它只服务训练，可在推理时移除。也可匹配归一化特征的余弦距离，但需说明零向量处理。原始均方误差对表示的尺度和坐标敏感，即使两个模型实现相近函数，隐藏向量也未必逐坐标相似。

为弱化宽度限制，可对一条有效长度为 T 的序列，把行向量归一化的特征写成 $\bar{H}_S \in \mathbb{R}^{T \times d_S}$、$\bar{H}_T \in \mathbb{R}^{T \times d_T}$，匹配关系 $R_S = \bar{H}_S \bar{H}_S^\top$ 与 $R_T = \bar{H}_T \bar{H}_T^\top$，例如最小化 $T^{-2}\|R_S - R_T\|_F^2$。两端关系矩阵均为 $T \times T$，但计算与存储随长度平方增长。注意力蒸馏还要定义头的映射、因果掩码和行归一化，不能直接把无掩码的关系损失套到因果注意力上。

DistilBERT 联合使用语言模型、蒸馏和表示距离；MiniLM 主要蒸馏自注意力关系，说明监督不必要求教师与学生完全同宽 [11, 12]。引入更多目标未必更好：中间特征可能限制学生形成更适合自身容量的表示。应保留只用输出监督的基线，单独消融特征项及其权重。

若教师与学生 tokenizer 不同，一个教师词元可能对应多个学生词元。相同数组位置不再表示同一事件，隐藏表示的位置也同样错位。文本级序列蒸馏是较直接的选择：保存教师输出的字符串，再由学生 tokenizer 编码。若要在字符串或字节边界上聚合概率，片段概率通常涉及多个条件概率的乘积以及可能的多条切分路径，不能简单把几个词元 logits 相加或平均。特征蒸馏则需先定义共同文本片段、池化和层映射；投影宽度不能自动解决时间位置对齐。

教师更大也不保证更适合学生。高熵词表分布、长轨迹和多层关系可能超出学生可表达的自由度；提高蒸馏权重可能挤压真实标签信号。报告教师和学生的层数、宽度、词表、上下文、目标层及有效词元数，才能解释收益来自监督还是额外容量和数据。

4.4 序列级蒸馏与前缀分布

设 $q_T(y \mid x)$ 是实际保存的教师序列分布，包含所用解码、长度限制及筛选规则。序列蒸馏以完整教师序列为硬目标：

$$\begin{aligned}\mathcal{L}_{\text{seq}}(x) &= -\mathbb{E}_{y\sim q_T(\cdot|x)} \log p_S(y \mid x) \\ &= -\mathbb{E}_{y\sim q_T} \sum_{t=1}^{|y|} \log p_S(y_t \mid x, y_{<t}).\end{aligned} \tag{4.11}$$

这里将 EOS 包含在 y 内，并假定序列分布归一化。于是 $\mathcal{L}_{\text{seq}} = H(q_T) + \mathrm{KL}(q_T \| p_S)$。若先按长度除以 $|y|$，则改变了不同序列的权重，不能不加说明地沿用同一个 KL 等式。

教师按原始概率采样时，q_T 可等于其序列模型；采用贪心或束搜索时，它更接近被解码器选出的目标；经过验证器筛选后，又变成条件于通过筛选的分布。机器翻译中的序列蒸馏表明，这类目标可降低学生面对的输出多样性和搜索难度 [13]，但也可能固化教师的格式偏好和错误。

对教师序列做 teacher forcing 时，学生始终读取教师给出的历史词元。部署时，学生读取自己的历史词元；一旦走错，后续状态可能远离训练分布，这就是这里所说的暴露偏差。序列蒸馏本身并未让学生在自己的错误前缀上受监督。它与“从谁的分布抽取前缀”是两个独立设计选择。

4.4.1 在学生访问的状态上查询教师

在每轮更新前固定一个学生快照 $\bar{\theta}$，让它生成序列；从生成序列的有效位置抽取状态 $h = (x, \hat{y}_{<t})$。记这些状态的分布为 $d_{\bar{\theta}}$，一种正向 KL 在线目标为

$$\mathcal{L}_{\text{on}}(\theta; \bar{\theta}) = \mathbb{E}_{h\sim d_{\bar{\theta}}} \left[\tau^2 \, \mathrm{KL}\Big(p_T^{(\tau)}(\cdot \mid h) \Big\| p_{S,\theta}^{(\tau)}(\cdot \mid h) \Big) \right]. \tag{4.12}$$

训练时把采样的前缀视为常量，只对当前学生输出求梯度。该目标明确表示一次基于已采样状态的监督更新；它不是把 d_θ 也当成可微函数后得到的完整策略梯度。若使用后者，还需处理采样分布随参数变化的项。状态按词元还是按序列平均、是否混入真实前缀，都应写进配置。在线蒸馏研究强调在学生自生成状态上提供监督的价值 [14]；采用何种 KL 方向或其他散度仍是另一个选择。

如果教师只能返回单个词元 a，可在固定状态上采样 $a \sim p_T^{(\tau)}(\cdot \mid h)$，用

$$\hat{\ell}(h, a) = -\tau^2 \log p_{S,\theta}^{(\tau)}(a \mid h) \tag{4.13}$$

估计软目标交叉熵。对教师采样取期望后，其学生梯度与式 (4.12) 相同，但单样本方差通常更大。若教师返回 argmax、用不同温度采样，或只保留通过筛选的词元，目标分布已经改变；这些都不能笼统写成“教师分布目标”。

Listing 4.2 固定采样状态的一轮在线蒸馏

```
with no_grad():
    trajectories = student_snapshot.generate(prompts, rollout_config)
    states = valid_prefixes(trajectories, include_eos=True)
    log_p_t = log_softmax(teacher(states).float() / tau, dim=-1)
log_p_s = log_softmax(student(states).float() / tau, dim=-1)
loss = tau**2 * mean(sum(exp(log_p_t) * (log_p_t - log_p_s), -1))
optimizer.zero_grad()
loss.backward()
optimizer.step()
```

教师须真正读取学生生成的同一前缀，不能拿预先缓存在教师前缀上的 logits 替代。学生每次更新后，访问的状态又会变化，因此缓存的时效、生成长度、错误前缀中的教师可靠性和额外调用量都要检查。

4.5 从答案到可验证的过程蒸馏

对问题 x、答案 y 和过程 r，先决定学生应生成什么。若训练输出按“过程、答案”排列，概率可分解为 $p_S(r \mid x)p_S(y \mid x, r)$。相应监督损失可写为

$$\mathcal{L}_{\text{reason}} = \lambda_r \mathcal{L}_{\text{CE}}(r \mid x) + \lambda_y \mathcal{L}_{\text{CE}}(y \mid x, r), \tag{4.14}$$

其中两个交叉熵各自按相应有效词元归一化，权重需随定义报告。答案单独蒸馏则优化 $\mathcal{L}_{\text{CE}}(y \mid x)$，两者的输入条件和推理成本不同。长过程带来的额外词元不能在训练或部署预算中消失。

验证器 $v(x, r, y)$ 可以来自程序执行、单元测试、数学等价性检查或独立判别器，但它怎样影响训练必须另作定义。

- **筛选或加权**。先生成过程，按验证结果接受样本或赋予非负权重，再训练式 (4.14)。离散执行器只参与数据选择，无需对它求导；同时记录通过率、拒绝原因和筛选后的题型覆盖。
- **奖励**。让学生采样 (r, y) 后使用验证结果作为奖励，需要强化学习或其他带采样估计的优化方法。它不等于在普通交叉熵后加一个能够自动反传的标量。
- **可微代理**。可以训练判别器或在连续表示上构造代理损失，但必须说明梯度究竟流向何处、如何越过离散生成、代理与真实执行结果的误差。一个判别器内部可微，并不使整段离散轨迹可微。

答案通过检查不证明中间步骤正确：偶然猜中、错误抵消或只覆盖部分行为的测试都可能漏检。对代码要保留独立测试及运行环境；对方程要核对变形条件、定义域和中间状态；开放问答则需保存来源证据与拒答条件。推理轨迹可提供训练信息，但“更长、更像推理”不是能力指标。可将答案蒸馏、未经筛选的过程蒸馏、经过同一验证器筛选的过程蒸馏作对照，同时固定学生预算，并测答案正确率、生成长度和验证成本。

4.6 怎样证明蒸馏值得付出成本

比较时至少保留真实硬标签训练、教师文本训练和所选软目标三类中与任务相关的基线。固定学生结构、初始化、训练切分和有效词元预算；软目标另报告温度、权重、目标层、教师生成配置及筛选率。多个种子反映训练波动，独立测试集用于最后判断；教师未覆盖的分布、少数语言和错误前缀也应有适量检查。

教师成本进入总账

固定学生更新步数不等于固定总成本。分别记录教师训练或取得成本、生成和 logits 缓存、过滤、学生更新、超参搜索与评测。若一个教师复用于多个学生，要写明摊销假设；若只对学生预算作公平比较，要将教师支出单列。软标签存储也可能成为瓶颈：保存 M 个位置、词表大小 V、每项 b 字节，稠密缓存仅数值部分就需 MVb 字节。

本书附录在固定合成六分类任务上，使用同一学生初始化、同一小批次序列和同一更新次数，比较硬标签与 $\tau = 2, \alpha = 0.5$ 的教师到学生 KL。这一次固定种子中，蒸馏学生的准确率和校准均未优于硬标签学生。它只说明一个具体教师、数据和预算下软标签没有自动带来收益，不能外推到语言模型或其他温度。协议、实测数字和限制见附录 B。

4.7 练习与提示

1. 给定教师 logits $(4, -1, 0.5)$、学生 logits $(0.2, 2, -0.7)$ 和 $\tau = 2$，计算两个 KL 方向，并核对正向缩放损失的梯度。提示：先算稳定 softmax，再按定义求和；导数应为 $\tau(p_S - p_T)$，分量之和为零。
2. 若教师在学生前缀上返回 argmax，训练目标与教师分布蒸馏有何区别？提示：分别写出 one-hot 目标与概率目标；区分前缀采样分布和下一词元监督分布。
3. 设计 tokenizer 不同的对照。提示：先用文本级序列蒸馏建立可执行基线；若引入片段概率或特征对齐，明确事件、边界与附加参数，并固定学生词元预算。
4. 为什么增加教师过程样本可能降低测试表现？提示：检查重复、错误轨迹、题型覆盖、污染、长度预算和验证器的漏检；答案通过率不能代替步骤正确率。

第 5 章

先决定更新什么：参数高效与低显存训练

第 4 章选择监督信号，本章选择允许信号改变的参数。同一个蒸馏目标既可全参数训练，也可只更新低秩增量。低显存训练还可以少存优化器状态、重算激活或把状态移到其他设备。这些办法改变不同的账目，不能只凭“可训练参数百分比”判断总成本。本书解释这些选择的原理与边界；已有模型的详细适配流程可继续参阅《现代小模型微调》。

5.1 四种节省，对应四本账

设基座共有 N 个参数，仅更新 N_{train} 个。全参数训练中，权重、梯度和 Adam 状态都随 N 增长；冻结基座后，不必为冻结参数保存梯度和更新状态，但基座权重与前向计算仍在。若采用新加的适配器，N_{train} 指新增可训练参数数目；此时运行模型的总参数是基座加适配器，不能把它误写为基座的一个子集。

表 5.1 节省的对象、代价与应记录的量。

对象	方法例子	代价与验证
梯度与更新状态	LoRA、只训练连接器	激活未必同步变小；测峰值内存和实际更新矩阵
优化器状态	低精度状态、梯度投影	检查数值误差、投影刷新成本和收敛轨迹
保存的激活	检查点、微批与累积	多做前向或增加调度；记录步时与有效 token/s
设备常驻状态	分片、CPU 卸载	通信或主机带宽可能成瓶颈；记录端到端耗时

梯度累积把大批量拆成多个微批。若一次优化步累积 a 个微批，每批有 b 条、每条 T 个有效词元，该步共见 abT 个词元。长度不一时，令第 k 个微批的词元损失总和与有效数分别为 S_k, M_k，目标应为

$$\mathcal{L}_{\text{step}} = \frac{\sum_{k=1}^{a} S_k}{\sum_{k=1}^{a} M_k}. \tag{5.1}$$

只有 M_k 相同时，把每批平均损失再除以 a 才与此相同。累积降低单次激活峰值，不减少相同总词元预算下的主要计算；依赖 batch 统计的算子、随机性和数值舍入也可能使它与一次性大批量不完全等价。激活检查点则少存中间结果、在反向前重算相应前向，以计算换内存。

5.1.1 冻结权重为什么仍要反向传播

对线性层 $u = W_0 x$，设输出端梯度为 $g = \partial \mathcal{L}/\partial u$。链式法则给出

$$\frac{\partial \mathcal{L}}{\partial x} = W_0^\top g, \qquad \frac{\partial \mathcal{L}}{\partial W_0} = g x^\top. \tag{5.2}$$

冻结 W_0 可以省去右式的参数梯度与后续更新；若 x 来自更早的可训练模块，左式仍必须计算。非线性层、归一化和注意力还可能需要保存或重算其反向所需的中间量。因此，冻结参数不等于把整段前向包进 `no_grad`：那会同时切断向输入的梯度。

只有某个冻结前缀的输入也不需要梯度，且其之前没有待训练模块时，才可在该前缀使用不建图的前向。例如只训练最后的分类头时，冻结编码器常可如此处理；在早期层插入适配器后，就应重新检查梯度路径。验证方法是明确列出应有梯度的参数，做一次反向后检查其梯度与参数变化，而不只看损失是否下降。

5.2 LoRA 的低秩增量限制了什么

对线性层 $W_0 \in \mathbb{R}^{m\times n}$，低秩适配（Low-Rank Adaptation，LoRA）[15] 使用

$$W = W_0 + sBA, \qquad s = \frac{\alpha_{\mathrm{L}}}{r}, \qquad B \in \mathbb{R}^{m\times r}, \quad A \in \mathbb{R}^{r\times n}. \tag{5.3}$$

r 是所选的正整数秩预算，α_{L} 控制增量尺度。W_0 冻结，只训练 A, B，新增参数为 $r(m+n)$，而不是 mn。增量 $\Delta W = sBA$ 的秩至多为 r；其列空间与行空间随两个因子共同学习，并非训练前就固定的一组方向。这个约束是归纳偏置，不是“所有任务更新天然低秩”的证明。

对输入 x，输出为 $u = W_0 x + sB(Ax)$。沿梯度 $g = \partial \mathcal{L}/\partial u$，

$$\frac{\partial \mathcal{L}}{\partial B} = s\, g(Ax)^\top, \qquad \frac{\partial \mathcal{L}}{\partial A} = s\, B^\top g x^\top, \tag{5.4}$$

$$\frac{\partial \mathcal{L}}{\partial x} = W_0^\top g + sA^\top B^\top g. \tag{5.5}$$

这说明低秩分支既要学习因子，也要把梯度送往更早的层。通常初始化 $B = 0$、A 为非零随机矩阵，使初始模型与基座相同；第一步 B 可获得非零梯度，而 A 的梯度先为零，后续随 B 改变。若两个因子都为零，在只有这一双线性分支且无额外扰动时，两个因子的梯度都为零，训练无法启动。

Listing 5.1 冻结基座但保留输入梯度的低秩层

```
W0.requires_grad_(False)
A = random_matrix(rank, input_width)
B = zeros(output_width, rank)
u = linear(x, W0) + (alpha_lora / rank) * linear(linear(x, A), B)
loss = task_loss(u, target)
loss.backward()
# W0.grad is absent; A and B remain trainable.
# Do not detach x or put the base branch inside no_grad().
```

伪代码中的 A, B 应注册为可训练参数，优化器只接收这些参数。真实实现还要处理 bias、dropout、混合精度与融合算子；这些配置需随适配器保存。

5.2.1 手算更新状态，不把它当整模压缩率

取 $m = n = 2048, r = 16$，原矩阵有 4 194 304 个参数，低秩因子共 65 536 个，为原矩阵的 1.5625%。在 BF16 参数、BF16 梯度和两个 FP32 Adam 矩、无额外 master weights 的账本下，每个可训练参数对应 12 字节状态。于是：

- 全量更新该矩阵的状态为 48 MiB；
- LoRA 因子的状态为 0.75 MiB，另加冻结 BF16 基座的 8 MiB，合计 8.75 MiB。

这只计算一个矩阵，未计激活、临时工作区、bias 和其他层；若实现保留 FP32 master weights，还需另加。LoRA 的比例主要解释更新状态为何减少，并不说明模型文件、前向计算或整个训练峰值会按同一比例下降。

秩也不是唯一旋钮。更新注意力投影还是前馈矩阵，是否包括 embedding、归一化和输出头，都会改变容量与内存。同样的 r 在不同宽度、不同目标矩阵集合下不是同一参数预算。可先比较目标层，再在固定总新增参数预算下分配各层秩；更复杂的分配与重参数化方法不应替代简单 LoRA 基线。

推理时，浮点基座可将增量合并为 $W' = W_0 + sBA$，消除独立分支。合并结果仍是原形状的稠密矩阵，基座参数量没有降低。保留独立适配器则便于任务切换，但增加分支与访存；合并前后的输出等价性需在关闭 dropout、相同精度和容差下核验。

5.3 QLoRA、LoftQ 与部署量化的关系

QLoRA[16] 将冻结基座用低比特格式存储，在计算时按实现恢复近似数值，并训练 LoRA 增量。若 $\widehat{W}_0$ 表示解码后的基座，计算近似为 $\widehat{W}_0 x + sB(Ax)$。存储位宽、计算 dtype 与累加精度是三项配置；“4-bit 基座”不意味着所有激活和矩阵乘都以 4-bit 执行。

其 4-bit NormalFloat（NF4）使用非均匀码本；双重量化进一步压缩量化常数；分页优化器涉及状态的内存管理。这三项分别影响数值表示、元数据和状态放置。NF4 不能不经适配就交给任意 INT4 乘法单元，分页也可能引入主机与设备之间的数据迁移。低比特基座仍须提供对输入的反向路径，不能因其权重冻结就切断前层适配器的梯度。

低显存适配与 QAT 的优化对象不同

常见 QLoRA 固定基座及其量化配置，只学习增量；典型量化感知训练（QAT）在前向中使用目标量化或模拟量化，按指定方案优化浮点影子权重、缩放或其他参数，使模型适应部署误差。两者可以组合，广义的量化感知适配也有交叉；仅训练一个低比特基座上的适配器，不能证明产物已经适配目标设备的量化格式、激活精度和内核。第 6 章再解释 QAT 的梯度近似。

量化后的基座与浮点基座不同，增量既可能学习任务迁移，也可能补偿量化残差。LoftQ[17] 将量化与低秩初始化协同考虑，其核心思想可用下列近似问题说明：

$$\min_{Q \in \mathcal{Q}_b, A, B} \|W_0 - Q - sBA\|_F^2. \tag{5.6}$$

$\mathcal{Q}_b$ 是给定位宽与量化格式允许的解码矩阵集合。这里展示的是初始化的重构目标，不是最终任务损失。固定 Q 时，低秩因子可近似残差 $W_0 - Q$；更新量化近似时，又可考虑已有低秩项。与“先独立量化、再把零增量接上”相比，这给了初始误差另一种分配方式，但不能保证任意残差都能由小秩表示，更不能由权重重构误差直接保证任务性能。

5.3.1 合并是一次需要验证的模型变换

若把低比特基座与增量合并，常见路径是先恢复浮点近似，得到 $W_{\text{merged}} = \widehat{W}_0 + sBA$，再按部署格式量化。第二次量化通常产生 $Q_{\text{deploy}}(W_{\text{merged}}) \neq W_{\text{merged}}$，因此不能要求与未合并的双分支逐位相同。另一条路径是运行时保留独立分支，其支持的算子、精度与额外开销需单独检查。

至少保存并比较浮点原模型、量化基座、量化基座加适配器，以及最终部署产物。用小批真实输入检查合并前后 logits，再检查独立任务集、模型文件与真实延迟。记录基座哈希、量化配置、适

配器配置、校准数据版本和运行时版本；适配器不能任意接到另一个同名但版本不同的基座上。

5.4 前缀、梯度投影与状态分片

软提示在输入侧增加可训练向量；前缀方法常在层内引入可学习的键和值。两者保留基座，但可能增加有效上下文或缓存。若任务需要改变已有表示的内部计算，少量提示向量未必足够；应在相同数据和推理预算下与 LoRA 或可承受的全参数基线比较。

梯度投影节省的是优化器处理梯度的维数。以左投影为例，令完整梯度 $G_t \in \mathbb{R}^{m\times n}$，投影基 $P_t \in \mathbb{R}^{m\times r}$ 的列正交，则

$$\widetilde{G}_t = P_t^\top G_t \in \mathbb{R}^{r\times n}, \tag{5.7}$$

$$U_t = \mathrm{AdamStateUpdate}(\widetilde{G}_t), \qquad W_{t+1} = W_t - \eta P_t U_t. \tag{5.8}$$

这是帮助理解维数的简化写法。它保留完整参数矩阵，投影状态只需随较小维数增长，但不必然消除完整梯度的临时峰值。每步更新可为低秩，跨步累加及子空间刷新仍可能形成更高秩的总更新，这与把最终增量固定为 BA 有区别。具体方法还须规定左右投影、刷新频率、尺度和刷新后的状态继承。GaLore 是这一方向的实现 [18]。

不能把 $U_t \in \mathbb{R}^{r\times n}$ 直接从 $W_t \in \mathbb{R}^{m\times n}$ 中相减，也不能把“先做 Adam 再投影”与“先投影再做 Adam”等同：逐坐标二阶状态是非线性的。报告中要包含投影矩阵、分解与刷新时间，才能判断状态节省是否值得。

ZeRO[19] 在设备之间分片优化器状态、梯度或参数。它改变每张卡持有的状态，而非更新的参数集合。通信、临时聚合和负载不均会带来峰值，不能把完整显存简单除以设备数。只有单设备时，CPU 卸载是以带宽换设备容量，并未减少整个机器的物理状态总量。

5.5 按瓶颈选择，并留下反例

若现有模型已具备所需知识，先建立原模型的任务与部署基线，再以小规模 LoRA 检查数据管线和适配是否有效；权重存储仍超预算时，评价量化基座，并单列它在未适配前造成的退化。若目标是改变词表、持续预训练或修复结构剪枝后的表示，低秩限制未必合适，应在可承受规模上保留全参数试验。LoRA 是基线，不能被当成任务能力上限。

考虑两个方案：A 只训练少量参数却保存长序列激活，B 更新更多参数但使用短微批与重算。A 的更新状态更少，峰值内存仍可能高于 B。这是账本允许的反例，不是硬件实测。应分别记录总参数、可训练参数、常驻权重、激活策略、峰值设备内存、每万有效词元耗时，以及合并和部署成本。若目标是更快训练，最终应比较达到同一验证质量所需的总时间；若目标是装入设备，首先检查容量门槛。

练习与提示

1. 取 $m = 4096, n = 1024, r = 32$，求 LoRA 新增参数比例与因子状态。提示：比例分母为 mn，状态按 12 字节/参数计，冻结权重另计。
2. 在“早期 LoRA– 冻结线性层 – 损失”的计算图上，写出冻结层输入梯度。提示：使用式 (5.2)；再判断 `no_grad` 会切断哪条路径。
3. 若两个微批分别有 8 与 32 个有效词元，其平均损失应怎样合并？提示：不能给两个微批相同权重。
4. 合并如何验收？提示：浮点产物核对 logits 容差；重新量化后测误差、质量、内存和延迟。

第 6 章

压缩改变什么：量化、剪枝与低秩分解

压缩是在给定任务与运行时上，寻找代价更低且误差可接受的模型表示。减少更新状态、删除模型维度、降低存储位宽和降低实际延迟是不同结果。上一章的 LoRA 保留原基座，只限制更新；本章考察部署表示本身怎样改变。我们沿着“表示、误差、实现、验证”展开：先说明数值误差怎样产生，再决定哪些误差值得修复，最后检查产物是否真的节省资源。

6.1 从一个量化器开始

均匀仿射量化将实数映射到有限整数集合，再恢复成近似实数：

$$q = \mathrm{clip}\left(\mathrm{round}(x/s) + z, q_{\min}, q_{\max}\right), \qquad \hat{x} = s(q - z), \quad s > 0. \tag{6.1}$$

本节取整数零点 z，s 是步长，$[q_{\min}, q_{\max}]$ 是编码范围。没有裁剪时，最近舍入使 $|x - \hat{x}| \le s/2$；发生饱和时，这个界不再成立。对称量化围绕零配置范围，非对称量化允许平移。零点、比例因子、舍入约定与累加精度都属于量化格式，不能只报告位宽。

例如用对称整数范围 $[-7, 7]$、步长 0.1 表示 $x = 0.26$，可得 $q = 3, \hat{x} = 0.3$；若输入为 1.2，被裁成 0.7，误差便远大于 0.05。这是手算，说明异常值裁剪与舍入是两种误差来源。增大范围能减少饱和，却会在相同编码数下加大步长；缩小范围则做相反折中，应该用目标分布选择。

按整个张量、输出通道、组或词元选择步长，可缩小局部动态范围，但增加比例因子、零点及布局元数据。若每 g 个权重共享一个 16-bit 比例因子，忽略边界时，仅这一项就增加 $16/g$ bit/权重。例如 4-bit 权重、$g = 128$ 时为 4.125 bit/权重，尚未计零点、打包对齐和高精度保留层。因此 $N/2$ 字节只是理想的 4-bit 权重编码数，不能直接当模型文件或运行内存大小。

所谓 W4A8KV8 还须说明哪些矩阵为 4-bit、激活是静态还是动态量化、KV 在何时压缩、计算与累加用什么精度。权重格式不变时，改变上下文长度可能让 KV 或工作区成为新瓶颈；文件变小也不保证峰值内存和延迟同步下降。

6.2 PTQ 与 QAT：何时引入量化误差

训练后量化（PTQ）从已有模型出发，选择量化格式及其参数。它可以只估计统计量，也可以局部优化缩放、裁剪、舍入或等价变换；不能以“完全没有梯度”定义它。量化感知训练（QAT）在训练或恢复训练的前向中引入目标量化，优化指定模型参数，使它们适应量化误差。两者之间存在使用局部重构训练的过渡方案，报告时写出实际更新对象、数据量和步数，比只贴标签更有用。

6.2.1 校准输出为什么比裸权重误差更贴近计算

对线性层 $W \in \mathbb{R}^{m\times n}$，令 $X \in \mathbb{R}^{n\times K}$ 的每列为一条校准激活。常用重构目标与带阻尼的二阶矩阵为

$$\min_{\widehat{W}\in\mathcal{Q}_b} \frac{1}{K}\left\|(W-\widehat{W})X\right\|_F^2, \qquad H_\lambda = \frac{2}{K}XX^\top + \lambda I, \quad \lambda > 0. \tag{6.2}$$

$\mathcal{Q}_b$ 是量化格式允许的解码矩阵集合。对单个输出行的重构误差，未加阻尼的 Hessian 正是 $2XX^\top/K$；阻尼缓解样本不足和相关通道造成的病态，但也改变了局部近似。

GPTQ[20] 借助这类二阶信息，在逐步量化时修正尚未固定的权重。直观上，一个权重的误差可由与它共同作用于校准激活的其他权重部分抵消。以列向量表示某一行的局部扰动 δw，若第 i 个分量被迫变成 ϵ_i，在二次模型下解

$$\min_{\delta w} \frac{1}{2}\delta w^\top H_\lambda \delta w \quad \text{使} \quad \delta w_i = \epsilon_i \tag{6.3}$$

会得到

$$\delta w = \frac{\epsilon_i}{(H_\lambda^{-1})_{ii}}(H_\lambda^{-1})_{:i}. \tag{6.4}$$

它来自拉格朗日乘子条件，展示相关方向的补偿为何出现。实际逐列算法还要约束已量化分量、分组和数值实现，不能直接把此式当成完整 GPTQ 伪代码。其目标仍是校准重构，不是直接保证最终任务正确率。

SmoothQuant[21] 的等价变换按本书的列向量约定写为

$$WX = (WS)(S^{-1}X), \qquad S = \mathrm{diag}(s_1,\dots,s_n), \quad s_i > 0. \tag{6.5}$$

增大 s_i 会缩小相应输入通道的激活，同时放大权重第 i 列。两端在浮点中保持同一映射，量化后则重新分配误差。部署时还需把变换融合进合适的相邻算子，或承担显式缩放开销；不能只改变激活而不补偿权重，也不能凭恒等式断言误差必然下降。

AWQ[22] 使用激活统计识别对输出更重要的权重通道，通过缩放改善低比特表示；它不等于“所有重要权重永久存为高精度”。这些方法处理的误差来源与内核要求不同，选择前应先确认目标设备支持的格式。

6.2.2 QAT 如何绕过舍入的导数

常见 QAT 保留浮点影子权重 w，前向使用量化再解码的 $\widehat{w} = Q_b(w)$。舍入几乎处处导数为零，直接使用其真实导数无法得到通常需要的更新。一种常见近似是直通估计器（STE）：对可表示区间 $[a,b]$，用

$$\frac{\partial \widehat{w}}{\partial w} \approx \mathbf{1}\{a < w < b\}, \qquad \frac{\partial \mathcal{L}}{\partial w} \approx \frac{\partial \mathcal{L}}{\partial \widehat{w}}\mathbf{1}\{a < w < b\}. \tag{6.6}$$

不同算法可对裁剪区间、缩放和零点采用不同近似；这些不是原离散目标的精确梯度。训练时仍可能保存浮点影子权重、梯度与优化器状态，所以部署位宽低并不意味着 QAT 的训练内存低。

前向模拟须尽量匹配部署的粒度、范围、舍入与激活精度；导出后若又换成另一种量化器，训练时适应的误差未必仍对应实际误差。EfficientQAT[23] 展示了量化感知恢复训练的具体路径，其成本和质量结论仍依赖模型、数据、位宽及实现条件。应把 PTQ 与 QAT 的校准、恢复训练、搜索和导出成本全部记入预算，不能无条件宣称后者更好。

Listing 6.1 模拟量化训练的概念流程；STE 由算子实现

```
for batch in training_data:
    q_weight = fake_quant(weight_fp, scales, zero_points)
    prediction = model_forward(batch.inputs, q_weight)
    loss = task_loss(prediction, batch.targets)
    optimizer.zero_grad()
    loss.backward()  # fake_quant uses a declared surrogate gradient
    optimizer.step()
freeze_quantization_config()
export_with_target_runtime_format()
check_exported_model_on_held_out_data()
```

校准数据不是测试数据

量化范围、逐层位宽和恢复训练步数都是模型选择。校准集要覆盖代表性的语言、长度、任务和异常激活，且记录来源；测试集只在方案冻结后使用。若用测试分数反复挑校准集或位宽，必须重新留出独立评测。还要测试分布变化，因为窄校准集上的低重构误差可能只是局部拟合。

6.3 低秩补偿：应该保留哪些误差方向

令量化残差 $E = W - \widehat{W}$，则

$$\|EX\|_F^2 = \operatorname{tr}\big(E(XX^\top)E^\top\big). \tag{6.7}$$

裸权重误差 $\|E\|_F$ 对所有输入方向赋相同权重，输出误差则强调激活实际访问的方向。校准样本未覆盖的方向可能几乎不受约束。这解释了为何量化误差的谱本身不够：还需看输入分布怎样加权这些方向，以及后续网络对误差怎样响应。

若在量化基座上增加秩不超过 r 的补偿 $R = BA$，目标可写为

$$\min_{\operatorname{rank}(R)\le r} \|(E-R)X\|_F^2. \tag{6.8}$$

当 $C = XX^\top$ 正定时，$C^{1/2}$ 可逆，且

$$\|(E-R)X\|_F^2 = \|(E-R)C^{1/2}\|_F^2. \tag{6.9}$$

记 $[F]_r$ 为矩阵 F 的秩-r 截断 SVD，则在没有其他结构约束时，一个最优解为

$$R_* = [EC^{1/2}]_r C^{-1/2}. \tag{6.10}$$

原因是右乘可逆矩阵保持秩，问题可变为对 $EC^{1/2}$ 的普通最优低秩逼近。若 C 奇异，未观测方向无法由此确定；改用 $C + \lambda I$ 会得到带权重残差正则的另一问题，不能假称仍是原校准目标的同一最优解。

小残差方向也可能值得优先修复

取 $E = \operatorname{diag}(3, 1)$、$C = \operatorname{diag}(1, 100)$，只能补偿一个方向。按裸残差 SVD，保留幅度为 3 的第一方向，剩余输出误差为 100；按激活加权目标，保留第二方向，剩余误差为 9。这是二维手算，说明“残差最大的方向”与“输出最敏感的方向”可以不同；它不证明某个真实网络的任务性能。

式 (6.10) 是单层线性重构的参照，不是全模型能力恢复定理。训练适配器时还存在优化误差、跨层误差传播和数据偏差；推理时独立低秩分支也有开销。比较不同秩，既要测校准重构，也要测独立任务、跨域输入和生成稳定性，并计入因子存储与运行时成本。

6.4 剪枝：零值不等于结构消失

非结构化剪枝把单个权重置零；若仍按原形状存为稠密张量，文件与矩阵乘开销通常都不改变。采用通用稀疏格式还要存位置。例如只看数值与索引，16-bit 值配 32-bit 索引时，每个非零需 48 bit；在 50% 非零率下，平均每个原位置已需 24 bit，尚未计行指针，反而多于原始 16-bit 稠密存储。这只是一个存储算例，专用位图和半结构化格式的开销不同。

半结构化模式如 2:4 对分组内非零数量施加约束，只有满足硬件、布局、矩阵形状与精度要求的内核才能使用对应稀疏通路。结构化剪枝则删除通道、头、层或行列，直接改变稠密矩阵形状，但也未必按删除比例加速：对齐、算子融合、窄矩阵利用率与系统中其他开销都可能成为瓶颈。

SparseGPT[24] 以近似二阶局部重构进行一次性稀疏化；LLM-Pruner[25] 研究结构依赖下的剪枝与恢复；SliceGPT[26] 利用变换和切片改变网络宽度。三者施加的表示约束不同，不能用同一个“稀疏率”替代配置，更不能把论文中的数值直接当成本设备的加速比。

6.4.1 重要性分数是一种局部模型

删除标量权重 w_i 对应 $\delta w_i = -w_i$。泰勒展开给出

$$\Delta \mathcal{L} \approx -\frac{\partial \mathcal{L}}{\partial w_i} w_i + \frac{1}{2} w_i^2 H_{ii}, \tag{6.11}$$

其中仅写出单参数扰动，H_{ii} 是损失 Hessian 的对角项。一阶分数常取 $|w_i \partial \mathcal{L}/\partial w_i|$ 的样本平均，或采用其他统计；二阶分数则考虑曲率。它们都依赖数据与近似。驻点附近一阶梯度可能很小，删除重要参数仍有明显二阶代价；一次删除多个参数还会出现交叉项，故零梯度不等于无用，也不能把逐参数排名无条件外推到大规模联合删除。

结构依赖也要显式处理。例如前馈网络 $W_{\text{down}}\phi(W_{\text{up}}x)$ 中，删除一个中间神经元须同时删除 W_{up} 对应行与 W_{down} 对应列；门控前馈网络还需同步处理门控投影。删除注意力头要核对输出投影及共享 KV 头的关系，改变残差宽度则会牵连归一化与多层接口。只在单个矩阵上删行，可能连合法模型都得不到。

6.4.2 从局部算子收益到整模收益

若某部分原占端到端耗时比例 f，该部分加速为原来的 s 倍，其余不变且无新增开销，则整体加速比为

$$S_{\text{total}} = \frac{1}{(1-f) + f/s}. \tag{6.12}$$

例如 $f = 0.6, s = 2$ 时，整体仅约为 1.43 倍。这是理想计时模型的手算；真实稀疏实现还可能增加索引、调度和布局转换成本。应在最终运行时分别测预填充与解码、目标 batch 和长度，并保存延迟分布。理论 FLOPs 或非零数下降，只能解释潜在收益。

6.5 低秩分解与层裁剪

对 $W \in \mathbb{R}^{m \times n}$，截断奇异值分解给出

$$W \approx U_r \Sigma_r V_r^{\top} = BA, \qquad N_{\text{factor}} = r(m+n). \tag{6.13}$$

在权重 Frobenius 范数下，秩-r 截断 SVD 是最优逼近，其平方误差为被舍弃奇异值平方之和。对激活加权误差或任务损失，不能直接沿用这一保证。与 LoRA 不同，这里用 BA 替换原矩阵，而非保留 W_0 再增加分支。

只有 $r < mn/(m+n)$，因子参数数目才少于原矩阵。在忽略 bias 与调度的单词元矩阵向量乘中，乘加数也从 mn 变为 $r(m+n)$；但额外中间张量、两个 kernel 启动及窄矩阵利用率可能抵消节省。例如 $m = n = 1024, r = 600$ 虽然小于满秩，因子却有 1 228 800 个参数，多于原矩阵的 1 048 576 个。因此先检查实际存储和目标内核，再选择秩。

层裁剪减少串行深度，但残差流、归一化和跨层分工会一起变化。不存在脱离模型和任务的“中间层总是最不重要”规则。可先在验证集逐层消融，定义依赖组，逐步删除并用固定预算恢复训练。最后除平均任务分外，还检查长上下文、遗忘、校准、输出格式与失败样本；训练后测量的最终模型才是部署候选。

6.6 把压缩顺序当作实验变量

先适配后量化，保留浮点适配的自由度，却在部署时引入新误差；先量化后适配，让增量适应量化基座，但可能限制修复空间；结构剪枝后蒸馏，可用教师恢复输出，但需计入教师与恢复训练成本。联合处理这些问题值得研究，但没有适用于所有预算的固定最优顺序。应从同一原模型出发，固定数据切分与可比较的总预算，保留每个阶段的产物及测量。

一个可执行的压缩对照可按如下顺序推进：

1. 保存原模型、运行时与任务基线，明确容量门槛及可接受的质量损失；
2. 在训练或校准切分上确定格式、范围、秩或剪枝结构，保留模型选择记录；
3. 导出最终格式，在目标运行时验证输入输出、算子支持与实际存储；
4. 方案冻结后测独立任务、峰值内存及真实延迟，必要时回到前一步修复再重新冻结。

最小报告应包含原精度、目标位宽或结构、组大小、范围、校准来源、恢复训练、文件字节数、设备与运行时、预热及重复次数、输入输出长度、峰值内存与延迟分布。附录 B 用本地小网络完成一次部署对照，检验“文件变小是否更快”；它不证明任何 LLM 量化方案在其他设备上的加速比。权重打包、内核与服务系统的进一步细节分别参阅《智能体模型量化压缩与部署》《Kernel 与 CUDA 编程》与《大模型推理加速与 AI 基础设施》。

练习与提示

1. 在式 (6.5) 中令 S 某个对角元为零会发生什么？提示：先检查可逆性，再谈量化收益。
2. 改变二维残差算例的 C，求何时修复第一方向更好。提示：比较两个剩余加权平方误差，裸残差谱保持不变。
3. 对一个 2048×1024 矩阵，求可减少因子参数的最大整数秩。提示：使用严格不等式 $r < mn/(m+n)$。
4. 设计“稀疏率更高但模型更慢”的对照。提示：固定任务质量、设备和输入，分别记录存储格式、内核、布局转换与端到端延迟，不能只对比非零数。

第 7 章

架构与系统：把复杂度还原成成本

部署预算不仅在训练结束后筛选模型，也会反过来决定训练对象。权重放得下却无法容纳缓存，或参数减少后内核利用率下降，都可能让一条统计上合理的路线失效。分析时应区分算术量、读写量、常驻状态和串行步骤，再用实际运行时验证。

7.1 注意力的计算与存储不是同一件事

对单层因果自注意力，忽略批量与多头的重排，可写为

$$\mathrm{Attn}(Q,K,V)=\mathrm{softmax}\left(QK^{\top}/\sqrt{d_h}+M\right)V. \tag{7.1}$$

其中 Q,K,V 是查询、键和值，d_h 为单头维度；掩码 M 在可访问位置为零，在未来位置为负无穷。长度为 T、总查询维度为 d 时，注意力配对计算约随 T^2d 增长。朴素实现还显式存储每个头的 $T\times T$ 分数矩阵，但这不等于任何正确实现都必须保存完整矩阵。FlashAttention[27, 28] 通过分块、在线归一化及重算减少高带宽内存读写，同时保持所定义注意力的结果，仅有浮点舍入差异。

因此“注意力是二次的”需要补一句：二次的是标准全注意力的配对算术量；中间张量的显存峰值还取决于实现。高效注意力不会消除所有训练激活，也不会自动消除自回归推理的历史 KV 状态。短序列时投影、前馈层和启动开销可能更重要，不能只看 T^2 项。

7.2 SSM 与混合架构改变了状态的形状

状态空间模型（SSM）用递归状态汇总历史。简化的一层为

$$h_t=\overline{A}_t h_{t-1}+\overline{B}_t x_t,\qquad y_t=C_t h_t+D_t x_t. \tag{7.2}$$

这里 h_t 是递归状态，x_t 是当前输入，$\overline{A}_t,\overline{B}_t,C_t$ 控制状态更新与读出，$D_t x_t$ 表示直接通路。Mamba[29] 引入输入依赖的选择机制并配合硬件实现；Mamba-2[30] 研究结构化状态空间与特定注意力形式之间的对偶。式 (7.2) 是理解状态更新的示意，不包含它们全部离散化、分组和门控细节。

递归推理状态可在固定架构下不随历史长度线性增长，但有限状态并非无损保存任意长输入。任务需要精确检索远处细节时，状态容量与压缩方式会成为限制。并行训练使用扫描或分块，实际吞吐仍受内核、数据类型和硬件影响；“线性复杂度”不保证任何输入长度下都快。

混合模型让部分层或头使用注意力，其他部分使用状态空间更新，Hymba[31] 是一种具体设计。它仍有注意力缓存，此外还要持有递归状态。公平比较应固定训练数据、累计处理 token、词表、参数预算和目标设备，并披露训练计算差异，再比较检索、长文建模及真实延迟；来自不同训练语料的两个模型不能证明纯架构优劣。

7.3 KV cache 的完整账本

对每层具有相同 KV 头配置、每个请求保留 T 个 token 的批量，未压缩缓存载荷为

$$M_{\mathrm{KV}} = 2bLTH_{\mathrm{kv}}d_h s_{\mathrm{KV}}. \tag{7.3}$$

系数 2 对应键和值；b 是批量中同时保留的请求数，L 是层数，H_{kv} 是每层的 KV 头数，d_h 是每头维度，s_{KV} 是每元素字节数。T 包括已经缓存的提示和生成 token，不等于提示长度上限。这里不含分页余量、比例因子、索引、对齐及分配器开销。动态请求应把 bT 换成同一时刻各请求保留长度之和，而不是用请求平均长度替代峰值；不同层的窗口或头数不同时还需逐层求和。

手算：权重量化不改变这个 KV 项

取 $b = 2, L = 24, T = 4096, H_{\mathrm{kv}} = 4, d_h = 64, s_{\mathrm{KV}} = 2$，则缓存载荷为 201 326 592 B，即 192 MiB。若 KV 头增至 16，该项变为 768 MiB；上下文减半或批量减半，则各减半。把权重从 16-bit 改为 4-bit，不会自动把 KV 从 2 B 改成 0.5 B。若真的量化 KV，还需另计比例因子、对齐以及压缩和读取成本。

多头注意力的 KV 头与查询头相同；分组查询注意力（grouped-query attention, GQA）让一组查询头共享一个 KV 头，多查询注意力（multi-query attention, MQA）让全部查询头共享一组键和值。它们降低缓存和读写，但改变结构与训练条件。对已有检查点，不能把更改头数当成无损运行时开关。滑窗只保留窗口状态，压缩或淘汰缓存则改变历史信息的可访问方式；判断能力代价时要用跨窗口依赖任务，而不只是短句准确率。

服务容量还需计算权重、KV、当前算子激活、临时工作区、递归状态和运行时余量。只报告权重文件大小，既不能确定峰值内存，也不能推断可接纳的并发。缓存共享、前缀复用和分页可以改变具体分配，应注明共享是否真正发生，以及基准是否把共享命中率固定住。

训练显存与 KV 不是同一账本。标准 teacher-forced 训练常保存反向所需激活，而不是照推理服务的持久 KV 账本累加。第 2 章的参数状态估计和本节的缓存估计分别回答“能否训练”和“能否服务”。

7.4 预填充与解码为何有不同瓶颈

预填充一次处理输入提示，能形成较大的矩阵乘；逐 token 解码在每一步读权重与缓存，低 batch 时容易受带宽限制。简化 Roofline 下界为

$$t \gtrsim \max\left(\frac{F}{P_{\mathrm{eff}}}, \frac{B_{\mathrm{move}}}{\mathrm{BW}_{\mathrm{eff}}}\right), \tag{7.4}$$

F 为 FLOPs，B_{move} 为移动字节数，有效算力和带宽都不等于硬件宣传峰值。该式忽略启动、同步、依赖、排队及调度，适合解释下界，不能当作最终延迟预测器。

首 token 延迟（time to first token, TTFT）从声明的请求起点计到首个输出 token；每输出 token 时间（time per output token, TPOT）在本书中指首 token 之后的平均生成间隔。以首个输出 token 已包含在 TTFT 的约定，对 $n_{\mathrm{out}} \geq 2$，从同一起点到最后一个输出 token 的完成延迟为

$$t_{\mathrm{complete}} = t_{\mathrm{TTFT}} + (n_{\mathrm{out}} - 1)t_{\mathrm{TPOT}}. \tag{7.5}$$

这里 t_{TPOT} 是该请求实际观测间隔的平均值，因而该分解是计时恒等式；用一次短试跑的平均 TPOT 预测其他长度时才是近似。TTFT 是否包含网络、排队与分词需单独声明；后处理与流关闭的额外时间另计端到端完成时间，不加入此处的 t_{complete}。$n_{\mathrm{out}} = 1$ 时只有 TTFT，没有后续间隔，不能除以

$n_{\text{out}} - 1$ 计算 TPOT；$n_{\text{out}} = 0$ 则是失败、取消或无输出请求，另行计数。以文本形式生成的拒答仍有输出 token，不能因为语义上拒答就记成零输出。

7.5 怎样选择架构或运行时

优先把目标工作负载写成长度分布、并发、批量、精度和质量门槛。先使用被目标运行时完整支持的基线，再考察 GQA、局部注意力、SSM 或混合结构。没有对应量化内核、算子图出现回退、权重反复解码或线程设置不匹配，都可能使理论收益消失。

在本系列中，内核实现与在线调度另有分册。本书要求的证据粒度是：给定模型版本和训练配方，导出固定产物，在目标设备上分别测预填充、解码、峰值内存和持续运行。若没有移动设备或 GPU，交付明确的复现方案，不以桌面 CPU 分类实验推断手机生成速度。

练习与提示

1. 若 batch 内四个请求保留长度分别为 512、1024、2048、4096，式 (7.3) 应如何修改？提示：用长度总和代替 bT。
2. 给一个“FLOPs 下降、延迟上升”的可能场景。提示：两次窄矩阵乘替代一次大矩阵乘，或引入不受支持的算子回退。
3. 只有权重文件大小与总 tokens/s 两项数据，能否判断 TTFT 改善？说明还缺哪些计时边界。

第 8 章

从语言建模到可靠行为

预训练优化下一 token 预测，不自动定义如何遵循指令、选择工具或停止生成。行为适配使用更明确的输入输出与偏好信号，仍受小模型容量、数据覆盖和推理预算限制。本章保留理解这条路径所必需的目标与边界；细致的 LoRA、QLoRA 和偏好训练实践由《现代小模型微调》继续展开。

8.1 继续预训练、SFT 与偏好不是同一目标

继续预训练用领域文本改变预测分布；监督微调（supervised fine-tuning, SFT）以示范回答训练条件策略；偏好优化让模型在候选回答中更倾向被选中的一方。三者可以串联，但顺序、数据比例和损失掩码必须记录。领域原文、指令示范与偏好对提供的是不同监督，不应仅因为数据量少就选择其中某一阶段。

对提示 x 与回答 y，将部署所需的终止符包含在 y 内，padding 不计损失。仅回答部分计损失的 SFT 为

$$\mathcal{L}_{\mathrm{SFT}} = -\mathbb{E}_{(x,y)} \sum_{t=1}^{|y|} \log \pi_\theta(y_t \mid x, y_{<t}), \tag{8.1}$$

这里先对每个回答求和，再对样本取期望，长回答因而贡献更多 token 项。若需要每个回答等权，可在和式外除以 $|y|$；若需要整个批次的 token 平均，应以批内监督 token 总数为分母，两者不能混用。若把系统提示和用户文本也计入，训练目标已改变；若聊天模板、终止符或工具消息格式与部署不一致，即使训练损失正常，也可能循环输出或泄漏角色标记。

建立 SFT 基线时，先用少量样本做过拟合检查，确认答案 mask、分词、位置和停止条件；再用真实验证集选择步数。过拟合检查只排查管线，不是模型能力结果。格式正确率、任务正确率和安全拒答质量应分别记录。

8.2 DPO：从偏好对优化概率比

直接偏好优化（direct preference optimization, DPO）把偏好学习写成一个概率比目标。给定同一提示的偏好对 (x, y^+, y^-) 与冻结参考策略 π_{ref}，定义

$$a_\theta(x, y) = \log \pi_\theta(y \mid x) - \log \pi_{\mathrm{ref}}(y \mid x), \tag{8.2}$$

$$\mathcal{L}_{\mathrm{DPO}} = -\mathbb{E} \log \sigma\left(\beta \left[a_\theta(x, y^+) - a_\theta(x, y^-)\right]\right), \tag{8.3}$$

其中 $\beta > 0$，σ 是 sigmoid 函数，序列对数概率由回答 token 的对数概率求和。DPO[32] 在相应偏好模型假设下，从带 KL 约束的奖励最大化推导这一分类式目标，避免在该训练环节单独运行近端策略优化（proximal policy optimization, PPO）；并不保证任意离线数据上的训练都稳定或都优于强化学习。参考策略需要对训练回答具有相应支持；常规实现使用同一 tokenizer 和聊天格式计算两套对数概率。

参考策略、长度和难度分布会影响概率比。随意将求和改成按长度平均，便不再是原式，需要单独说明动机和验证。若偏好主要奖励长回答，模型可能变得冗长；若偏好来自不可靠评审，学生会复制其判断偏差。小模型还可能因有限容量在偏好遵循与事实性之间产生权衡。

一个不需要训练的梯度检查

若偏好差值为零，单对损失为 $\log 2$；增大正确方向的差值，损失应下降。交换偏好标签，应改变梯度方向。实现时分别检查策略与参考模型的回答掩码，冻结参考参数；若 tokenizer 不同，应先解决共同事件空间与序列边界问题，不能直接逐 token 相减。

8.3 推理过程监督与测试时计算

自然语言推理轨迹常称思维链（chain of thought, CoT），可以提供中间结构，也可能包含冗余、错误或事后解释。第 4 章已区分答案蒸馏、过程蒸馏与学生分布采样。DeepSeek-R1[33] 的报告提供了强化学习和向较小模型迁移的具体实例；不能由此推出任意小模型只要延长思考便会变强。

可执行程序、代数等价检查和单元测试能验证部分结果，但验证器并不总能检查过程，且可能存在漏洞。使用拒绝采样时，要同时报告生成候选数、接受率、每题教师或工具调用数；使用强化学习（reinforcement learning, RL）时还要报告探索、奖励建模和更新成本。一次学生训练中没调用教师，不表示前置教师数据是免费的。把验证器用于挑选测试答案还改变了最终系统，需与单次学生生成分别报告。

以正确率 – 生成 token– 延迟三条曲线评价推理比构造一个混合单位的“推理效率比值”更透明。如果确需单一效用，先将 token、时间、金钱等转换成同一成本或进行明确归一化，并说明权重如何来自场景约束。错误回答的长解释不会因为 token 多而获得能力加分。

8.4 停止、拒答与工具边界

工具调用模型至少要学会选择工具、生成合法参数、读取工具结果以及停止。把无工具时的猜测与工具检索成功混在一个分数里，会掩盖系统依赖。评测应分别包括工具不可用、结果为空、响应冲突和输入分布外等情况。

拒答率也不是单调越高越好。记录有害请求上的适当拒答与正常请求上的误拒，并单独检查隐私、越权和事实性。训练目标只是行为塑造的一部分；部署端权限、输入边界和输出校验同样需要测试。对低置信度输入转人工或调用更大模型，是预算和可靠性之间的一种可评估路线。

练习与提示

设计“短过程 SFT、长过程 SFT、仅答案 SFT”三组比较，固定训练 token 与学生更新预算；再用同一最大输出长度评测。思考：如果只固定样本条数，长过程组额外获得了什么？若验证器只看最后数字，又会遗漏什么？

第 9 章

小型多模态与端侧工作负载

模型能在文本任务上满足预算，不代表接入相机或麦克风后仍然可用。图像编码、音频流、连接器、视觉 token 和后处理共同决定系统成本。训练设计要围绕真实输入链路，而不是只计算语言骨干的参数。

9.1 编码器、桥接器与语言骨干

典型视觉语言模型（vision-language model, VLM）把图像 I 经编码器 g_v 转成特征，再由连接器 P 对齐到语言骨干的输入空间：

$$v = P(g_v(I)), \qquad p(y \mid x, I) = \prod_t p_\theta(y_t \mid x, v, y_{<t}). \tag{9.1}$$

连接器可以是线性层、多层感知机或重采样模块。输出维度相同只保证张量能连接，不保证视觉语义、位置关系与语言表示已经对齐。MobileVLM[34] 提供移动场景下的具体设计；不同系统中冻结和训练哪些组件需分别核对，不能概括为“只训练一个连接器就能得到所有视觉能力”。

一种易于分析的训练顺序是：先固定两个预训练骨干建立桥接基线，再比较解冻少量层或全部参数。第一阶段测特征是否可被语言侧利用，后续阶段测联合更新的收益与遗忘。每一步保持图像分辨率、token 数、样本与训练预算可追踪。

9.2 视觉 token 为什么会扩大文本成本

若图像使用边长为 p 的非重叠 patch，输入高宽为 H, W，未重采样前 token 数约为 $\lceil H/p \rceil \lceil W/p \rceil$，还可能有特殊 token。把长宽各翻倍，会使 patch 数约增为四倍；全注意力的视觉 – 视觉配对部分可能约增为十六倍。实际系统常使用多裁剪、动态分辨率或局部编码，因此这个式子是特定简化布局下的估算。

若每帧压缩后保留 n_v 个 token，视频取 f 帧，语言侧输入约增加 fn_v 个 token；这会增加预填充工作与缓存。音频特征的步长、流式分块和保留历史也会形成同类账本，不能把“每秒输入”直接当成文本 token。

池化或丢弃视觉 token 可以减小输入，却可能删除小字、计数对象和精细空间关系。把分类平均分保持住并不表示光学字符识别（optical character recognition, OCR）与定位无损。实验应分别测小目标、文档布局、计数、空间关系与低光照，并保留预处理和裁剪参数。

例如一个**教学输入账本**含 512 个文本 token、8 帧图像、每帧 64 个桥接 token，则语言侧在生成前处理约 $512 + 8 \times 64 = 1024$ 个 token。若改为每帧 256 个，则变为 2560 个。这个差异会传入预填充与 KV 账本，但编码器若仍逐帧处理同样分辨率，其耗时不会随桥接 token 同比例缩小。总内存还应计入编码器权重、图像或音频缓冲、桥接激活以及同时驻留的语言权重。

传感器链路还需要明确时间语义。音频的采样率以 Hz 计，视频帧率以帧/秒计，编码器再把一

表 9.1 多模态成本应按输入链路分项测量。

环节	控制变量	典型失败
采集与预处理	分辨率、帧率、裁剪、音频采样率	图像内容被裁掉，预处理耗时被漏计
编码与桥接	编码器版本、精度、输出 token 数	桥接瓶颈或不支持的算子回退
语言生成	提示、上下文、输出上限、缓存精度	视觉 token 占满上下文，长输出拖慢响应
后处理与动作	解析器、规则、工具权限	文本看似正确却产生非法坐标或动作

个时间窗口转换为若干特征 token；这几种计数不能互换。流式系统应记录窗口长度、重叠、保留历史、丢帧策略与传感器时间戳。仅测语言模型从输入张量到输出的时间，会遗漏等待窗口填满和输入队列积压造成的延迟。

9.3 端侧训练必须考虑更新后的生命周期

离线运行可以减少原始数据外传，但日志、缓存、模型记忆和适配器仍可能泄漏信息。许可与来源应覆盖图像、文字、音频以及标注，不能仅为语言语料建数据卡。

设备更新还面对存储、供电、网络和回滚约束。发布一个新适配器时，应记录基座版本、量化格式、签名或哈希、兼容运行时及可回退版本。持续训练的新任务增益应与旧任务保持、误拒、校准和更新失败后的恢复一起测试。一次成功的实验室更新不能证明长时间个性化学习安全。

9.4 热、能耗与异构硬件

中央处理器（CPU）、图形处理器（GPU）与神经网络处理器（NPU）对位宽、形状和算子支持不同。NPU 名义上支持 INT8，并不表示整张模型图都能在其上运行；图中一个回退节点就可能增加数据搬运与同步。持续运行会触发温控调频，短预热后的最快一次值不足以代表用户体验。

有可用功率测量时，报告采样来源及频率，分别给出整机能量与扣除空闲基线后的估计：

$$E_{\text{total}} = \int_0^{t_{\text{end}}} P(t)\,dt, \qquad E_{\text{incremental}} = \int_0^{t_{\text{end}}} [P(t) - P_{\text{idle}}]\,dt. \tag{9.2}$$

焦耳是能量单位，瓦是功率单位。平均功率较低但完成时间更长，能量仍可能更高。空闲基线需在相同设备设置与相近热状态下测量，后台任务和低频采样都会带来误差；增量能量是带有这些假设的估计。没有可靠功率数据就标记“未测”，不可由延迟直接换算能耗。

练习与提示

一个视频系统将每帧从 256 个视觉 token 压到 64 个。设计固定视频内容与输出预算的对照，同时记录编码耗时、TTFT、缓存、OCR 与计数。提示：编码器本身若未改变，视觉 token 减少不会按比例降低所有环节耗时。

第 10 章

评测：把分数变成可复现证据

小模型项目最容易在评测阶段失去可解释性：数据、提示模板、推理长度、量化格式和硬件同时变化，最后只剩一个平均分。本章把评测拆成能力、可靠性和系统成本三条证据线。基准分数只能回答它所覆盖的任务，不能替代真实工作负载的回归测试。

10.1 四层评测框架

表 10.1 从模型行为到设备成本的四层评测。

层次	可记录的指标	必须固定或披露的条件
基础能力	留出损失、PPL、知识与语言任务	tokenizer、示范数、提示模板、污染检查、测试版本
任务与推理	数学、代码、事实性、长上下文、校准	采样温度、生成上限、停止规则、验证器
系统效率	文件大小、峰值内存、TTFT、TPOT、吞吐、能耗	设备、运行时、内核、线程、批量、输入输出长度
风险与可靠性	适当拒答、误拒、隐私、偏见、分布外失败率	攻击预算、人工或自动评审协议、样本版本

大规模多任务语言理解评测（Massive Multitask Language Understanding, MMLU）采用多学科选择题，GSM8K 采用数学文字题；语言模型整体评测（Holistic Evaluation of Language Models, HELM）强调多维度，而基于模型评审的对话比较还涉及评审偏差。这些评测覆盖的能力与误差来源不同 [35, 36, 37, 38]。压缩工作还应在同一产物上记录质量与系统代价。引用一个分数时，明确它是论文报告、独立复现还是本次实测；仅列论文名称不等于已经统一评测条件。

评测切分至少要区分训练集、用于选超参数的验证集、一次性使用的测试集，以及用于检查领域转移的分布外集合（out-of-distribution, OOD）。测试数据去重和来源审计只能减少已知泄漏，不能证明模型从未见过公开题目。对无法审计训练语料的基座，应同时使用新构造的私有留出题与真实任务案例，并说明其覆盖局限。

10.2 先逐任务报告，再考虑汇总

准确率、负对数似然、延迟和拒答率既有不同单位，也有不同优化方向，不能直接求平均、最小值或方差。即使都写成百分数，知识正确率和有害请求拒答率也不必具有同一业务含义。优先展示每个任务的原始指标、样本数和区间，并列出必须独立通过的质量门槛。

确需总分时，先在实验前定义方向一致的无量纲映射 $z_k = g_k(a_k)$ 与权重 $w_k \geq 0, \sum_k w_k = 1$，再计算 $\sum_k w_k z_k$。映射可以由场景的最低可接受值与目标值确定，不能看过结果后调整。只有在这些

分数可比较时，才有理由附上

$$\bar{z} = \frac{1}{K}\sum_{k=1}^{K} z_k, \qquad z_{\min} = \min_k z_k, \qquad v_z = \frac{1}{K}\sum_k (z_k - \bar{z})^2. \tag{10.1}$$

v_z 描述所选任务之间的离散程度，不是运行随机性或总体估计误差；不同任务上的分数通常也不是独立同分布样本。平均值不能抵消安全或关键任务未过门槛的事实。

对同一任务，压缩前后的变化可写为

$$\Delta a_k = a_{k,\text{comp}} - a_{k,\text{ref}}, \qquad r_t = \frac{t_{\text{comp}}}{t_{\text{ref}}}, \qquad r_m = \frac{M_{\text{comp}}}{M_{\text{ref}}}. \tag{10.2}$$

若 a_k 是准确率，Δa_k 应注明是百分点差；若是损失，改善方向相反。r_t, r_m 在分母为正且测量范围一致时才有意义，小于 1 表示对应成本下降。与其宣称一个“总体提升”，更可核查的结论是：哪些任务变化在容许范围内，哪些资源约束确实通过。

10.3 校准、失败样例和统计不确定性

分类或选择任务有明确定义的类别概率时，可同时报告负对数似然（negative log-likelihood, NLL）与 Brier 分数。设第 i 题有 C 类，预测概率为 p_{ic}，真值为 y_i，则一种常用多分类 Brier 定义是

$$\text{Brier} = \frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C}\left(p_{ic} - \mathbf{1}[y_i = c]\right)^2. \tag{10.3}$$

是否再除以类别数会改变数值，报告时需固定。NLL 与 Brier 同时受正确性和概率质量影响，不能仅凭其中一个下降就断言各置信度区间都已校准。

令 $c_i = \max_c p_{ic}$，按事先给定的置信度区间分成非空集合 B_j，期望校准误差（expected calibration error, ECE）的常见估计为

$$\text{ECE} = \sum_j \frac{|B_j|}{n}\left|\text{acc}(B_j) - \text{conf}(B_j)\right|. \tag{10.4}$$

其中前者是箱内正确率，后者是平均置信度。ECE 依赖分箱和样本数，应连同可靠性图与每箱数量报告。生成答案的口头“有九成把握”或某个 token 的最大概率，并不自动成为整份答案正确的概率；必须另行定义置信度提取与答案判定协议。

对固定题集比较两个模型，保留逐题对应结果，使用配对分析；例如对样本对做 bootstrap 重采样，估计准确率差或延迟差的区间。题目若按用户、文档或对话成组，应按组重采样，避免把相关片段当作独立样本。对于低失败率，零次观测失败也不证明风险为零，需报告样本数及适当的二项区间。随机种子之间的训练差异与固定模型在有限测试集上的抽样误差是两层不确定性，不能只报一个混合方差。

资源允许时重复不同初始化或数据顺序，并预先说明如何汇总；资源不足时将结果标成单次观测。模型、步数和阈值在验证集上确定后才使用测试集。对自动评审，保存原始输出、评审提示、模型版本、顺序与拒答结果；对人工评审，记录抽样、盲评和一致性。失败样例至少区分答错、格式失败、超时、工具错误、误拒与适当拒答。

10.4 遗忘与鲁棒性需要各自的对照

继续预训练或任务适配后，在同一通用留出集上重测原模型与新模型，逐项报告遗忘，而不只报告新领域增益。新任务和旧任务的分数没有共同尺度时，不应直接相减成一个“净收益”。鲁棒性测试可给固定题目加入拼写扰动、格式变化或无关上下文；必须保持应有答案不变，避免把题意改变也算成模型不稳。

分布外评测至少明确变化轴，如新来源、新时间段、新语言或新设备输入。安全测试同时包括应拒请求与正常请求，分别记录适当拒答和误拒；隐私测试还要声明攻击者可见信息、查询次数和成功判据。以上测试给出指定威胁模型下的证据，不构成对任意输入的保证。

10.5 真实延迟的测量协议

沿用第 7 章的计时边界：首 token 已包含在 TTFT 内，t_{complete} 从同一起点计到最后一个输出 token。对一个输出 $n_{\text{out}} \geq 2$ 个 token 的请求，有

$$t_{\text{complete}} = t_{\text{TTFT}} + (n_{\text{out}} - 1)t_{\text{TPOT}}, \tag{10.5}$$

$$t_{\text{TPOT}} = \frac{t_{\text{complete}} - t_{\text{TTFT}}}{n_{\text{out}} - 1}. \tag{10.6}$$

单 token 请求只有 TTFT，TPOT 记为不适用；零输出请求与超时另计失败。分块流式接口若一次返回多个 token，应说明测得的是块到达间隔还是模型内部 token 时间。对全部请求先求中位数再套入上式，通常不再满足同一恒等关系；应先逐请求计算，再汇总各指标。

最小测量顺序是：固定设备与供电状态，加载固定产物；按预定次数预热；在相同输入集上交替测原精度与压缩产物；每次保存输入长度、实际输出长度、TTFT、完成时间与失败状态。固定最大生成长度并不保证实际生成长度相同，还应给固定长度性能测试与正常停止的任务测试分别标注目的。异步 GPU 计时需在声明的边界同步或使用设备事件，避免只测到任务入队。

记录运行时、内核、线程、批量、并发、预热和重复次数，分别报告中位数与尾延迟。端到端计时是否包含模型加载、分词、排队、网络与后处理，要逐项写明；若计入最后 token 之后的后处理或流关闭，应另报该时间，不套用上面的 token 间隔恒等式。冷启动、稳态单请求和并发负载是不同场景；总生成 token 除以服务区间得到的吞吐，也不等于单个用户的 TPOT。

内存至少区分文件字节、进程峰值、设备分配峰值与设备保留峰值；采样式工具可能漏掉短时峰值。若只测到 CPU 内存，不能将其称为 GPU 显存。能耗沿用第 9 章的积分边界，注明整机或器件、空闲基线与仪器。没有相应设备或测量能力的项目写“未测”，并保存复现方案。

一份可复查的结果记录

模型与检查点哈希；数据版本、切分和污染检查；训练配置、预算、种子及选择规则；压缩格式、组大小与校准集；设备、运行时和计时边界；逐样本结果、区间计算方法、失败样例与原始日志路径。结论应能沿这些字段回到一次具体运行，而不是只有一张截图。

10.6 练习与提示

1. 一个模型的准确率为 80%，平均延迟为 2 秒。为何两者的平均数没有解释？**提示：**先确定约束和效用映射，保留原单位。
2. 某请求 TTFT 为 0.2 秒，生成 5 个 token，后续平均间隔为 0.05 秒，完成时间是多少？**提示：**为 0.4 秒；首 token 不再计一次解码间隔。
3. 两个模型在同一 100 题上分别答对 80 题和 82 题。只凭这两个总数能否判断变化稳定？**提示：**还需逐题对应、题目相关性和区间；也要区分有限题集误差与训练种子差异。

第 11 章

统一训练配方：从预算到可部署模型

本章用约束、基线和对照组织可追踪流程。数据量、推理预算与教师强度应分别比较；无需训练、压缩不值得或预算不足，也都是有效结论。

11.1 先写约束，再选算法

表 11.1 候选路径由已测瓶颈决定，不表示算法优先级。

观测到的约束	可比较的干预	采用前的验证
训练峰值超限	减微批、重计算、PEFT、低精度状态或卸载	哪一项占内存；总训练时间是否仍可接受
权重与运行状态装不下	运行时支持的 PTQ、QAT、结构压缩或更小基座	权重、缓存、工作区和质量是否同时过门槛
解码延迟超限	量化内核、缓存策略、线程或批量设置；必要时换架构	先判明带宽、算术或调度瓶颈，再测尾延迟
领域能力不足	提示或检索基线、监督适配、继续预训练、蒸馏	缺知识还是缺行为；数据质量、许可和遗忘
推理任务失败	答案或过程监督、验证器、更多测试时计算	预算增加带来的收益、错误类型和教师成本
多租户或多任务	适配器、前缀、模块化路由	切换、并发、合并误差和隔离性

从可行域反推规模：几百兆字节的设备未必适合十亿参数浮点基座，长上下文也可能让缓存超限。候选需满足质量、许可与算子支持。原精度无法在目标设备运行时，可在其他设备保留质量参照，但目标设备的原精度性能须标为未测，不能虚构相对加速。

11.2 一条可执行的基线流水线

1. **写规格：**固定目标设备、内存、TTFT 与 TPOT 门槛、输入输出长度、并发、质量下限、风险及许可证要求。区分硬约束与可以权衡的偏好。
2. **选并测基线：**锁定模型、tokenizer 和聊天格式。先测未适配模型或最简单可用方案，保存原精度质量与目标设备系统记录；不要预设某个参数区间。
3. **版本化数据：**只建立任务所需的数据桶，保存来源、许可、切分、去重、污染检查和采样比例。校准、验证与测试职责分开。
4. **限定预算：**预先固定累计输入 token、监督 token、更新步数、搜索次数和种子。按第 2 章账本估算，再测完整步骤修正内存与时间。
5. **比较监督与更新：**用硬标签或纯 SFT 建立参照，再根据任务缺口选择一种蒸馏或适配干预。该

参照是一次可比较基线，不能称为任务上限。

6. **比较压缩：**先检查目标运行时可用格式，以同一原模型和独立校准集比较一种可行压缩；必要时才增加恢复训练或其他结构。顺序本身也可成为实验变量。
7. **接入目标运行时：**重新加载导出产物，测文件、峰值内存、TTFT、TPOT、吞吐与失败率；有测量能力时再记录能耗和热稳定性。
8. **冻结并回归：**方案只在验证集上选择，冻结后对能力、遗忘、分布外、风险及长上下文做测试，生成配置、哈希、日志与限制清单。

固定学生预算可比较监督方法；项目总成本还需加入教师生成、缓存、筛选与搜索。一次试跑通过只支持进入下一阶段，不能替代最终回归。

11.3 把一次运行写成可追踪记录

将一次运行视为一个有父节点的产物：它从哪个检查点、哪份数据和哪组配置出发，最后生成了什么。下面是**记录格式示例，非本书实测**；尖括号和空值必须由实际运行填入，预算字段须在开始前确定。

Listing 11.1 运行记录的最小结构；路径均相对于记录根目录。

```
{
  "run_id": "<unique-id>",
  "parent_run_id": null,
  "status": "planned",
  "source": {"code_commit": "<id>", "config": "config.json"},
  "model": {"checkpoint_sha256": "<hash>",
            "tokenizer_sha256": "<hash>"},
  "data": {"manifest": "data.json", "split_sha256": "<hash>"},
  "budget": {"input_tokens": null, "supervised_tokens": null,
             "updates": null, "search_trials": null},
  "training": {"seed": null, "precision": "<format>"},
  "environment": "environment.json",
  "command": "command.txt",
  "outputs": {"checkpoint": null, "metrics": null,
              "sample_results": null, "raw_log": null},
  "decision": {"gate": "<predeclared-rule>",
               "passed": null, "reason": null}
}
```

配置文件补充架构、损失归一化、优化器、学习率、微批、累积、训练或校准方法及停止规则；环境文件记录操作系统、设备、库、运行时与内核版本。数据清单记录来源许可、过滤版本、各桶数量与哈希。不能用一个数据集名称代替实际切分。

运行结束时，将状态写为完成或失败，填入实际 token、步数、耗时、产物哈希和日志路径；计划预算与实际用量分字段保留。压缩运行用父节点指向被压缩检查点，评测运行指向导出产物，而非仅指向训练目录。失败运行同样保留错误与部分日志。这样，指标变化才能追溯到一条明确差异，而不依赖文件夹名中的“final”。

11.4 按失败位置回退

先检查输入和管线，再修改算法。小批过拟合仍失败时，排查标签移动、掩码、格式与优化器，暂停大规模训练；训练峰值超限时，按状态账本处理主导项，并重新核实步时。领域增益伴随通用

退化时，回到数据配比、回放和停止步数，而不是继续堆叠监督。

压缩后若未过质量门槛，回到未压缩父节点，先检查校准分布、格式和敏感层，再比较提高精度或有限恢复训练。质量通过而延迟没有改善时，检查算子回退、打包、缓存和线程；只有实际瓶颈与候选干预吻合时才继续。不要用同一份测试集反复挑选补丁。

发布采用可回退检查点：保存上一个通过全部门槛的模型、tokenizer、适配器、量化配置与运行时组合；先在验证环境重载，再以相同请求回归。若上线环境出现门槛外退化，恢复这一完整组合并保存触发日志。只有权重文件而缺少兼容配置，通常不足以重建旧行为。

11.5 训练与压缩的联合目标

串联训练和压缩便于归因，也可能错过需要共同适应的解。令 q 为量化或稀疏配置，m 为硬件映射与内核，$\mathcal{W}$ 为固定工作负载，可用下式表达候选目标：

$$\begin{aligned} \min_{\theta,q,m} \quad & \mathcal{L}_{\text{task}}(f_{\theta,q}) + \lambda_{\text{KD}}\mathcal{D}(f_T, f_{\theta,q}) \\ & + \lambda_q \mathcal{E}_q(\theta, q) + \lambda_m \mathcal{C}(m; \mathcal{W}) \\ \text{s.t.} \quad & M(\theta, q, m; \mathcal{W}) \le B_M, \\ & t_{\text{TTFT}}(\theta, q, m; \mathcal{W}) \le B_F, \\ & t_{\text{TPOT}}(\theta, q, m; \mathcal{W}) \le B_P. \end{aligned} \tag{11.1}$$

各 λ 为事先定义的尺度或权重，$\mathcal{E}_q$ 为误差代理，$\mathcal{C}$ 为系统成本代理；三种预算分别约束内存、首 token 与后续生成。该式是问题定义，不是已经求出的通用算法。误差代理降低不保证任务改善，系统代理降低也不保证实测变快；最终门槛要在导出产物上检查。小预算项目可先固定 q, m 优化 θ，再逐项扩展搜索，避免无法归因的联合调参。

11.6 消融要回答一个有限的问题

表 11.2 选择一条主轴，其他预算固定；不要求运行全部组合。

主轴	最小对照示例	解释结果所需的记录
数据	原始与过滤，等累计训练 token	唯一来源覆盖、重复暴露、污染与各桶比例
监督	硬标签与温度蒸馏，等学生预算	温度 τ、教师调用、校准与逐样本错误
更新	全参或可行参照与低秩更新	可训练参数、激活峰值、搜索与实际总时间
压缩	原精度与一种可用格式	校准集、文件、峰值、质量和真实延迟
推理预算	两个输出上限或候选数	实际 token、停止率、验证器成本与准确率
部署	同一产物的两种设置	设备、线程、并发、输入长度与热状态

等 token、等更新与等时间不能总是兼得。按统计收益、优化效率或项目成本确定主预算，并披露其余差异。本书固定预算下的蒸馏与部署对照见附录 B 。

11.7 练习与提示

1. 为标称 8 GB 内存手机写约束表。**提示：**扣除系统与应用余量，分开权重、缓存和工作区。
2. 文件减半而 TPOT 上升，应先检查什么？**提示：**检查内核、打包与算子回退，再决定干预。
3. 为失败训练填写运行记录。**提示：**保留父节点、命令、实际预算、异常与回退决定。

第 12 章

从研究问题到可证伪实验

小模型研究的难点不是缺少候选方法，而是如何在有限预算下辨认真正起作用的变量。一个可检验的问题要说明：预期发生什么、改变什么、与谁比较、看什么指标，以及怎样的结果会推翻预期。本章的方案均为研究设计，不表示本书已经运行这些实验或观察到相应收益。

12.1 一条弹性的十二周学习路线

下面按每周可投入的时间、已有基础与设备能力安排六个阶段，十二周只是规划示例。每阶段先完成一个小而完整的交付，再决定是否扩展；不要求在一周内复现多种大型训练，更不以把所有方法跑一遍为完成标准。没有合适设备时，可先完成手算、代码检查与可执行配置，明确保留未运行项。

参考时段	阅读与问题	最小可检查交付
1–2 周	语言建模、tokenizer、缩放和显存	推导损失与一个预算账本；跑通小网络；注明理论代理与实测差异
3–4 周	数据来源、去重、污染与硬标签基线	版本化切分和来源清单；固定一个学生与评测协议
5–6 周	选择一条能力获得分支	在 SFT、温度蒸馏或继续预训练中选一项，与简单基线作受控比较
7–8 周	选择一条适配或压缩分支	在可用方法中选一项，记录质量、可训练参数、文件与实际预算
9–10 周	架构与部署成本	测或设计预填充、解码、峰值内存对照；检查目标运行时支持
11–12 周	一个机制假设与最终报告	完成一个对照、失败分析与限制；保存命令、配置、样本结果和复现入口

若任务耗时超过计划，优先缩小模型或减少无关方法，保留数据隔离、对照与日志。若只能交付一个实验，选最可能改变项目决策的问题。论文中的大规模结果用于理解机制与条件，不能由一个小实验自动推广。

12.2 先定义失败，再投入计算

设主指标差为 Δ，在实验前确定有实际意义的最小收益 $\delta > 0$、质量容许退化、资源上限与区间估计方法。若 Δ 的区间上界仍小于 δ，就否定了“至少取得该幅度收益”的预设主张；若区间跨越

门槛，应报告证据不足，不能把未显著误写成方法等价。过小样本或过高随机性需要通过先导实验评估；样本量不能从结果好坏倒推。

以下五个问题各自需要独立验证集和最终测试集。选择超参数、校准样本或位宽使用验证集，测试集用于冻结后的检查。保留无干预基线与成本账本，即使结果不支持假设，也能解释下一步应该改变什么。

12.2.1 激活加权谱能否预测低秩修复收益

假设。在相同量化格式与恢复预算下，量化残差的激活加权谱，比裸残差谱更能预测低秩补偿的任务收益。令 $E = W - \widehat{W}$，校准的未中心化二阶矩为 $\Sigma_X = XX^\top/n$；比较 E 与 $E\Sigma_X^{1/2}$ 的奇异值分布。后者与第 6 章的输出重构误差有关，但这种关系本身不证明下游任务收益。

干预与对照。固定基座、量化配置、校准数据、适配器位置和恢复训练 token，在事先选定的秩集合上运行补偿。分别用裸谱特征和加权谱特征建立同等复杂度的预测器，并保留无补偿组；预测器拟合使用一组模型或层，评估使用预先留出的组，避免把同一层的相邻秩结果随意分到两侧。

指标与失败条件。主要指标是在留出组上预测任务分数变化的误差，两种预测器采用相同拟合预算；次要指标包括域外质量、重构误差和恢复训练成本。若加权特征对预测误差的改善上界未达预定 δ，该预测优势主张在此范围内被否定。若只降低校准重构误差而未改善任务，不得改称“能力已经恢复”。单模型层内相关性不能支持跨模型普遍性。

12.2.2 校准选择与位分配是否存在互补收益

假设。在固定校准样本数、总存储与搜索预算下，覆盖性样本选择和混合位宽的联合方案，在任务分数上的收益超出两个单项收益之和，即存在正交互效应。选择变量是校准子集 $\mathcal{S}$ 与逐层位宽 b_l，而质量必须在固定验证分布上测量，不能在被选择的容易样本上自我评分。可写成

$$\begin{aligned} \min_{\mathcal{S},\{b_l\}} \quad & \mathcal{L}_{\mathrm{val}}(\widehat{f}_{\mathcal{S},b}), \\ \text{s.t.} \quad & |\mathcal{S}| \le n_{\mathrm{cal}}, \quad \sum_l N_l b_l + B_{\mathrm{meta}} \le B_{\mathrm{bits}}. \end{aligned} \tag{12.1}$$

其中 N_l 为该层量化权重数，B_{meta} 是比例因子等元数据占用；无法用内核实现的位分配不列入候选。

干预与对照。采用二乘二比较：随机与覆盖性选样，各自配统一与混合位宽。覆盖规则在看测试结果前冻结，四组使用同一基座、候选配置数、校准 token 上限和目标运行时。混合位宽组与统一位宽组匹配实际总存储预算，而非只匹配未加权平均位宽。

指标与失败条件。主要指标为冻结后测试质量的交互增益。以越高越好的任务分数为 a_{ij}，$i = 1$ 表示覆盖性选样、$j = 1$ 表示混合位宽，则交互增益为 $a_{11} - a_{10} - a_{01} + a_{00}$；同时报告域外质量、文件字节、延迟与搜索成本。若交互增益的区间上界未达预定门槛，便不支持互补优势；若验证集收益在独立测试上消失，应优先考虑选择过拟合。存储相同而内核延迟超限，也足以使该方案失去部署价值。

12.2.3 短轨迹是否比长轨迹更适合给定学生

假设。对指定推理任务与学生容量，经过验证的短轨迹在相同学生训练预算下，可在较低生成成本上达到不低于长轨迹的质量。这里检验的是轨迹表示与预算分配，不将输出长度本身当作推理能力。

干预与对照。从同一提示与教师来源构造仅答案、短自然语言过程、长过程及程序或方程轨迹；固定题目划分、难度覆盖、学生初始化、更新与累计输入 token 上限。不同长度导致独立样本数或重复暴露不同，需逐项记录，并用另一组等题目暴露对照检查影响。验证器版本与过滤规则保持一

致，教师生成和筛选另记账。

指标与失败条件。统一最大输出长度，另外扫描推理预算，画准确率 – 实际生成 token 与准确率 – 延迟曲线，记录中间步骤错误、格式失败及验证器成本。预先给定质量非劣界与成本降低门槛；若短轨迹在相近成本下持续低于非劣界，便否定该条件下的替代主张。若优势只来自教师更强、题目更容易或额外学生更新，则不能归因为轨迹更紧凑。

12.2.4 受限回放能否缓解端侧持续学习的遗忘

假设。在给定更新与存储预算下，小规模旧任务回放可比纯新任务更新更好地保持旧能力，同时保留可接受的新任务增益。

干预与对照。按同一时间顺序提供任务，比较不更新、只用新数据更新，以及在固定总训练 token 中分配一部分给旧数据回放三组。固定基座、可训练模块、每轮更新上限与设备；回放组额外占用的持久存储单列，并审计其中是否包含不应留存的数据。最终质量在不参与回放的旧、新任务留出集上测。

指标与失败条件。分别报告新任务变化、各旧任务退化、误拒和校准；有设备时测更新能量、峰值内存及一次受控更新失败后的恢复时间。若相对纯新数据组，旧任务退化改善不足预定门槛，或新任务增益跌破质量下限，则该回放配置未达到主张。若可靠性或隐私探针超过预设风险上限，即使均分提高也不采用。回退机制通过一次演练只证明该次路径可用，不证明所有故障均可恢复。

12.2.5 视觉 token 压缩在什么任务上开始失效

假设。在指定输入分辨率与训练预算下，学习式重采样比简单池化能用更少视觉 token 保持 OCR、计数和空间关系质量，但收益有任务边界。

干预与对照。固定视觉编码器、语言骨干、图像和视频帧采样，在同一 token 数网格上比较不压缩、池化与学习式重采样。训练使用相同数据和更新预算；新增连接器的参数与计算另报。用验证集确定方案后，在独立的文档、小字、计数、空间关系及低光照集合测试，避免只比较易分类图像。

指标与失败条件。逐任务报告质量与区间，并测完整链路的编码时间、TTFT、峰值内存与实际 token。若学习式方法在相同 token 数下未取得预设质量收益，或者为保质量增加的编码代价抵消语言侧节省，就否定所宣称的收益范围。若 OCR 已越过允许退化线，不能用其他任务均分补偿；应将该 token 数标为不适用于相应工作负载。

12.3 让失败成为下一次实验的输入

结论应绑定数据、任务、模型、硬件和修订日期；把渐近复杂度、内存流量与内核时间分开；将教师、校准、搜索和恢复训练计入预算。对没有支持的假设，保存反例与失败条件，再检查是机制不成立、作用范围更窄，还是实验尚无足够分辨率。不要在看到结果后悄悄更换主要指标，并把新故事当成原假设的成功。

12.4 练习与提示

1. 从上述问题选一个，写明一个主指标、最小有意义效应、对照和预算。**提示：**“质量更好、成本更低”还不足以决定成功或失败。
2. 为量化 – 低秩补偿实验构造三个失败原因。**提示：**区分校准覆盖不足、残差不可由给定秩表达、局部重构与任务目标不一致。
3. 差值区间覆盖退化与改善时如何报告？**提示：**证据不足不等于等价；扩大样本或限缩结论。

附录 A

公式速查表

本表只列正文中反复使用的定义与近似。符号沿用对应章节，局部重复字母依上下文解释；每个公式的假设与适用范围仍以正文为准。近似账本不能替代峰值测量，代理目标也不能替代任务评测。

单位与口径提醒

参数量是计数，不能把总参数、可训练参数和每 token 活跃参数互换。1 B 为 8 bit；1 MiB = 2^{20} B，1 GiB = 2^{30} B，而 GB 通常按 10^9 B 计。FLOPs 是计算量，FLOPs/s 是算力，均不等于实际完成时间。能量用焦耳，功率用瓦，token 数依赖 tokenizer。

KV 式只计算键和值的载荷，需另加分页、比例因子、索引、对齐与运行时开销。动态批量的峰值来自同一时刻保留长度之和；不能将批量请求数 b 同每元素字节数混用。改变权重位宽，不自动改变缓存精度。

TPOT 是本书规定的首 token 之后平均间隔，不含首 token 的重复计数；完成延迟止于最后一个输出 token，之后的后处理或流关闭另计。$n_{\text{out}} = 1$ 时标为不适用，$n_{\text{out}} = 0$ 或超时单独记录状态；不同工具采用其他定义时，应先转换口径再比较。跨任务平均分只在定义了共同尺度与方向后使用，不能混合准确率、损失和秒数。

主题	公式与条件
带监督掩码的下一词元损失	$\mathcal{L}_{\text{LM}} = -\dfrac{\sum_i m_i \log p_\theta(x_i \mid x_{<i})}{\sum_i m_i}$；$m_i \in \{0,1\}$，分母非零，因果边界与 padding 规则固定。
困惑度	$\text{PPL} = \exp(\mathcal{L}_{\text{LM}})$，损失用自然对数；只在同 tokenizer、文本处理和监督范围下直接比较。
因果自注意力	$\text{softmax}(QK^\top/\sqrt{d_h} + M)V$；掩码禁止访问未来位置。
二项缩放近似	$L(N, D) = E + AN^{-\alpha} + BD^{-\beta}$；固定数据、架构与优化范围下的经验模型。
训练计算代理	$C \approx \kappa ND$，常用粗估 $\kappa \approx 6$；D 是累计处理 token，非仅答案监督 token。
计算最优的边际条件	$\alpha AN^{-\alpha} = \beta BD^{-\beta}$，同时满足 $C = \kappa ND$；不再额外加入独立计算缩放项。
训练峰值账本	$M_{\text{peak}} \approx M_W + M_G + M_O + M_{\text{master}} + M_A + M_{\text{tmp}} + M_{\text{runtime}}$. 主权重可选，激活与工作区依赖实现。
正向温度蒸馏	$\mathcal{L}_{\text{KD}} = \tau^2 \,\text{KL}(p_T^{(\tau)} \| p_S^{(\tau)})$；定义在可对齐的输出空间与给定前缀上，温度 $\tau > 0$。
LoRA 更新	$W' = W + \dfrac{\alpha}{r} BA$；减少可训练状态不等于减少基座总参数。
均匀仿射量化	$q = \text{clip}(\text{round}(x/s) + z, q_{\min}, q_{\max})$, $\hat{x} = s(q - z), \quad s > 0.$ 对称的有符号约定常取 $z = 0$；一般仿射式允许非零零点。
GPTQ 型局部目标	$\min_{\widehat{W}} \|(W - \widehat{W})X\|_F^2$；$X$ 为校准激活，不是测试集。

主题	公式与条件（续）
激活加权误差	$\|EX\|_F^2 = \mathrm{tr}(EXX^\top E^\top)$，$E = W - \widehat{W}$。
低秩参数门槛	$W \in \mathbb{R}^{m\times n}$，用秩 r 因子需 $r(m+n)$ 个参数；仅在 $r < mn/(m+n)$ 时少于原矩阵。
未压缩 KV 载荷	$M_{\mathrm{KV}} = 2bLTH_{\mathrm{kv}}d_h s_{\mathrm{KV}}$；$b$ 为请求数，s_{KV} 为每元素字节数，其余符号见第 7 章。
变长请求的 KV 载荷	$M_{\mathrm{KV}} = 2LH_{\mathrm{kv}}d_h s_{\mathrm{KV}} \sum_{i=1}^{b} T_i$；同层配置、无共享、稠密存储时成立。
DPO	$a_\theta(x,y) = \log \pi_\theta(y \mid x) - \log \pi_{\mathrm{ref}}(y \mid x)$；记 $a_\theta^{\pm} = a_\theta(x, y^{\pm})$，则 $\mathcal{L}_{\mathrm{DPO}} = -\mathbb{E}\log\sigma(\beta[a_\theta^{+} - a_\theta^{-}])$。对数概率按回答 token 求和。
完成延迟	$t_{\mathrm{complete}} = t_{\mathrm{TTFT}} + (n_{\mathrm{out}} - 1)t_{\mathrm{TPOT}}$；首 token 已计入 TTFT，$n_{\mathrm{out}} \geq 2$。单 token 请求只有 TTFT。
整机与增量能量	$E_{\mathrm{total}} = \int_0^{t_{\mathrm{end}}} P(t)\,dt$，$E_{\mathrm{incremental}} = \int_0^{t_{\mathrm{end}}} [P(t) - P_{\mathrm{idle}}]\,dt$。仪器、采样与空闲基线需说明。

附录 B

最小实验与复现方案

本附录区分三类证据：手算及独立数值核验、已经运行的合成分类训练与部署对照、尚未在目标生成设备上运行的方案。原始训练与部署记录的 UTC 时间戳为 2026 年 10 月 21 日；本版在 2026 年 10 月 23 日复核保存的检查点与统计结果，沿用原始实验的种子和教师配置。

B.1 目录、命令与复核边界

固定配置在 `manuscript/experiments/config.json`，软件硬件、来源哈希、数据、检查点和逐次延迟均保存在本主题的 `output/experiments/modern_small_model_training-jingfeng/`。详细命令见随书实验 README。基本运行顺序为：

```
# Run from this book's directory, with Python and pinned requirements installed.
python manuscript/experiments/run.py
python manuscript/experiments/summarize.py
python manuscript/experiments/verify_formulas.py
python manuscript/experiments/audit_saved_results.py
```

主运行会覆盖实验输出目录；复核脚本只读取保存的模型与记录，另写审计 JSON，不重新训练或重新测时。它检查源码哈希、数据再生成和划分独立性、学生批次与预算，并用独立标量参考重算准确率、负对数似然、Brier 与 ECE，再检查逐次延迟和 RSS 汇总。原始训练使用 Python 3.13.3、PyTorch 2.8.0、NumPy 2.2.6；本版复核使用独立 Python 3.12 环境和相同 PyTorch、NumPy 版本，完整环境留在输出目录。

B.2 固定预算下的蒸馏对照

本实验是离线生成的非线性六分类教学任务，不是语言模型能力评测。输入为 32 维高斯特征，经固定随机非线性映射、乘性交互和正弦特征生成类别概率，再混入 10% 均匀分布并随机抽取标签。固定数据种子 14051；训练、验证、测试集分别为 4096、1024、3072 个样本。所有划分在训练前生成，验证和测试标签均不参与参数更新。教师和学生是两层隐藏层的 Tanh MLP；隐藏宽度分别为 192 和 64，参数量分别为 44550 和 6662。学生参数初始化种子 18051，两个条件共享同一初始权重。教师固定训练 1000 次更新；两个学生固定各训练 600 次更新，每批 128，共享完整的小批次索引序列。优化器均为 AdamW，权重衰减为 0.001；教师和学生学习率分别为 0.003 和 0.002，使用最终检查点，无提前停止或结果筛选。蒸馏条件取 $\tau = 2$、$\alpha = 0.5$，采用教师到学生的 KL，乘以 τ^2。学生更新预算相同，蒸馏另付教师训练、缓存和额外损失计算的成本。

Brier 分数按每个样本对六个类别的平方误差求和，再对样本取均值；ECE 使用 15 个固定等宽置信度分箱，计算分箱准确率和最大类别概率均值之差的绝对值，再按样本数加权。ECE 依赖分箱，不能代替可靠性图或多次重复的不确定性分析。教师训练集准确率为 99.10%，验证集准确率为 36.52%，

表 B.1 本次固定种子的测试结果；Accuracy 单位为百分比，其余越低越好。

Method	Accuracy	NLL	Brier	ECE
Teacher	35.29	2.443	0.947	0.353
Hard labels	44.08	1.462	0.704	0.036
Distillation	43.46	1.534	0.724	0.130

训练集和验证集 NLL 分别为 0.118 和 2.359。这种训练与留出数据的差距与过拟合相符；本实验没有进一步区分容量、优化和数据噪声各自的因果作用。固定教师的测试表现低于硬标签学生，蒸馏没有提升测试准确率，且 NLL、Brier 和 ECE 均变差。因此本例说明教师更大、存在软标签都不能保证学生受益，不能据此概括所有蒸馏方法。本次教师训练耗时 0.898 秒，缓存 4096 条训练样本的教师输出耗时 0.018 秒，FP32 logits 缓存占 98304 字节。硬标签和蒸馏学生的更新阶段分别耗时 0.489 和 0.506 秒。这些单次墙钟时间不含数据生成和评测，并受进程调度影响；教师成本没有摊销到假定的多个学生。

B.3 压缩文件与部署代价的实测区别

同一蒸馏学生分别保存为 FP32 状态字典和动态 INT8 Linear 状态字典。部署设备为 Apple M1、8 GB 统一内存，使用 CPU、单线程、PyTorch 2.8.0、NumPy 2.2.6 和 QNNPACK；未启用 GPU。动态量化只替换 Linear，偏置及非线性运算保留浮点，运行时量化激活；这不是全模型每个张量都采用 INT8 的保证。每次前向固定输入 32×32 个特征值、输出 32×6 个 logits，这是分类前向的教学替代任务，没有生成文本的输入或输出 token 长度。每个条件使用三个全新独立进程，每进程先预热 100 次，再计时 1000 次；每对进程交替先运行 FP32 和 INT8。计时只覆盖 eager CPU 前向，不包含模型加载、预处理和输出后处理。表中延迟为 3000 次观测合并后的中位数与第 95 百分位，不是跨设备性能结论。

表 B.2 本地分类模型部署对照。文件为含序列化元数据的权重状态字典；RSS 为整个进程。

Format	File/KiB	Peak RSS/MiB	Median/μs	P95/μs	Acc./%
FP32	28.71	185.62	42.04	116.38	43.46
INT8	11.50	188.88	139.92	376.42	43.20

峰值 RSS 由各进程的 `getrusage` 高水位取得，macOS 原始单位为字节，表中取三个进程的最大值；没有将导入时内存从结果中相减。它包括 Python、PyTorch、模型、输入和后端工作区，不是张量专用内存，也不是训练显存。按固定 batch 分块进行的完整测试集评估发生在记录部署峰值之后。FP32 三个进程各自的延迟中位数为 41.71、42.35、42.25 微秒，RSS 为 161.48、182.39、185.62 MiB。INT8 三个进程各自的延迟中位数为 149.81、140.88、134.33 微秒，RSS 为 151.94、188.88、188.59 MiB。本次文件缩小为原来的 0.401 倍，但前向延迟中位数增至 3.33 倍。运行时常驻成本和量化开销可能盖过小矩阵收益；本实验只证明本配置未获得加速，未定位全部瓶颈。测试 logits 的最大和平均绝对差分别为 0.06644 和 0.01154，预测类别一致率为 99.45%。没有测量 GPU 显存、能耗或文本生成吞吐，不能将这些分类前向结果外推到 LLM 服务。

B.4 独立数值参考与边界核验

`verify_formulas.py` 以 Python 标量指数、对数及求和实现独立参考，并与 PyTorch 的 KL 及自动微分核对。取教师 logits $(4, -1, 0.5)$、学生 logits $(0.2, 2, -0.7)$、$\tau = 2$，有

$$D_{\mathrm{KL}}(p_\tau \| q_\tau) = 0.78038035, \qquad D_{\mathrm{KL}}(q_\tau \| p_\tau) = 1.06086891.$$

教师到学生方向乘 τ^2 后为 3.12152140。对学生 logits 的梯度为

$$(-1.10440103,\ 1.06989390,\ 0.03450713),$$

与独立公式 $\tau(q_\tau - p_\tau)$ 在双精度误差范围内一致。这同时给出 KL 方向不能交换的反例；脚本另验证 τ^2 并不让所有有限温度的梯度完全相等。

训练静态账本取 $N = 1.5 \times 10^9$：BF16 参数与梯度各 3 GB，FP32 Adam 一阶和二阶矩各 6 GB，合计 18 GB（约 16.76 GiB）；若另留 FP32 主副本，再加 6 GB，合计 24 GB（约 22.35 GiB）。这些数值不含激活、工作区、分配器保留和通信缓冲，不能直接作为显卡容量需求。只把参数存储改为理想 INT4、其余账本不变时为 15.75 GB，这仅是区分各项的算术反例，不表示可以直接对 INT4 权重做普通 Adam 更新。

KV cache 假定每层保存稠密完整的 K 和 V，取 $b = 2, L = 24, T = 4096, H_{\mathrm{kv}} = 4, d_h = 64, s = 2$，由

$$M_{\mathrm{KV}} = 2bLTH_{\mathrm{kv}}d_h s = 201326592 \text{ bytes} = 192 \text{ MiB}.$$

脚本以缩小后的显式 BF16 K/V 张量独立验证元素数和字节数，并核对零 batch、零 token、上下文减半和 batch 减半边界。误将 24 个 query heads 代入本例的 4 个 KV heads 会高估六倍。此式不包含分页填充、块元数据、窗口截断、共享和缓存量化。全部原始配置、逐次延迟、独立进程记录、环境版本及方程检查位于本主题 `output/experiments/modern_small_model_training-jingfeng/`。结果仅来自一个固定种子，没有估计训练种子引起的方差。

B.5 生成模型与端侧的未运行方案

合成分类没有自回归前缀和 KV cache，不能验证真实生成延迟。配套 `experiments/run_llama_cpp_plan.sh` 给出一个可执行的 CPU 生成内核对照：提供本地许可允许使用的 F16 GGUF 模型和同一版本的 llama.cpp 工具，量化为 Q8_0，分别以固定线程、预热、512 个预填充 token 和 128 个生成 token 做五次重复，保存工具版本、文件哈希、原始 JSON 与进程峰值内存。

该方案使用 `llama-bench` 的合成 token 基准；预填充与生成是分开测试，后者并非携带 512 token 提示的完整用户请求。因此它报告的是指定内核形状的吞吐和峰值进程内存，不能直接称为 TTFT、逐请求 TPOT、端到端延迟或任务质量。若要验收生成应用，还需按第 10 章的逐 token 时间戳协议，固定真实提示、聊天模板、输出长度和停止条件，保存请求级事件；量化前后再使用相同留出任务测质量与校准。

本机未提供目标手机/NPU、相应模型产物和可靠功率仪表，因而这一方案未执行，也没有填入预期速度或能耗。真实端侧验收须将预处理、传感器编码、模型生成、后处理、持续运行和温控一起纳入，并明确整机功率与空闲基线；QAT、剪枝和多设备比较同样保留为后续有预算的对照，不能从本例推断效果。

附录 C

术语表与阅读清单

C.1 术语表

以下术语用于快速查阅；具体算法、计时边界与实现条件见对应章节。英文缩写并不构成独立技术规格，例如同称 4-bit 的两份产物可能采用不同格式与内核。

Token / tokenizer	词元及将文本映射为词元序列的分词器。累计处理 token、监督 token、唯一文本量与重复暴露次数是不同计数。
有效 token	对数据所提供学习信号的非正式概括，没有跨语料、模型和任务统一可观测的计量单位。本书记录清洗后 token、去重簇、来源覆盖与暴露次数，并用受控实验判断收益，不把“质量”直接折算为一个确定 token 数。
PPL	困惑度（perplexity），在本书自然对数约定下为平均负对数似然的指数。换 tokenizer 或监督范围后不可直接按数值高低比较。
KL / dark knowledge	KL 是 Kullback–Leibler 散度，方向不可互换；dark knowledge 指教师输出分布中不同类别或词元的相对概率结构，不意味着教师总是正确。
暴露偏差	正确前缀上的训练与自身输出前缀上的部署之间的分布错配。序列蒸馏也需说明前缀来自何种策略。
PEFT / LoRA	参数高效微调（parameter-efficient fine-tuning）及低秩适配（low-rank adaptation）。主要减少更新参数和相关状态，不自动压缩基座或免除激活反传。
PTQ	训练后量化（post-training quantization）：在已有模型上选择量化表示与参数，可包含统计校准或有限局部优化；不能以“完全禁止梯度”定义。
QAT	量化感知训练（quantization-aware training）：在训练或恢复训练中引入目标量化误差，使模型适应低精度表示；需要对应导出格式与内核验证。
结构化剪枝	删除通道、头、层、行列或块，使计算形状或可识别模式发生变化。只有运行时能够利用该结构，才可能得到实际加速。
KV / GQA / MQA	KV 指键和值；分组查询注意力让一组查询头共享 KV，多查询注意力让全部查询头共享。缓存账本同时依赖请求数、历史长度、层数、KV 头数与元素字节数。
TTFT / TPOT	首 token 延迟（time to first token）与每输出 token 时间（time per output token）。本书 TPOT 指首 token 之后的平均间隔；一个输出 token 时不计算 TPOT。计时是否包括排队、分词与网络需说明。
Prefill / decode	预填充一次处理提示，解码逐步生成后续 token。两阶段的算术强度、缓存访问与调度条件不同，应分别测量。
SFT / DPO	监督微调（supervised fine-tuning）使用示范回答；直接偏好优化（direct preference optimization）使用偏好对和参考策略的概率比。二者目标与数据职责不同。
CoT / RL	思维链（chain of thought）是自然语言推理轨迹；强化学习（reinforcement learning）通过奖励与交互或采样优化策略。长轨迹和更高奖励都需独立的正确性验证。
VLM / OCR	视觉语言模型（vision-language model）联合处理视觉与文本；光学字符识别（optical character recognition）关注图像中的文字。视觉 token 压缩可能对后者更敏感。
OOD / ECE	分布外（out-of-distribution）评测明确改变来源、时间或其他分布轴；期望校准误差（expected

	calibration error）比较分箱置信度与正确率，依赖分箱与样本量。
量化位宽	描述数值载荷的位数。比例因子、零点、组大小、累加精度、打包与内核同样决定文件、误差和速度。
蒸馏成本	至少分开记录教师生成或 logits 缓存、筛选、学生训练、搜索与评测；已有教师的原始训练成本可另列背景，不与本次边际调用成本混为一项。

C.2 常见误区

1. 参数更少就一定更快：内存带宽、KV cache、矩阵形状与内核可能占主导。
2. 4-bit 是统一规格：不同格式、元数据和运行时不能仅按位宽比较。
3. 长 CoT 就代表可靠推理：轨迹可能包含错误；最终答案通过也不证明每一步成立。
4. 平均基准分代表部署质量：关键任务、风险、尾延迟与能耗需要独立检查，异量纲指标不能直接平均。
5. 教师更大就一定能教好学生：教师偏差、学生容量、tokenizer 和采样前缀分布同样关键。
6. 没有观察到显著差异就已经证明等价：还需检查样本量、区间与预先设定的等价或非劣门槛。

C.3 主干阅读顺序

建议按以下顺序建立概念，再回到正文的实验协议。每组阅读先回答一个问题，不以方法名称数量为进度：

1. **目标与预算**。Transformer、Kaplan 与 Chinchilla[2, 3, 4]：预测目标是什么，计算约束怎样影响参数与数据分配？
2. **数据与小模型配方**。教材化数据、FineWeb、DataComp-LM 与 SmolLM2[9, 5, 7, 39]：哪些结论来自受控数据比较，哪些仍受模型或训练规模影响？
3. **蒸馏与分词**。温度蒸馏、DistilBERT、MiniLM、序列蒸馏、学生分布采样与 SentencePiece[10, 11, 12, 13, 14, 8]：监督对象与前缀分布分别是什么，何时不能逐 token 对齐？
4. **更新状态**。LoRA、QLoRA、GaLore 与 ZeRO[15, 16, 18, 19]：减少的是哪些状态，新增了什么计算或通信？
5. **模型表示**。GPTQ、SmoothQuant、AWQ、EfficientQAT、SparseGPT、LLM-Pruner 与 SliceGPT[20, 21, 22, 23, 24, 25, 26]：局部误差目标如何联系任务质量，产物能否被实际内核使用？
6. **架构与评测**。FlashAttention、Mamba 系列、Hymba 与多维评测 [27, 28, 29, 30, 31, 35, 36, 37, 38]：渐近复杂度之外，哪些状态与测量边界改变了比较？
7. **行为与多模态**。DPO、DeepSeek-R1 与 MobileVLM[32, 33, 34]：额外监督与系统组件分别贡献什么，教师和传感器成本如何入账？

完整参考文献由主文件通过 Biber 生成。本附录提供阅读路线；具体性能只在原论文条件下解释，原始文献与本地归档入口见本书来源记录。

参考文献

[1] P. Zhang, G. Zeng, T. Wang, and W. Lu. *TinyLlama: An Open-Source Small Language Model*. Version v2. Jan. 4, 2024. arXiv: 2401.02385v2. url: https://arxiv.org/abs/2401.02385v2 (visited on 10/23/2026).

[2] A. Vaswani, N. Shazeer, N. Parmar, et al. *Attention Is All You Need*. Version v7. June 12, 2017. arXiv: 1706.03762v7. url: https://arxiv.org/abs/1706.03762v7 (visited on 10/23/2026).

[3] J. Kaplan, S. McCandlish, T. Henighan, et al. *Scaling Laws for Neural Language Models*. Version v1. Jan. 23, 2020. arXiv: 2001.08361v1. url: https://arxiv.org/abs/2001.08361v1 (visited on 10/23/2026).

[4] J. Hoffmann, S. Borgeaud, A. Mensch, et al. *Training Compute-Optimal Large Language Models*. Version v1. Mar. 29, 2022. arXiv: 2203.15556v1. url: https://arxiv.org/abs/2203.15556v1 (visited on 10/23/2026).

[5] G. Penedo, H. Kydlíček, L. Ben Allal, et al. *The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale*. Version v2. June 25, 2024. arXiv: 2406.17557v2. url: https://arxiv.org/abs/2406.17557v2 (visited on 10/23/2026).

[6] K. Lee, D. Ippolito, A. Nystrom, et al. "Deduplicating Training Data Makes Language Models Better". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2022, pp. 8424–8445. doi: 10.18653/v1/2022.acl-long.577. arXiv: 2107.06499v2. url: https://aclanthology.org/2022.acl-long.577/ (visited on 10/23/2026).

[7] J. Li, A. Fang, G. Smyrnis, et al. *DataComp-LM: In search of the next generation of training sets for language models*. Version v4. June 17, 2024. arXiv: 2406.11794v4. url: https://arxiv.org/abs/2406.11794v4 (visited on 10/23/2026).

[8] T. Kudo and J. Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2018, pp. 66–71. doi: 10.18653/v1/D18-2012. arXiv: 1808.06226v1. url: https://aclanthology.org/D18-2012/ (visited on 10/23/2026).

[9] S. Gunasekar, Y. Zhang, J. Aneja, et al. *Textbooks Are All You Need*. Version v2. June 20, 2023. arXiv: 2306.11644v2. url: https://arxiv.org/abs/2306.11644v2 (visited on 10/23/2026).

[10] G. Hinton, O. Vinyals, and J. Dean. *Distilling the Knowledge in a Neural Network*. Version v1. Mar. 9, 2015. arXiv: 1503.02531v1. url: https://arxiv.org/abs/1503.02531v1 (visited on 10/23/2026).

[11] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. Version v4. Oct. 2, 2019. arXiv: 1910.01108v4. url: https://arxiv.org/abs/1910.01108v4 (visited on 10/23/2026).

[12] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*. Version v2. Feb. 25, 2020. arXiv: 2002.10957v2. url: https://arxiv.org/abs/2002.10957v2 (visited on 10/23/2026).

[13] Y. Kim and A. M. Rush. "Sequence-Level Knowledge Distillation". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2016, pp. 1317–1327. doi: 10.18653/v1/D16-1139. arXiv: 1606.07947v4. url: https://aclanthology.org/D16-1139/ (visited on 10/23/2026).

[14] R. Agarwal, N. Vieillard, Y. Zhou, et al. *On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes*. Version v3. June 23, 2023. arXiv: 2306.13649v3. url: https://arxiv.org/abs/2306.13649v3 (visited on 10/23/2026).

[15] E. J. Hu, Y. Shen, P. Wallis, et al. *LoRA: Low-Rank Adaptation of Large Language Models*. Version v2. June 17, 2021. arXiv: 2106.09685v2. url: https://arxiv.org/abs/2106.09685v2 (visited on 10/23/2026).

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

[17] Y. Li, Y. Yu, C. Liang, et al. *LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models*. Version v4. Oct. 12, 2023. arXiv: 2310.08659v4. url: https://arxiv.org/abs/2310.08659v4 (visited on 10/23/2026).

[18] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian. *GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection*. Version v2. Mar. 6, 2024. arXiv: 2403.03507v2. url: https://arxiv.org/abs/2403.03507v2 (visited on 10/23/2026).

[19] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. *ZeRO: Memory Optimizations Toward Training Trillion Parameter Models*. Version v3. Oct. 4, 2019. arXiv: 1910.02054v3. url: https://arxiv.org/abs/1910.02054v3 (visited on 10/23/2026).

[20] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. *GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers*. Version v2. Oct. 31, 2022. arXiv: 2210.17323v2. url: https://arxiv.org/abs/2210.17323v2 (visited on 10/23/2026).

[21] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han. *SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models*. Version v7. Nov. 18, 2022. arXiv: 2211.10438v7. url: https://arxiv.org/abs/2211.10438v7 (visited on 10/23/2026).

[22] J. Lin, J. Tang, H. Tang, et al. *AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration*. Version v6. June 1, 2023. arXiv: 2306.00978v6. url: https://arxiv.org/abs/2306.00978v6 (visited on 10/23/2026).

[23] M. Chen, W. Shao, P. Xu, et al. *EfficientQAT: Efficient Quantization-Aware Training for Large Language Models*. Version v3. July 10, 2024. arXiv: 2407.11062v3. url: https://arxiv.org/abs/2407.11062v3 (visited on 10/23/2026).

[24] E. Frantar and D. Alistarh. *SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot*. Version v3. Jan. 2, 2023. arXiv: 2301.00774v3. url: https://arxiv.org/abs/2301.00774v3 (visited on 10/23/2026).

[25] X. Ma, G. Fang, and X. Wang. *LLM-Pruner: On the Structural Pruning of Large Language Models*. Version v3. May 19, 2023. arXiv: 2305.11627v3. url: https://arxiv.org/abs/2305.11627v3 (visited on 10/23/2026).

[26] S. Ashkboos, M. L. Croci, M. G. do Nascimento, T. Hoefler, and J. Hensman. *SliceGPT: Compress Large Language Models by Deleting Rows and Columns*. Version v2. Jan. 26, 2024. arXiv: 2401.15024v2. url: https://arxiv.org/abs/2401.15024v2 (visited on 10/23/2026).

[27] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. Version v2. May 27, 2022. arXiv: 2205.14135v2. url: https://arxiv.org/abs/2205.14135v2 (visited on 10/23/2026).

[28] T. Dao. *FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning*. Version v1. July 17, 2023. arXiv: 2307.08691v1. url: https://arxiv.org/abs/2307.08691v1 (visited on 10/23/2026).

[29] A. Gu and T. Dao. *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*. Version v2. Dec. 1, 2023. arXiv: 2312.00752v2. url: https://arxiv.org/abs/2312.00752v2 (visited on 10/23/2026).

[30] T. Dao and A. Gu. *Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality*. Version v1. May 31, 2024. arXiv: 2405.21060v1. url: https://arxiv.org/abs/2405.21060v1 (visited on 10/23/2026).

[31] X. Dong, Y. Fu, S. Diao, et al. *Hymba: A Hybrid-head Architecture for Small Language Models*. Version v1. Nov. 20, 2024. arXiv: 2411.13676v1. url: https://arxiv.org/abs/2411.13676v1 (visited on 10/23/2026).

[32] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. Version v3. May 29, 2023. arXiv: 2305.18290v3. url: https://arxiv.org/abs/2305.18290v3 (visited on 10/23/2026).

[33] DeepSeek-AI, D. Guo, D. Yang, et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. Version v2. Jan. 22, 2025. arXiv: 2501.12948v2. url: https://arxiv.org/abs/2501.12948v2 (visited on 10/23/2026).

[34] X. Chu, L. Qiao, X. Lin, et al. *MobileVLM : A Fast, Strong and Open Vision Language Assistant for Mobile Devices*. Version v2. Dec. 28, 2023. arXiv: 2312.16886v2. url: https://arxiv.org/abs/2312.16886v2 (visited on 10/23/2026).

[35] D. Hendrycks, C. Burns, S. Basart, et al. *Measuring Massive Multitask Language Understanding*. Version v3. Sept. 7, 2020. arXiv: 2009.03300v3. url: https://arxiv.org/abs/2009.03300v3 (visited on 10/23/2026).

[36] K. Cobbe, V. Kosaraju, M. Bavarian, et al. *Training Verifiers to Solve Math Word Problems*. Version v2. Oct. 27, 2021. arXiv: 2110.14168v2. url: https://arxiv.org/abs/2110.14168v2 (visited on 10/23/2026).

[37] P. Liang, R. Bommasani, T. Lee, et al. *Holistic Evaluation of Language Models*. Version v2. Nov. 16, 2022. arXiv: 2211.09110v2. url: https://arxiv.org/abs/2211.09110v2 (visited on 10/23/2026).

[38] L. Zheng, W.-L. Chiang, Y. Sheng, et al. *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. Version v4. June 9, 2023. arXiv: 2306.05685v4. url: https://arxiv.org/abs/2306.05685v4 (visited on 10/23/2026).

[39] L. Ben Allal, A. Lozhkov, E. Bakouch, et al. *SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model*. Version v1. Feb. 4, 2025. arXiv: 2502.02737v1. url: https://arxiv.org/abs/2502.02737v1 (visited on 10/23/2026).