北京師範大學

学术硕士研究生
学位论文

基于优化状态特性的显存高效零阶优化算法

作　　者：　余梓铭

学　　号：　202321081035

导师姓名、职称：　李　嘉　讲　师

学　科　专　业：　计算机科学与技术

培　养　单　位：　人工智能学院

提　交　日　期：　二〇二六年五月

北京师范大学学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：余梓铭　　日期：2026 年 5 月 18 日

学位论文使用授权书

学位论文作者完全了解北京师范大学有关保留和使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。保密的学位论文在解密后适用于本授权书。

本人签字：余梓铭　　日　期：2026年5月18日

导师签名：李嘉　　日　期：2026年5月18日

基于优化状态特性的显存高效零阶优化算法

摘 要

预训练大模型在垂直领域的应用依赖下游任务微调，该过程对显存的巨大消耗已成为制约其普惠化的核心瓶颈。其中，反向传播中的激活值缓存与优化器状态（如动量等）存储是两大主要开销来源。零阶优化通过前向传播估计梯度，无需反向传播，从而避免激活值缓存，是显存高效训练的代表性技术之一。零阶优化的核心流程包含梯度估计与优化器状态更新两个关键步骤，然而现有算法在这两个步骤均面临挑战。在梯度估计方面，梯度估计方差会随着模型参数量线性增长，从而显著制约收敛速度。在优化器状态更新方面，尽管基于动量的优化器（如 AdamW）能够有效加速收敛，但其优化器状态会引入可观的显存开销；现有的优化器状态量化技术虽能压缩显存，但当量化精度低于 4 比特时往往会导致训练崩溃。上述问题的根源在于现有算法未能充分挖掘优化状态的内蕴特性。

本文以优化状态特性为统一视角，从梯度估计与优化器状态更新两个关键步骤出发，系统研究显存高效的零阶优化算法。主要研究成果如下：

（1）针对零阶优化梯度估计方差大、收敛慢的问题，本文提出一种基于随机子空间的零阶梯度估计算法。该算法利用梯度矩阵的低秩结构特性，通过构造列正交投影矩阵在低维子空间内生成扰动并进行梯度估计，将梯度估计方差从依赖模型总参数量降至仅依赖子空间维度。理论分析的结果证明了算法的收敛性及对反向传播梯度的逼近能力。多种大语言模型微调实验表明，该算法在显存开销与现有零阶算法相当的前提下，收敛速度与微调性能均显著优于同类算法。

（2）针对基于动量的优化器显存开销过大，且现有量化方法在精度低于 4 比特时极易导致训练崩溃的问题，本文提出一种基于极坐标向量量化的优化器状态压缩算法。该算法利用动量值分布的圆对称性与准高斯特性，构建二维极坐标量化框架：针对有符号一阶动量采用角度均匀分布码本确保方向精度，针对无符号二阶动量设计第一象限映射、自适应角度分配及轴偏移防除零机制。在图像分类与自然语言建模等多项预训练与微调任务上的实验表明，该算法在将优化器状态压缩至 1.5-2 比特后仍能稳定训练，达到与全精度版本相当的性能。

（3）本文将所提出的算法与 AdamW 等基于动量的优化器集成，在大语言模型微调任务中开展验证。结果表明，该方案在显著降低显存开销的同时保持收敛速度和模型性能，为资源受限场景下的大模型微调提供了技术支撑。

关键词：大模型微调，零阶优化，显存高效优化，低秩子空间，向量量化

Memory-Efficient Zeroth-Order Optimization Algorithms Based on the Properties of Optimization States

ABSTRACT

Deploying pre-trained large-scale models for domain-specific applications depends on task-specific fine-tuning, but the large GPU memory consumption of this process is a major bottleneck. This memory overhead comes from two sources: cached activation values during backpropagation and stored optimizer states (e.g., first-order momentum and second-order moments). Zeroth-order (ZO) optimization eliminates activation storage by estimating gradients through forward passes only, without backpropagation. This makes it a promising approach for memory-efficient training. ZO optimization has two key steps: gradient estimation and optimizer state updating. However, existing algorithms face challenges in both steps. First, the variance of gradient estimation grows linearly with model size, which limits convergence speed. Second, momentum-based optimizers like AdamW can speed up convergence, but their optimizer states need a lot of memory. Also, existing methods for quantizing optimizer states often cause training to fail when the precision is below 4 bits. These problems happen because existing algorithms do not fully use the inherent properties of optimization states.

This thesis uses the properties of optimization states as a unified perspective to study memory-efficient ZO algorithms from two key steps: gradient estimation and optimizer state updating. The main contributions are as follows:

(1) To address the high variance and slow convergence of ZO gradient estimation, this thesis proposes a random subspace ZO algorithm. The algorithm uses the low-rank structure of gradient matrices. It builds column-orthogonal projection matrices to generate perturbations and estimate gradients in low-dimensional subspaces. This reduces the variance of gradient estimation from depending on total parameter count to only depending on subspace dimension. Theoretical analysis shows that the algorithm converges and can approximate backpropagation gradients. Experiments on various large language models show that the proposed algorithm is significantly better than existing ZO methods in convergence speed and fine-tuning performance, with similar

memory cost.

(2) To address the large memory cost of momentum-based optimizer states and the problem that existing quantization methods often cause training to fail below 4 bits, this thesis proposes an optimizer state compression algorithm based on polar vector quantization. The algorithm uses the circular symmetry and quasi-Gaussian distribution of momentum values. It builds a two-dimensional polar quantization framework. For signed first-order momentum, it uses angularly uniform codebooks to ensure directional accuracy. For unsigned second-order momentum, it uses first-quadrant mapping, adaptive angle allocation, and an axis-offset mechanism to prevent division by zero. Experiments on image classification and language modeling tasks, for both pre-training and fine-tuning, show that the algorithm can train stably at 1.5–2 bits, with performance similar to full-precision versions.

(3) This thesis integrates the proposed algorithms with momentum-based optimizers such as AdamW and tests them on large language model fine-tuning tasks. Results show that the proposed scheme greatly reduces memory cost while keeping convergence speed and model performance, providing technical support for large-scale model fine-tuning in resource-limited scenarios.

KEY WORDS: Large-Scale Model Fine-tuning, Zeroth-Order Optimization, Memory-Efficient Optimization, Low-Rank Subspace, Vector Quantization

目 录

本文符号说明 X

第 1 章 绪论 1

1.1 研究背景与意义 1

1.2 国内外研究现状 2

1.2.1 无需反向传播的零阶优化 2

1.2.2 进一步降低零阶优化显存的技术 3

1.3 本文主要工作 4

1.4 论文结构安排 5

第 2 章 相关理论基础 7

2.1 优化理论与算法基础 7

2.1.1 优化问题的数学表述 7

2.1.2 深度学习训练显存占用 7

2.1.3 零阶优化算法 8

2.1.4 基于动量的优化算法 10

2.2 量化压缩在深度学习中的应用 11

2.2.1 标量量化与向量量化 11

2.2.2 模型张量的量化压缩 12

2.3 低秩子空间在深度学习中的应用 14

2.3.1 低秩矩阵与近似理论 15

2.3.2 通用权重子空间假说 16

2.3.3 基于权重低秩的优化方法 16

2.3.4 基于梯度低秩的优化方法 17

2.4 本章小结 17

第 3 章 基于低秩子空间的大模型零阶梯度估计算法 18

3.1 引言 18

3.2 算法介绍 19

3.2.1 低秩扰动的零阶梯度估计器 19

3.2.2 子空间延迟更新策略 22

3.2.3 参数高效微调方法集成 23

3.3 理论分析 23
3.3.1 符号约定与投影构造 23
3.3.2 辅助引理 24
3.3.3 梯度估计性质 25
3.3.4 二次函数上的精确分析 26
3.3.5 收敛性分析 27
3.4 实验结果与分析 32
3.4.1 实验设置 32
3.4.2 性能评估 33
3.4.3 消融实验 34
3.4.4 进一步分析 35
3.5 本章小结 38
第 4 章 基于极坐标向量量化的优化器状态压缩算法 39
4.1 引言 39
4.2 算法介绍 40
4.2.1 低精度优化器 40
4.2.2 从标量量化到向量量化 42
4.2.3 二维向量量化框架 44
4.2.4 数据驱动的极坐标码本设计 45
4.3 实验结果与分析 49
4.3.1 实验设置 49
4.3.2 主要结果 49
4.3.3 消融实验 51
4.3.4 进一步分析 54
4.4 本章小结 55
第 5 章 算法集成 56
5.1 引言 56
5.2 实验设置 56
5.3 实验结果 57
5.3.1 SGDM 优化器 57
5.3.2 AdamW 优化器 58
5.4 分析与讨论 59
5.4.1 时间效率分析 59
5.4.2 参考基准分析 60

5.4.3 显存优化的协同效应 60
5.5 本章小结 60
第 6 章 总结与展望 61
6.1 工作总结 61
6.2 未来展望 62
附录 A SubZero 算法补充内容 64
附录 B Polaris 算法补充内容 69
参考文献 75
在学期间完成的相关学术成果 82
致 谢 83
答辩委员会名单 84

插图清单

图 1.1 本文研究内容 5
图 3.1 LLM 微调过程中梯度矩阵的低秩结构 20
图 3.2 不同模型与设置下的训练损失曲线对比 33
图 3.3 提示微调方案下 OPT-1.3B 模型在 SST-2 数据集上的实验结果对比 37
图 3.4 OPT-1.3B 在 SST-2 数据集上采用提示微调时的训练指标对比 37
图 4.1 标量量化的局限性与向量量化的优势 42
图 4.2 LLaMA 预训练过程中 AdamW 一阶动量的分布特性分析 43
图 4.3 二维向量量化映射的可视化对比 44
图 4.4 优化器状态的极坐标向量量化和反量化示意图 45
图 4.5 引理证明几何示意图 46
图 4.6 C4 和 OpenWebText 数据集上的验证困惑度曲线 50
图 4.7 在 Alpaca 数据集上微调大语言模型的训练损失曲线 52
图 4.8 LLaMA-130M 模型动量状态在不同量化技术下的性能对比图 55
图 5.1 训练损失曲线（SGDM，OPT-1.3B） 58
图 5.2 训练损失曲线（SGDM，OPT-13B） 58
图 5.3 训练损失曲线（AdamW，OPT-1.3B） 59
图 5.4 训练损失曲线（AdamW，OPT-13B） 59

附表清单

表 3.1 不同投影矩阵生成方案的性能对比（准确率%） 21
表 3.2 RoBERTa-large 在 SST-2 上全参数微调的显存开销对比..... 21
表 3.3 SuperGLUE 基准上 OPT-13B 微调性能对比..... 33
表 3.4 不同模型与微调方案的性能对比 34
表 3.5 非可微目标下的微调性能对比 35
表 3.6 正交投影矩阵效果对比 35
表 3.7 子空间更新频率 F 与秩 r 的影响..... 35
表 3.8 PEFT 方案中非方阵重塑策略的效果 35
表 3.9 OPT-13B 微调的峰值显存 (GB) 与运行时间 (分钟) 对比..... 36
表 3.10 OPT 系列模型在 SST-2 数据集上微调的峰值显存与运行时间对比 ... 36
表 3.11 随机种子对 OPT-1.3B 提示微调的影响 37
表 3.12 批量大小对 RoBERTa-large 全参数微调的影响 38
表 4.1 LLaMA-130M 模型训练过程中 AdamW 优化器状态静态量化误差对比48
表 4.2 C4 与 OpenWebText 预训练任务结果对比 50
表 4.3 ImageNet-1K 分类任务结果对比 51
表 4.4 Alpaca 微调后 GLUE 基准上的性能对比..... 52
表 4.5 2.0 比特 AdamW 动量不同向量量化策略的验证困惑度对比 53
表 4.6 1.5 比特量化下不同优化器与缩放因子 α 的验证困惑度对比..... 53
表 4.7 不同分块大小对验证困惑度与显存占用的影响 54
表 5.1 算法集成验证实验方案 57
表 5.2 SGDM 优化器下的不同实验方案性能对比..... 58
表 5.3 AdamW 优化器下的不同实验方案性能对比 59

算法清单

算法 2.1　MeZO 算法[16]……9
算法 2.2　低比特 AdamW 优化器……14
算法 3.1　GenerateProjMatrix(m, n, r)……22
算法 3.2　PerturbParams($\mathcal{W}, \mathcal{U}, \mathcal{V}, r, \varepsilon, s$)……22
算法 3.3　SubZero 算法……22
算法 4.1　一阶动量码本搜索（有符号数据）……47
算法 4.2　二阶动量码本搜索（无符号数据）……47
算法 B.1　低比特 Adafactor 优化器……71
算法 B.2　Polaris 低比特 AdamW 优化器……72
算法 B.3　Polaris 低比特 Adafactor 优化器……73

本文符号说明

a, A	非粗体字母，表示标量
$[T]$	集合 $\{1, 2, \cdots, T\}$
$\boldsymbol{a} \in \mathbb{R}^d$	粗体小写字母，表示 d 维列向量
$(a_1, \cdots, a_d)^\top$	列向量的行表示形式
x_i	向量 $\boldsymbol{x}$ 的第 i 个元素
$\boldsymbol{X} \in \mathbb{R}^{m \times d}$	粗体大写字母，表示 $m \times d$ 矩阵
$\boldsymbol{x}_i$	矩阵 $\boldsymbol{X}$ 的第 i 列向量
$\boldsymbol{A}^\top$	矩阵 $\boldsymbol{A}$ 的转置
$\boldsymbol{I}_m$	$m \times m$ 单位矩阵
$\mathrm{vec}(\boldsymbol{A})$	矩阵 $\boldsymbol{A}$ 的向量化操作（垂直堆叠各列）
$\mathrm{bdiag}(\boldsymbol{A}_1, \cdots, \boldsymbol{A}_l)$	以 $\boldsymbol{A}_1, \cdots, \boldsymbol{A}_l$ 为对角块的分块对角矩阵
$\boldsymbol{A} \otimes \boldsymbol{B}$	矩阵 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 的 Kronecker 积
$\boldsymbol{A} \odot \boldsymbol{B}$	矩阵 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 的逐元素乘积（Hadamard 积）
$\langle \boldsymbol{A}, \boldsymbol{B} \rangle$	矩阵 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 的内积
$\|\boldsymbol{x}\|_p$	向量 $\boldsymbol{x}$ 的 ℓ_p-范数
$\|\boldsymbol{x}\|$ 或 $\|\boldsymbol{x}\|_2$	向量 $\boldsymbol{x}$ 的 ℓ_2-范数，即 $\sqrt{\sum_{i=1}^d x_i^2}$
$\|\boldsymbol{A}\|$ 或 $\|\boldsymbol{A}\|_2$	矩阵 $\boldsymbol{A}$ 的谱范数
$\|\boldsymbol{A}\|_F$	矩阵 $\boldsymbol{A}$ 的 Frobenius 范数，即 $\sqrt{\langle \boldsymbol{A}, \boldsymbol{A} \rangle}$
$\mathcal{N}(\mu, \sigma^2)$	均值为 μ、方差为 σ^2 的一元正态分布
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	均值为 $\boldsymbol{\mu}$、协方差矩阵为 $\boldsymbol{\Sigma}$ 的多元正态分布
$\mathbb{E}[\boldsymbol{x}]$	随机变量 $\boldsymbol{x}$ 的期望
$\mathrm{Var}[\boldsymbol{x}]$	随机变量 $\boldsymbol{x}$ 的方差
$C_L^{s,p}(S)$	集合 S 上 s 阶光滑且 p 阶 L-光滑的函数类

第 1 章 绪论

1.1 研究背景与意义

近年来，以 GPT-5[1]、Qwen-3[2]、DeepSeek-V3[3] 为代表的千亿级参数大语言模型在语言理解、文本生成等任务中取得了显著进展[4]。随着大模型向垂直领域渗透，个性化需求、隐私保护及实时响应的本地化微调需求持续增长[5-7]。为满足这些需求，在显存资源受限的消费级显卡（如 RTX 4090）甚至端侧设备上开展资源高效型微调任务已成为学术界与工业界的共同目标[8-10]。

然而，大模型微调对显存资源提出了极高要求[11]。以 AdamW[12] 为主流的一阶优化算法在反向传播过程中需同时存储模型参数、激活、梯度以及优化器状态，导致显存占用随模型规模急剧增长[13]，难以直接应用于单机单卡或移动嵌入式等资源受限场景[14-15]。例如，全参微调 175B 参数的 GPT-3 需要约 1200GB 显存[6]，远超主流消费级 GPU 的物理容量。

为应对这一问题,学术界与工业界围绕显存高效训练开展了大量研究[8,15-17]。零阶优化是其中的代表性方法之一，其通过前向传播估计梯度，无需反向传播，避免了激活值存储，具有将训练显存降至接近推理水平的潜力[16,18]。零阶优化的核心流程包含梯度估计与优化器状态更新两个关键步骤，然而现有算法在这两个步骤均面临挑战。在梯度估计中，梯度估计方差随模型参数量线性增长，严重制约收敛速度；在优化器状态更新中，基于动量的优化器（如 AdamW）虽能有效加速收敛，但其优化器状态会引入显著的显存开销，甚至可能抵消零阶优化在激活值上的节省。针对梯度估计方差问题，现有研究发展了稀疏扰动[19]、参数高效适配[16]及随机投影[20]等策略减少梯度估计方差；针对优化器状态压缩问题，现有研究通过分解（如 GaLore[15]）、划分（如 Adam-mini[21]）以及量化（如 4 比特 AdamW[17]）等手段压缩优化器状态。

这些显存高效优化算法的涌现，反映出大模型优化算法的研究重心正从单纯追求收敛速度与精度，转向在极端资源受限下寻求性能、稳定性与效率的平衡。现有算法在适配消费级硬件时仍面临理论与实践挑战：零阶优化算法以放弃精确梯度为代价换取显存节省，但其梯度估计的高方差会损害优化过程的稳定性与收敛效率，尤其在参数量极高的大模型中，这一缺陷被显著放大[22]；优化器状态量化压缩技术虽为降低动量存储开销提供了可行路径，但在极低比特（低于 4 比特）场景下，量化误差的累积极易破坏训练动态，导致模型性能下降或训练崩溃。

因此，当前显存高效训练领域的核心问题在于如何在突破硬件显存物理极

限的同时，弥补其带来的稳定性与收敛性损失。本研究基于此背景展开，深入挖掘优化变量的状态特性，从梯度估计与优化器状态更新两个关键环节出发，系统研究显存高效的零阶优化算法。在梯度估计方面，利用梯度矩阵的低秩结构特性，引入随机子空间理论[15,23-24]，在不引入额外显存开销的前提下降低梯度估计方差，使其兼具低显存与高效率的优点；在优化器状态更新方面，依据动量值的分布特性，探索表示能力更强的向量量化技术[25-26]，突破传统标量量化在优化器状态压缩上的比特瓶颈，为零阶优化框架内动量状态极低比特存储提供可行方案。通过将对优化状态特性的洞察转化为算法设计，本研究致力于构建以零阶优化为核心的显存高效优化算法体系，为绿色、普适、可持续的大模型发展提供算法基础。

1.2 国内外研究现状

现有显存高效的零阶优化算法研究主要沿两条技术路径展开：梯度估计方法改进与优化器状态压缩。前者通过设计高效的零阶梯度估计策略，在降低估计方差的同时避免反向传播中的激活值存储；后者通过压缩优化器状态，降低基于动量的优化器的显存占用。本节系统回顾这两类算法的研究现状与核心挑战，剖析其在理论机制和工程实现上的局限，凝练当前研究空白，并明确本文的研究动机与技术切入点。

1.2.1 无需反向传播的零阶优化

零阶优化（Zeroth-Order Optimization, ZO）算法通过前向传播估计梯度，避免了反向传播中的中间激活值存储，是显存高效优化的重要技术路径。零阶优化算法的核心思想是通过对模型权重进行微小扰动并观测损失函数的变化来估计梯度方向。MeZO[16] 首次将零阶优化成功应用于大语言模型（Large Language Model, LLM）微调，其仅依赖前向传播和基于训练损失值的有限差分估计梯度。由于不需要存储中间激活值，MeZO 显著降低了显存占用，在多个任务上实现了与基于反向传播的优化算法相当的微调性能。实验表明，该算法实现了最高 12 倍的显存节省，展示了零阶优化算法在资源受限场景下的应用价值。

然而，ZO 算法的核心问题在于梯度估计的高方差。理论上，其梯度估计的方差与模型参数的维度成线性关系[22,27]。当应用于参数量动辄数十亿甚至千亿的 LLM 时，巨大的参数维度会导致梯度估计产生剧烈抖动，严重拖慢模型收敛速度，并可能损害最终收敛精度，使其在效率和性能上落后于一阶算法[22,28]。

理论研究表明，减少梯度方差可加速收敛[29]。针对这一高方差问题，当前研究主要从两个维度展开改进。一是调整训练批次大小，实验表明增大批次大小

可有效抑制梯度噪声及其方差[22]，但这种方法导致计算时间和显存消耗呈超线性增长，实际应用效率受限。二是通过稀疏的参数扰动策略降低有效扰动维度，包括基于随机掩码的稀疏扰动[19]和块坐标扰动[18]；或直接压缩可训练参数的空间，包括结合参数高效微调（Parameter Efficient Fine-Tuning, PEFT）框架[16,18]以及引入额外的张量分解适配器[30]。值得关注的是，近期理论研究[20,31]提出利用随机投影技术将高维参数扰动映射至低秩子空间，可在理论上突破 ZO 梯度估计方差对参数维度的线性依赖。但该技术需要存储与模型参数维度相当的投影矩阵，对于参数量超过千亿的现代 LLM 而言，其显存开销仍然难以承受。因此，如何设计一种存储代价较小的低秩投影机制，从而在获得低秩子空间方差减少收益的同时不引入显著的显存开销，是当前零阶优化领域需要解决的关键问题。

1.2.2 进一步降低零阶优化显存的技术

在零阶优化框架内，当与基于动量的优化器（如 AdamW）结合时，动量机制的引入通常能有效加速收敛并提升模型性能，但优化器动量状态的存储会引入显著的显存开销，甚至可能抵消零阶优化在前向传播中对激活显存的节省。因此，优化器状态的压缩成为零阶优化显存高效训练的关键环节之一[32]。现有优化器状态压缩工作可分为分解、划分和量化三条路线。

基于分解的方法通过结构近似减少状态存储。Adafactor[33] 以外积近似二阶矩，避免显式维护全维度二阶动量；GaLore[15] 从梯度投影入手，在低秩子空间中运行 Adam，从源头降低优化器状态规模；其改进版本 Q-GaLore[34] 在此基础上进一步对权重与投影矩阵进行低比特量化，压缩显存。

基于划分的方法通过对参数集合进行覆盖或分块，让多个参数共享统计量，从而减少需要维护的独立状态。SM3[35] 使用覆盖统计量近似二阶矩；Adam-mini[21] 通过按 Hessian 最小稠密子块进行参数分块，实现二阶矩的共享与复用，降低独立状态规模。

基于量化的方法是最直接的压缩手段，可与分解和划分方法结合使用。其通过将高精度优化器状态压缩为低比特表示以降低存储开销[36]。当前应用最广泛的是标量量化。例如，Dettmers 等人[37] 提出的 8 比特优化器采用分块动态量化策略，将一阶和二阶动量压缩至 8 比特，在保持训练性能的同时减少显存占用；Li 等人[17] 进一步将 AdamW 中的二阶动量压缩至 4 比特，并提出无零点映射方法以缓解因零点引入的量化误差。

然而，标量量化的表示能力存在理论上限。当压缩率进一步提升至 4 比特以下时，其有限的离散取值空间难以精确捕捉优化器状态的细微变化，累积的量化误差易导致训练过程不稳定。

与此相对，向量量化在理论上具有更强的表示能力。它将多个元素组成向

量进行联合量化，利用维度间的相关性，在同等比特率下能实现更低的失真。Bucklew 等人[25] 已指出，对于近似服从高斯分布或具有球对称特性的随机变量，采用极坐标或球坐标的多维向量量化在均方误差意义上优于传统的一维标量量化。目前，向量量化已在模型推理的权重和 KV 缓存压缩上取得成功（如 AQLM[38]、CommVQ[39] 等工作），实现了 2-3 比特下的高保真压缩。近年来，Tian 等人[40] 尝试将二维几何映射应用于快速二维向量量化结构，但由于理论局限尚不成熟，其在 AdamW 状态量化中仅能实现约 3.3 比特的表示。

尽管向量量化在模型推理压缩上取得了进展，但其在模型训练，特别是针对优化器状态的压缩上，尚未得到充分研究。核心难点在于优化器状态是高频更新的动态变量，参与逐元素运算而非矩阵乘法运算，对量化误差的容忍度远低于相对静态的模型权重。现有研究尚未系统性地探索如何为 AdamW 等优化器的动态状态设计高效、稳定的向量量化方案。因此，如何设计面向优化器状态的码本结构、更新策略与误差补偿机制，以在维持训练稳定性的前提下将动量状态压缩至 2-3 比特的极低水平，是一个需要深入研究的方向。

1.3 本文主要工作

尽管现有显存高效的零阶优化算法在降低训练开销方面已取得一定进展，但在实际应用中仍面临两方面局限：一是零阶优化在大模型场景下的梯度估计方差过大问题，现有方法难以在不增加额外存储的前提下实现稳定收敛；二是优化器状态量化在极低比特条件下的稳定性问题，目前尚缺乏面向动量状态的有效压缩方案。究其根本，现有研究未能充分挖掘优化状态自身的内蕴特性。

基于以上分析，本文以优化状态的内蕴特性为研究主线，从梯度估计与优化器状态更新两个关键环节出发，探索面向大模型微调的显存高效零阶优化算法。整体研究架构如图 1.1所示，主要研究内容涵盖以下三个方面：

(1) 基于低秩子空间的大模型零阶梯度估计算法——SubZero。针对零阶优化算法在大模型微调中梯度估计方差大、收敛性能受限的问题，本文提出了基于随机子空间的零阶梯度估计算法（Random Subspace Zeroth-Order Optimization, SubZero）。该算法利用梯度矩阵的低秩结构，通过构造列正交投影矩阵在低维子空间内生成扰动，将梯度估计方差从依赖模型总参数量降至仅依赖子空间维度。理论分析与实验结果表明，SubZero 在显存开销与 MeZO 相当的前提下，其梯度估计能够紧密逼近反向传播梯度，显著降低了梯度估计方差。多种大语言模型微调实验结果表明，该算法在显存开销与现有零阶算法相当的前提下，收敛速度与微调性能均显著优于同类算法，为提升零阶优化在大模型微调中的梯度估计质量提供了有效途径。

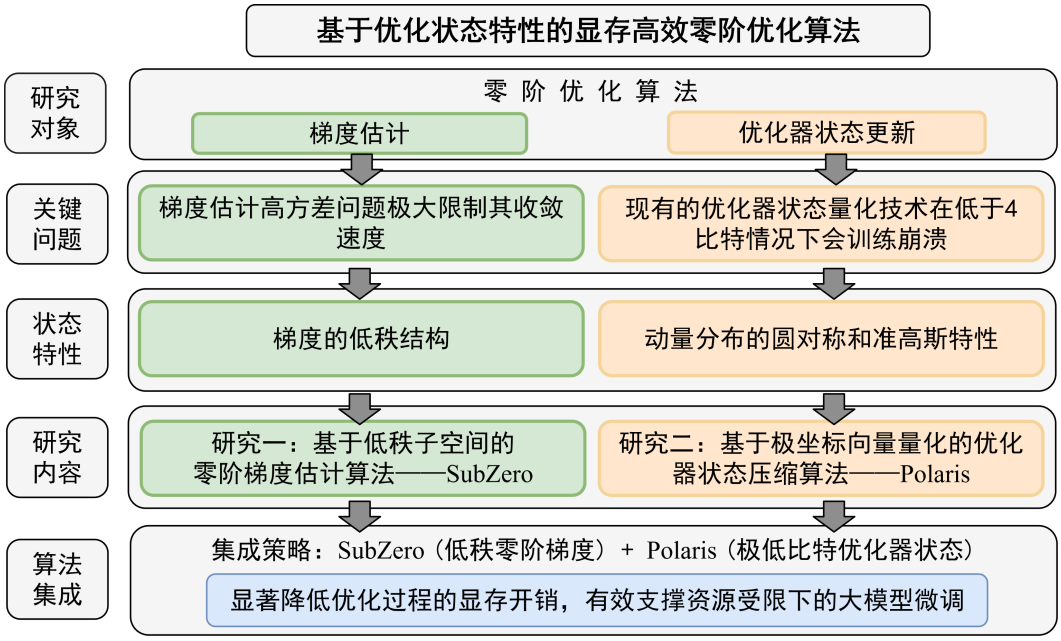


图 1.1　本文研究内容

(2) 基于极坐标向量量化的优化器状态压缩算法——Polaris。针对基于动量的优化器状态显存占用大、且现有量化方法在低于 4 比特时易引发训练崩溃的问题，本文提出了基于极坐标向量量化的优化器状态压缩算法（Polar Vector Quantization-based Optimizer State Compression, Polaris）。该算法利用动量值分布的圆对称性与准高斯特性，构建二维极坐标量化框架：针对有符号一阶动量采用角度均匀分布码本确保方向精度，针对无符号二阶动量设计第一象限映射、自适应角度分配及轴偏移防除零机制。在涵盖图像分类、自然语言建模等一系列预训练与微调任务中的实验结果表明，Polaris 在将优化器状态压缩至 1.5 比特与 2 比特后仍能稳定训练，达到了与全精度版本相当的性能表现，为在零阶优化框架中实现优化器状态极低比特存储提供了关键技术手段。

(3) 基于 SubZero 与 Polaris 的集成验证。本文进一步将 SubZero 与 Polaris 进行算法集成，在大语言模型微调任务中验证二者的协同显存优化效果。围绕梯度估计器、优化器类型与优化器状态压缩三个正交维度展开实验，系统评估不同组合方案在显存占用、收敛速度及模型性能等方面的表现。实验结果表明，SubZero 与 Polaris 的协同应用能够在显著降低显存开销的同时保持收敛速度和模型性能，验证了二者在显存效率与优化性能上的显著协同优势，为资源受限场景下的大模型微调提供了显存高效的实用路径。

1.4 论文结构安排

本文主要研究基于优化状态特性的显存高效零阶优化算法，共分为六个章节，各章节内容安排如下：

第一章，介绍大模型微调面临的显存挑战及其研究意义，回顾零阶优化算法与优化器状态压缩技术的国内外研究现状，分析梯度估计方差大与量化比特受限等核心挑战，在此基础上阐述本文的研究内容与创新点。

第二章，介绍优化理论与算法基础，包括优化问题的数学表述、深度学习训练显存占用分析、零阶优化与基于动量的优化算法原理。综述量化压缩以及低秩子空间在深度学习中的应用，为后续研究提供理论支撑与技术背景。

第三章，针对零阶优化梯度估计方差大的问题，提出基于随机子空间的零阶梯度估计算法——SubZero。介绍分层低秩扰动机制、子空间懒惰更新策略等核心组件，并进行算法收敛性分析。通过多种大语言模型及微调方案的实验验证算法的有效性与优越性。

第四章，针对基于动量的优化器状态显存占用大、且现有量化方法在低于 4 比特时易引发训练崩溃的问题，提出基于极坐标向量量化的优化器状态压缩算法——Polaris。阐述二维极坐标量化框架、有符号与无符号数据的设计原则等核心内容，通过静态误差分析与动态训练实验验证算法在极低比特下的稳定性与性能。

第五章，将 SubZero 与 Polaris 进行系统集成，在大语言模型微调任务上验证协同显存优化效果。围绕梯度估计器、优化器类型、优化器状态压缩三个正交维度设计多组对比实验，量化 SubZero 低秩梯度估计与 Polaris 极坐标向量量化的协同显存优化效应，验证两种技术在显存效率与优化性能上的协同优势。

第六章为总结与展望。总结全文工作，分析当前研究存在的局限性及可改进之处，为未来的研究工作明确方向。

第 2 章 相关理论基础

本章系统阐述支撑本文研究的核心理论基础，构建相关理论框架。首先，从深度学习训练的优化问题数学表述出发，深入剖析零阶优化算法和基于动量的优化算法的原理与显存占用；其次，针对大模型训练中的显存瓶颈，系统梳理量化压缩在深度学习训练和推理中的应用，重点阐述标量量化与向量量化的数学基础及在极低比特优化器状态压缩中的应用潜力；进而，深入探讨低秩子空间理论在深度学习中的多重应用，特别是梯度矩阵低秩性对优化算法设计的启示。这些理论为本文算法设计提供理论基础。

2.1 优化理论与算法基础

2.1.1 优化问题的数学表述

深度学习模型训练本质上是一个大规模非凸随机优化问题。给定训练数据集 $\mathcal{D}=\{(\boldsymbol{x}_i,y_i)\}_{i=1}^{n}$，其中 $\boldsymbol{x}_i\in\mathbb{R}^{d_{\text{in}}}$ 为输入样本，$y_i\in\mathcal{Y}$ 为对应标签，n 为样本总数。深度学习模型由参数向量 $\boldsymbol{\theta}\in\mathbb{R}^{d}$ 表征，其中 d 为参数量，在现代大语言模型中通常达到数十亿甚至千亿级别。模型训练的目标是最小化经验风险函数：

$$\min_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})=\frac{1}{n}\sum_{i=1}^{n}\ell(f(\boldsymbol{x}_i;\boldsymbol{\theta}),y_i), \tag{2-1}$$

其中 $\ell:\mathcal{Y}\times\mathcal{Y}\to\mathbb{R}_{+}$ 为损失函数（如交叉熵损失），$f(\cdot;\boldsymbol{\theta})$ 为模型预测函数。由于目标函数 $\mathcal{L}(\boldsymbol{\theta})$ 高度非凸且参数量 d 巨大，直接求解全局最优解在计算上不可行。因此，实际训练中采用基于迭代的随机优化算法，通过以下通用更新规则逼近局部最优解：

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\eta_t\boldsymbol{g}_t, \tag{2-2}$$

其中 $\eta_t>0$ 为学习率，$\boldsymbol{g}_t$ 为第 t 次迭代时目标函数关于参数的随机梯度估计。它通常是在一个小批量数据上计算得到的梯度。

2.1.2 深度学习训练显存占用

在深度学习训练中，显存占用主要由四个部分构成：模型参数、优化器状态、梯度和激活值[11,41-42]。以参数量为 d、使用 AdamW 优化器、混合精度（FP16/BF16）训练为例，各组件显存占用分析如下：

- 模型参数：在混合精度训练中，模型参数通常以 16 位浮点数（FP16 或 BF16）存储，占用显存为 $2d$ 字节。为保持数值稳定性，优化器通常维护一份 FP32

格式的参数副本，额外占用 $4d$ 字节。因此，参数相关显存总共约 $6d$ 字节。

- 优化器状态：这是 AdamW 等自适应优化器显存消耗的主要来源。AdamW 需维护一阶动量（动量项）$\boldsymbol{m}_t$ 和二阶动量（自适应学习率项）$\boldsymbol{v}_t$，两者均以 FP32 格式存储，各占用 $4d$ 字节。因此，优化器状态显存开销为 $8d$ 字节，是模型参数显存（FP16）的 4 倍。对于 70B 参数的 LLaMA 模型，仅优化器状态就需消耗约 560GB 显存，远超单卡容量。
- 梯度：在反向传播过程中，梯度通常以 FP16 格式存储，占用 $2d$ 字节。若采用梯度累积或分布式训练，可能需要额外的梯度缓冲区。
- 激活值：激活值是反向传播计算梯度所必需的中间结果，其显存占用与模型深度、批次大小、序列长度及隐藏维度密切相关。对于 Transformer 架构，每层激活值主要包括三类：一是注意力层的查询、键、值张量；二是注意力分数矩阵，其显存需求随序列长度呈平方增长，是长序列训练的主要瓶颈；三是前馈网络的中间结果，其维度通常为隐藏维度的数倍。堆叠全部层后，大模型的激活值显存往往远超参数显存。

综合上述分析，使用 AdamW 训练大模型时，单设备总显存需求可近似为：

$$\text{总显存} \approx \underbrace{6d}_{\text{参数}} + \underbrace{8d}_{\text{优化器状态}} + \underbrace{2d}_{\text{梯度}} + \text{激活值显存}. \tag{2-3}$$

以 7B 参数模型为例，总显存需求可达 120–150GB，远超消费级 GPU（如 RTX 4090 的 24GB）容量[11]，甚至需要多卡 A100（80GB）才能容纳。这一“显存墙”问题构成了本文研究的核心动机。

2.1.3 零阶优化算法

与依赖反向传播的一阶优化算法不同，零阶优化算法仅通过查询目标函数值来估计梯度，无需反向传播计算梯度，从根本上避免了存储中间激活值[43]。这使得零阶优化算法在显存受限场景下具有独特优势，其显存占用可降至接近推理水平。

最经典的零阶梯度估计方法基于同时扰动随机逼近（Simultaneous Perturbation Stochastic Approximation, SPSA）。对于光滑函数 $\mathcal{L}(\boldsymbol{\theta})$，其梯度可通过以下有限差分公式近似：

$$\hat{\nabla}\mathcal{L}(\boldsymbol{\theta};\boldsymbol{u}) = \frac{\mathcal{L}(\boldsymbol{\theta}+\mu\boldsymbol{u}) - \mathcal{L}(\boldsymbol{\theta}-\mu\boldsymbol{u})}{2\mu}\boldsymbol{u}, \tag{2-4}$$

其中 $\mu > 0$ 为扰动幅度（平滑参数），$\boldsymbol{u} \in \mathbb{R}^d$ 为随机扰动向量，通常服从标准高斯分布 $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$ 或 Rademacher 分布（取值为 ± 1）。该估计器的期望等于真实梯度的平滑版本：

$$\mathbb{E}_{\boldsymbol{u}}[\hat{\nabla}\mathcal{L}(\boldsymbol{\theta};\boldsymbol{u})] = \nabla\mathcal{L}_{\mu}(\boldsymbol{\theta}) \approx \nabla\mathcal{L}(\boldsymbol{\theta}), \tag{2-5}$$

其中 $\mathcal{L}_{\mu}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{u}}[\mathcal{L}(\boldsymbol{\theta} + \mu\boldsymbol{u})]$ 为高斯平滑后的目标函数。

基于该梯度估计器的理论分析表明，SPSA 的收敛速率与参数维度 d 密切相关。事实上，这一维度依赖性在零阶优化算法中普遍存在。针对前述光滑非凸优化问题 (2-1)，现有文献给出了明确的复杂度界。Nesterov 和 Spokoiny[27] 提出的零阶梯度下降算法，利用两点高斯随机梯度估计器，其收敛速率为 $\mathcal{O}(d/K)$（K 为迭代次数），达到 ϵ-驻点所需的函数查询复杂度高达 $\mathcal{O}(dn/\epsilon)$，其中 n 为训练样本数量。此外，Ghadimi 和 Lan[44]提出的零阶随机梯度下降算法，虽将收敛速率提升至 $\mathcal{O}(\sqrt{d/K})$，但其函数查询复杂度仍为 $\mathcal{O}(d/\epsilon^2)$。这种对维度 d 的依赖意味着：当模型参数量达到数十亿规模时，所需的函数查询次数将呈线性增长，导致计算开销难以承受且收敛效率极低。因此，如何突破维度依赖性是零阶优化应用于大模型的关键挑战。

算法 2.1 MeZO 算法[16]

输入： 模型参数 $\boldsymbol{\theta} \in \mathbb{R}^d$，损失函数 $\mathcal{L}$，总步数 T，扰动尺度 ε，学习率调度 $\{\eta_t\}$，批量大小 B

输出： $\boldsymbol{\theta}$

for $t = 0, 1, \cdots, T-1$ **do**
　采样小批量 $\mathcal{B}^t \subset \mathcal{D}$ 和随机种子 s^t
　$\boldsymbol{\theta} \leftarrow \text{PerturbParams}(\boldsymbol{\theta}, \varepsilon, s^t)$, $\ell_+^t \leftarrow \mathcal{L}(\boldsymbol{\theta}; \mathcal{B}^t)$　// 正向扰动并计算损失
　$\boldsymbol{\theta} \leftarrow \text{PerturbParams}(\boldsymbol{\theta}, -2\varepsilon, s^t)$, $\ell_-^t \leftarrow \mathcal{L}(\boldsymbol{\theta}; \mathcal{B}^t)$　// 反向扰动并计算损失
　$\boldsymbol{\theta} \leftarrow \text{PerturbParams}(\boldsymbol{\theta}, \varepsilon, s^t)$　// 恢复原始参数
　$projected\_grad \leftarrow (\ell_+^t - \ell_-^t)/(2\varepsilon)$　// 投影梯度估计
　用种子 s^t 重置随机数生成器　// 复现扰动向量
　for 每个参数 $\theta_i \in \boldsymbol{\theta}$ **do**
　　生成 $z_i \sim \mathcal{N}(0,1)$, $\theta_i \leftarrow \theta_i - \eta_t \cdot projected\_grad \cdot z_i$　// 原地参数更新
　end for
end for
子程序 $\text{PerturbParams}(\boldsymbol{\theta}, \epsilon, s)$:
　用种子 s 重置随机数生成器　// 确保扰动可复现
　for 每个参数 $\theta_i \in \boldsymbol{\theta}$
　　生成 $z_i \sim \mathcal{N}(0,1)$, $\theta_i \leftarrow \theta_i + \epsilon \cdot z_i$　// 原地修改参数
　end for
　return $\boldsymbol{\theta}$

MeZO[16] 是首个将零阶优化算法应用于 LLM 微调的工作，其微调显存开销与推理阶段相近，算法流程详见算法 2.1。MeZO 将经典零阶随机梯度下降改造为原地操作，通过随机种子复现技术即时生成扰动向量，无需持久化存储，将扰动向量的显存开销从 $\mathcal{O}(d)$ 降至 $\mathcal{O}(1)$。在优化器设计上，MeZO 直接复用 SGD[45] 或 Adam[46] 等标准优化器的状态变量（如动量缓冲区），无需额外设计专用优化器。更重要的是，MeZO 仅需前向传播计算损失值，完全避免了反向传播所需的中间激活值存储。从显存占用角度分析，MeZO 的显存主要包括模型参数（FP16）

$2d$、可选的动量状态（FP32）$4d$（若使用带动量的 SGD 作为优化器），以及极小的运行时开销 $\mathcal{O}(1)$，总显存约为 $2d$ 至 $6d$ 字节。在单卡 A100（80GB）上，MeZO 可训练 30B 参数的模型，而基于反向传播的微调仅能训练 2.7B 参数的模型[16]。实验表明，MeZO 在多种任务上性能与基于反向传播的优化算法相当，且兼容 LoRA 等参数高效微调技术，还可直接优化准确率等不可微目标。

尽管 MeZO 在显存效率上取得了显著突破，但其局限性仍不容忽视。首先，MeZO 依赖于全空间随机扰动，其理论收敛速率受参数维度 d 的制约，函数查询复杂度为 $\mathcal{O}(d/\epsilon^2)$[16,44]。当 d 达到数十亿时，所需的函数查询次数极高，导致收敛缓慢，需要极小的学习率和精细的扰动幅度调参。其次，研究表明，在复杂下游任务（如多步推理等）上，MeZO 的微调性能与基于反向传播的优化算法仍存在一定差距[16]。因此，如何在保持显存效率的同时突破维度依赖性，仍是零阶优化应用于大模型的关键挑战。

2.1.4 基于动量的优化算法

基于动量的优化算法通过累积历史梯度信息来加速收敛，是大模型训练的主流选择[47]。代表性的带动量的 SGD[48]（SGD with Momentum, SGDM）引入一阶动量状态 $\boldsymbol{m}_t$，其更新规则可简化为：

$$\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + (1-\beta)\boldsymbol{g}_t, \tag{2-6}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \boldsymbol{m}_t, \tag{2-7}$$

其中 $\beta \in [0,1)$ 为动量系数（通常为 0.9）。此时优化器状态显存为 $4d$ 字节（FP32）。

Adam[46]及其变体 AdamW[12]通过维护一阶动量 $\boldsymbol{m}_t$ 和二阶动量 $\boldsymbol{v}_t$ 两个状态向量，为每个参数自适应调整学习率。以 AdamW[12] 为例，其更新规则可简化为：

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t, \quad \boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2, \tag{2-8}$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t \left(\frac{\boldsymbol{m}_t/(1-\beta_1^t)}{\sqrt{\boldsymbol{v}_t/(1-\beta_2^t)} + \epsilon} + \lambda \boldsymbol{\theta}_{t-1} \right), \tag{2-9}$$

其中 β_1, β_2 为衰减率（通常分别为 0.9 和 0.999），$\epsilon \approx 10^{-8}$ 为数值稳定常数，λ 为权重衰减系数。从显存占用角度分析，AdamW 需存储 $\boldsymbol{m}_t$ 和 $\boldsymbol{v}_t$ 两个 FP32 状态向量，显存开销为 $8d$ 字节。以 7B 参数模型为例，AdamW 优化器状态的存储需求可达 56GB，远超消费级 GPU 容量。这一严峻的显存挑战催生了对优化器状态压缩技术的广泛研究，量化便是其中的代表性方向。

2.2 量化压缩在深度学习中的应用

量化是模型压缩领域中的核心技术[11]，旨在通过降低数值精度（例如从 32 位浮点数 FP32 降至 8 位整数 INT8 或更低）来减少模型的存储与计算开销，同时尽量保持模型性能[49]。该技术特别适用于资源受限的设备（如移动设备、嵌入式系统）以及大规模深度学习模型的训练与推理场景。量化的数学本质在于构建映射函数，将连续空间中的实数离散化至有限码本集合，通常借助缩放因子与零点完成线性或非线性变换[50]。量化技术的优势主要体现在三个方面：首先，在显存效率方面，通过减少参数、梯度及优化器状态的存储需求，有效缓解大模型训练的“显存墙”问题[17,37]；其次，在计算性能方面，利用低精度整数运算单元（如 Tensor Core）提升计算吞吐量，降低推理延迟[13,51-52]；最后，在能源效率方面，降低数据搬运与计算过程中的能量消耗，提升能效比[11]。

2.2.1 标量量化与向量量化

量化技术可分为标量量化（Scalar Quantization, SQ）与向量量化（Vector Quantization, VQ）[53-54]。两者在编码方式、率失真性能及计算复杂度上存在显著差异，适用于不同的应用场景。

标量量化独立处理每个数值元素，是最基础且应用最广泛的量化形式。给定浮点数 $x \in \mathbb{R}$，量化函数 $\texttt{quantize}(x)$ 将其映射到有限码本 $\mathcal{C} = \{c_1, c_2, \cdots, c_k\} \subset \mathbb{R}$ 中最近的值，即 $\texttt{quantize}(x) = \arg\min_{c \in \mathcal{C}} |x - c|$。以神经网络权重 $\boldsymbol{W}$ 为例，标量量化通常通过对称均匀量化实现：

$$\mathrm{SQ}(\boldsymbol{W}) = \mathrm{clamp}\left(\left\lfloor \frac{\boldsymbol{W}}{s} \right\rceil, -2^{b-1}, 2^{b-1} - 1\right), \quad s = \frac{\max(|\boldsymbol{W}|)}{2^{b-1} - 1}, \tag{2-10}$$

其中 s 为缩放因子，$\lfloor \cdot \rceil$ 表示四舍五入运算符，b 为量化位宽。从信息论角度，标量量化的均方误差随比特率呈指数衰减[55-56]。但当比特率极低（如 1 或 2 比特）时，码本容量极小，难以准确拟合复杂分布，量化误差急剧增大。尽管 GPTQ[57]、AWQ[58]、QuaRot[52] 等算法通过逐层量化、异常值保护及旋转操作提升了精度，但标量量化技术通常仅限于 3–4 比特级别。当涉及到极低比特压缩时，标量量化的均方误差在高压缩率下迅速恶化。

相比之下，向量量化将多个元素组成向量进行联合量化，利用维度间的相关性，在相同比特率下获得更低的量化失真。给定待量化权重 $\boldsymbol{W} \in \mathbb{R}^{p \times q}$，VQ 将其重塑为维度为 $(p \cdot q/k, k)$ 的 $\boldsymbol{W}'$，对于每个 k 维行向量，用码本 $\boldsymbol{C} \in \mathbb{R}^{2^n \times k}$ 中最近向量的索引替换。量化过程可表示为：

$$\mathrm{VQ}(\boldsymbol{W}') = \left\{ \underset{j \in 2^n}{\mathrm{argmin}} \|\boldsymbol{W}'_{i,:} - \boldsymbol{C}_{j,:}\|_F \mid i = 1, \cdots, p \cdot q/k \right\}, \tag{2-11}$$

码本 $\boldsymbol{C}$ 的行向量代表聚类中心，当前研究提出了多种优化策略：VPTQ[59] 和

GPTVQ[60] 分别通过 K-Means 和期望最大化算法聚类源向量；AQLM[38]利用逐层训练码本以获得更高准确性；QuIP#[61] 则应用 Hadamard 旋转抑制异常值并引入预定义全局码本。向量量化在理论上有更好的率失真性能，尤其适合 2 位及以下的低比特场景，但主要挑战在于码本搜索复杂度为 $\mathcal{O}(2^n \cdot k)$，当码本规模较大时需要近似最近邻搜索加速。

综上所述，标量量化实现简单但在极低比特下误差较大，向量量化利用维度相关性可获得更好的压缩性能但计算开销较高。

2.2.2 模型张量的量化压缩

在大模型训练过程中，需要存储的主要张量包括模型权重、激活值、梯度和优化器状态[13]。量化通过对这些张量进行低精度表示，可显著降低显存占用。根据是否参与训练过程，量化方法主要分为训练后量化（Post-Training Quantization, PTQ）与量化感知训练（Quantization-Aware Training, QAT）两类范式[11,62]。

PTQ 方法无需重新训练即可部署低比特模型，在高效性上具有明显优势[11]。例如，GPTQ[57]采用二阶误差补偿策略，对大语言模型权重进行逐层校准，在 4 比特量化下保持 90% 以上原始精度；SmoothQuant[63] 通过激活值平滑缩放解决异常值问题，显著提升了 Transformer 模型的 8 比特量化效果；Duquant[51] 利用旋转和置换变换减轻激活值异常值的影响，在 4 位权重与激活值量化条件下实现了先进性能。然而，PTQ 在极低比特（$\leqslant$ 4 比特）场景中仍存在明显的精度损失，目前最先进的方法在 4 比特量化下仍难以完全恢复模型的原始精度，这表明硬件计算支持与实际可达到的模型性能之间仍存在显著差距[51-52]。

QAT 方法将优化环节前置至训练阶段，在前向传播中将权重和激活值量化为低比特表示，反向传播使用直通估计器近似量化操作的梯度，从而实现对量化模型的优化[11]。BitNet 系列[64-65]是极低比特大模型量化的开创性工作。BitNet[64] 训练具有二值权重的神经网络，BitNet b1.58[65] 训练具有三值权重的神经网络，两者均考虑激活值为整数表示，进一步减少显存开销并加速前向过程。BitNet A4.8[66] 结合了 1 比特权重和 4 比特激活值，展示了在 LLM 中实现高效训练和推理的可能性。这些研究不仅推动 QAT 技术发展，也为量化技术在大模型中的未来应用开辟新思路。

需要强调的是，PTQ 和 QAT 主要针对模型权重和激活值进行量化，并不涉及优化器状态的压缩。即使在 QAT 框架下进行全训练流程优化，AdamW 等自适应优化器仍需维护 FP32 格式的一阶和二阶动量状态，显存开销并未减少。因此，PTQ/QAT 与优化器状态量化是正交的技术方向，前者面向推理部署或训练过程中的权重/激活压缩，后者面向训练过程中的优化器显存优化。本文关注的优化器状态量化可与 PTQ/QAT 结合使用，进一步降低训练显存峰值。

基于上述量化范式，不同类型张量的量化难度、研究热度及实际收益存在显著差异，这取决于其生命周期、显存占比及对量化误差的敏感度。

模型权重量化研究最为成熟。权重在训练完成后相对静态，分布稳定，适合 PTQ 方法。由于权重在整个训练和推理过程中持久存在，其量化收益可直接转化为存储和带宽的节省，因此成为量化研究的重要目标[11]。

激活值量化主要用于推理加速和部分训练场景。激活值在前向传播中动态生成，分布随输入变化，量化难度高于权重[11]。在训练场景中，混合精度训练已广泛采用 FP16/BF16 存储激活值，将显存占用减半。然而，由于数值稳定性考虑，激活值通常不低于 8 比特，更低比特会导致梯度计算失真，影响收敛[14]。

混合精度训练是量化神经网络内多种张量的代表性工作[13]。该方法通过使用 16 位浮点数（FP16 或 BF16）存储前向激活、反向梯度及参数，将相关显存占用减半。其中 BF16（Brain Floating Point）通过牺牲部分尾数精度换取更大的动态范围，在大模型训练中比 FP16 更稳定。然而，为避免梯度下溢和舍入误差，混合精度训练仍需维护 FP32 格式的优化器状态（一阶和二阶动量）及主参数副本。因此，优化器状态显存并未减少，仍是主要瓶颈。以 AdamW 为例，优化器状态需维护两个 FP32 动量向量，显存开销达 $8d$ 字节，占训练总显存的 50% 以上。这一限制促使研究者探索直接针对优化器状态的压缩技术[17,37]。

梯度量化研究相对较少，这一现象可从三个方面解释。首先，梯度的生命周期短，在反向传播计算后立即用于参数更新，仅存在于单次迭代中，其显存占用在更新后即可释放。其次，梯度的显存占比较低，对于 AdamW 优化器，梯度以 FP16 格式存储仅需 $2d$ 字节，远低于优化器状态的 $8d$ 字节（FP32），压缩收益有限。最后，梯度累积操作对量化误差较为敏感，低比特表示可能导致更新方向偏差，影响训练稳定性。因此，尽管有少量工作探索梯度量化，但其并非显存优化的重点。

优化器状态量化是当前大模型训练显存优化的重要方向。对于 AdamW 等自适应优化器，需维护一阶动量 $\boldsymbol{m}_t$ 和二阶动量 $\boldsymbol{v}_t$ 两个 FP32 状态向量，显存开销达 $8d$ 字节，占训练总显存的 50% 以上[17,37]。因此，直接针对优化器状态进行量化压缩是减少显存的重要手段。

低比特优化器采用低精度存储与高精度计算相结合的策略。具体而言，优化器状态（如 AdamW 的动量项）在静态存储时采用低精度格式，而在计算更新时临时反量化为高精度，从而显著降低优化器的静态显存占用。低比特 AdamW 优化器的基本流程如算法 2.2 所示。为了形式化描述这一过程，本文定义两个核心函数接口：$\texttt{quantize}(\cdot)$ 将高精度张量压缩为低比特表示（返回码本索引与缩放因子），$\texttt{dequantize}(\cdot)$ 则从低比特表示恢复为高精度张量近似值。

该算法的核心在于每步迭代中，动量状态以量化格式 $\boldsymbol{m}_t^q, \boldsymbol{v}_t^q$ 存储，仅在更新

算法 2.2 低比特 AdamW 优化器

输入： 步数 T, 学习率 η, 超参数 β_1, β_2, 衰减参数 λ, 常数 ϵ
输出： $\boldsymbol{\theta}_T$

初始化 $\boldsymbol{\theta}_0, \boldsymbol{m}_0^q \leftarrow 0, \boldsymbol{v}_0^q \leftarrow 0$
for $t = 1, \cdots, T$ **do**
　$\boldsymbol{g}_t \leftarrow \nabla_\theta f(\boldsymbol{\theta}_{t-1})$　// 计算梯度
　$\boldsymbol{m}_{t-1}, \boldsymbol{v}_{t-1} \leftarrow \texttt{dequantize}(\boldsymbol{m}_{t-1}^q), \texttt{dequantize}(\boldsymbol{v}_{t-1}^q)$　// 反量化恢复动量
　$\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t, \boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2$　// 更新动量
　$\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t/(1-\beta_1^t), \hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t/(1-\beta_2^t)$　// 偏差校正
　$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta\left(\frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t}+\epsilon} + \lambda\boldsymbol{\theta}_{t-1}\right)$　// 更新参数
　$\boldsymbol{m}_t^q, \boldsymbol{v}_t^q \leftarrow \texttt{quantize}(\boldsymbol{m}_t), \texttt{quantize}(\boldsymbol{v}_t)$　// 量化存储动量
end for

时临时反量化为高精度格式。8 比特 AdamW 优化器[37]采用分块动态量化以保持训练稳定性；4 比特 AdamW 优化器[17]在此基础上进一步压缩动量状态，并针对分母项提出无零点映射策略以避免除零异常。

优化器状态量化与传统的权重或激活量化存在本质区别，主要体现在以下方面。第一，运算类型差异。权重、激活量化主要面向矩阵乘法运算，量化误差在累加求和过程中往往能相互抵消或得到抑制。然而，优化器更新规则涉及敏感的逐元素除法操作：

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t\left(\frac{\boldsymbol{m}_t/(1-\beta_1^t)}{\sqrt{\boldsymbol{v}_t/(1-\beta_2^t)}+\epsilon} + \lambda\boldsymbol{\theta}_{t-1}\right), \tag{2-12}$$

其中二阶动量 $\boldsymbol{v}_t$ 位于分母位置。这一特性对量化精度提出了极高要求，微小的量化误差若导致分母异常，极易引发数值不稳定甚至训练发散。第二，动态分布特性。权重在训练完成后相对静态，而优化器状态每步都在更新，分布随训练进程持续变化，早期训练阶段与后期训练阶段的统计特性差异显著，要求量化策略能够适应分布漂移。第三，误差累积效应。优化器状态参与多步迭代更新，单步量化误差会通过动量机制累积。

尽管优化器状态量化取得了显著进展，但仍存在以下局限。首先，标量量化在 4 比特以下表示能力受限，难以精确捕捉优化器状态的细微变化。其次，向量量化虽在理论上具有更强的表示能力，已在模型推理的权重压缩上实现 2-3 比特高保真压缩，但在优化器状态压缩上几乎仍是空白。最后，现有研究尚未系统性地探索如何为 AdamW 等优化器的动态状态设计高效、稳定的向量量化方案。

2.3 低秩子空间在深度学习中的应用

低秩子空间方法通过将模型参数或梯度分解为低秩矩阵的乘积，在高维参数空间中识别有效的低维流形，为大规模神经网络的高效优化提供了重要理论

依据。近期研究表明，大模型的学习主要发生在一个显著的低秩子空间内[6,15]。本节从低秩的基本概念出发，系统介绍通用权重子空间的研究、基于低秩的参数高效微调和梯度子空间优化技术。

2.3.1 低秩矩阵与近似理论

在线性代数中，矩阵的秩定义为该矩阵线性无关的行向量或列向量的最大数量[67]。对于矩阵 $\boldsymbol{W} \in \mathbb{R}^{m\times n}$，若其秩 $r \ll \min(m,n)$，则称该矩阵为低秩矩阵。低秩矩阵可通过奇异值分解（Singular Value Decomposition, SVD）表示为：

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathsf{T}}, \tag{2-13}$$

其中 $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_r > 0$ 为奇异值，$\boldsymbol{U} \in \mathbb{R}^{m\times r}$ 和 $\boldsymbol{V} \in \mathbb{R}^{n\times r}$ 为正交矩阵[68]。

低秩结构的直观含义是：高维数据的有效信息实际上集中在少数几个主导方向上，其余方向的贡献可忽略不计。这一特性在深度学习中表现为：神经网络的权重矩阵、梯度矩阵等均呈现出显著的低秩特性[6,15]，为模型压缩和高效优化提供了理论基础。

低秩近似的核心问题是如何用一个低秩矩阵 $\tilde{\boldsymbol{W}}$ 来逼近原始矩阵 $\boldsymbol{W}$，使得近似误差最小化。形式化地，给定矩阵 $\boldsymbol{W} \in \mathbb{R}^{m\times n}$ 和目标秩 $k \ll \min(m,n)$，低秩近似问题可表示为：

$$\tilde{\boldsymbol{W}} = \operatorname{argmin}_{\operatorname{rank}(\boldsymbol{X})\leqslant k} \|\boldsymbol{W} - \boldsymbol{X}\|_F, \tag{2-14}$$

其中 $\|\cdot\|_F$ 表示 Frobenius 范数。Eckart-Young-Mirsky 定理[69] 为上述问题提供了最优解：截断 SVD 是最佳低秩逼近方法。具体而言，保留前 k 个最大的奇异值及其对应的奇异向量，可得：

$$\tilde{\boldsymbol{W}}_k = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathsf{T}} = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^{\mathsf{T}}, \tag{2-15}$$

其中 $\boldsymbol{U}_k \in \mathbb{R}^{m\times k}$、$\boldsymbol{\Sigma}_k \in \mathbb{R}^{k\times k}$、$\boldsymbol{V}_k \in \mathbb{R}^{n\times k}$ 分别为截断后的左奇异向量、奇异值和右奇异向量矩阵。该近似满足以下误差界：

$$\|\boldsymbol{W} - \tilde{\boldsymbol{W}}_k\|_F = \sqrt{\sum_{i=k+1}^{\min(m,n)} \sigma_i^2}. \tag{2-16}$$

这一定理为深度学习中的低秩方法提供了关键理论支撑：当矩阵的奇异值快速衰减时，仅需保留少量主成分即可实现高精度近似。在神经网络中，权重矩阵和梯度矩阵通常呈现显著的奇异值衰减特性[15]，这意味着使用低秩表示不会引入过大的近似误差。

2.3.2 通用权重子空间假说

近年来，关于深度神经网络过度参数化与泛化性能之间关系的探讨催生了通用权重子空间假说。Kaushik 等人[70] 通过对超过 1100 个在不同数据集、初始化策略及超参数配置下训练的神经网络进行系统性分析发现，尽管这些网络的任务背景迥异，其学到的权重表征最终均收敛于一个共享的低维子空间。

该发现深刻揭示了神经网络学习本质的几何解释。首先，在泛化性层面，该假说解释了为何参数量远超样本规模的模型仍能有效避免过拟合，原因在于其有效解空间被局限在极低维度的子空间内。其次，在架构主导性层面，研究指出网络架构对可学习子空间范围的影响程度显著超越数据本身，从而解释了不同初始化点最终能获得相似表示的现象，并为迁移学习的成功提供了理论依据。最后，从优化理论视角看，若学习过程本质上发生在共享子空间内，则在低秩子空间中进行参数寻优不仅可行，且更贴合神经网络的学习本质，这为 LoRA、权重共享及稀疏训练等技术的有效性提供了底层逻辑支撑。

上述理论为显存高效优化算法提供了坚实的先验支撑，目前基于低秩子空间的优化技术主要沿基于权重低秩的参数高效微调和基于梯度低秩的全参数优化两条路径演进。

2.3.3 基于权重低秩的优化方法

基于上述低秩理论，参数高效微调（Parameter-Efficient Fine-Tuning, PEFT）中的低秩方法应运而生，旨在仅更新少量参数即可实现与全量微调相当的性能。Hu 等人[6] 提出低秩适配器（Low-Rank Adaptation, LoRA），是目前最流行的低秩微调方法之一。LoRA 假设权重更新矩阵具有低秩结构，将 $\boldsymbol{\Delta W}$ 分解为两个低秩矩阵的乘积：

$$\boldsymbol{W}' = \boldsymbol{W} + \boldsymbol{\Delta W} = \boldsymbol{W} + \boldsymbol{BA}, \quad \boldsymbol{B} \in \mathbb{R}^{m\times r}, \boldsymbol{A} \in \mathbb{R}^{r\times n}, \tag{2-17}$$

其中 $r \ll \min(m, n)$ 为秩超参数（通常取 8–64）。LoRA 仅需训练 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，可训练参数量降至原模型的 0.05%–3%，从而极大减少了微调时优化器状态的显存占用。

Lialin 等人[71] 提出 ReLoRA，利用周期性重置低秩分解技术，将 LoRA 的思想应用于神经网络预训练。该技术通过周期性地将低秩适配器合并回主权重，并重新初始化适配器，实现了更稳定的低秩训练。

LoRA 和 ReLoRA 虽然验证了低秩假设的有效性，但两者均作用于权重空间，且需要冻结部分参数或依赖全参数预训练初始化。与此不同，另一类方法选择将低秩投影应用于梯度空间，从而实现真正的全参数学习。

2.3.4 基于梯度低秩的优化方法

Zhao 等人[15]提出 GaLore，首次将低秩投影应用于梯度而非权重。GaLore 在 ReLoRA 思路的基础上，考虑用梯度矩阵的截断奇异值分解构造低秩投影矩阵，实现了基于动量的优化器状态的低秩存储。其核心思想是将梯度矩阵 $\boldsymbol{G}_t \in \mathbb{R}^{m\times n}$ 投影到低秩子空间：

$$\tilde{\boldsymbol{G}}_t = \boldsymbol{P}_t^{\mathsf{T}} \boldsymbol{G}_t \boldsymbol{Q}_t, \quad \boldsymbol{P}_t \in \mathbb{R}^{m\times r}, \boldsymbol{Q}_t \in \mathbb{R}^{n\times r}, \tag{2-18}$$

其中投影矩阵 $\boldsymbol{P}_t$ 和 $\boldsymbol{Q}_t$ 通过梯度 SVD 定期更新。优化器状态仅需在低秩空间 $\mathbb{R}^{r\times r}$ 中维护，显存开销从 $\mathcal{O}(mn)$ 降至 $\mathcal{O}(r^2)$。GaLore 在 LLaMA-7B 预训练任务上实现了与 Adam 相当的性能，同时优化器状态显存减少高达 65.5%[15]。

后续改进如 APOLLO[72] 采用随机投影替代 SVD，避免了显式存储投影矩阵，仅需保存随机种子，实现了更极致的显存节省，同时保持与 AdamW 相当的性能。

2.4 本章小结

本章构建了支撑本文研究的理论框架。在优化算法方面，介绍了零阶优化算法与基于动量的优化算法的基本原理及显存占用分析。在量化压缩方面，阐述了标量量化与向量量化的数学基础，以及量化技术在深度学习中的常见应用。在低秩子空间方面，从低秩矩阵定义与近似理论出发，结合通用权重子空间假说，系统介绍了一些基于低秩分解和低秩近似的优化方法。上述理论为本文后续章节的算法设计提供了基础支撑。

第 3 章 基于低秩子空间的大模型零阶梯度估计算法

本文以优化状态特性为统一视角，从梯度估计与优化器状态更新两个关键环节出发，系统研究显存高效的零阶优化算法。本章聚焦第一个关键环节——梯度估计，针对现有零阶优化算法梯度估计方差随模型参数量线性增长、导致收敛缓慢的问题，提出基于随机子空间的零阶梯度估计算法 SubZero。

3.1 引言

在 LLM 微调过程中，以 Adam[46]或 AdamW[12]为代表的一阶优化（First-Order Optimization, FO）算法虽能取得良好性能，但其深度依赖反向传播机制。在计算梯度时，FO 算法必须在显存中持续缓存前向传播产生的中间激活值，导致显存开销随模型规模与序列长度剧增[18,73]。为显著提升显存效率，MeZO[16]首次将零阶优化（ZO）算法引入 LLM 微调。该算法完全摒弃了反向传播，仅利用前向传播的损失值进行有限差分梯度估计。由于无需存储任何中间激活值，其微调时的显存占用降至与推理阶段相当的极低水平。然而，ZO 梯度估计的方差与模型参数维度呈线性正相关，在参数量巨大的 LLMs 中，这种极高方差的梯度估计会导致收敛极其缓慢，性能较 FO 算法存在显著差距[19,22,28]。

降低 ZO 梯度估计方差通常可加速收敛[29]。针对高方差问题，现有技术主要可分为两类：第一种技术在训练过程中逐步增大批量大小，以减少 ZO 梯度估计中的梯度噪声和方差[22,28]，但这会导致训练后期因批量过大而产生显著的运行时间和显存开销；第二种技术通过稀疏参数扰动（如随机稀疏剪枝掩码[19]和块坐标扰动[18]）或参数高效微调[16,18]、张量化适配器[30]等技术减少可训练参数量。近期理论研究[20,31]探索了在随机子空间内进行低维扰动以降低梯度方差、提升收敛速率的算法。其核心改进在于对 SPSA（见公式 (2-4)）中的全空间扰动机制进行了重构，通过投影矩阵 $\boldsymbol{P}$ 将扰动向量 $\tilde{\boldsymbol{z}}$ 限制在低维子空间中生成：

$$\tilde{\boldsymbol{z}} = \boldsymbol{P}\boldsymbol{z}, \tag{3-1}$$

其中 $\boldsymbol{P} \in \mathbb{R}^{d\times q}$ 为随机投影矩阵（元素服从 $\mathcal{N}(0,1)$），$\boldsymbol{z} \in \mathbb{R}^{q}$ 为服从 $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$ 的低维扰动向量，$q < d$ 为子空间维度。对应的子空间梯度估计器为：

$$\hat{\nabla}\mathcal{L}(\boldsymbol{w}, \boldsymbol{P}; \mathcal{B}) = \frac{\mathcal{L}(\boldsymbol{w}+\varepsilon\boldsymbol{P}\boldsymbol{z};\mathcal{B}) - \mathcal{L}(\boldsymbol{w}-\varepsilon\boldsymbol{P}\boldsymbol{z};\mathcal{B})}{2\varepsilon}\boldsymbol{P}\boldsymbol{z}. \tag{3-2}$$

然而，LLM 参数规模庞大，全参数微调时 $\boldsymbol{P}$ 矩阵维度达 $q \times d$（d 为总参数量）[74]；在 LoRA 等参数高效微调方案中矩阵规模依然可观[6]。这导致显存需求与计算复杂度急剧上升。

受此启发，本章提出首个应用于 LLM 微调的随机子空间零阶梯度估计算法 SubZero，旨在应对高维模型微调中的显存与性能平衡挑战。核心引入了一种专为 LLM 架构定制的低秩扰动梯度估计机制：摒弃传统全秩扰动，采用分层低秩策略，通过在每一层组合列正交矩阵与高斯随机矩阵生成低秩扰动矩阵。此外，针对现有子空间算法计算开销大的问题，本算法采用分层特定小矩阵替代大型投影矩阵，并引入延迟更新策略（周期性生成扰动），进一步减少额外显存和计算开销。

理论层面，本章重构梯度估计等价形式，揭示与传统零阶[16]及随机子空间算法[20] 的本质差异，证明估计梯度能紧密逼近反向传播真实梯度，并建立收敛性保证。实验表明，SubZero 兼容全参数微调及 LoRA 等参数高效微调多种训练范式，在多种大语言模型微调任务上性能均显著优于传统零阶优化算法。

综上，本章的主要贡献如下：

（1）分析了梯度矩阵的低秩结构特性，提出了一种基于随机子空间的零阶梯度估计算法 SubZero，通过在低维子空间内生成扰动并进行梯度估计，将梯度估计方差从依赖模型总参数量降至仅依赖子空间维度，显著降低了梯度估计方差。

（2）分析了 SubZero 梯度估计的等价形式，揭示了其与传统零阶算法及随机子空间算法的本质差异，理论证明了估计梯度能紧密逼近反向传播真实梯度，且方差显著低于传统零阶算法，并建立了与 SGD 结合时的收敛性理论保证。

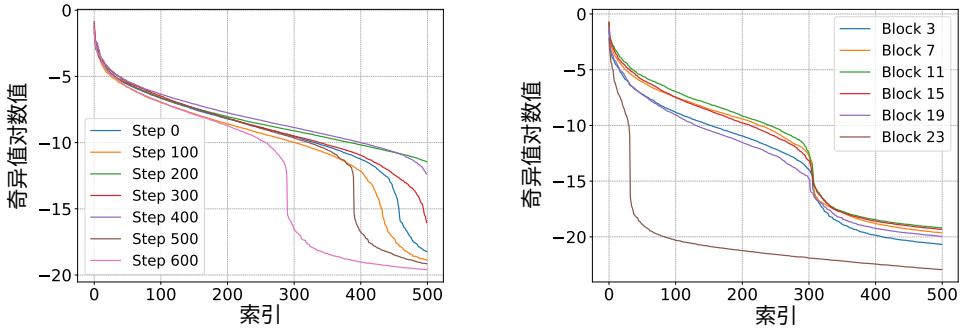
（3）通过在全参数微调及多种 PEFT 方案（包括 LoRA、前缀微调和提示微调）中的实验，验证了 SubZero 在保持与 MeZO 相当显存开销的前提下，显著提升了模型性能（如在 LLaMA-7B 上提升 7.1%），展现了良好的兼容性与实用性。

3.2 算法介绍

本节首先详述本章提出的 SubZero——一种面向 LLM 微调的随机子空间零阶梯度估计算法，随后阐述其在四种代表性微调方案中的集成。

3.2.1 低秩扰动的零阶梯度估计器

本章算法的核心思想是：在低维子空间中探索更新方向，可显著降低梯度估计方差[20,31]，相比 MeZO 等在原始空间中的估计算法更具优势。这一思路的动机源于近期研究的共同发现：LLM 微调过程中的反向传播梯度会快速收敛至一个低维子空间[15-16,23-24]。为验证该现象，本章在 OPT-1.3B 架构上使用 SST-2 数据集进行实证分析，发现各层梯度矩阵在整个优化过程中均呈现显著的低秩结构。如图 3.1所示，对梯度矩阵进行奇异值分解后观察到明显的谱衰减现象——仅少数奇异值主导整个梯度谱，且该模式在不同层和训练阶段保持一致。



(a) 第 22 层 `k_proj` 梯度矩阵的奇异值谱

(b) 第 100 步时各层 `q_proj` 梯度矩阵的奇异值谱

图 3.1　LLM 微调过程中梯度矩阵的低秩结构

基于上述观察，本章提出专为 LLM 微调定制的随机子空间零阶优化算法 SubZero。该算法通过在固有低维子空间中执行梯度估计，同时实现两个目标：（1）显著降低梯度估计方差；（2）规避传统子空间算法[20,31]因显式维护投影矩阵（如公式 3-2的矩阵 $\boldsymbol{P}$）而引入的额外显存开销。

分层随机子空间扰动。 LLM 主要由执行矩阵乘法的全连接层构成。记第 i 层可训练参数矩阵为 $\boldsymbol{W}_i \in \mathbb{R}^{m_i \times n_i}$，下面说明其低秩扰动矩阵 $\tilde{\boldsymbol{Z}}_i \in \mathbb{R}^{m_i \times n_i}$ 的设计方法。

本章提出分层低秩扰动策略，区别于先前针对整个模型参数的随机子空间算法[20,31]。每步训练中：1. 生成低维随机矩阵 $\boldsymbol{Z}_i \in \mathbb{R}^{r \times r}$（$r \ll \min\{m_i, n_i\}$）2. 对两个服从 $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ 的高斯随机矩阵进行 QR 分解，生成列正交投影矩阵 $\boldsymbol{U}_i \in \mathbb{R}^{m_i \times r}$ 和 $\boldsymbol{V}_i \in \mathbb{R}^{n_i \times r}$（见算法 3.1）

组合上述矩阵得到低秩扰动：

$$\tilde{\boldsymbol{Z}}_i = \boldsymbol{U}_i \boldsymbol{Z}_i \boldsymbol{V}_i^{\mathsf{T}}, \tag{3-3}$$

其中 $\tilde{\boldsymbol{Z}}_i$ 位于由 $\boldsymbol{U}_i$ 和 $\boldsymbol{V}_i$ 张成的子空间内，$\boldsymbol{Z}_i$ 为服从 $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ 的低维随机扰动矩阵。SubZero 的投影矩阵仅需通过权重矩阵、激活矩阵、历史 ZO 梯度矩阵或随机矩阵的 QR 分解获得，其带来的额外显存开销可忽略不计。表 3.1 展示了不同投影矩阵生成方案在 OPT-1.3B/LLaMA2-7B（SST-2）和 OPT-13B（RTE）上的性能对比。实验结果表明，本章提出的随机矩阵方案性能最优。相比之下，基于权重和历史 ZO 梯度的方案虽可行但性能显著受损；激活矩阵因依赖批量大小而效果不佳，需更复杂的处理机制。

设模型含 l 层，参数矩阵集合 $\mathcal{W} = \{\boldsymbol{W}_i\}_{i=1}^{l}$，扰动矩阵集合 $\tilde{\mathcal{Z}} = \{\tilde{\boldsymbol{Z}}_i\}_{i=1}^{l}$。类似公式 (2-4) 和 (3-2)，计算损失差值：

$$\rho = \frac{\mathcal{L}(\mathcal{W} + \varepsilon\tilde{\mathcal{Z}}; \mathcal{B}) - \mathcal{L}(\mathcal{W} - \varepsilon\tilde{\mathcal{Z}}; \mathcal{B})}{2\varepsilon}. \tag{3-4}$$

（注：集合乘以标量表示对标量乘以集合中每个元素；集合相加表示对应元

表 3.1　不同投影矩阵生成方案的性能对比（准确率%）

投影源	OPT-1.3B	LLaMA2-7B	OPT-13B
权重矩阵	91.5	91.7	65.3
激活矩阵	51.5	52.9	53.1
ZO 梯度	89.6	92.0	67.5
随机矩阵	**93.4**	**94.5**	**74.0**

素相加。此仅为数学表达，实践中通过两次全模型前向传播计算 ρ。）第 i 层的梯度估计为：

$$\hat{\nabla}\mathcal{L}(\boldsymbol{W}_i;\mathcal{B}) = \rho\tilde{\boldsymbol{Z}}_i = \rho\boldsymbol{U}_i\boldsymbol{Z}_i\boldsymbol{V}_i^{\top}. \tag{3-5}$$

在第 3.3节中，本文分析了该梯度估计的有效性：定理 3.1证明了估计梯度 (3-5) 与一阶算法中反向传播计算的原始梯度距离紧密；定理 3.2表明相比 MeZO[16]的梯度估计 (3-2)，本章的算法具有更小的方差和角度误差。

将估计梯度 (3-5) 代入任意一阶优化器（如 SGD）：

$$\boldsymbol{W}_i^{t+1} = \boldsymbol{W}_i^{t} - \eta^t\hat{\nabla}\mathcal{L}(\boldsymbol{W}_i^{t};\mathcal{B}^t) = \boldsymbol{W}_i^{t} - \eta^t\rho^t\boldsymbol{U}_i^{t}\boldsymbol{Z}_i^{t}\boldsymbol{V}_i^{t\top}. \tag{3-6}$$

本章采用 SGD 作为 SubZero 的优化器。尽管优化器的选择与零阶梯度估计机制在理论上相互正交，但为最大限度地提升显存效率，本章沿用 MeZO[16]的设置采用 SGD。第 3.3节的定理 3.3保证了该组合的收敛性并给出了收敛速率界。

表 3.2　RoBERTa-large 在 SST-2 上全参数微调的显存开销对比

方法	显存开销（GB）
SGD	6.063
MeZO[16]	2.683
S-RGF[20]	23.845
SubZero	**2.690**

表 3.2 展示了在相同实验设置下（包括分层扰动和子空间维度匹配），SubZero 与现有随机子空间算法 S-RGF[20] 的显存开销对比。在 RoBERTa-large 全参数微调 SST-2 任务上，S-RGF 的显存占用约为 SGD 的 4 倍、MeZO 的 8.8 倍，而 SubZero 与 MeZO 相当。

算法 3.1 GenerateProjMatrix(m, n, r)

输入：参数矩阵维度 $m \times n$，子空间秩 r

生成随机矩阵 $\boldsymbol{R}_1 \in \mathbb{R}^{m\times r}$ 和 $\boldsymbol{R}_2 \in \mathbb{R}^{n\times r}$，元素服从 $\mathcal{N}(0,1)$
$\boldsymbol{U}, \_ \leftarrow$ QR_Decomposition($\boldsymbol{R}_1$)
$\boldsymbol{V}, \_ \leftarrow$ QR_Decomposition($\boldsymbol{R}_2$)
return $\boldsymbol{U}, \boldsymbol{V}$

算法 3.2 PerturbParams($\mathcal{W}, \mathcal{U}, \mathcal{V}, r, \varepsilon, s$)

输入：模型参数集 $\mathcal{W}$，投影矩阵集 $\mathcal{U}, \mathcal{V}$，秩 r，扰动尺度 ε，随机种子 s

用随机种子 s 重置随机数生成器
for $i = 1, 2, \cdots, l$ **do**
　生成扰动矩阵 $\boldsymbol{Z}_i \in \mathbb{R}^{r\times r}$，服从 $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
　$\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i + \varepsilon \boldsymbol{U}_i \boldsymbol{Z}_i \boldsymbol{V}_i^{\mathsf{T}}$
end for
return $\mathcal{W}$

算法 3.3 SubZero 算法

输入：第 i 层参数矩阵 $\boldsymbol{W}_i \in \mathbb{R}^{m_i\times n_i}$ $(i = 1, \cdots, l)$，损失 $\mathcal{L}$，总步数 T，扰动尺度 ε，学习率 $\{\eta^t\}$，子空间更新频率 T_0，秩 r

输出：$\mathcal{W}^T$　// 符号约定：$\mathcal{W}^t = \{\boldsymbol{W}_i^t\}, \mathcal{U}^t = \{\boldsymbol{U}_i^t\}, \mathcal{V}^t = \{\boldsymbol{V}_i^t\}$

for $t = 0, 1, \cdots, T-1$ **do**
　采样小批量 $\mathcal{B}^t$ 和随机种子 s^t
　for $i = 1, 2, \cdots, l$ **do**
　　if $t \bmod T_0 \equiv 0$ **then**
　　$\boldsymbol{U}_i^t, \boldsymbol{V}_i^t \leftarrow$ GenerateProjMatrix(m_i, n_i, r)
　　else
　　$\boldsymbol{U}_i^t \leftarrow \boldsymbol{U}_i^{t-1}, \boldsymbol{V}_i^t \leftarrow \boldsymbol{V}_i^{t-1}$
　　end if
　end for
　$\mathcal{W}^t \leftarrow$ PerturbParams($\mathcal{W}^t, \mathcal{U}^t, \mathcal{V}^t, r, \varepsilon, s^t$), $\ell_+^t \leftarrow \mathcal{L}(\mathcal{W}^t; \mathcal{B}^t)$
　$\mathcal{W}^t \leftarrow$ PerturbParams($\mathcal{W}^t, \mathcal{U}^t, \mathcal{V}^t, r, -2\varepsilon, s^t$), $\ell_-^t \leftarrow \mathcal{L}(\mathcal{W}^t; \mathcal{B}^t)$
　$\mathcal{W}^t \leftarrow$ PerturbParams($\mathcal{W}^t, \mathcal{U}^t, \mathcal{V}^t, r, \varepsilon, s^t$)
　$\rho^t \leftarrow (\ell_+^t - \ell_-^t)/(2\varepsilon)$
　用种子 s^t 重置随机数生成器
　for $i = 1, 2, \cdots, l$ **do**
　　重生成扰动矩阵 $\boldsymbol{Z}_i^t \in \mathbb{R}^{r\times r}$，元素服从 $\mathcal{N}(0,1)$
　　$\boldsymbol{W}_i^{t+1} \leftarrow \boldsymbol{W}_i^t - \eta^t \rho^t (\boldsymbol{U}_i^t \boldsymbol{Z}_i^t \boldsymbol{V}_i^{t\mathsf{T}})$
　end for
end for

3.2.2 子空间延迟更新策略

根据公式 (3-6)，第 t 步第 i 层参数矩阵的梯度估计 $\hat{\nabla}\mathcal{L}(\boldsymbol{W}_i^t; \mathcal{B}^t)$ 位于由 $\boldsymbol{U}_i^t$ 和 $\boldsymbol{V}_i^t$ 定义的子空间内：$\boldsymbol{U}_i^t$ 张成列空间，$\boldsymbol{V}_i^t$ 决定行空间。若迭代生成这些矩阵，将引入额外计算开销。

LLM 微调需兼顾计算效率与梯度子空间近似精度：过短的更新间隔（如每步更新）导致高计算成本并限制子空间探索；过长间隔则降低近似精度，无法捕捉梯度子空间的演化特性。为此，本节提出延迟子空间更新策略：每 $F > 1$ 步训练仅在首步重新生成 $\boldsymbol{U}_i$ 和 $\boldsymbol{V}_i$，后续 $F-1$ 步保持不变（见算法 3.3第 4-7 行）。通过 QR 分解两个独立随机矩阵生成列正交矩阵 $\boldsymbol{U}_i$ 和 $\boldsymbol{V}_i$（见算法 3.1），该策略在

各项实验中均展现出优异的运行效率与稳定的性能表现。

SubZero 每层仅维护三个小矩阵：扰动矩阵 $\boldsymbol{Z}_i \in \mathbb{R}^{r\times r}$ 和列正交矩阵 $\boldsymbol{U}_i \in \mathbb{R}^{m_i\times r}$、$\boldsymbol{V}_i \in \mathbb{R}^{n_i\times r}$。由于 $r \ll \min\{m_i, n_i\}$，此设计显著提升显存效率。结合原地操作与分层参数更新（细节见附录 A.1），SubZero 在保持性能的同时大幅降低 GPU 显存占用。例如，在 SST-2 上全参数微调 OPT-1.3B[75]时，SubZero 仅需 6.8GB 显存，而 SGD 需 11.5GB，显存效率提升 1.6 倍（见图 3.3(d)）。

算法 3.3 总结了 SubZero 的单步训练流程，主要包含三个阶段：

- 利用算法 3.1 获取或复用投影矩阵；
- 执行算法 3.2 以扰动参数并估算损失差值 ρ；
- 基于公式 (3-6) 逐层更新参数。

3.2.3 参数高效微调方法集成

本节描述 SubZero 在全参数微调[74]和三种主流参数高效微调（PEFT）方案中的集成：LoRA[6]、前缀微调[76]和提示微调[77]。SubZero 通常可轻松集成至这些方案，但对极端非方阵（行列数差异极大）存在挑战——这在 LoRA 中尤为突出：其使用两个低秩矩阵 $\boldsymbol{A}_i \in \mathbb{R}^{m_i\times k}$ 和 $\boldsymbol{B}_i \in \mathbb{R}^{k\times n_i}$（$k \ll \min\{m_i, n_i\}$，如 $k = 8$ 而 $\min\{m_i, n_i\} = 2048$[18]）近似全矩阵 $\boldsymbol{W}_i' \in \mathbb{R}^{m_i\times n_i}$。此时无法找到 $r \ll k$ 满足公式 (3-3)，导致 SubZero 难以直接应用。

为克服此限制，本节提出重塑策略：将原始非方阵转换为近似方阵。例如，将 $\boldsymbol{A}_i \in \mathbb{R}^{m_i\times k}$ 重塑为 $\boldsymbol{A}_i' \in \mathbb{R}^{m_i'\times k'}$，满足 $m_i k = m_i' k'$ 且 m_i' 接近 k'。此重塑使公式 (3-3) 可应用于低秩扰动（$r \ll \min\{m_i', k'\}$）。第 3.4.3节表 3.8验证了该策略的有效性。

3.3 理论分析

本节从理论上分析 SubZero 为何能够降低梯度估计方差并加速收敛。在分析前，首先定义必要的数学符号，并给出若干辅助引理。

3.3.1 符号约定与投影构造

设神经网络包含 l 个可训练层，第 i 层参数矩阵维度为 $m_i \times n_i$，定义：

$$\boldsymbol{P} = \mathrm{bdiag}(\boldsymbol{V}_1 \otimes \boldsymbol{U}_1, \cdots, \boldsymbol{V}_l \otimes \boldsymbol{U}_l), \tag{3-7}$$

$$\boldsymbol{z} = [\mathrm{vec}(\boldsymbol{Z}_1)^\top, \cdots, \mathrm{vec}(\boldsymbol{Z}_l)^\top]^\top, \tag{3-8}$$

$$\tilde{\boldsymbol{z}} = [\mathrm{vec}(\tilde{\boldsymbol{Z}}_1)^\top, \cdots, \mathrm{vec}(\tilde{\boldsymbol{Z}}_l)^\top]^\top. \tag{3-9}$$

其中 $\boldsymbol{U}_i \in \mathbb{R}^{m_i\times r}$ 和 $\boldsymbol{V}_i \in \mathbb{R}^{n_i\times r}$ 为列正交矩阵（满足 $\boldsymbol{U}_i^\top \boldsymbol{U}_i = \boldsymbol{V}_i^\top \boldsymbol{V}_i = \boldsymbol{I}_r$），$r$ 为子空间秩，$\otimes$ 表示 Kronecker 积，$\mathrm{bdiag}(\cdot)$ 表示块对角矩阵，$\mathrm{vec}(\cdot)$ 表示矩阵向量化

操作。整个投影矩阵 $\boldsymbol{P} \in \mathbb{R}^{d\times q}$ 满足 $\boldsymbol{P}^{\mathsf{T}}\boldsymbol{P} = \boldsymbol{I}_q$，其中 $d = \sum_{i=1}^{l} m_i n_i$ 为总参数量，$q = lr^2$ 为子空间维度。

3.3.2 辅助引理

引理 3.1 (层投影矩阵的列正交性)： 设 $\tilde{\boldsymbol{Z}} = \boldsymbol{U}\boldsymbol{Z}\boldsymbol{V}^{\mathsf{T}}$，其中 $\boldsymbol{U} \in \mathbb{R}^{m\times r}$，$\boldsymbol{Z} \in \mathbb{R}^{r\times r}$，$\boldsymbol{V} \in \mathbb{R}^{n\times r}$，且 $\boldsymbol{U}^{\mathsf{T}}\boldsymbol{U} = \boldsymbol{V}^{\mathsf{T}}\boldsymbol{V} = \boldsymbol{I}_r$。则有 $\mathrm{vec}(\tilde{\boldsymbol{Z}}) = \boldsymbol{P}\mathrm{vec}(\boldsymbol{Z})$ 且 $\boldsymbol{P}^{\mathsf{T}}\boldsymbol{P} = \boldsymbol{I}_{r^2}$，其中 $\boldsymbol{P} = \boldsymbol{V} \otimes \boldsymbol{U}$。

证明 由于 $\mathrm{vec}(\boldsymbol{U}\boldsymbol{Z}\boldsymbol{V}^{\mathsf{T}}) = (\boldsymbol{V} \otimes \boldsymbol{U})\mathrm{vec}(\boldsymbol{Z})$，只需证明 $(\boldsymbol{V} \otimes \boldsymbol{U})^{\mathsf{T}}(\boldsymbol{V} \otimes \boldsymbol{U}) = \boldsymbol{I}_{r^2}$。实际上，

$$(\boldsymbol{V} \otimes \boldsymbol{U})^{\mathsf{T}}(\boldsymbol{V} \otimes \boldsymbol{U}) = (\boldsymbol{V}^{\mathsf{T}} \otimes \boldsymbol{U}^{\mathsf{T}})(\boldsymbol{V} \otimes \boldsymbol{U}) = (\boldsymbol{V}^{\mathsf{T}}\boldsymbol{V}) \otimes (\boldsymbol{U}^{\mathsf{T}}\boldsymbol{U}) = \boldsymbol{I}_r \otimes \boldsymbol{I}_r = \boldsymbol{I}_{r^2}.$$

∎

引理 3.2 (块对角投影矩阵的列正交性)： 设块对角矩阵 $\boldsymbol{P} = \mathrm{bdiag}(\boldsymbol{P}_1, \boldsymbol{P}_2, \cdots, \boldsymbol{P}_l)$，且 $\tilde{\boldsymbol{z}}_i = \boldsymbol{P}_i\boldsymbol{z}_i$，其中 $\boldsymbol{P}_i^{\mathsf{T}}\boldsymbol{P}_i = \boldsymbol{I}_{r^2}$，$i = 1, 2, \cdots, l$。则 $\tilde{\boldsymbol{z}} = \boldsymbol{P}\boldsymbol{z}$，且 $\boldsymbol{P}^{\mathsf{T}}\boldsymbol{P} = \boldsymbol{I}_{lr^2}$，其中 $\tilde{\boldsymbol{z}} = [\tilde{\boldsymbol{z}}_1^{\mathsf{T}}, \cdots, \tilde{\boldsymbol{z}}_l^{\mathsf{T}}]^{\mathsf{T}}$，$\boldsymbol{z} = [\boldsymbol{z}_1^{\mathsf{T}}, \cdots, \boldsymbol{z}_l^{\mathsf{T}}]^{\mathsf{T}}$。

证明 易验证 $\tilde{\boldsymbol{z}} = \boldsymbol{P}\boldsymbol{z}$。此外，

$$\boldsymbol{P}^{\mathsf{T}}\boldsymbol{P} = \mathrm{bdiag}(\boldsymbol{P}_1^{\mathsf{T}}, \cdots, \boldsymbol{P}_l^{\mathsf{T}})\mathrm{bdiag}(\boldsymbol{P}_1, \cdots, \boldsymbol{P}_l) = \mathrm{bdiag}(\boldsymbol{P}_1^{\mathsf{T}}\boldsymbol{P}_1, \cdots, \boldsymbol{P}_l^{\mathsf{T}}\boldsymbol{P}_l) = \boldsymbol{I}_{lr^2}.$$

∎

以下给出高斯分布的几个基本性质，它们在后续证明中将被反复使用。

定义 3.1 (标准高斯向量)： 称 $\boldsymbol{z}$ 为 n 维标准高斯向量（记作 $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}_n)$），若其概率密度函数为 $p(\boldsymbol{z}) = \frac{1}{\kappa}e^{-\frac{1}{2}\|\boldsymbol{z}\|^2}$，其中 $\kappa > 0$ 满足 $\int_{\mathbb{R}^n} \frac{1}{\kappa}e^{-\frac{1}{2}\|\boldsymbol{z}\|^2} d\boldsymbol{z} = 1$。

定义 3.2 (卡方分布)： 设 $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}_n)$，则称 $x = \|\boldsymbol{z}\|^2$ 服从自由度为 n 的卡方分布，记作 $x \sim \chi^2(n)$。

引理 3.3 (高斯分布的正交不变性)： 设 $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}_n)$，则对任意正交矩阵 $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ 和连续函数 f，有 $\mathbb{E}_{\boldsymbol{z}}[f(\boldsymbol{z})] = \mathbb{E}_{\boldsymbol{z}}[f(\boldsymbol{Q}\boldsymbol{z})]$。

引理 3.4 (卡方分布的矩)： 若 $x \sim \chi^2(n)$，则

$$\mathbb{E}[x] = n, \quad \mathrm{Var}[x] = 2n.$$

引理 3.5 (Hessian Lipschitz 函数的二次逼近)： [27] 设 $f \in C_{L_2}^{2,2}(\mathbb{R}^n)$（即 f 的 Hessian 矩阵是 L_2-Lipschitz 连续的）。则对任意 $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$，有

$$|f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle - \frac{1}{2}\langle \nabla^2 f(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle| \leqslant \frac{L_2}{6}\|\boldsymbol{y} - \boldsymbol{x}\|^3.$$

引理 3.6 (高斯范数的矩)： [27] 设 $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}_n)$。当 $0 \leqslant t \leqslant 2$ 时，有 $\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{z}\|^t] \leqslant n^{t/2}$；当 $t \geqslant 2$ 时，有 $n^{t/2} \leqslant \mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{z}\|^t] \leqslant (n + t)^{t/2}$。

引理 3.7 (一个二次型的期望)： 设 $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}_n)$，则对任意 $\boldsymbol{y} \in \mathbb{R}^n$，有

$$\mathbb{E}_{\boldsymbol{z}}[\|\langle \boldsymbol{y}, \boldsymbol{z}\rangle \boldsymbol{z}\|^2] = (n+2)\|\boldsymbol{y}\|^2.$$

证明 注意到对任意正交矩阵 $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$，有

$$\|\langle \boldsymbol{y}, \boldsymbol{Q}\boldsymbol{z}\rangle \boldsymbol{Q}\boldsymbol{z}\|^2 = \|\langle \boldsymbol{Q}^{\mathsf{T}}\boldsymbol{y}, \boldsymbol{z}\rangle \boldsymbol{z}\|^2, \quad \|\boldsymbol{Q}^{\mathsf{T}}\boldsymbol{y}\| = \|\boldsymbol{y}\|.$$

根据引理3.3，可取 $\boldsymbol{y} = [1, 0, \cdots, 0]^{\mathsf{T}}$，只需证明 $\mathbb{E}_{\boldsymbol{z}}[\|\langle \boldsymbol{y}, \boldsymbol{z}\rangle \boldsymbol{z}\|^2] = n+2$。利用引理3.4可得

$$\begin{aligned}\mathbb{E}_{\boldsymbol{z}}[\|\langle \boldsymbol{y}, \boldsymbol{z}\rangle \boldsymbol{z}\|^2] &= \mathbb{E}_{\boldsymbol{z}}\left[\sum_{i=1}^{n} \boldsymbol{z}_1^2 \boldsymbol{z}_i^2\right] = \sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{z}_1^2 \boldsymbol{z}_i^2] \\ &= \mathbb{E}_{\boldsymbol{z}_1}[\boldsymbol{z}_1^4] + \mathbb{E}_{\boldsymbol{z}_1}[\boldsymbol{z}_1^2] \sum_{i=2}^{n} \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{z}_i^2] = n+2.\end{aligned}$$

∎

3.3.3 梯度估计性质

SubZero 使用对称差分进行梯度估计。下述定理给出其关键性质。

定理 3.1 (梯度估计性质)： 对于 SubZero 的梯度估计，以下两条性质成立：

(a) 梯度估计可等价表示为

$$\hat{g}_{\varepsilon}(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z}) = \frac{f(\boldsymbol{x} + \varepsilon \boldsymbol{P}\boldsymbol{z}) - f(\boldsymbol{x} - \varepsilon \boldsymbol{P}\boldsymbol{z})}{2\varepsilon} \boldsymbol{P}\boldsymbol{z}, \tag{3-10}$$

其中 $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q), \varepsilon > 0$。

(b) 设 $f \in C_{L_2}^{2,2}(\mathbb{R}^d)$（即 f 的 Hessian 矩阵 Lipschitz 常数为 L_2），则估计梯度与子空间内真实梯度的偏差满足

$$\boldsymbol{\Phi}(\boldsymbol{x}) = \left\| \mathbb{E}_{\boldsymbol{z}}[\hat{g}_{\varepsilon}(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})] - \boldsymbol{P}\boldsymbol{P}^{\mathsf{T}} \nabla f(\boldsymbol{x}) \right\|_2 \leqslant \frac{\varepsilon^2}{6} L_2 (q+4)^2.$$

证明 **(a)** 由引理3.1和引理3.2，结论显然成立。

(b) 记 $a_{\boldsymbol{z}}(\tau) = f(\boldsymbol{x} + \tau \boldsymbol{z}) - f(\boldsymbol{x}) - \tau \langle \nabla f(\boldsymbol{x}), \boldsymbol{z}\rangle - \frac{\tau^2}{2} \langle \nabla^2 f(\boldsymbol{x})\boldsymbol{z}, \boldsymbol{z}\rangle$。引理3.5表明

$$|a_{\boldsymbol{z}}(\pm\varepsilon)| \leqslant \frac{\varepsilon^3}{6} L_2 \|\boldsymbol{z}\|^3.$$

注意到

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{z}}[\hat{g}_{\varepsilon}(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})] - \boldsymbol{P}\boldsymbol{P}^{\mathsf{T}} \nabla f(\boldsymbol{x}) \\ &= \frac{\boldsymbol{P}}{2\kappa\varepsilon} \int_{\mathbb{R}^q} [f(\boldsymbol{x} + \varepsilon \boldsymbol{P}\boldsymbol{z}) - f(\boldsymbol{x} - \varepsilon \boldsymbol{P}\boldsymbol{z}) - 2\varepsilon \langle \nabla f(\boldsymbol{z}), \boldsymbol{P}\boldsymbol{z}\rangle] \boldsymbol{z} e^{-\frac{1}{2}\|\boldsymbol{z}\|^2} d\boldsymbol{z}.\end{aligned}$$

因此，依据引理3.6，有

$$\begin{aligned}&\|\mathbb{E}_{\boldsymbol{z}}[\hat{g}_{\varepsilon}(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})] - \boldsymbol{P}\boldsymbol{P}^{\mathsf{T}} \nabla f(\boldsymbol{x})\| \\ &\leqslant \frac{1}{2\kappa\varepsilon} \int_{\mathbb{R}^q} |f(\boldsymbol{x} + \varepsilon \boldsymbol{P}\boldsymbol{z}) - f(\boldsymbol{x} - \varepsilon \boldsymbol{P}\boldsymbol{z}) - 2\varepsilon \langle \nabla f(\boldsymbol{z}), \boldsymbol{P}\boldsymbol{z}\rangle| \|\boldsymbol{z}\| e^{-\frac{1}{2}\|\boldsymbol{z}\|^2} d\boldsymbol{z}\end{aligned}$$

$$= \frac{1}{2\kappa\varepsilon}\int_{\mathbb{R}^q} |a_{\boldsymbol{P}\boldsymbol{z}}(\varepsilon) - a_{\boldsymbol{P}\boldsymbol{z}}(-\varepsilon)|\|\boldsymbol{z}\| e^{-\frac{1}{2}\|\boldsymbol{z}\|^2} d\boldsymbol{z}$$
$$\leqslant \frac{\varepsilon^2 L_2}{6\kappa}\int_{\mathbb{R}^q} \|\boldsymbol{z}\|^4 e^{-\frac{1}{2}\|\boldsymbol{z}\|^2} d\boldsymbol{z} \leqslant \frac{\varepsilon^2}{6} L_2 (q+4)^2.$$

∎

定理3.1的 (a) 部分给出了 SubZero 梯度估计的等价形式，其与标准随机子空间零阶优化算法的关键区别在于：SubZero 利用了神经网络的层结构，通过块对角投影矩阵 $\boldsymbol{P}$ 实现层间解耦。这种设计保持了参数更新的层间独立性，同时显著降低了子空间维度。

定理3.1的 (b) 部分表明，当设置 $\varepsilon = \frac{1}{q+4}$ 时，梯度估计偏差 $\boldsymbol{\Phi}(\boldsymbol{x})$ 的上界为常数 $\frac{L_2}{6}$，与总参数量 d 无关。这一性质对大语言模型优化至关重要——当 d 达到十亿级时，传统零阶算法（如 MeZO）的梯度估计误差随 d 线性增长，而 SubZero 的误差仅取决于子空间维度 q（通常 $q \ll d$）。

3.3.4 二次函数上的精确分析

为深入理解 SubZero 的优化行为，本节在严格凸二次函数 $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x}$ 上进行精确分析（$\boldsymbol{H} \in \mathbb{R}^{d\times d}$ 为正定矩阵）。该设定具有理论合理性：预训练大语言模型的微调通常在局部极小值附近进行，该区域可被二次函数良好近似[78]。

定理 3.2 (二次函数上的梯度估计)： 设 $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x}$ 且 $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$，则 SubZero 的梯度估计满足：

$$\mathbb{E}_{\boldsymbol{z}}[\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})] = \boldsymbol{P}\boldsymbol{P}^\top \nabla f(\boldsymbol{x}), \tag{3-11}$$

$$\mathbb{E}_{\boldsymbol{z}}[\|\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\|^2] = (q+2)\|\boldsymbol{P}^\top \nabla f(\boldsymbol{x})\|^2, \tag{3-12}$$

$$\mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle \nabla f(\boldsymbol{x}), \hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\rangle^2}{\|\boldsymbol{P}^\top \nabla f(\boldsymbol{x})\|^2 \|\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\|^2}\right] = \frac{1}{q}. \tag{3-13}$$

证明 容易验证 $\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z}) = \boldsymbol{P}\langle \boldsymbol{P}^\top \nabla f(\boldsymbol{x}), \boldsymbol{z}\rangle \boldsymbol{z}$。因此 $\mathbb{E}_{\boldsymbol{z}}[\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})] = \boldsymbol{P}\boldsymbol{P}^\top \nabla f(\boldsymbol{x})$。结合引理3.7，得到 $\mathbb{E}_{\boldsymbol{z}}[\|\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\|^2] = (q+2)\|\boldsymbol{P}^\top \nabla f(\boldsymbol{x})\|^2$。注意到对任意正交矩阵 $\boldsymbol{Q} \in \mathbb{R}^{q\times q}$，有

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle \nabla f(\boldsymbol{x}), \hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\rangle^2}{\|\boldsymbol{P}^\top \nabla f(\boldsymbol{x})\|^2 \|\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\|^2}\right] &= \mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle \boldsymbol{P}^\top \nabla f(\boldsymbol{x}), \boldsymbol{z}\rangle^2}{\|\boldsymbol{P}^\top \nabla f(\boldsymbol{x})\|^2 \|\boldsymbol{z}\|^2}\right] \\
&= \mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle \boldsymbol{P}^\top \nabla f(\boldsymbol{x}), \boldsymbol{Q}\boldsymbol{z}\rangle^2}{\|\boldsymbol{P}^\top \nabla f(\boldsymbol{x})\|^2 \|\boldsymbol{Q}\boldsymbol{z}\|^2}\right] \\
&= \mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle \boldsymbol{Q}^\top \boldsymbol{P}^\top \nabla f(\boldsymbol{x}), \boldsymbol{z}\rangle^2}{\|\boldsymbol{Q}^\top \boldsymbol{P}^\top \nabla f(\boldsymbol{x})\|^2 \|\boldsymbol{z}\|^2}\right].
\end{aligned}$$

根据引理3.3，可取 $\boldsymbol{P}^{\mathsf{T}}\nabla f(\boldsymbol{x}) = [1, 0, \cdots, 0]^{\mathsf{T}}$。于是

$$\mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle \nabla f(\boldsymbol{x}), \hat{g}_{\varepsilon}(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\rangle^2}{\|\boldsymbol{P}^{\mathsf{T}}\nabla f(\boldsymbol{x})\|^2 \|\hat{g}_{\varepsilon}(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\|^2}\right] = \mathbb{E}_{\boldsymbol{z}}\left[\frac{\boldsymbol{z}_1^2}{\|\boldsymbol{z}\|^2}\right] = \frac{1}{q}.$$

∎

定理3.2揭示了 SubZero 在二次函数上的三个关键优势：

1. **无偏性**（式 (3-11)）：梯度估计的期望精确等于子空间 $\boldsymbol{P}$ 内的真实梯度 $\boldsymbol{P}\boldsymbol{P}^{\mathsf{T}}\nabla f(\boldsymbol{x})$。

2. **低方差**（式 (3-12)）：梯度估计的方差与子空间维度 q 线性相关。相比 MeZO（方差随 d 线性增长），当 $q \ll d$ 时 SubZero 显著降低了方差。例如，当 $d = 10^9$ 而 $q = 10^4$ 时，方差降低约 10^5 倍。

3. **准确的方向性**（式 (3-13)）：估计梯度与真实梯度的期望平方余弦相似度为 $\frac{1}{q}$，仅取决于子空间维度 q。这保证了参数更新方向的高度准确性，对优化效率至关重要。

3.3.5 收敛性分析

基于上述性质，本节给出 SubZero 的收敛性保证。本节的收敛性分析遵循由简到繁的框架：首先研究固定子空间内的局部收敛性质，随后分析随机子空间的统计性质，最后考虑子空间周期性切换对全局优化的影响，建立 SubZero 的全局收敛保证。

固定子空间内的局部收敛。为简化分析（遵循 Zhao 等人[15]的方法），本节先假设投影矩阵 $\boldsymbol{P}$ 在优化过程中固定。该假设合理，因为 SubZero 采用延迟更新策略：$\boldsymbol{P}$ 在多个迭代步内保持不变。本节给出必要的引理。

引理 3.8 (投影保持光滑性)： 设 $h(\boldsymbol{y}) = f(\boldsymbol{x} + \boldsymbol{P}\boldsymbol{y})$，其中 $f \in C_{L_1}^{1,1}(\mathbb{R}^d)$，$\boldsymbol{P}^{\mathsf{T}}\boldsymbol{P} = \boldsymbol{I}$，则 $h \in C_{L_1}^{1,1}(\mathbb{R}^q)$。

证明 若 f 是 L_1-光滑的，则对任意 $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathbb{R}^q$，

$$\begin{aligned}
\left\|\nabla h(\boldsymbol{y}_1) - \nabla h(\boldsymbol{y}_2)\right\| &= \left\|\boldsymbol{P}^{\mathsf{T}}\nabla f(\boldsymbol{x} + \boldsymbol{P}\boldsymbol{y}_1) - \boldsymbol{P}^{\mathsf{T}}\nabla f(\boldsymbol{x} + \boldsymbol{P}\boldsymbol{y}_2)\right\| \\
&\leqslant \left\|\boldsymbol{P}^{\mathsf{T}}\right\| \left\|\nabla f(\boldsymbol{x} + \boldsymbol{P}\boldsymbol{y}_1) - \nabla f(\boldsymbol{x} + \boldsymbol{P}\boldsymbol{y}_2)\right\| \\
&\leqslant L_1 \left\|\boldsymbol{P}(\boldsymbol{y}_1 - \boldsymbol{y}_2)\right\| \\
&= L_1 \left\|\boldsymbol{y}_1 - \boldsymbol{y}_2\right\|.
\end{aligned}$$

∎

引理 3.9 (随机梯度估计的范数上界)： [27] 设 $f \in C_{L_1}^{1,1}(\mathbb{R})$，则对任意 $\boldsymbol{x} \in \mathbb{R}$，有

$$\mathbb{E}_{\boldsymbol{z}}[\|g_{\varepsilon}(\boldsymbol{x})\|^2] = \mathbb{E}_{\boldsymbol{z}}\left[\left\|\frac{f(\boldsymbol{x} + \varepsilon\boldsymbol{z}) - f(\boldsymbol{x})}{\varepsilon}\right\|^2\right]$$

$$\leqslant 4(n+4)\|\nabla f_\varepsilon(\boldsymbol{x})\|^2 + 3\varepsilon^2 L_1^2(f)(n+4)^3.$$

以及

$$\|\nabla f(\boldsymbol{x})\|^2 \leqslant 2\|\nabla f_\varepsilon(\boldsymbol{x})\|^2 + \frac{\varepsilon^2}{2} L_1^2(f)(n+6)^3,$$

其中 $f_\varepsilon(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}}[f(\boldsymbol{x}+\varepsilon\boldsymbol{z})]$。

引理 3.10 (固定子空间下的收敛性)： 设 $\boldsymbol{y}^* = \operatorname{argmin}_{\boldsymbol{x}\in\mathbb{R}^q} h(\boldsymbol{y})$，其中 $h \in C_{L_1}^{1,1}(\mathbb{R}^q)$ 且 h 非凸。令 $\mathcal{E}_k = (\boldsymbol{z}_0, \boldsymbol{z}_1, \cdots, \boldsymbol{z}_k)$，$\boldsymbol{z}_k \sim \mathcal{N}(0, \boldsymbol{I}_q)$，步长 $\eta = \frac{1}{4(q+4)L_1}$。$\{\boldsymbol{y}_k\}_{k>0}$ 是由算法生成的序列。记 $\phi_0 = h(\boldsymbol{y}_0)$，对 $k \geqslant 1$，$\phi_k = \mathbb{E}_{\mathcal{E}_{k-1}}[h(\boldsymbol{y}_k)]$。则对于固定的 $\boldsymbol{P}$，有

$$\phi_{k+1} - \phi_k \leqslant -\frac{1}{4}\eta \mathbb{E}_{\mathcal{E}_k}\left[\|\nabla h(\boldsymbol{y}_k)\|^2\right] + \frac{\varepsilon^2(q+6)^3}{8} L_1^2 + \frac{3\varepsilon^2(q+4)}{32} L_1.$$

证明 若子空间 $\boldsymbol{P} \in \mathbb{R}^{d\times q}$ 固定，优化目标可重写为 $\min_{\boldsymbol{y}\in\mathbb{R}^q} h(\boldsymbol{y}) := f(\boldsymbol{x}+\boldsymbol{P}\boldsymbol{y})$。考虑随机梯度搜索算法 $\mathcal{RG}_\varepsilon(\varepsilon > 0)$：1) 生成 $\boldsymbol{z}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$，并计算对应的 $g_\varepsilon(\boldsymbol{y}_k)$；2) 更新 $\boldsymbol{y}_{k+1} = \boldsymbol{y}_k - \eta_k g_\varepsilon(\boldsymbol{y}_k)$。

考虑估计一次迭代后函数 h_ε 的变化。由于 h 是 L_1-光滑的，且 h_ε 的 Lipschitz 常数 $L_\varepsilon \leqslant L_1$[27]，有

$$h_\varepsilon(\boldsymbol{y}_{k+1}) \leqslant h_\varepsilon(\boldsymbol{y}_k) - \eta_k\langle\nabla h_\varepsilon(\boldsymbol{y}_k), g_\varepsilon(\boldsymbol{y}_k)\rangle + \frac{1}{2}\eta_k^2 L_\varepsilon\|g_\varepsilon(\boldsymbol{y}_k)\|^2.$$

对 $\boldsymbol{z}_k$ 取期望，得到

$$\mathbb{E}_{\boldsymbol{z}_k}[h_\varepsilon(\boldsymbol{y}_{k+1})] \leqslant h_\varepsilon(\boldsymbol{y}_k) - \eta_k\|\nabla h_\varepsilon(\boldsymbol{y}_k)\|^2 + \frac{1}{2}\eta_k^2 L_\varepsilon\,\mathbb{E}_{\boldsymbol{z}_k}[\|g_\varepsilon(\boldsymbol{y}_k)\|^2].$$

由 $h \in C^{1,1}(\mathbb{R}^q)$ 及引理3.9，有

$$\begin{aligned}\mathbb{E}_{\boldsymbol{z}_k}[h_\varepsilon(\boldsymbol{y}_{k+1})] \leqslant\ & h_\varepsilon(\boldsymbol{y}_k) - \eta_k\|\nabla h_\varepsilon(\boldsymbol{y}_k)\|^2 \\ & + \frac{1}{2}\eta_k^2 L_1\left(4(q+4)\|\nabla h_\varepsilon(\boldsymbol{y}_k)\|^2 + 3\varepsilon^2 L_1^2(q+4)^3\right).\end{aligned}$$

取 $\eta_k = \hat{\eta} = \frac{1}{4(q+4)L_1}$，则

$$\mathbb{E}_{\boldsymbol{z}_k}[h_\varepsilon(\boldsymbol{y}_{k+1})] \leqslant h_\varepsilon(\boldsymbol{y}_k) - \frac{1}{2}\hat{\eta}\|\nabla h_\varepsilon(\boldsymbol{y}_k)\|^2 + \frac{3\varepsilon^2}{32} L_1(q+4).$$

对 $\mathcal{E}_k$ 取期望，得

$$\phi_{k+1} \leqslant \phi_k - \frac{1}{2}\hat{\eta}\mathbb{E}_{\mathcal{E}_k}[\|\nabla h_\varepsilon(\boldsymbol{y}_k)\|^2] + \frac{3\varepsilon^2(q+4)}{32} L_1.$$

由引理3.9，$\mathbb{E}_{\mathcal{E}_k}[\|\nabla h(\boldsymbol{y}_k)\|^2] \leqslant 2\mathbb{E}_{\mathcal{E}_k}[\|\nabla h_\varepsilon(\boldsymbol{y}_k)\|^2] + \frac{\varepsilon^2(q+6)^3}{2} L_1^2$。因此，

$$\phi_{k+1} - \phi_k \leqslant -\frac{1}{4}\hat{\eta}\mathbb{E}_{\mathcal{E}_k}\left[\|\nabla h(\boldsymbol{y}_k)\|^2\right] + \frac{\varepsilon^2(q+6)^3}{8} L_1^2 + \frac{3\varepsilon^2(q+4)}{32} L_1. \tag{3-14}$$

∎

随机子空间的统计性质。接下来，本节分析随机子空间的统计性质。下述引

理表明，若投影矩阵由 Gram-Schmidt 正交化得到，则其期望满足特定形式。

引理 3.11 (随机投影性质 I)： 设矩阵 $\boldsymbol{A} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_r) \in \mathbb{R}^{n\times r}$ 由独立同分布列向量构成，且 $\boldsymbol{a}_k \sim \mathcal{N}(0, \boldsymbol{I}_n)$。对 $\boldsymbol{A}$ 执行 Gram-Schmidt 正交化过程：$\boldsymbol{u}_k = \boldsymbol{a}_k - \sum_{s=1}^{k-1}\langle \boldsymbol{a}_k, \boldsymbol{e}_s\rangle \boldsymbol{e}_s$，$\boldsymbol{e}_k = \frac{\boldsymbol{u}_k}{\|\boldsymbol{u}_k\|}$。记号 $[\boldsymbol{a}_k]_i \leftrightarrow [\boldsymbol{a}_k]_j$ 表示交换 $\boldsymbol{a}_k$ 的第 i 个和第 j 个元素，其他元素不变；$[\boldsymbol{a}_k]_i = -1\times[\boldsymbol{a}_k]_i$ 表示仅将 $\boldsymbol{a}_k$ 的第 i 个元素乘以 -1，其他元素不变。设 $f(\boldsymbol{A}, \boldsymbol{U}, \boldsymbol{E})$ 是矩阵 $\boldsymbol{A}$、$\boldsymbol{U} = (\boldsymbol{u}_1, \cdots, \boldsymbol{u}_r)$ 和 $\boldsymbol{E} = (\boldsymbol{e}_1, \cdots, \boldsymbol{e}_r)$ 的函数，则

1. 若进行 $[\boldsymbol{a}_k]_i \leftrightarrow [\boldsymbol{a}_k]_j$ 或 $[\boldsymbol{a}_k]_i = -1\times[\boldsymbol{a}_k]_i$ 操作，$\mathbb{E}[f]$ 保持不变。
2. $[\boldsymbol{a}_k]_i \leftrightarrow [\boldsymbol{a}_k]_j \Rightarrow [\boldsymbol{u}_k]_i \leftrightarrow [\boldsymbol{u}_k]_j$ 且 $[\boldsymbol{e}_k]_i \leftrightarrow [\boldsymbol{e}_k]_j$。
3. $[\boldsymbol{a}_k]_i = -1\times[\boldsymbol{a}_k]_i \Rightarrow [\boldsymbol{u}_k]_i = -1\times[\boldsymbol{u}_k]_i$，$[\boldsymbol{e}_k]_i = -1\times[\boldsymbol{e}_k]_i$，且对 $i \neq j$，$[\boldsymbol{u}_k]_j$、$[\boldsymbol{e}_k]_j$ 不变。
4. $\mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k\rangle}\right] = \frac{1}{n}$。
5. $\mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i[\boldsymbol{u}_k]_j}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k\rangle}\right] = 0$，其中 $i \neq j$。

证明 根据实分析，在高斯分布下 $\boldsymbol{A}$ 几乎处处满秩，且 $\boldsymbol{u}_k$、$\boldsymbol{e}_k$ 几乎处处非零。

1. 由于 $\boldsymbol{a}_k$ 独立同分布，结论显然成立。
2. 对 $k = 1$，结论显然。假设对 $k = 1, 2, \cdots, k-1$ 成立（$k \geqslant 2$），则 $[\boldsymbol{a}_k]_i \leftrightarrow [\boldsymbol{a}_k]_j$ 时，$[\boldsymbol{u}_k]_i = [\boldsymbol{a}_k]_i - \sum_{s=1}^{k-1}\langle \boldsymbol{a}_k, \boldsymbol{e}_s\rangle [\boldsymbol{e}_s]_i$，$[\boldsymbol{u}_k]_j = [\boldsymbol{a}_k]_j - \sum_{s=1}^{k-1}\langle \boldsymbol{a}_k, \boldsymbol{e}_s\rangle [\boldsymbol{e}_s]_j$，$[\boldsymbol{e}_k]_i = [\boldsymbol{u}_k]_i/\|\boldsymbol{u}_k\|$，$[\boldsymbol{e}_k]_j = [\boldsymbol{u}_k]_j/\|\boldsymbol{u}_k\|$。由强归纳假设，有 $[\boldsymbol{u}_k]_i \leftrightarrow [\boldsymbol{u}_k]_j$，$[\boldsymbol{e}_k]_i \leftrightarrow [\boldsymbol{e}_k]_j$。
3. 对 $k = 1$，结论显然。假设对 $k = 1, 2, \cdots, k-1$ 成立（$k \geqslant 2$），则

$$[\boldsymbol{a}_k]_i = -1\times[\boldsymbol{a}_k]_i$$
$$\Rightarrow \begin{cases} [\boldsymbol{u}_k]_i = [\boldsymbol{a}_k]_i \times(-1) - \sum_{s=1}^{k-1}\langle \boldsymbol{a}_k, \boldsymbol{e}_s\rangle [\boldsymbol{e}_s]_i \times(-1) = [\boldsymbol{u}_k]_i \times(-1), \\ [\boldsymbol{u}_k]_j = [\boldsymbol{u}_k]_j \times 1 \quad (i \neq j), \end{cases}$$

进而

$$\begin{cases} [\boldsymbol{e}_k]_i = [\boldsymbol{u}_k]_i/\|\boldsymbol{u}_k\| = [\boldsymbol{e}_k]_i \times(-1), \\ [\boldsymbol{e}_k]_j = [\boldsymbol{e}_k]_j \times 1 \quad (i \neq j). \end{cases}$$

由强归纳假设得证。

4. 由于 $0 \leqslant \frac{[\boldsymbol{u}_k]_i^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k\rangle} \leqslant 1$，期望存在。由性质 (2)，$[\boldsymbol{a}_k]_i \leftrightarrow [\boldsymbol{a}_k]_j$ 时 $\frac{[\boldsymbol{u}_k]_i^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k\rangle}$ 与 $\frac{[\boldsymbol{u}_k]_j^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k\rangle}$ 互换，因此所有 i 的期望相等。于是

$$\mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k\rangle}\right] \times n = \sum_{s=1}^{n}\mathbb{E}\left[\frac{[\boldsymbol{u}_k]_s^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k\rangle}\right] = \mathbb{E}\left[\frac{\langle \boldsymbol{u}_k, \boldsymbol{u}_k\rangle}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k\rangle}\right] = 1,$$

故 $\mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k\rangle}\right] = \frac{1}{n}$。

5. 由 $\left|\frac{[\boldsymbol{u}_k]_i[\boldsymbol{u}_k]_j}{\langle \boldsymbol{u}_k,\boldsymbol{u}_k\rangle}\right| \leqslant \left|\frac{[\boldsymbol{u}_k]_i^2+[\boldsymbol{u}_k]_j^2}{2\langle \boldsymbol{u}_k,\boldsymbol{u}_k\rangle}\right| \leqslant 1$，期望存在。根据性质 (3)，将 $[\boldsymbol{a}_k]_i$ 乘以 -1 会使 $[\boldsymbol{u}_k]_i$ 变号而 $[\boldsymbol{u}_k]_j$ 不变，从而 $\frac{[\boldsymbol{u}_k]_i[\boldsymbol{u}_k]_j}{\langle \boldsymbol{u}_k,\boldsymbol{u}_k\rangle}$ 变号。由于分布对称，期望为 0。

∎

引理 3.12 (随机投影性质 II)： 设 $\boldsymbol{A} \in \mathbb{R}^{n\times r}$ 的元素为独立同分布的标准正态随机变量。对 $\boldsymbol{A}$ 执行 Gram-Schmidt 正交化得到列正交矩阵 $\boldsymbol{Q} \in \mathbb{R}^{n\times r}$（列向量为 $\boldsymbol{e}_1,\cdots,\boldsymbol{e}_r$）和上三角矩阵 $\boldsymbol{R}$。则对每个 $k = 1,2,\cdots,r$，有

$$\mathbb{E}[\boldsymbol{e}_k\boldsymbol{e}_k^{\mathsf{T}}] = \frac{1}{n}\boldsymbol{I}_n,$$

其中 $\boldsymbol{I}_n$ 为 n 阶单位矩阵。

证明 由 Gram-Schmidt 过程，$\boldsymbol{e}_k = \boldsymbol{u}_k/\|\boldsymbol{u}_k\|$，$\boldsymbol{u}_k = \boldsymbol{a}_k - \sum_{s=1}^{k-1}\langle \boldsymbol{a}_k,\boldsymbol{e}_s\rangle \boldsymbol{e}_s$，故 $\boldsymbol{e}_k\boldsymbol{e}_k^{\mathsf{T}} = \frac{\boldsymbol{u}_k\boldsymbol{u}_k^{\mathsf{T}}}{\langle \boldsymbol{u}_k,\boldsymbol{u}_k\rangle}$。其 (i,j) 元为 $\mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i[\boldsymbol{u}_k]_j}{\langle \boldsymbol{u}_k,\boldsymbol{u}_k\rangle}\right]$。由引理3.11，当 $i=j$ 时该期望为 $1/n$，当 $i\neq j$ 时为 0。因此 $\mathbb{E}[\boldsymbol{e}_k\boldsymbol{e}_k^{\mathsf{T}}] = \frac{1}{n}\boldsymbol{I}_n$。 ∎

引理 3.13 (随机投影性质 III)： 设 $\boldsymbol{A} \in \mathbb{R}^{n\times r}$ 的元素为独立同分布的标准正态随机变量。对 $\boldsymbol{A}$ 进行 QR 分解得到列正交矩阵 $\boldsymbol{Q} \in \mathbb{R}^{n\times r}$。则

$$\mathbb{E}[\boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}}] = \frac{r}{n}\boldsymbol{I}_n.$$

证明 由线性性及 $\boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}} = \sum_{k=1}^{r}\boldsymbol{e}_k\boldsymbol{e}_k^{\mathsf{T}}$，结合引理3.12得

$$\mathbb{E}[\boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}}] = \sum_{k=1}^{r}\mathbb{E}[\boldsymbol{e}_k\boldsymbol{e}_k^{\mathsf{T}}] = \sum_{k=1}^{r}\frac{1}{n}\boldsymbol{I}_n = \frac{r}{n}\boldsymbol{I}_n.$$

∎

引理 3.14 (随机投影性质 IV)： 设 $\boldsymbol{A}_1 \in \mathbb{R}^{m\times r}$、$\boldsymbol{A}_2 \in \mathbb{R}^{n\times r}$ 的元素均为独立同分布的标准正态随机变量。分别对它们进行 QR 分解得到列正交矩阵 $\boldsymbol{Q}_1 \in \mathbb{R}^{m\times r}$、$\boldsymbol{Q}_2 \in \mathbb{R}^{n\times r}$。定义 $\boldsymbol{P} = \boldsymbol{Q}_2 \otimes \boldsymbol{Q}_1$，则

$$\mathbb{E}[\boldsymbol{P}\boldsymbol{P}^{\mathsf{T}}] = \frac{r^2}{mn}\boldsymbol{I}_{mn}.$$

证明 由 Kronecker 积的性质，

$$\mathbb{E}[\boldsymbol{P}\boldsymbol{P}^{\mathsf{T}}] = \mathbb{E}[(\boldsymbol{Q}_2\otimes\boldsymbol{Q}_1)(\boldsymbol{Q}_2^{\mathsf{T}}\otimes\boldsymbol{Q}_1^{\mathsf{T}})] = \mathbb{E}[(\boldsymbol{Q}_2\boldsymbol{Q}_2^{\mathsf{T}})]\otimes\mathbb{E}[(\boldsymbol{Q}_1\boldsymbol{Q}_1^{\mathsf{T}})] = \frac{r}{n}\boldsymbol{I}_n\otimes\frac{r}{m}\boldsymbol{I}_m = \frac{r^2}{mn}\boldsymbol{I}_{mn}.$$

∎

子空间切换下的全局收敛。最后，结合实际优化过程中子空间的延迟更新机制，本节给出 SubZero 的全局收敛性定理及其证明。

定理 3.3 (全局收敛性)： 设 $f \in C_{L_1}^{1,1}(\mathbb{R}^d)$ 为非凸函数，且存在下界 f^*。记随机噪声序列为 $\mathcal{E}_k = (\boldsymbol{z}_0,\boldsymbol{z}_1,\cdots,\boldsymbol{z}_k)$，其中 $\boldsymbol{z}_k \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I}_q)$；记投影矩阵序列为 $\mathcal{P}_j = (\boldsymbol{P}_0,\boldsymbol{P}_1,\cdots,\boldsymbol{P}_j)$，其中 $\boldsymbol{P}_j$ 由式 (3-7) 定义，且具有固定的更新频率 F。则由 SubZero

算法生成的序列 $\{\boldsymbol{x}_k\}_{k>0}$ 满足：

$$\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}_{\mathcal{E}_k,\mathcal{P}_{\lfloor k/F\rfloor}}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right]\leqslant\epsilon,$$

其中，若扰动尺度 ε 满足 $\varepsilon\leqslant\mathcal{O}\left(\frac{\epsilon^{1/2}}{q^{3/2}d^{1/2}L_1^{3/2}}\right)$，则迭代次数 T 的量级为 $T=\mathcal{O}\left(\frac{d}{\epsilon}\right)$。此处 $T=KF$，K 表示子空间更新的总次数。

证明 设 $\mathcal{P}_j=(\boldsymbol{P}_0,\boldsymbol{P}_1,\cdots,\boldsymbol{P}_j)$，其中 $\boldsymbol{P}_j$ 是由算法生成的投影矩阵序列，且 $j\leqslant K$。根据引理3.8和引理3.10，当子空间固定时，可通过变换 $h(\boldsymbol{y})=f(\boldsymbol{x}+\boldsymbol{P}\boldsymbol{y})$ 将原问题 $f\in C_{L_1}^{1,1}(\mathbb{R}^d)$ 转化为 $h\in C_{L_1}^{1,1}(\mathbb{R}^q)$。考虑更新规则：

$$\begin{aligned}\boldsymbol{y}_{j,0}&=0,\quad h_j(\boldsymbol{y})=f(\boldsymbol{x}_{jF}+\boldsymbol{P}_j\boldsymbol{y}),\quad\forall j\in\{0,1,\cdots,K-1\},\\ \boldsymbol{y}_{j,k}&=\boldsymbol{y}_{j,k-1}-\eta\widehat{\nabla}h_j(\boldsymbol{y}_{j,k-1}),\quad\forall k\in\{0,1,\cdots,F\},\\ \boldsymbol{x}_{jF+k}&=\boldsymbol{x}_{jF}+\boldsymbol{P}_j\boldsymbol{y}_k.\end{aligned}$$

在第 j 个子空间内，投影矩阵 $\boldsymbol{P}_j$ 保持不变，因此可在当前子空间内累积 ϕ 的变化。由引理3.10，

$$\begin{aligned}\phi_{(j+1)F}-\phi_{jF}&\leqslant-\frac{1}{4}\hat{\eta}\sum_{i=0}^{F-1}\mathbb{E}_{\mathcal{E}_{jF+i}}\left[\|\nabla h_j(\boldsymbol{y}_{j,i})\|^2\right]+\frac{\varepsilon^2(q+6)^3}{8}FL_1^2+\frac{3\varepsilon^2(q+4)}{32}FL_1\\&\leqslant-\frac{1}{4}\hat{\eta}\mathbb{E}_{\mathcal{E}_{jF}}\left[\|\nabla h_j(\boldsymbol{y}_{j,0})\|^2\right]+\frac{\varepsilon^2(q+6)^3}{8}FL_1^2+\frac{3\varepsilon^2(q+4)}{32}FL_1.\end{aligned}$$

注意到 $\nabla h_j(\boldsymbol{y}_{j,0})=\boldsymbol{P}_j^{\mathsf{T}}\nabla f(\boldsymbol{x}_{jF})$。对历史投影矩阵 $\mathcal{P}_j$ 取期望，并利用引理3.14（$\mathbb{E}[\boldsymbol{P}_j\boldsymbol{P}_j^{\mathsf{T}}]=\frac{q}{d}\boldsymbol{I}$，且 $\boldsymbol{P}_j$ 与 $\boldsymbol{x}_{jF}$ 独立），得

$$\begin{aligned}&\mathbb{E}_{\mathcal{P}_{j+1}}[\phi_{(j+1)F}]-\mathbb{E}_{\mathcal{P}_j}[\phi_{jF}]\\&\leqslant-\frac{1}{4}\hat{\eta}\mathbb{E}_{\mathcal{E}_{jF},\mathcal{P}_j}\left[\|(\boldsymbol{P}_j)^{\mathsf{T}}\nabla f(\boldsymbol{x}_{jF})\|^2\right]+\frac{\varepsilon^2(q+6)^3}{8}FL_1^2+\frac{3\varepsilon^2(q+4)}{32}FL_1\\&=-\frac{q}{4d}\hat{\eta}\mathbb{E}_{\mathcal{E}_{jF},\mathcal{P}_j}\left[\|\nabla f(\boldsymbol{x}_{jF})\|^2\right]+\frac{\varepsilon^2(q+6)^3}{8}FL_1^2+\frac{3\varepsilon^2(q+4)}{32}FL_1.\end{aligned}$$

假设 $f(\boldsymbol{x})\geqslant f^*$ 对所有 $\boldsymbol{x}\in\mathbb{R}^d$ 成立，并令 $T=KF$，将上述不等式对 j 求和得到

$$\begin{aligned}\mathbb{E}_{\mathcal{P}_{K-1}}[\phi_T]\leqslant\mathbb{E}_{\mathcal{P}_0}[\phi_0]&-\frac{q}{4d}\hat{\eta}\sum_{j=0}^{K-1}\mathbb{E}_{\mathcal{E}_{jF},\mathcal{P}_j}\left[\|\nabla f(\boldsymbol{x}_{jF})\|^2\right]\\&+T\frac{\varepsilon^2(q+6)^3}{8}L_1^2+T\frac{3\varepsilon^2(q+4)}{32}L_1.\end{aligned}$$

由于 $\mathbb{E}_{\mathcal{P}_{K-1}}[\phi_T]\geqslant f^*$，移项得

$$\frac{q}{4d}\hat{\eta}\sum_{j=0}^{K-1}\mathbb{E}_{\mathcal{E}_{jF},\mathcal{P}_j}\left[\|\nabla f(\boldsymbol{x}_{jF})\|^2\right]\leqslant\mathbb{E}_{\mathcal{P}_0}[\phi_0]-f^*+T\frac{\varepsilon^2(q+6)^3}{8}L_1^2+T\frac{3\varepsilon^2(q+4)}{32}L_1.$$

代入 $\hat{\eta}=\frac{1}{4(q+4)L_1}$，得

$$\frac{q}{16d(q+4)L_1}\sum_{j=0}^{K-1}\mathbb{E}_{\mathcal{E}_{jF},\mathcal{P}_j}\left[\|\nabla f(\boldsymbol{x}_{jF})\|^2\right]\leqslant\mathbb{E}_{\mathcal{P}_0}[\phi_0]-f^*+T\frac{\varepsilon^2(q+6)^3}{8}L_1^2+T\frac{3\varepsilon^2(q+4)}{32}L_1.$$

因此，

$$\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}_{\mathcal{E}_k,\mathcal{P}_{\lfloor k/F\rfloor}}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right]\leqslant\frac{16(q+4)dL_1(\mathbb{E}_{\mathcal{P}_0}[\phi_0]-f^*)}{qT}+\frac{2\varepsilon^2(q+6)^3(q+4)d}{q}L_1^3+\frac{3\varepsilon^2(q+4)^2d}{2q}L_1^2.$$

为使 $\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}[\|\nabla f(\boldsymbol{x}_k)\|^2]\leqslant\epsilon$，可取

$$\varepsilon\leqslant\mathcal{O}\left(\frac{\epsilon^{1/2}}{q^{3/2}d^{1/2}L_1^{3/2}}\right),$$

则期望迭代步数的上界为 $\mathcal{O}\left(\frac{d}{\epsilon}\right)$。 ∎

3.4 实验结果与分析

本节通过综合实验评估 SubZero 的有效性。本节首先介绍实验设置，包括数据集、模型、对比算法和实现细节。然后展示主实验结果，对比 SubZero 与主流零阶优化算法的性能。接着通过消融实验验证关键设计选择。最后分析 SubZero 的收敛行为和计算效率。

3.4.1 实验设置

模型与数据集。实验使用两类语言模型：（1）中等规模掩码语言模型 RoBERTa-large[79]；（2）大规模自回归语言模型 OPT-1.3B/13B[75]、LLaMA2-7B[80]和 Mistral-7B[81]。下游任务涵盖 SuperGLUE 基准[82]（包括 BoolQ[83]、CB[84]、COPA[85]、MultiRC[86]、ReCoRD[87]、RTE[88]、WiC[89]、WSC[90]）、SST-2[91]、SQuAD[92] 和 DROP[93]等。对于 RoBERTa-large，额外使用 SST-5[91]、MNLI[94]和 SNLI[95]评估非可微目标优化能力。参照 MeZO[16]实验设置，每个任务随机抽取 1000 个样本用于训练，500 个用于验证，1000 个用于测试（RoBERTa-large 每类 512 个样本）。

评估指标方面，分类任务与多项选择任务采用准确率（Accuracy, %），生成任务（SQuAD、DROP）采用 F1 分数（%），所有指标均以百分比形式呈现。

微调方案。遵循 MeZO[16]的设置，本节评估四种微调方案：全参数微调（FT）[74]和三种参数高效微调（PEFT）方法——LoRA[6]、前缀微调（Prefix Tun-

ing）[76]和提示微调（Prompt Tuning）[77]。其中 LoRA 秩设为 8，LoRA 的系数 α 设为 16，前缀微调的词元长度设为 5，提示微调的词元长度设为 10。

对比算法。零阶优化算法包括 MeZO[16]、ZO-AdaMU[28]、S-MeZO[19]、Hi-ZOO[96]和 LOZO[73]。作为首个且最流行的 LLM 零阶优化算法，MeZO 是本章的主要对比基线，同时本章使用一阶优化器 SGD 作为性能基准。同时对比零样本、上下文学习（In-Context Learning, ICL）[97]和线性探测（Linear Probing, LP）[98]。

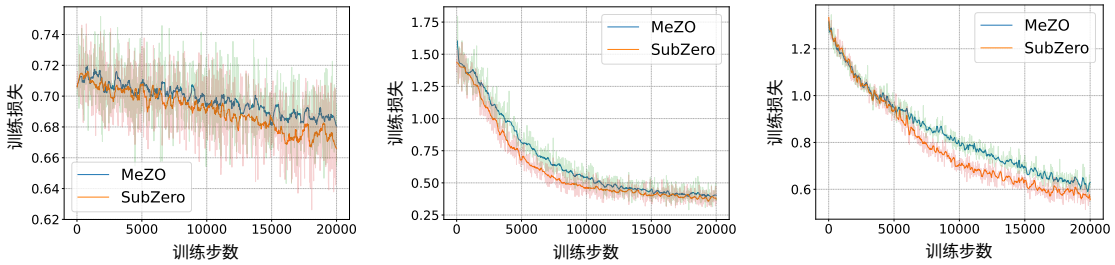
实现细节。所有实验在单张 A800 GPU 上使用 PyTorch 2.1.0+CUDA 11.8 实现。默认批量大小为 16，使用无动量 SGD 优化器。SubZero 采用延迟更新策略，子空间更新频率在 $\{500, 1000, 2000\}$ 中选择，秩 r 根据模型规模在 $\{4, 8, 16, 32, 64, 128\}$ 中选择。借助范数对齐技巧（见附录 A.1），SubZero 可直接沿用 MeZO 的超参数设置。完整超参数搜索网格见附录 A.2。

3.4.2 性能评估

SuperGLUE 基准上的性能对比。遵循 MeZO[16]的设置，本节在 SuperGLUE 基准[82]上使用 OPT-13B 评估 SubZero。该基准涵盖分类、多项选择和生成等多种任务类型（见表 3.3）。零阶算法应用于全参数微调（FT）和 LoRA 两种方案。

表 3.3　SuperGLUE 基准上 OPT-13B 微调性能对比

任务类型	分类任务							多项选择		生成任务		平均相对
任务	SST-2	RTE	CB	BoolQ	WSC	WIC	MultiRC	COPA	ReCoRD	SQuAD	DROP	百分比
SGD(FT)	94.9	82.3	85.7	78.4	65.3	65.8	74.2	90.0	82.4	88.0	35.5	-
零样本	58.8	59.6	46.4	59.0	38.5	55.0	46.9	80.0	81.2	46.2	14.6	-
上下文学习	87.0	62.1	57.1	66.9	39.4	50.5	53.1	87.0	82.5	75.9	29.6	-
线性探测	93.4	68.6	67.9	59.3	63.5	60.2	63.5	55.0	27.1	3.7	11.1	-
MeZO(FT)	92.1	71.5	71.4	74.4	61.5	60.0	60.1	87.0	82.0	84.2	31.2	0%
ZO-AdaMU(FT)	92.1	72.9	67.9	73.0	61.5	60.7	63.0	**89.0**	**83.0**	82.4	**32.0**	0.46%
S-MeZO(FT)	92.3	**76.9**	**75.0**	**76.5**	61.1	58.2	**63.3**	87.0	71.2	77.9	31.9	-0.10%
HiZOO(FT)	91.3	69.3	69.4	67.3	63.5	59.4	55.5	88.0	81.4	81.9	31.3	-2.15%
LOZO(FT)	**92.9**	73.6	71.4	70.7	63.5	60.2	60.3	87.0	81.7	**84.5**	30.7	0.10%
SubZero(FT)	92.1	74.0	73.2	75.3	**65.4**	**60.8**	61.0	88.0	82.3	**84.5**	**32.0**	**1.89%**
MeZO(LoRA)	92.2	74.4	69.6	75.2	64.4	59.7	58.2	87.0	82.0	82.9	31.0	0%
ZO-AdaMU(LoRA)	88.0	72.0	71.6	72.6	60.1	56.4	58.9	88.0	**83.2**	76.8	**32.4**	-1.78%
S-MeZO(LoRA)	90.8	62.2	**75.0**	72.9	51.9	55.8	56.4	86.0	69.9	76.4	31.7	-5.79%
HiZOO(LoRA)	90.6	67.5	69.6	70.5	63.5	60.2	60.2	87.0	81.9	**83.8**	31.2	-1.16%
SubZero(LoRA)	**93.8**	**75.5**	71.4	**76.1**	**65.4**	**60.3**	**60.3**	**89.0**	81.9	83.7	31.3	**1.57%**



(a) WIC: OPT-13B, FT　(b) SQuAD: OPT-13B, LoRA　(c) CB: LLaMA2-7B, Prompt

图 3.2　不同模型与设置下的训练损失曲线对比

表 3.3展示了关键结果，最优零阶算法以粗体标出。结果表明：零阶算法显著优于零样本、上下文学习和线性探测等基线算法，凸显了其提升预训练模型下游任务性能的能力。经超参数精细调优后，作为首个 LLM 零阶优化器的 MeZO 表现出强劲竞争力。在 FT 方案中，除 SubZero 外，仅 ZO-AdaMU 和 LOZO 能超越 MeZO。SubZero 在所有任务和微调方案中均持续优于 MeZO：在 FT 方案中，SubZero 相比 MeZO 平均提升 1.89%，且在所有任务上均保持稳定优势；在 LoRA 方案中，平均提升 1.57%，显著优于 S-MeZO（-5.79%）和 ZO-AdaMU（-1.78%）。在 FT 方案中，S-MeZO 在多个分类任务上表现良好，但在 ReCoRD 任务上即使经过超参数调优仍表现不佳。排除 ReCoRD 后，SubZero 在 FT 方案中仍以 2.05% 对 1.20%、在 LoRA 方案中以 1.74% 对-4.90% 的优势领先 S-MeZO。

跨模型与跨方案的泛化性。此外，本节针对 SST-2 任务微调 OPT-1.3B，针对 CB 任务微调 LLaMA2-7B 和 Mistral-7B，并基于全参数微调 (FT) 及三种 PEFT 方案（LoRA、前缀微调、提示微调）对 SubZero 进行了评估。如表 3.4 所示，SubZero 在所有模型和微调方案中均优于 MeZO。值得注意的是，当 MeZO 在提示微调方案中表现欠佳时，SubZero 却表现出色，性能接近 SGD 优化器。

表 3.4　不同模型与微调方案的性能对比

	LLaMA2-7B				Mistral-7B				OPT-1.3B			
	FT	LoRA	前缀	提示	FT	LoRA	前缀	提示	FT	LoRA	前缀	提示
SGD	69.6	75.0	69.6	69.6	73.2	75.0	69.6	62.5	93.2	93.0	93.1	90.7
MeZO	64.3	73.2	69.6	60.7	62.5	69.6	58.3	57.1	92.3	92.8	91.6	85.9
SubZero	**71.4**	**75.0**	**76.8**	**66.1**	**64.3**	**73.2**	**64.3**	**62.5**	**93.4**	**92.9**	**92.2**	**89.1**

图 3.2(a)-(c) 展示了多条训练损失曲线，表明 SubZero 通常比 MeZO 收敛更快且达到更低损失值。

非可微目标优化。遵循 MeZO[16]，本节使用全参数微调方案，在 RoBERTa-large 和 OPT-13B 上应用 SubZero 优化两个非可微目标：分类任务的准确率和 SQuAD 任务的 F1 分数。基线方法还包括使用可微交叉熵目标配合 Adam、MeZO 和 SubZero 的结果。如表 3.5所示，在两个非可微目标下，SubZero 持续优于 MeZO，且与交叉熵目标下的性能差距缩小，表明其对不可导目标的鲁棒性。

3.4.3 消融实验

本节通过三项消融实验验证 SubZero 关键技术的有效性。具体分析涵盖正交投影矩阵、子空间秩与更新频率以及 PEFT 非方阵重塑策略对模型性能的影响，旨在证实本章所提各核心设计的必要性。

正交投影矩阵的有效性。表 3.6 系统验证了列正交约束在子空间构建中的关键作用。实验结果显示，在 RTE 和 WSC 任务上，采用基于 QR 分解生成的列

表 3.5　非可微目标下的微调性能对比

模型	RoBERTa-large				OPT-13B
任务	SST-2	SST-5	SNLI	MNLI	SQuAD
零样本	79.0	35.5	50.2	48.8	46.2
交叉熵 (Adam)	93.9	55.9	88.7	83.8	84.2
交叉熵 (MeZO)	92.9	53.2	83.0	77.0	84.2
交叉熵 (SubZero)	93.4	54.0	84.7	77.4	84.5
准确率/F1 (MeZO)	92.4	46.5	81.9	73.9	80.2
准确率/F1 (SubZero)	**92.7**	**47.1**	**83.0**	**74.8**	**81.1**

正交投影矩阵，相较于直接使用高斯随机矩阵，准确率分别显著提升了 6.5% 和 5.5%。这一性能增益归因于正交基能够更有效地保持子空间内的几何结构，避免扰动方向的冗余，从而提升梯度估计质量。

表 3.6　正交投影矩阵效果对比

数据集	是否正交	准确率
RTE	✗	67.5
	✓	**74.0**
WSC	✗	59.6
	✓	**65.1**

表 3.7　子空间更新频率 F 与秩 r 的影响

$F\backslash r$	32	64	128
500	**72.6**	70.0	72.2
1000	73.6	71.8	**74.0**
2000	72.2	**73.3**	72.2
20000	70.4	**71.1**	68.6

子空间维度与更新频率。表 3.7分析了子空间秩 r 和更新频率 F 的影响。SubZero 对秩的选择具有鲁棒性，在 $r \in [32, 128]$ 范围内性能波动小于 3%。但当 $F = 20000$（即单个子空间贯穿训练）时性能显著下降，验证了子空间更新策略的必要性——过于稀疏的子空间切换会降低对优化轨迹的适应能力。

非方阵重塑策略。对于 PEFT 中存在的高度非方阵（如 LoRA 的低秩矩阵 $A \in \mathbb{R}^{d\times r}, r \ll d$），本章提出重塑策略以适配低秩扰动。表 3.8 展示了 OPT-1.3B 在 SST-2 任务上的验证结果。实验表明，重塑策略平均提升 5.3%，尤其在提示微调中提升达 14.9%，验证了第 3.2.3 节中重塑机制的有效性。

表 3.8　PEFT 方案中非方阵重塑策略的效果

算法	LoRA	前缀	提示
MeZO	92.8	91.6	85.9
SubZero(无重塑)	92.1	89.4	74.2
SubZero(有重塑)	**92.9**	**92.2**	**89.1**

3.4.4 进一步分析

显存效率与计算开销。尽管 SubZero 在生成投影矩阵时因 QR 分解引入了额外计算与显存开销，但实验表明该资源成本严格可控。本节对比了零阶算法

MeZO 和 SubZero、SGD 以及仅推理方法（零样本学习和上下文学习）在 OPT-13B 上的显存消耗与运行时间。如表3.9所示，零阶算法（包括 SubZero）相比 SGD 实现超过 1.8 倍的显存降低。值得注意的是，SubZero 的显存占用与 MeZO 高度接近，同时性能更优。本章在 MeZO 和 SubZero 实验中均采用逐层参数更新策略（见附录A.1），这使得全参数微调和 LoRA 方案在保留一位小数时显存占用几乎相同。

表 3.9　OPT-13B 微调的峰值显存 (GB) 与运行时间 (分钟) 对比

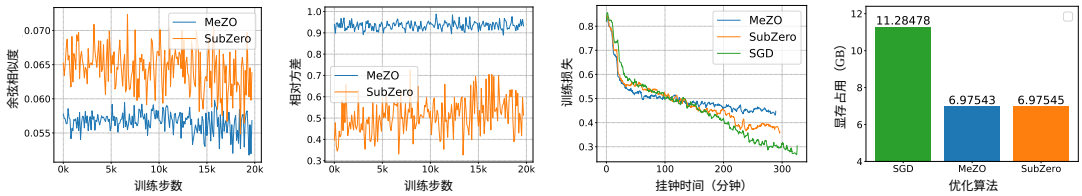
任务	SST-2		WIC		SQuAD	
方法	显存	时间	显存	时间	显存	时间
零样本/上下文学习	24.2	0	24.8	0	27.2	0
SGD(全参数微调)	48.9	190.3	48.9	257.3	122.7	623.7
MeZO(全参数微调)	26.1	324.9	26.6	370.5	37.4	670.2
SubZero(全参数微调)	26.5	337.3	27.1	385.3	37.8	690.5
MeZO(LoRA)	26.1	123.9	26.6	171.6	37.4	476.7
SubZero(LoRA)	26.1	130.3	26.6	179.7	37.4	486.5

表3.10展示了不同规模 OPT 系列模型上的实验结果，所有模型微调 20K 步。实验中，参数量低于 6.7B 的模型使用 FP32 精度，6.7B 及以上规模的模型使用 BF16 精度。具体而言，额外时间开销低于 8.5%（6.7B 模型峰值 8.36%），显存开销在 BF16 精度下仍低于 1.8%。这说明随着模型复杂度增加，QR 分解的计算成本渐近可忽略，从而确立了 SubZero 的实用可扩展性。

表 3.10　OPT 系列模型在 SST-2 数据集上微调的峰值显存与运行时间对比

模型	显存 (GB)			时间 (秒)		
	MeZO	SubZero	额外开销 (%)	MeZO	SubZero	额外开销 (%)
OPT-1.3B	6.80	6.88	**1.18**	11215	11683	**4.18**
OPT-2.7B	12.40	12.60	**1.61**	21579	22335	**3.51**
OPT-6.7B	13.98	14.20	**1.57**	9833	10655	**8.36**
OPT-13B	26.08	26.53	**1.73**	18668	19245	**3.09**

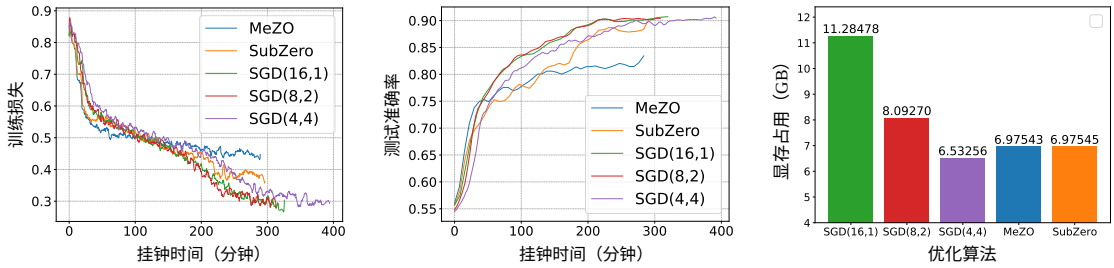
梯度估计质量。如图 3.3 所示，SubZero 显著改善了梯度估计的质量。具体而言，图 3.3(a) 展示了估计梯度 $\hat{\boldsymbol{g}}$ 与真实期望梯度 $\boldsymbol{g}$ 之间的余弦相似度 $\mathbb{E}[\cos(\hat{\boldsymbol{g}}, \boldsymbol{g})]$，SubZero 相比 MeZO 有大幅提升。图 3.3(b) 展示了梯度估计的相对方差 $\mathrm{Var}[\|\hat{\boldsymbol{g}}\|^2]/\|\boldsymbol{g}\|^2$，SubZero 同样显著降低。这种更高质量的梯度估计直接转化为更快的训练收敛速度，如图 3.3(c) 所示。此外，图 3.3(d) 表明 SubZero 在显著提升性能的同时，保持了与 MeZO 接近的极低峰值显存开销。



(a) 余弦相似度　(b) 相对方差　(c) 训练损失　(d) 显存开销

图 3.3　提示微调方案下 OPT-1.3B 模型在 SST-2 数据集上的实验结果对比

与一阶优化器的内存-性能权衡。在正式实验中，本章为一阶优化器和零阶优化器设置了相同的批量大小。此处本节调整 SGD 的梯度累积步数，使其内存占用与零阶优化器匹配，进而比较其收敛速度与性能。其中 SGD(BS, GA) 表示批量大小为 BS、梯度累积步数为 GA 的 SGD。实验设置与图 3.3 相同，实验结果如图 3.4 所示（所有算法均训练 20K 步）。在相近内存占用下，SubZero 的收敛速率几乎与 SGD 相当，超越 MeZO，并达到与 SGD 相当的测试准确率。



(a) 训练损失　(b) 测试准确率　(c) 显存开销

图 3.4　OPT-1.3B 在 SST-2 数据集上采用提示微调时的训练指标对比

稳定性分析。本节进一步考察了随机种子和批次大小对 SubZero 的影响。

首先，本节在提示微调方案下使用三个随机种子对 OPT-1.3B 模型在 SST-2 数据集上进行微调。结果如表3.11所示，超参数设置见表A.2。不同随机种子下，MeZO 的方差较大，而 SubZero 方差小且平均性能更优。

表 3.11　随机种子对 OPT-1.3B 提示微调的影响

种子	42	0	1234	平均
MeZO	85.9	83.3	80.7	83.3
SubZero	**89.1**	**89.4**	**89.2**	**89.2**

接着，本节使用 RoBERTa-large 模型在全参数微调方案下于 SST-2 数据集上考察批量大小对零阶优化器的影响。结果如表3.12所示。表3.5中训练轮次为 100K，而表3.12中为 20K。其余超参数与表3.5一致，详见附录A.2。对于零阶优化器，较大批量总能获得更好性能。在各批量大小下，SubZero 均展现出优于 MeZO 的微调性能。

表 3.12　批量大小对 RoBERTa-large 全参数微调的影响

批量大小	算法	SST-2	SST-5	SNLI	MNLI	平均
16	MeZO	91.7	44.7	77.3	**53.0**	66.7
	SubZero	**91.9**	**45.9**	**77.5**	52.8	**67.0**
32	MeZO	92.9	45.4	78.3	53.2	67.5
	SubZero	**93.0**	**45.5**	**79.6**	**54.0**	**68.0**

3.5 本章小结

本章针对传统零阶优化梯度估计方差高、收敛慢的问题，提出了基于随机子空间的零阶梯度估计算法 SubZero，在维持推理级显存开销的基础上改善了梯度估计质量。理论分析建立了该算法对反向传播梯度的逼近性保证，推导了其估计方差较 MeZO 等传统算法的显著降低，并给出了与 SGD 结合时的收敛保证。实验验证涵盖全参数微调与 LoRA、前缀微调等多种参数高效微调方案，结果表明 SubZero 在精度与收敛速度上均有提升。本章工作为资源受限场景下的大模型微调提供了一种可行的零阶梯度估计方案。

第 4 章 基于极坐标向量量化的优化器状态压缩算法

上一章聚焦第一个关键环节——梯度估计，提出了基于随机子空间的零阶梯度估计算法 SubZero，有效降低了梯度估计方差，提升了零阶优化算法的梯度估计质量。本章聚焦第二个关键环节——优化器状态更新，针对基于动量的优化器状态显存占用大、且现有量化方法在低于 4 比特时易引发训练崩溃的问题，提出基于极坐标向量量化的优化器状态压缩算法 Polaris。

4.1 引言

优化器状态量化是降低大模型训练显存的重要手段[11,17,37]。现有研究主要聚焦标量量化（Scalar Quantization, SQ），即独立地对每个浮点数进行低位宽编码（如将 32 位压缩至 4 位）[17,37]。然而，当比特宽度进一步降低至 4 比特以下时，标量量化面临根本性局限：由于标量量化的矩形网格无法适配优化器状态的非均匀分布，导致极低比特下量化误差急剧增大，进而严重制约模型训练的稳定性与最终性能。

向量量化（Vector Quantization, VQ）为突破这一瓶颈提供了自然思路。作为一种经典的数据压缩技术，其核心思想是将多个标量参数组合成向量，并通过共享的码本进行联合编码。相较于标量量化，VQ 通过联合编码多个维度的参数，能更好地捕捉数据的内在几何结构，理论上可提供更高的压缩上限[25-26]。

尽管向量量化在模型推理与权重压缩领域成果斐然[39,60]，但其在模型训练，特别是针对优化器状态的压缩上，几乎仍是空白。现有研究尚未系统性地探索如何为 AdamW 等优化器的动态状态设计高效、稳定的向量量化方案。直接套用通用向量量化技术（如基于欧氏距离最小化的 K-means 聚类[39,60]）在优化器状态压缩场景下并非最优解，主要面临两方面挑战：一方面，优化器状态是高频更新的动态变量，而在线 K-means 涉及昂贵的迭代计算开销，显著增加训练延迟；另一方面，通用 VQ 通常以最小化均方误差（MSE）为目标，可能导致重建误差低但训练稳定性差，这与优化器的数值敏感性需求并不兼容。具体而言，一阶动量决定更新方向，需降低角度误差以保证梯度方向准确；二阶动量位于分母位置，需严格避开零点以防除零错误引发训练发散。因此，设计面向优化器状态的专用向量量化方案，需在计算开销、量化误差控制和训练稳定性之间取得平衡。

针对上述挑战，本章提出基于极坐标向量量化的优化器状态压缩算法——Polaris。优化器状态的一阶和二阶动量呈现显著的非均匀分布特性，极坐标的半径与角度解耦表示天然适配这一结构：角度划分控制梯度更新方向，通过均匀角

量化降低方向误差；半径划分控制步长幅度，通过非均匀径向量化确保二阶动量远离零点以防数值发散。该设计支持半径和角度的自适应比特分配，结构化网格避免昂贵的训练与搜索开销，使 1.5-2 比特优化器首次在预训练任务上达到与全精度相当的效果。

综上，本章主要贡献如下：

（1）针对传统一维标量量化在极低比特（如 2 比特）下性能显著退化的问题，深入分析了优化器动量的准高斯分布与圆对称性特征，提出了基于极坐标向量量化的优化器状态压缩框架——Polaris。该框架基于上述分布特性构建了适配 AdamW/Adafactor 的极坐标向量量化方案，显著降低了优化器状态的显存开销。

（2）针对有符号动量数据，从理论上严格推导了角度均匀采样的合理性，并给出了量化误差的上界估计；针对无符号动量数据，提出第一象限映射机制，设计了幅值自适应的角度分配策略，即为大模值分量分配更多码字以保障量化精度，为小模值分量分配较少码字以提升码本利用率；同时引入轴偏移技术，有效规避除零风险，确保训练过程的数值稳定性。基于上述优化器动量向量量化的设计原则，进一步提出了数据驱动的极坐标码本搜索算法。静态实验表明，Polaris 在表征能力上显著优于传统一维标量量化基线。

（3）通过在自然语言处理与计算机视觉两大领域的多项预训练及微调任务中的实验，验证了 1.5 比特与 2 比特版本的 AdamW 及 Adafactor 在显存占用大幅降低的同时性能与 16/32 比特全精度版本相当。

4.2 算法介绍

本节详细阐述基于向量量化的优化器状态压缩算法。首先，构建低精度优化器的通用计算框架，明确量化与反量化算子在优化循环中的嵌入时机。其次，通过分析优化器状态的经验分布，揭示传统标量量化在极低比特场景下的局限性，确立极坐标向量量化的理论基础。随后，深入探讨码本设计原则，经理论推导证明角度均匀性的优势，并提出基于数据驱动的码本搜索算法。最后，通过静态实验验证了该算法的有效性。

4.2.1 低精度优化器

如第 4.1节所述，优化器状态的高昂显存成本给大规模模型训练带来了挑战[15,17,37]。为缓解这一问题，本章遵循现有的低比特优化范式[17,37]，采用低精度存储与高精度计算相结合的策略。具体而言，优化器状态（如 AdamW[12]的一阶和二阶动量）在静态存储时采用低精度格式，而在计算更新时临时反量化为高精度，从而显著降低优化器的静态显存占用，详细算法流程见算法 2.2。本节以

最常用的 AdamW[12]为例，具体展示该框架的实现细节。

低比特 AdamW 优化器的核心流程如下。在第 t 次迭代中，给定小批量梯度 $\boldsymbol{g}_t$，首先通过反量化函数恢复先前的高精度一阶和二阶动量：

$$\boldsymbol{m}_{t-1} = \texttt{dequantize}(\boldsymbol{m}_{t-1}^q), \quad \boldsymbol{v}_{t-1} = \texttt{dequantize}(\boldsymbol{v}_{t-1}^q), \tag{4-1}$$

其中 $\boldsymbol{m}_{t-1}^q$ 和 $\boldsymbol{v}_{t-1}^q$ 分别为低精度存储的动量状态。随后执行标准的动量更新：

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t, \quad \boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2. \tag{4-2}$$

遵循[15,37]，在基于 Transformer 的模型训练期间，仅量化线性层等二维参数对应的优化器状态，而嵌入层及一维参数（如偏置和归一化参数）的优化器状态保持全精度。值得注意的是，极低比特量化往往导致张量范数显著收缩。为此，本章在参数更新步骤引入了比例因子 α 予以补偿：

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t \left(\frac{\alpha \cdot \boldsymbol{m}_t/(1-\beta_1^t)}{\sqrt{\boldsymbol{v}_t/(1-\beta_2^t)} + \epsilon} + \lambda \boldsymbol{\theta}_{t-1} \right), \tag{4-3}$$

其中 η_t 为学习率，λ 为权重衰减系数。具体而言，对于量化层，α 设为特定值（如 2.0）；对于嵌入层等非量化层，则设 $\alpha = 1.0$ 以应用标准更新规则。该策略的必要性与有效性将在消融实验中验证。最后，本章将新产生的动量通过量化函数压缩为低精度版本存储：

$$\boldsymbol{m}_t^q = \texttt{quantize}(\boldsymbol{m}_t), \quad \boldsymbol{v}_t^q = \texttt{quantize}(\boldsymbol{v}_t), \tag{4-4}$$

以此大大降低大模型训练过程中的峰值显存。

针对另一广泛使用的优化器 Adafactor[33]，沿用相似的量化框架即可实现其低比特版本，仅需在计算流程中嵌入反量化与量化操作。与 AdamW 不同，Adafactor 通过分解二阶动量矩阵已显著降低显存开销。鉴于分解后的二阶动量累加器显存占用远小于一阶动量，保持它们为全精度即可，本章仅对其一阶动量状态实施量化。详细算法流程见附录B.3。

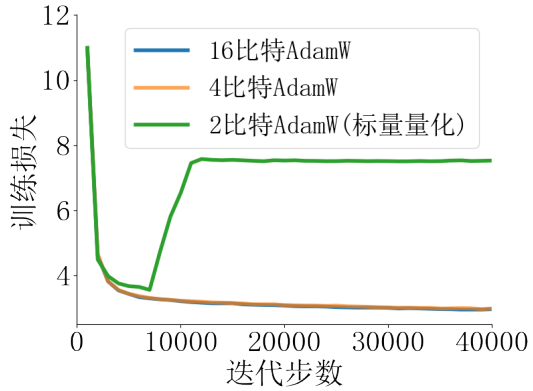
值得注意的是，优化器状态量化与传统的权重或激活量化存在本质区别。后者主要面向矩阵乘法运算，量化误差在累加求和过程中往往能相互抵消或得到抑制。然而，优化器更新规则（以 AdamW 为例）涉及敏感的逐元素除法操作（即 $\hat{\boldsymbol{m}}_t/(\sqrt{\hat{\boldsymbol{v}}_t}+\epsilon)$），其中二阶动量位于分母位置。这一特性对量化精度提出了极高要求：微小的量化误差若导致分母异常，极易引发数值不稳定甚至训练发散。因此，该框架的成功关键在于 `quantize` 与 `dequantize` 的具体实现必须保证极高的重建精度。下文将重点讨论如何设计高精度的量化映射以满足这一严苛要求。

4.2.2 从标量量化到向量量化

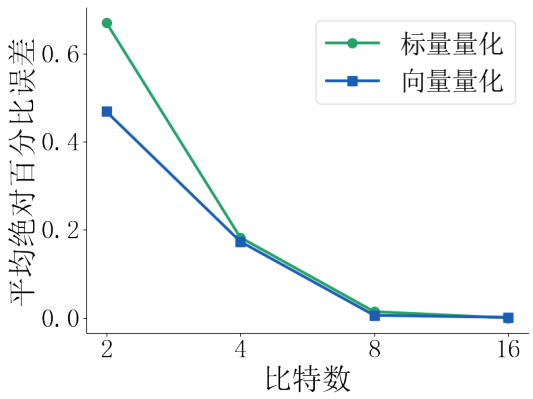
为深入理解标量量化在极低比特场景下的局限性，并阐明向量量化的优势，本节依次从训练现象、量化误差、分布特性三个层面展开分析。

训练稳定性问题。本节在 GPT-2 (124M) 模型的 OpenWebText 预训练任务上评估了现有基于标量量化的低比特优化器性能。如图 4.1(a) 所示，当采用 2 比特量化时，AdamW 和 Adafactor 优化器在训练初期即出现损失发散，最终训练崩溃。这一现象表明，标量量化的矩形网格结构难以适配优化器状态的非均匀分布：在 2 比特的极端压缩比下，仅有 4 个离散值可用，无法有效表示复杂的动量状态。

量化误差对比。为量化评估不同方法的表征能力，本节在 LLaMA-130M 模型的真实训练数据上测量了标量量化（SQ）与向量量化（VQ）的重建误差。如图 4.1(b) 所示，在不同比特宽度下，二维向量量化的平均百分比误差（MAPE）均低于标量量化。特别是在极低比特（低于 4 比特）设置下，VQ 相较于 SQ 具有显著的重建精度优势。



(a) 不同低比特优化器在 GPT-2 预训练任务中的损失曲线对比



(b) 不同比特宽度下 SQ 与 VQ 的误差对比

图 4.1　标量量化的局限性与向量量化的优势

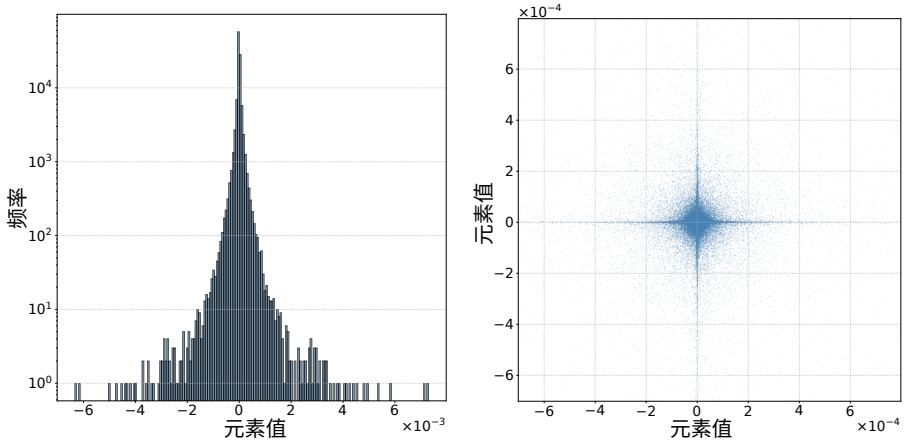
上述实验结果表明，向量量化在表征能力上优于标量量化，但尚未解释其根本原因。下文从优化器状态的分布特性出发，揭示 SQ 失效的内在机制。

优化器状态分布特性分析。为确定最佳的向量维度及量化几何结构，本节对 LLaMA 预训练过程中的优化器状态进行经验分布分析。

首先，从训练过程中的 LLaMA 模型中提取 AdamW 一阶动量张量，选取具有代表性的注意力层权重 `q_proj` 作为分析对象。其次，保留原始数据的量级与符号信息以反映真实训练动态。最后，将序列中相邻元素两两配对，形成坐标点集合 $\{(m_{2i}, m_{2i+1})\}$，以分析元素间的相关性。

基于上述处理流程，本节绘制了一维频率直方图与二维联合散点图，如图 4.2 所示。观察发现：

- 一维边缘分布呈现单峰形态，大部分数值集中在均值附近，表现出明显的**准高斯分布特性**；
- 二维联合分布呈现出以中心为原点的近似**圆对称形态**，散点云在角度方向上分布均匀，未表现出明显的椭圆拉伸。



(a) 一维频率直方图　　(b) 二维联合散点图

图 4.2　LLaMA 预训练过程中 AdamW 一阶动量的分布特性分析

上述特性揭示了传统标量量化（$k = 1$）的局限性：其矩形网格划分假设各维度独立且分布均匀，与圆对称分布不匹配，导致角落处存在大量冗余表示区域。相比之下，极坐标系统天然契合圆对称分布。因此，本章选择 $k = 2$ 并采用二维极坐标向量量化技术，以匹配优化器状态的内在几何结构。

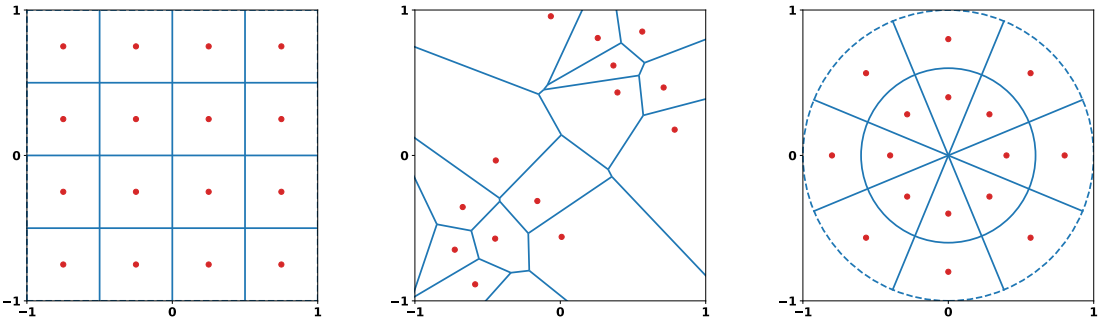
向量量化的几何优势。向量量化通过联合编码多个维度的参数，能更好地捕捉数据的内在几何结构，理论上可提供更高的压缩上限[25-26]。针对优化器状态的圆对称分布特性，极坐标表示具有天然优势。

如图 4.3 所示（红点表示码本向量，蓝线表示决策边界），三种量化映射方式具有本质不同的几何结构：(a) 一维标量量化的矩形网格忽略数据的圆对称性，导致角落冗余；(b) 通用二维向量量化（如 K-means 聚类）虽能通过不规则聚类适应分布，但计算开销巨大且难以控制码本零点；(c) 极坐标量化采用同心圆结构，兼具几何一致性与计算效率。

极坐标量化的核心优势包括：

- 径向-角度解耦：半径对应幅值信息，角度对应方向信息，分别适配一阶动量（有符号）和二阶动量（无符号）的不同特性；
- 结构化效率：极坐标码本具有规则的几何结构，避免 K-means 等方法的昂贵搜索开销；
- 数值稳定性：通过角度偏移可严格避开坐标轴，防止二阶动量除零风险。

综上所述，标量量化在极低比特下失效的根本原因在于其矩形网格与优化器状态的圆对称分布不匹配；而二维极坐标向量量化通过几何结构上的天然适



(a) 一维标量映射　　(b) 通用二维聚类　　(c) 极坐标结构化

图 4.3　二维向量量化映射的可视化对比

配，为极低比特优化器设计提供了可行路径。

4.2.3 二维向量量化框架

本节首先建立通用的 k 维量化理论框架，然后具体实例化为二维极坐标量化技术。

通用量化框架。给定实矩阵 $\boldsymbol{X} \in \mathbb{R}^{k\times N}$（其中 $N = n/k$），本节将 $\boldsymbol{X}$ 的列视为 N 个 k 维向量。为了形式化描述量化与反量化过程，本节定义以下四个核心算子用于构建 `quantize` 和 `dequantize` 函数：

- 缩放算子 $\mathcal{M}$：计算块级缩放因子。将向量划分为不相交的块，对于每个块，计算其中所有向量的最大 2-范数。设 $\mathcal{M}(\boldsymbol{X}) \in \mathbb{R}^N$ 为缩放因子向量，其第 i 个元素 $\mathcal{M}(\boldsymbol{X})_i$ 对应包含 $\boldsymbol{x}_i$ 的块的最大范数。
- 归一化算子 $\mathcal{N}$：基于缩放因子进行归一化。定义 $\mathcal{N}(\boldsymbol{X})_i = \boldsymbol{x}_i / \mathcal{M}(\boldsymbol{X})_i$，将原始向量映射到单位超球体内。
- 码本映射 $\mathcal{R}$：定义单射映射 $\mathcal{R} : \mathbb{T}_{kb} \to \mathbb{R}^k$，其像构成码本，集合中的每个向量称为码字。其中 $\mathbb{T}_{kb} = \{0, 1, \cdots, 2^{kb} - 1\}$ 为索引空间。
- 索引映射 $\mathcal{I}$：定义 $\mathcal{I} : \mathbb{R}^k \to \mathbb{T}_{kb}$，为任意输入向量找到码本中最近的码字并返回其索引。

基于上述算子，本节构建 k 维 b-比特量化函数 `quantize` 与其对应的反量化函数 `dequantize`。量化过程输出索引与缩放因子，表示为：

$$\texttt{quantize}(\boldsymbol{X}) = (\mathcal{I} \circ \mathcal{N}(\boldsymbol{X}), \mathcal{M}(\boldsymbol{X})) : \mathbb{R}^{k\times N} \to \mathbb{T}_{kb}^N \times \mathbb{R}^N. \tag{4-5}$$

反量化过程则通过查表与重建恢复近似值，定义为：

$$\texttt{dequantize}(\texttt{quantize}(\boldsymbol{X})) = \mathcal{R}(\mathcal{I} \circ \mathcal{N}(\boldsymbol{X})) \odot \mathcal{M}(\boldsymbol{X}) : \mathbb{T}_{kb}^N \times \mathbb{R}^N \to \mathbb{R}^{k\times N}, \tag{4-6}$$

其中 $\odot$ 表示列向缩放。该框架通过向量维度参数 k 统一了不同的量化技术。标准一维标量量化[17,37] 对应于 $k = 1$ 的特殊情况，此时每个“向量”仅为标量，归

一化简化为除以块的最大绝对值。虽然在较高精度下有效，但这种标量量化技术在极低比特场景（如低于 4 比特）中因无法利用维度间相关性而引入显著误差。

二维极坐标量化流程。基于上述理论框架，本节具体介绍 $k = 2$ 的二维极坐标向量量化实现。如图 4.4 所示，该过程包含量化与反量化两个阶段：在量化侧，首先将张量重塑为二维向量序列并划分为不相交的块；随后对每个块计算最大 2-范数 $s = \max(\|\boldsymbol{v}\|_2)$ 作为缩放因子（对应算子 $\mathcal{M}$），并将块内向量除以 s 映射到单位圆内（对应算子 $\mathcal{N}$）；接着在预定义码本中搜索最近邻码字（对应算子 $\boldsymbol{\mathcal{I}}$），仅存储码字索引与缩放因子 s。在反量化侧，根据索引查表取出码字 $\boldsymbol{c}$（对应 $\mathcal{R}$），并计算 $\hat{\boldsymbol{v}} = s \cdot \boldsymbol{c}$ 以恢复近似的高精度状态。

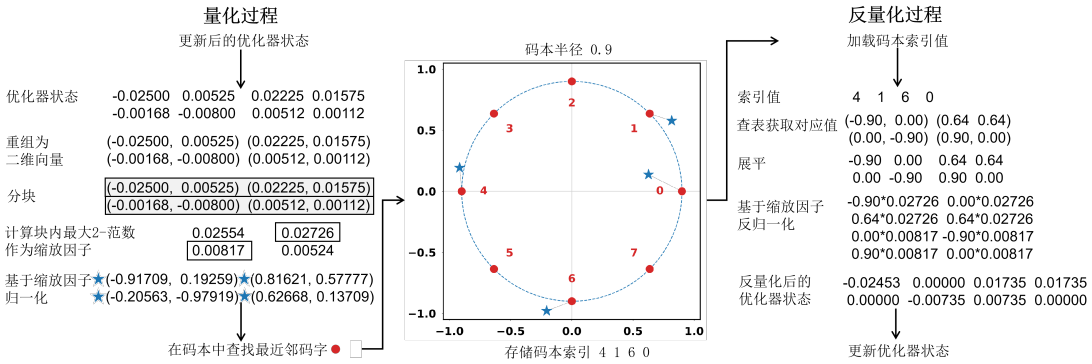


图 4.4　优化器状态的极坐标向量量化和反量化示意图

4.2.4 数据驱动的极坐标码本设计

码本的设计直接决定了量化的精度。在本章的二维极坐标量化技术中，码本由一组位于极坐标系下的点组成。为了确定这些点的具体位置（半径与角度），本节首先建立理论误差界，以此指导搜索策略，随后基于数据驱动框架搜索最优配置。

理论误差界与角度均匀性。本节首先形式化球对称采样对量化误差的控制能力。以下引理表明，当缩放因子接近 1 时，角度分布越均匀，量化误差越小。

引理 4.1： 设 $\boldsymbol{x} \in \mathbb{R}^2, Y \subseteq \mathbb{R}^2$ 且 $s > 0$。若 $\forall \boldsymbol{y} \in Y$，$\|\boldsymbol{x}\|_2 = s\|\boldsymbol{y}\|_2 > 0$ 且 $\boldsymbol{x}$ 与 $\boldsymbol{y}$ 的夹角不超过 $\phi \leqslant \frac{\pi}{2}$，则有

$$\|\boldsymbol{x} - \boldsymbol{y}\|_2 \leqslant \frac{2\sin(\phi/2) + |s-1|}{s}\|\boldsymbol{x}\|_2.$$

证明 如图 4.5 所示，设 O 为圆心，$\triangle ABC$ 为圆的内接三角形，且线段 AB 经过点 O，记 $\phi = \angle COB$。不失一般性，令 $\boldsymbol{y} = \overrightarrow{OB} \in Y$，$\boldsymbol{x} = s\overrightarrow{OC}$。基于圆内接三角形的几何性质，可得

$$\|\overrightarrow{BC}\|_2 = 2\|\boldsymbol{y}\|_2 \sin(\phi/2).$$

由于 2-范数满足三角不等式。因此，

$$\begin{aligned}\|\boldsymbol{x}-\boldsymbol{y}\|_2 &= \|\overrightarrow{OC}-\overrightarrow{OB}+(s-1)\overrightarrow{OC}\|_2 \\ &\leqslant \|\overrightarrow{BC}\|_2+\|(s-1)\overrightarrow{OC}\|_2 \\ &= \frac{2\sin(\phi/2)+|s-1|}{s}\|\boldsymbol{x}\|_2.\end{aligned}$$

∎

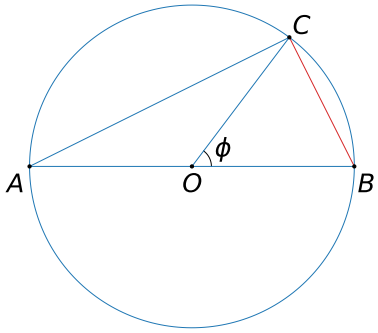


图 4.5　引理证明几何示意图

上述引理表明，当 $s \approx 1$ 时，相对量化误差主要由角度差 ϕ 决定。在本节设置中，若角度均匀分布，则最大夹角 ϕ 最小化。因此，角度均匀性成为本节设计量化映射的核心指导原则。基于此，本节固定了码字的角度分布模式（例如有符号输入采用 8 个均匀角度 $\Theta=\{j\pi/4\}$），从而将复杂的码本搜索问题简化为搜索半径集合及其码字分配的组合优化问题。

码本搜索算法框架。在固定角度分布模式后，本节使用真实训练过程中的优化器状态数据作为搜索集，通过最小化特定目标函数来寻找最优半径组合及码字分配策略。搜索过程需满足严格的比特宽度约束：对于 k 维 b 比特量化，码本总大小必须为 $|C|=2^{kb}$。例如，二维 2 比特量化对应 16 个码字，二维 1.5 比特量化对应 8 个码字。针对一阶动量和二阶动量的不同分布特性，本节分别设计搜索算法。

对于**一阶动量（有符号输入）**，码本覆盖整个平面，角度均匀分布于 $[0,2\pi)$。搜索变量为半径集合 $R=\{r_1,r_2,\cdots,r_{|R|}\}$ 及每个半径对应的码字数量 $K=\{k_1,k_2,\cdots,k_{|R|}\}$，需满足总码字数约束 $\sum_{j=1}^{|R|}k_j\cdot|R|=2^{kb}$。其目标是最小化逐元素重建误差。给定一批量化的动量样本 $\{\boldsymbol{m}^{(i)}\}$，搜索半径集合 R 使得下式最小化：

$$\mathcal{L}_1(R)=\sum_i\|\boldsymbol{m}^{(i)}-\texttt{dequantize}(\texttt{quantize}(\boldsymbol{m}^{(i)};R))\|_2^2. \tag{4-7}$$

搜索算法如算法 4.1 所示。

对于**二阶动量（无符号输入）**，码本仅覆盖第一象限，且需避免码字靠近坐标轴以防止除零风险[17]。搜索变量包括半径集合 R、每个半径对应的码字数量 K

算法 4.1 一阶动量码本搜索（有符号数据）

输入： 动量样本集 $\{\boldsymbol{m}^{(i)}\}_{i=1}^{N}$, 比特宽度 b, 向量维度 $k=2$
输出： 最优码本 $\mathcal{C}_{\text{best}}$

计算目标码本大小 $|\mathcal{C}| \leftarrow 2^{kb}$　// 如 2-bit→16 码字，1.5-bit→8 码字
固定基础角度分布 $\Theta_{\text{base}} = \{j\pi/4 \mid j = 0, \cdots, 7\}$
初始化最优误差 $e_{\text{best}} \leftarrow +\infty$，最优码本 $\mathcal{C}_{\text{best}} \leftarrow \varnothing$
for 迭代次数 $t = 1, \cdots, T$ **do**
　随机采样： 随机生成半径数量 $|R| \in \{1, 2, 3, 4\}$
　随机采样： 生成半径集合 $R = \{r_1, \cdots, r_{|R|}\}$，其中 $r_j \sim \text{Uniform}(0.1, 1.0)$
　约束投影： 排序并去重 R，确保 $0 < r_1 < r_2 < \cdots < r_{|R|} \leqslant 1$
　码字分配： 计算每个半径的码字数 $k_j = |\mathcal{C}|/(|R| \cdot 8)$，确保 $\sum_j k_j \cdot 8 = |\mathcal{C}|$
　构建码本 $\mathcal{C} = \{(r_j \cos\theta, r_j \sin\theta) \mid r_j \in R, \theta \in \Theta_{\text{base}}\}$
　计算重建误差 $e_t = \frac{1}{N}\sum_i \|\boldsymbol{m}^{(i)} - \texttt{dequantize}(\texttt{quantize}(\boldsymbol{m}^{(i)}; \mathcal{C}))\|_2^2$
　if $e_t < e_{\text{best}}$ **then**
　　更新最优解：$e_{\text{best}} \leftarrow e_t$, $\mathcal{C}_{\text{best}} \leftarrow \mathcal{C}$
　end if
end for

及角度偏移量 δ。总码字数同样需满足 $|\mathcal{C}| = 2^{kb}$ 约束。由于二阶动量最终用于计算更新量 $\boldsymbol{m}/\sqrt{\boldsymbol{v}}$，本节直接以更新量的误差为优化目标。给定样本对 $\{(\boldsymbol{m}^{(i)}, \boldsymbol{v}^{(i)})\}$，搜索半径集合 R 及码字分配 K 使得下式最小化：

$$\mathcal{L}_2(R, K) = \sum_i \left\| \frac{\boldsymbol{m}^{(i)}}{\sqrt{\boldsymbol{v}^{(i)}}} - \frac{\boldsymbol{m}^{(i)}}{\sqrt{\texttt{dequantize}(\texttt{quantize}(\boldsymbol{v}^{(i)}; R, K))}} \right\|_2^2. \tag{4-8}$$

此外，为避免除零风险，二阶动量搜索需引入角度偏移量 δ，确保所有角度位于 $(\delta, \pi/2 - \delta)$ 范围内。搜索算法如算法 4.2 所示。

算法 4.2 二阶动量码本搜索（无符号数据）

输入： 动量样本对 $\{(\boldsymbol{m}^{(i)}, \boldsymbol{v}^{(i)})\}_{i=1}^{N}$, 比特宽度 b, 向量维度 $k=2$
输出： 最优码本 $\mathcal{C}_{\text{best}}$

计算目标码本大小 $|\mathcal{C}| \leftarrow 2^{kb}$　// 如 2-bit→16 码字，1.5-bit→8 码字
初始化最优误差 $e_{\text{best}} \leftarrow +\infty$，最优码本 $\mathcal{C}_{\text{best}} \leftarrow \varnothing$
for 迭代次数 $t = 1, \cdots, T$ **do**
　随机采样： 随机生成半径数量 $|R| \in \{2, 3, 4\}$
　随机采样： 生成半径集合 $R = \{r_1, \cdots, r_{|R|}\}$，其中 $r_j \sim \text{Uniform}(0.1, 1.0)$
　约束投影： 排序并确保 $0 < r_1 < r_2 < \cdots < r_{|R|} \leqslant 1$
　随机采样： 生成角度偏移量 $\delta \sim \text{Uniform}(0.05, 0.2)$　// 弧度，避免靠近坐标轴
　码字分配： 为每个半径分配码字数 k_j，满足 $\sum_{j=1}^{|R|} k_j = |\mathcal{C}|$ 且 $k_j \geqslant 2$
　构建角度： 对第 j 个半径，生成角度 $\theta_{j,l} = \delta + \frac{l}{k_j+1}(\frac{\pi}{2} - 2\delta)$，$l = 1, \cdots, k_j$
　构建码本 $\mathcal{C} = \{(r_j \cos\theta_{j,l}, r_j \sin\theta_{j,l}) \mid r_j \in R, l = 1, \cdots, k_j\}$
　计算步长重建误差 $e_t = \frac{1}{N}\sum_i \left\| \frac{\boldsymbol{m}^{(i)}}{\sqrt{\boldsymbol{v}^{(i)}}} - \frac{\boldsymbol{m}^{(i)}}{\sqrt{\texttt{dequantize}(\texttt{quantize}(\boldsymbol{v}^{(i)}; \mathcal{C}))}} \right\|_2^2$
　if $e_t < e_{\text{best}}$ **then**
　　更新最优解：$e_{\text{best}} \leftarrow e_t$, $\mathcal{C}_{\text{best}} \leftarrow \mathcal{C}$
　end if
end for

为了定量评估量化重建质量，本节引入以下静态误差指标。给定输入矩阵 $\boldsymbol{X} \in \mathbb{R}^{k\times\frac{n}{k}}$，令 $\boldsymbol{Y} = \texttt{dequantize}(\texttt{quantize}(\boldsymbol{X}))$ 为其量化重建值。为了测量 $\boldsymbol{X}$ 与 $\boldsymbol{Y}$ 之间的差异，本节使用重塑函数 $h : \mathbb{R}^{k\times\frac{n}{k}} \to \mathbb{R}^{m\times\frac{n}{m}}$ 对矩阵元素进行重新分块。具体而言，h 首先将输入矩阵按列向量化，随后将向量划分为长度为 m 的连续段，最后重塑为 $\mathbb{R}^{m\times\frac{n}{m}}$ 中的矩阵。基于此，本节定义 m 维范数相对误差（m-NRE）和角度误差（AE）如下：

$$m\text{-NRE} = \text{mean}_i\left(\frac{\|h(\boldsymbol{X})_i - h(\boldsymbol{Y})_i\|_2}{\|h(\boldsymbol{X})_i\|_2 + \varepsilon}\right),\ \text{AE} = \arccos\left(\frac{\langle \boldsymbol{X}, \boldsymbol{Y}\rangle}{\|\boldsymbol{X}\|_F\|\boldsymbol{Y}\|_F}\right),$$

其中 $\text{mean}_i(\cdot)$ 表示对所有分块 $i = 1, \cdots, n/m$ 取平均值，ε 为防止除零的小正数。m-NRE 衡量幅值重建的相对精度，而 AE 反映方向保留程度。

本节将提出的二维极坐标向量量化技术与三种主流的一维标量量化映射进行对比：线性幂量化[17,99]、动态量化[37,100] 以及基于分位数的 NormalFloat 量化 (NF)[8]。鉴于 NF 主要针对权重量化设计，本节的静态分析主要聚焦于用于优化器状态压缩的线性和动态映射。各类映射的具体规格与可视化详见附录 B.4。

表 4.1 展示了 LLaMA-130M 模型第 8 层 `q_proj` 权重对应的 AdamW 优化器状态静态量化误差对比结果，对于每一列，最佳结果以**粗体**显示，次佳结果以下划线 显示。实验表明，本节提出的二维向量量化方法在所有指标上均优于基线方法，始终实现最低的相对误差和最高的余弦相似度。特别是在二阶动量量化上，本章的 2 比特方法将角度误差从基线的约 32 度降低至 16.6 度，优势显著。

表 4.1　LLaMA-130M 模型训练过程中 AdamW 优化器状态静态量化误差对比

	一阶动量			二阶动量		
量化方法	NRE-1 ↓	NRE-2 ↓	AE(°)↓	NRE-1 ↓	NRE-2 ↓	AE(°)↓
线性幂量化 2-bit	0.513	0.947	26.747	0.800	0.007	32.965
动态量化 2-bit	0.435	0.579	24.632	0.782	0.007	32.860
Polaris 2-bit	**0.394**	**0.421**	**22.631**	**0.295**	**0.002**	**16.665**
Polaris 1.5-bit	0.528	0.807	31.788	0.379	0.003	21.875

相较于传统一维标量量化方案，本章技术能够更准确地重建优化器状态的原始几何结构，这对维持极低比特下的训练稳定性至关重要。这种基于极坐标的码本设计不仅在理论上最小化了相对量化误差，更高度契合了优化器状态的非均匀分布特性。具体而言，针对一阶动量，该设计利用角度均匀性确保了梯度更新方向的准确性；针对作为分母的二阶动量，则通过角度偏移策略严格规避零点，消除了潜在的除零风险。

4.3 实验结果与分析

本节通过系统实验全面评估优化器状态压缩算法 Polaris 的有效性。首先介绍实验设置，涵盖数据集、模型架构、对比基线及实现细节。随后汇报主实验结果，对比 Polaris 与现有低比特优化算法的性能表现。最后通过消融实验验证关键设计组件的贡献。

4.3.1 实验设置

模型与数据集。预训练任务中，本节遵循 Wang 等人[99] 的设置，在 OpenWebText[101] 上训练 GPT-2 (124M) 共 40k 步；并参照 Zhao 等人[15] 的设置，在 C4[102] 上训练 LLaMA-2 (130M, 350M) 共 80k 步。微调任务采用 LLaMA-2-7B 与 Qwen2.5-7B 在 Alpaca[103] 数据集上进行指令微调，并在 GLUE[104] 基准上评估。视觉任务方面，基于 Zhou 等人[105] 的开源框架，在 ImageNet-1K[106] 上分别训练 ViT-B/16[107] 与 ResNet-50[108]，轮数分别为 150 轮与 100 轮。

对比基线。为全面验证 Polaris 的压缩潜力，本节将其与 16/32 比特全精度优化器、4 比特量化基线[17] 及其 2 比特扩展版本进行对比。优化器选取具有代表性的 AdamW[12] 与 Adafactor[33]。

实现细节。所有实验均在单张 NVIDIA A800 GPU 上完成。为严格控制变量，模型权重、梯度、激活值及学习率调度等超参数均与公开基线保持一致，仅替换优化器状态量化模块。量化设置方面，优化器状态张量 $\boldsymbol{X}$ 按块大小 64 进行分组，并输入至本章提出的极低比特量化器。为兼顾极致压缩与数值稳定性，缩放因子采用块大小 256 进行 8 比特动态量化（即双重量化策略[8]）。超参数配置方面，Adafactor 的学习率缩放因子设为 $\alpha = 2.0$；AdamW 在 2 比特与 1.5 比特设置下分别设为 $\alpha = 2.0$ 与 $\alpha = 2.5$。一阶动量衰减率统一设为 $\beta_1 = 0.9$，AdamW 的二阶动量衰减率设为 $\beta_2 = 0.95$。

4.3.2 主要结果

实验分别在 C4 数据集上训练 LLaMA-130M 与 350M 模型，以及在 OpenWebText 数据集上训练 GPT2-124M 模型。表 4.2 汇总了验证困惑度 (Validation Perplexity, VPPL)、挂钟时间 (Wall-Clock Time, WCT，单位：小时) 及优化器状态显存占用 (Memory Cost, MC，单位：MB) 的对比结果，其中 *Crash* 表示训练失败。结果表明，将 4 比特基线的一维标量量化技术直接扩展至 2 比特会导致训练崩溃。相比之下，本章提出的优化器状态压缩算法不仅确保了训练稳定性，且在效率与性能上展现出显著优势。具体而言，该算法支持低至 2.0 甚至 1.5 比特的稳定量化；与 16 比特基线相比，优化器显存占用最高降低 4.6×，同时保持了具有竞争力的挂钟时间。图 4.6 展示了在 C4 和 OpenWebText 数据集上的验证困惑

表 4.2　C4 与 OpenWebText 预训练任务结果对比

优化器	LLaMA-130M			LLaMA-350M			GPT2-124M		
	VPPL	WCT	MC	VPPL	WCT	MC	VPPL	WCT	MC
32-bit Adafactor	20.393	22.98	516.87	17.708	59.57	1429.64	19.556	31.34	476.45
4.0-bit Adafactor	20.497	22.99	228.42	17.481	59.70	411.77	19.658	31.30	200.23
2.0-bit Adafactor	*Crash*	-	-	*Crash*	-	-	*Crash*	-	-
2.0-bit Adafactor(Polaris)	20.243	23.04	208.80	16.790	59.98	330.66	19.890	31.37	173.75
1.5-bit Adafactor(Polaris)	20.273	23.14	203.74	16.712	59.50	312.63	20.164	31.32	168.69
16-bit AdamW	20.350	22.33	518.70	16.866	58.36	1420.72	19.645	31.30	951.90
4.0-bit AdamW	20.680	22.49	269.30	17.123	59.04	582.98	19.842	31.29	391.46
2.0-bit AdamW	*Crash*	-	-	*Crash*	-	-	*Crash*	-	-
2.0-bit AdamW(Polaris)	20.480	22.63	230.06	16.917	59.09	403.66	19.979	31.34	341.50
1.5-bit AdamW(Polaris)	20.904	22.59	219.94	17.157	59.14	367.62	20.665	31.39	331.38

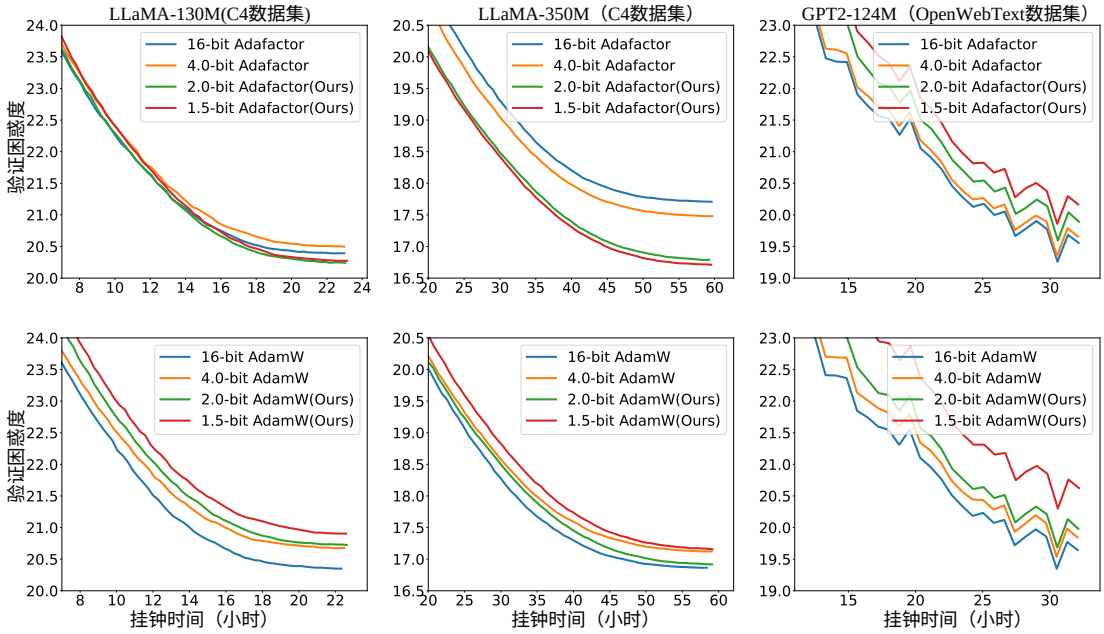


图 4.6　C4 和 OpenWebText 数据集上的验证困惑度曲线

度曲线，进一步印证了本算法的优越性。

值得注意的是，本算法在部分设置中表现出意外优势。例如，在使用 AdamW 进行 LLaMA-350M 预训练时，2 比特优化器不仅优于标准 4 比特基线，更缩小了与 16 比特全精度版本性能差距的 80% 以上。更为显著的是，在 LLaMA-130M 与 350M 的 Adafactor 预训练任务中，低比特优化器在训练后期的验证困惑度甚至超越了高精度基线。本文推测这一现象源于极低比特量化的固有正则化效应：引入的噪声有助于优化器逃离尖锐局部极小值，收敛至更平坦、泛化能力更强的极小值[47]。

在 ImageNet-1K 图像分类任务上，本节基于 ViT-Base/16 和 ResNet-50 模型进行评估。表 4.3 汇总了测试准确率 (Test Accuracy, TA，单位：%)、挂钟时间 (Wall-Clock Time, WCT，单位：小时) 及优化器状态 GPU 显存占用 (Memory Cost,

表 4.3　ImageNet-1K 分类任务结果对比

	ViT-Base/16			ResNet-50		
优化器	TA	WCT	MC	TA	WCT	MC
32-bit SGDM	-	-	-	75.41	29.94	94.21
32-bit Adafactor	80.72	58.80	326.11	77.60	31.99	214.71
4.0-bit Adafactor	80.57	59.19	59.09	76.85	32.37	135.66
2.0-bit Adafactor	*Crash*	-	-	*Crash*	-	-
2.0-bit Adafactor(Polaris)	79.53	59.35	26.25	76.72	32.39	125.89
32-bit AdamW	80.72	56.47	654.38	77.68	30.06	192.87
4.0-bit AdamW	79.28	56.85	103.39	75.65	31.24	24.91
2.0-bit AdamW	*Crash*	-	-	*Crash*	-	-
2.0-bit AdamW(Polaris)	79.68	57.32	49.39	76.30	31.21	10.93

MC，单位：MB) 的对比结果，其中 *Crash* 表示训练失败。需要说明的是，ViT 架构未采用 SGDM 优化器，因为广泛研究表明 SGDM 难以有效优化 ViT，常导致训练发散或收敛性能显著劣化[47,105]，因此 ViT 实验均以 AdamW 为基准。实验结果与大语言模型上的结论一致：直接扩展至 2 比特量化会导致训练失败，而本章算法确保了稳定训练。值得注意的是，本章算法带来了显著的效率提升。特别是在 ViT-Base/16 任务中，尽管 ViT 必须依赖显存开销较大的 AdamW 优化器，本章算法仍将其状态占用从 654.4 MB 大幅降至 49.4 MB，压缩比超过 13×，且 Top-1 准确率仅损失约 1%。更为重要的是，本章算法进一步将 AdamW 的显存占用压缩至远低于传统轻量级 SGDM 优化器的水平。在 ResNet-50 任务中，本章算法仅需 10.93 MB 显存，而 32-bit SGDM 需 94.21 MB。本章算法在确保复杂模型训练稳定性的同时，实现了极致的显存效率。

表 4.4 汇总了在 Alpaca 数据集上微调后，LLaMA2-7B 和 Qwen2.5-7B 在 GLUE 基准上的性能表现。表中记录了平均分数 (Average Score, AVG) 与总显存成本 (Total Memory Cost, TMC，单位：MB)，所有任务性能指标单位均为百分比 (%)。实验结果进一步验证了本章提出的低比特 Adafactor 优化器的有效性与可扩展性。图 4.7 展示了在 Alpaca 数据集上微调大语言模型的训练损失曲线。本章低比特算法的收敛轨迹与全精度基线紧密吻合，表明性能下降可忽略不计。

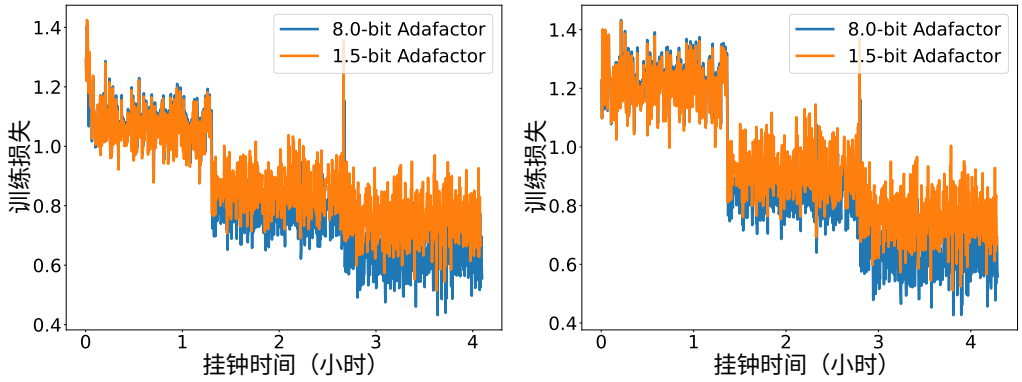
4.3.3 消融实验

为验证本章关键设计的有效性，本节围绕三个方面展开消融研究。第一，对比不同向量量化策略，验证极坐标码本设计相比传统 K-means 聚类的优势。第二，分析缩放因子 α 的敏感性，确定其最优取值范围。第三，探究分块大小的影响，权衡计算效率与重建质量。

向量量化策略对比。现有向量量化技术（如 K-means 聚类[39,60]）在优化器

表 4.4　Alpaca 微调后 GLUE 基准上的性能对比

模型	优化器	SST-2	RTE	COLA	MNLI	MRPC	AVG	TMC
LLaMA2-7B	原始	83.0	70.8	29.2	36.5	66.9	57.3	-
	8.0-bit Adafactor	91.5	74.7	35.1	53.9	65.0	64.0	54 220
	1.5-bit Adafactor(Polaris)	91.3	74.0	35.7	50.0	68.1	63.8	49 188
Qwen2.5-7B	原始	94.8	83.0	46.0	77.3	75.3	75.3	-
	8.0-bit Adafactor	94.6	80.1	46.6	70.4	76.5	73.6	62 969
	1.5-bit Adafactor(Polaris)	95.4	80.9	47.6	71.4	75.7	74.5	57 929



(a) LLaMA2-7B　　(b) Qwen2.5-7B

图 4.7　在 Alpaca 数据集上微调大语言模型的训练损失曲线

状态量化场景面临双重挑战：其一，在线 K-means 涉及昂贵的迭代计算，每次更新需遍历全部状态进行多轮聚类，显著增加训练延迟；其二，K-means 以最小化均方误差为目标，无法适配优化器对数值稳定性的特殊需求——一阶动量决定更新方向，需保证方向精度；二阶动量位于分母，需严格避开零点。

本节在 GPT-2-124M 模型的 OpenWebText 预训练任务上，对比三种向量量化策略：K-means 采用标准聚类初始化码本且训练过程在线更新码本；K-means-EMA 在初始化后以指数移动平均方式在线更新码本；本章提出的数据驱动极坐标码本搜索算法得到的码本在训练过程固定。实验固定使用 2.0 比特 AdamW 优化器，对一阶动量和二阶动量采用上述策略的交叉组合，共 9 组实验。每组运行至收敛或出现训练崩溃（*Crash* 表示训练崩溃），记录验证困惑度。

如表 4.5 所示，9 组实验中仅 2 组成功收敛，且二阶动量均采用本章算法：一阶动量采用 K-means-EMA 时验证困惑度为 20.401；一阶动量采用本章算法时降至 19.979。其余 7 组均出现训练崩溃。这一分布揭示关键洞察：二阶动量的量化策略是训练稳定性的决定性因素——由于二阶动量位于更新公式分母，量化误差易导致除零或数值爆炸，而本章的极坐标码本通过零点规避设计确保了数值稳定性。

值得注意的是，在二阶动量采用本章算法的前提下，一阶动量使用在线更新的 K-means 仍会崩溃，而 K-means-EMA 可收敛。这一现象表明，单纯的在线更

表 4.5　2.0 比特 AdamW 动量不同向量量化策略的验证困惑度对比

一阶动量 \ 二阶动量	K-means	K-means-EMA	本章方法 Polaris
K-means	*Crash*	*Crash*	*Crash*
K-means-EMA	*Crash*	*Crash*	20.401
本章方法 Polaris	*Crash*	*Crash*	**19.979**

新并不足以保证稳定性，关键在于码本中心更新的平滑性。K-means 每次迭代直接将样本均值赋值为新中心，中心位置随当前批次剧烈跳变，导致更新方向在相邻步间缺乏连续性；而 K-means-EMA 通过指数滑动平均融合历史信息，中心演化轨迹更为平缓，确保了码本在方向精度上的稳定。本章的极坐标码本进一步通过均匀角度分配优化方向表示，使困惑度再降低 0.422，验证了角度精度对收敛质量的提升效果。

缩放因子 α 的敏感性分析。极低比特量化导致张量范数显著收缩，需引入缩放因子 α 补偿更新幅度。然而，α 取值过大可能引发梯度爆炸，过小则导致更新不足。本节在 C4 数据集上训练 LLaMA-130M 模型，在 OpenWebText 数据集上训练 GPT-2-124M 模型，对 1.5 比特 Adafactor 和 AdamW 分别测试不同 α 取值。所有实验固定其他超参数，仅调整 α，记录最终验证困惑度。

表 4.6　1.5 比特量化下不同优化器与缩放因子 α 的验证困惑度对比

优化器	α	LLaMA-130M	GPT-2-124M
Adafactor	1.0	20.543	20.918
	2.0	**20.273**	20.164
	3.0	20.299	**19.890**
AdamW	1.0	21.016	21.291
	2.0	21.072	20.648
	2.5	**20.904**	20.665
	3.0	21.036	**20.624**

如表 4.6 所示，Adafactor 对 α 较为敏感，在 LLaMA-130M 上 $\alpha = 2.0$ 最优，GPT-2-124M 上 $\alpha = 3.0$ 最优。AdamW 在 LLaMA-130M 上 $\alpha = 2.5$ 最优，且在 2.0 至 3.0 范围内性能波动较小。该结果表明，α 需根据模型规模与数据特性调整，但本章默认取值在多数场景下具有鲁棒性。

分块大小的影响。分块大小决定了缩放因子的作用粒度。较小的分块能有效隔离异常值，提供更精细的逐块归一化，但会引入更多的缩放因子存储开销。较大的分块虽能压缩存储，但归一化范围易受极端值主导，从而导致重建精度损失[37]。本节在 C4 数据集上训练 LLaMA-130M 模型，在 OpenWebText 数据集上训练 GPT-2-124M 模型，系统对比了 64、128 与 256 三种分块尺寸的效果。实验中 LLaMA-130M 与 GPT-2-124M 的缩放因子分别固定为 $\alpha = 2.5$ 与 $\alpha = 2.0$，均采用

1.5 比特 AdamW 优化器并控制其他超参数一致，最终记录验证困惑度（VPPL）与优化器状态显存占用（MC，单位：MB）。

如表 4.7 所示，分块大小为 64 时 VPPL 最优；增大至 256 时性能出现显著退化。在显存方面，得益于本章采用的双重量化策略，缩放因子已被进一步压缩至 8 比特，因此不同分块配置下的 MC 差异极小（波动范围不足 2 MB）。尽管分块越细意味着需存储更多的 8 比特缩放因子实例，带来微弱的开销增长，但该结果充分验证了默认设置的合理性：在极低比特场景下，细粒度分块所引入的极小存储代价完全可被容忍，而其提供的高精度归一化对保障优化器状态的重建质量与训练稳定性至关重要。

表 4.7　不同分块大小对验证困惑度与显存占用的影响

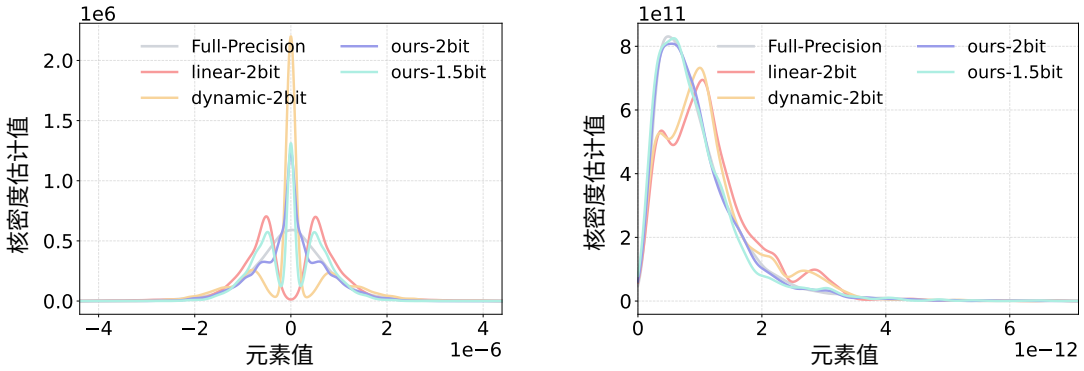
分块大小	LLaMA-130M		GPT-2-124M	
	VPPL	MC	VPPL	MC
64	**20.904**	219.938	**20.648**	331.375
128	21.445	218.673	21.385	330.086
256	22.703	218.063	23.025	329.465

4.3.4 进一步分析

时间开销。本章提出的 Polaris 算法分块归一化、极坐标向量量化与反量化操作均通过高度优化的 CUDA 算子实现。如表 4.2与表 4.3所示，在 LLaMA-130M 的 C4 预训练中，2 比特 Adafactor 相比 32 比特基线挂钟时间仅增加约 0.3%，1.5-bit 版本增加约 0.7%；AdamW 的 16 比特基线耗时 22.33 小时，2 比特与 1.5 比特版本分别为 22.63 与 22.59 小时，增幅约 1.2%–1.3%。在更大规模的 LLaMA-350M 上，低比特版本与全精度的时间差异在 1% 以内；ImageNet-1K 任务中，ViT-Base/16 的时间增幅约 0.9%–1.5%，ResNet-50 约 3.8%。此外，图 4.6中横坐标为挂钟时间的验证困惑度曲线显示，训练相同步数时，Polaris 与全精度基线的耗时基本持平。因此，Polaris 算法在显著压缩显存的同时，端到端效率与全精度版本基本持平，额外开销可忽略不计。

低比特量化下优化器状态的分布保持。图 4.8 直观呈现了本章算法能更忠实地复现一阶动量和二阶动量的原始全精度分布。该图以 LLaMA-130M 模型第 8 层 `q_proj` 权重的动量状态为分析对象，系统对比了不同量化技术的分布保持效果。其中，以全精度数据为参考基准，核密度估计（KDE）曲线通过平滑拟合直观呈现数据的概率分布特征，从而定性评估各量化技术对原始分布形态的保持能力。观察可知，相较于数据无关的一维标量量化方案，本章提出的二维向量量化技术能够更准确地拟合原始分布。该优势在处理高度偏态的二阶动量时尤为显著：即便在 1.5 比特的激进量化设置下，本章算法仍能稳定保持分布的主体

结构与尾部特征，而一维标量量化技术在此极端压缩比下难以有效捕捉数据的基本结构，导致分布失真。



(a) 一阶动量的核密度分布（KDE）　(b) 二阶动量的核密度分布（KDE）

图 4.8　LLaMA-130M 模型动量状态在不同量化技术下的性能对比图

4.4 本章小结

本章针对传统一维标量量化在极低比特下表征能力严重退化的问题，提出了基于极坐标向量量化的优化器状态压缩算法 Polaris，并构建了 AdamW 与 Adafactor 的 1.5/2.0 比特变体。在理论与设计层面，本章深入揭示了优化器动量的准高斯分布与圆对称特性，严格证明了角度均匀采样的合理性并推导了量化误差控制界。针对无符号动量，本章提出第一象限映射与幅值自适应码字分配策略，辅以轴偏移机制有效保障训练数值稳定性。涵盖语言模型预训练、微调及视觉任务的广泛实验表明，Polaris 在大幅压缩状态显存的同时，其 1.5/2.0 比特版本均实现了与 16/32 比特全精度相当的性能与收敛表现。本章工作为优化器状态的极低比特存储提供了一条可行的技术路径。

第 5 章 算法集成

第三章聚焦第一个关键环节——梯度估计，提出了基于随机子空间的零阶梯度估计算法 SubZero，有效降低了梯度估计方差。第四章聚焦第二个关键环节——优化器状态更新，提出了基于极坐标向量量化的优化器状态压缩算法 Polaris，实现了优化器状态极低比特压缩。本章将 SubZero 与 Polaris 进行系统集成，围绕梯度估计器、优化器类型与优化器状态压缩三个正交维度展开实验，系统评估不同组合方案在显存占用、收敛速度及模型性能等方面的表现，验证两种技术在显存效率与优化性能上的协同优势。

5.1 引言

在大语言模型微调中，优化器的选择直接影响收敛速度与最终性能。将零阶优化用于大语言模型微调的开创性工作 MeZO[16]为实现极致显存效率，默认采用不带动量的 SGD 优化器[45]。然而，相比原始 SGD，带动量的优化器通过累积历史梯度信息，能够有效平滑梯度噪声、加速收敛并提升模型泛化能力[12,48]。

SGDM[48]通过维护一阶动量，在梯度方向上累积历史更新，有效抑制震荡并加速收敛，其更新形式见式 (2-6)。一阶动量的引入使参数更新方向更加稳定，有效缓解了零阶梯度估计中的高方差问题。

AdamW[12]作为自适应学习率优化器，同时维护一阶和二阶动量，其更新形式见式 (2-8)。AdamW 通过二阶矩估计自适应调整各参数的学习率，并解耦权重衰减与梯度更新，在大语言模型微调中表现出卓越性能。

然而，动量状态的维护显著增加了显存占用。动量状态的存储开销与模型参数量成正比。当模型规模达到数十亿参数级别时，这一开销变得极为可观。以 AdamW 优化器为例，其需同时维护一阶和二阶动量，显存需求可达模型参数本身的两倍，成为制约大模型微调的关键瓶颈。

5.2 实验设置

本实验在 SST-2 情感分类数据集[91]上对 1.3B 和 13B 的 OPT 系列模型[75]进行全参数微调。实验设置与小节 3.4保持一致，从 SST-2 中采样 1000 条训练样本、500 条验证样本和 1000 条测试样本，共训练 15000 步。评估指标涵盖峰值显存占用（GB）、时间开销（分钟）、训练损失以及测试准确率（%）四个维度，用于全面衡量各方案在显存效率与优化性能方面的表现。

实验围绕三个正交维度展开：梯度估计器、优化器类型、优化器状态压缩。其中 MeZO+SGD 仅使用零阶梯度估计，不引入任何动量机制，其显存占用水平代表了大模型微调的理论下限，在本实验中作为显存效率的参考基准。

表 5.1　算法集成验证实验方案

方案	梯度估计器	优化器	优化器状态压缩
MeZO + SGD	MeZO	SGD	全精度
MeZO + SGDM	MeZO	SGDM	全精度
MeZO + AdamW	MeZO	AdamW	全精度
SubZero + SGDM	SubZero	SGDM	全精度
SubZero + AdamW	SubZero	AdamW	全精度
SubZero + SGDM + Polaris	SubZero	SGDM	2 比特压缩
SubZero + AdamW + Polaris	SubZero	AdamW	2 比特压缩

5.3 实验结果

5.3.1 SGDM 优化器

表 5.2给出了 SGDM 优化器下的不同实验方案性能对比结果。为便于分析，表格中特别标注了 MeZO+SGD 作为显存占用参考基准，并以 MeZO+SGDM 为基准计算各方案的相对下降比例。从表 5.2可以观察到以下关键现象。

关于显存优化效果，MeZO+SGD 因不存储任何动量状态，显存占用最低，但这牺牲了动量带来的收敛稳定性。相比之下，MeZO+SGDM 在 OPT-1.3B 上占用 5.86 GB，在 OPT-13B 上占用 50.02 GB。SubZero+SGDM+Polaris 通过极坐标向量量化将一阶动量压缩至 2 比特，显存占用分别降至 3.83 GB 和 30.73 GB，较 MeZO+SGDM 下降 34.6% 和 38.6%。这一结果表明，Polaris 的极坐标量化策略能够显著降低动量状态的显存开销。

关于训练动态分析，值得注意的是 SubZero+SGDM+Polaris 在大幅降低显存的同时，测试准确率与 SubZero+SGDM 持平，表明 2 比特极坐标量化对 SGDM 的优化动态影响极小。这验证了 Polaris 量化策略在极低比特下的稳定性。

从图 5.1和图 5.2可见，SubZero+SGDM 和 SubZero+SGDM+Polaris 的训练损失曲线几乎完全重合，且均显著优于 MeZO+SGDM。这说明 SubZero 的低秩梯度估计加速了收敛，Polaris 量化对训练动态几乎无影响，量化后的优化器在收敛速度和模型精度上均与全精度版本持平，验证了其在极低比特设置下的有效性。

表 5.2　SGDM 优化器下的不同实验方案性能对比

模型	方案	峰值显存	显存降幅	准确率	时间开销
OPT-1.3B	MeZO + SGD	3.40	-	85.89	41.97
	MeZO + SGDM	5.86	0%	87.38	42.49
	SubZero + SGDM	5.90	-0.7%	90.94	47.01
	SubZero + SGDM + Polaris	3.83	34.6%	90.25	47.23
OPT-13B	MeZO + SGD	26.08	-	90.71	213.56
	MeZO + SGDM	50.02	0%	91.85	237.60
	SubZero + SGDM	50.92	-1.8%	92.43	258.30
	SubZero + SGDM + Polaris	30.73	38.6%	91.97	260.11

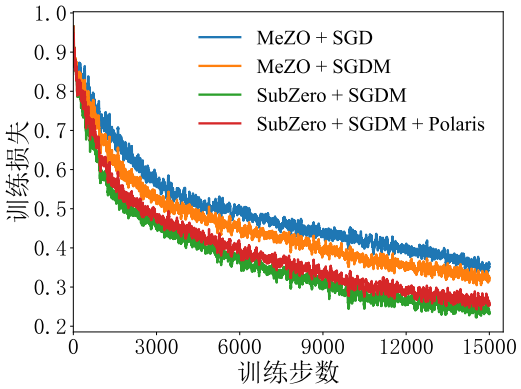


图 5.1　训练损失曲线（SGDM，OPT-1.3B）

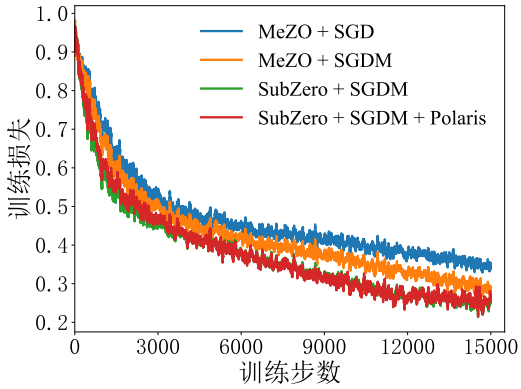


图 5.2　训练损失曲线（SGDM，OPT-13B）

5.3.2 AdamW 优化器

表 5.3给出了 AdamW 优化器下的不同实验方案性能对比结果。为便于分析，表格中特别标注了 MeZO+SGD 作为显存占用参考基准，并以 MeZO+AdamW 为基准计算各方案的相对下降比例。从表 5.3可以观察到以下关键现象。

关于显存优化效果，AdamW 需要同时存储一阶和二阶动量，显存开销显著高于 SGDM。MeZO+AdamW 在 OPT-1.3B 上占用 8.31 GB，在 OPT-13B 上占用 73.77 GB。SubZero+AdamW+Polaris 通过极坐标向量量化将一阶动量和二阶动量压缩至 2 比特，显存占用分别降至 4.42 GB 和 35.22 GB，较 MeZO+AdamW 下降 46.8% 和 52.3%。这一降幅超过 SGDM 场景，原因是 AdamW 的动量显存占用占比更大，压缩收益更显著。

关于训练动态分析，值得注意的是 SubZero+AdamW+Polaris 在大幅降低显存的同时，测试准确率与 SubZero+AdamW 持平，表明 2 比特极坐标量化对 AdamW 的优化动态影响极小。这验证了 Polaris 量化策略在极低比特下的稳定性。

从图 5.3和图 5.4可见，SubZero+AdamW 和 SubZero+AdamW+Polaris 的训练损失曲线几乎完全重合，且均显著优于 MeZO+AdamW。这说明 SubZero 的低秩

表 5.3　AdamW 优化器下的不同实验方案性能对比

模型	方案	峰值显存	显存降幅	准确率	时间开销
OPT-1.3B	MeZO + SGD	3.40	-	85.89	41.97
	MeZO + AdamW	8.31	0%	90.82	54.11
	SubZero + AdamW	8.35	-0.5%	91.85	57.78
	SubZero + AdamW + Polaris	4.42	46.8%	91.28	57.99
OPT-13B	MeZO + SGD	26.08	-	90.71	213.56
	MeZO + AdamW	73.77	0%	92.66	287.15
	SubZero + AdamW	74.86	-1.5%	92.88	309.71
	SubZero + AdamW + Polaris	35.22	52.3%	92.88	314.40

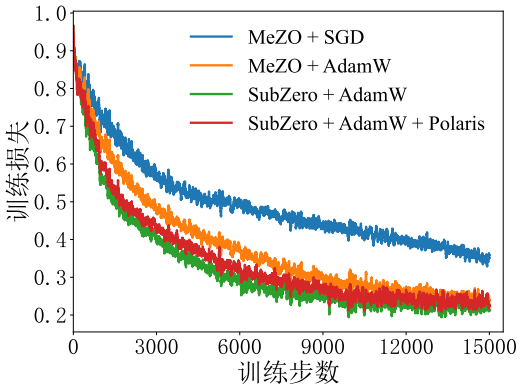


图 5.3　训练损失曲线（AdamW，OPT-1.3B）

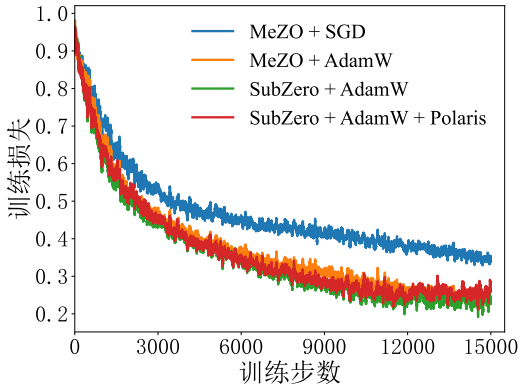


图 5.4　训练损失曲线（AdamW，OPT-13B）

梯度估计加速了收敛，同时 Polaris 量化对训练动态几乎无影响，量化后的优化器在收敛速度和模型精度上均与全精度版本持平，验证了其在极低比特设置下的有效性。

5.4 分析与讨论

5.4.1 时间效率分析

表 5.2和表 5.3的时间开销数据表明，SubZero 与 Polaris 所引入的额外计算开销均保持在较低水平。

在 SGDM 优化器下，SubZero+SGDM 相较于 MeZO+SGDM 在 OPT-1.3B 和 OPT-13B 上时间开销分别从 42.49 分钟增至 47.01 分钟、从 237.60 分钟增至 258.30 分钟，增幅约 10%。Polaris 量化引入的时间开销极小，SGDM+Polaris 相较于 SGDM 在两个模型上分别仅增加 0.22 和 1.81 分钟。

在 AdamW 优化器下，SubZero+AdamW 相较于 MeZO+AdamW 在 OPT-1.3B 和 OPT-13B 上时间开销分别从 54.11 分钟增至 57.78 分钟、从 287.15 分钟增至

309.71 分钟，增幅约 7%。Polaris 量化引入的时间开销同样极小，AdamW+Polaris 相较于 AdamW 在两个模型上分别仅增加 0.21 和 4.69 分钟。

结合小节 3.4与小节 4.3分析，SubZero 的额外开销源于低秩子空间投影与重构，Polaris 的额外开销源于极坐标量化与反量化。实验数据表明，两项技术的时间开销增幅均在可接受范围内。因此，集成方案未引入显著的额外训练开销。

5.4.2 参考基准分析

表 5.2和表 5.3中 MeZO+SGD 作为显存效率的理论参考基准，其 3.40 GB 和 26.08 GB 的显存占用代表零阶优化在不使用任何动量时的下限。实际应用中，无动量 SGD 的配置往往难以达到理想的收敛效果，特别是在大规模语言模型微调任务中。因此，本研究的重点在于在保证优化性能的前提下，尽可能逼近这一理论下限。例如，在 OPT-1.3B 上 SubZero+SGDM+Polaris 和 SubZero+AdamW+Polaris 分别将显存开销降至 3.83 GB 和 4.42 GB，与理论下限的差距缩小至 12.6% 和 29.9%，同时显著优于无动量 SGD 基线。

5.4.3 显存优化的协同效应

综合 SGDM 和 AdamW 的实验结果，可以得出以下结论。

SubZero 梯度估计与优化器类型正交。无论使用 SGDM 还是 AdamW，SubZero 都能在保持相近显存开销的前提下提升模型精度。这说明低秩子空间扰动策略不依赖于特定的优化器选择。

Polaris 压缩的收益与动量状态规模正相关。AdamW 因维护更多动量状态，压缩后显存降幅高于 SGDM。这一规律在实际应用中具有指导意义：优化器动量状态越复杂，Polaris 的压缩价值越大。

两种技术的组合在大幅压缩显存的同时保持了收敛速度和模型性能。在 OPT-13B 模型上，SubZero+AdamW+Polaris 的组合方案仅占用 35.22 GB 显存，较 MeZO+AdamW 下降 52.3%，同时保持了与全精度版本相当的收敛速度和测试准确率。这一方案为大模型微调提供了显存高效的实用路径。

5.5 本章小结

本章通过系统集成验证，量化了 SubZero 与 Polaris 协同优化的实际效果。SubZero 与 Polaris 的组合应用能够在显著降低显存开销的同时保持收敛速度和模型性能，为资源受限大模型微调提供了技术支撑。

第 6 章 总结与展望

6.1 工作总结

在资源受限的硬件环境下，实现大语言模型的高效微调已成为推动其普惠应用与规模化落地的关键所在。作为显存高效优化的代表性方法之一，零阶优化仅需通过前向传播即可估计梯度，避免了反向传播过程中的激活值缓存，从而大幅压缩了显存占用。然而，当前零阶优化方法在其两大核心步骤——梯度估计与优化器状态更新的执行过程中，仍面临显著挑战：一方面，其梯度估计方差随模型参数量呈线性增长，严重制约收敛速度；另一方面，尽管引入动量等优化器状态可有效加速训练，但维护这些状态变量会带来不可忽视的额外显存开销，部分抵消了零阶方法本身的显存优势。

针对上述挑战，本文深入挖掘零阶优化中梯度估计与优化器状态更新环节的内蕴特性，将对真实优化过程中的优化状态的洞察转化为针对性算法设计。本文的主要贡献如下：

（1）针对零阶优化梯度估计方差大、收敛慢的问题，本文提出基于随机子空间的零阶梯度估计算法 SubZero。该算法利用梯度矩阵的低秩结构，通过构造列正交投影矩阵，在低维子空间内生成扰动并估计梯度，将梯度估计方差从依赖模型总参数量降至仅依赖子空间维度。为平衡计算效率与估计精度，算法采用分层低秩扰动策略与延迟子空间更新机制，在避免存储高维投影矩阵的同时，实现高效子空间探索。理论分析证明了算法的收敛性及其对反向传播梯度的逼近能力。在多种大语言模型微调实验中，SubZero 在显存开销与现有零阶算法相当的前提下，收敛速度与微调性能均优于同类算法，且兼容全参数微调与 LoRA、前缀微调、提示微调等多种参数高效微调范式。上述结果表明，SubZero 有效提升了零阶优化在大模型微调中的梯度估计质量。

（2）针对基于动量的优化器状态显存占用大、且现有量化方法在低于 4 比特时易引发训练崩溃的问题，本文提出基于极坐标向量量化的优化器状态压缩算法 Polaris。该算法利用动量值分布的圆对称性与准高斯特性，构建二维极坐标量化框架。该框架遵循低精度存储与高精度计算相结合的原则，在参数更新时将低比特动量临时反量化为高精度，更新完成后重新量化存储，显著降低静态显存占用。对于一阶动量，采用角度均匀分布的码本结构以确保方向精度。对于二阶动量，设计第一象限映射机制，并采用自适应角度分配策略：为大模值分量分配更多码字以保障量化精度，为小模值分量分配较少码字以提升码本利用率，同时引入轴偏移机制以规避除零风险。静态实验表明，在极低比特设置下，二维向量量

化的重建误差显著低于标量量化。图像分类与自然语言建模等多项任务的实验结果表明，Polaris 将优化器状态压缩至 1.5-2 比特后仍能稳定训练，性能与全精度版本相当。上述结果表明，Polaris 为在零阶优化框架中实现优化器状态极低比特存储提供了关键技术。

（3）本文将所提出的算法与 AdamW 等基于动量的优化器进行算法集成，并在大语言模型微调任务中开展了系统性验证。实验围绕零阶梯度估计器、优化器类型与优化器状态压缩三个维度展开，通过对比多种方案组合在 OPT 系列模型上的全参数微调性能，评估了各方案的显存效率与模型性能。实验结果表明，SubZero 与 Polaris 的组合方案在大幅降低显存开销的同时保持了收敛速度和模型性能。

本文以优化状态特性为主线，围绕零阶优化技术，从梯度估计的低秩建模、优化器状态的极低比特压缩以及二者的算法集成验证三个方面，系统研究了显存高效的零阶优化算法。SubZero 通过利用梯度矩阵的低秩结构，实现了更高质量的梯度估计；Polaris 通过利用动量值分布的统计特性，实现了优化器状态的极低比特压缩。算法集成验证进一步表明，将上述技术协同应用于大模型微调，可显著降低显存开销，同时保持收敛速度和模型性能。上述成果为资源受限场景下的大模型微调提供了技术支撑。

6.2 未来展望

本文工作从梯度估计与优化器状态更新两个维度探索了显存高效的零阶优化算法。随着大模型在手机、座舱等边缘端应用场景的扩展，模型规模持续增长与算力异构化对零阶优化算法设计提出了更高要求。未来研究可从以下三个方向展开：

（1）零阶优化与全链路量化技术的结合。当前零阶梯度估计与动量量化压缩多独立发展，二者结合有望进一步降低显存开销。未来可构建零阶估计与量化训练的联合框架，在梯度扰动生成阶段引入量化感知机制，通过理论建模分析量化噪声与零阶方差的耦合关系，并设计抗噪正则化策略以提升鲁棒性。同时，面向权重、激活值与优化器状态等核心显存占用模块，探索统一的动态量化方法与误差控制策略。

（2）基于低秩子空间的高精度梯度估计与自适应优化加速。现有零阶优化算法在子空间内的优化动力学利用尚不充分，且受限于查询预算，梯度估计往往存在较大方差。未来研究将聚焦于提升子空间内的优化效率与估计精度：一方面，通过合理增加查询次数，获取低方差的梯度估计，克服零阶优化固有的噪声瓶颈；另一方面，充分利用投影至低秩子空间后的动量信息，结合 SGDM 或 Adam

等先进优化器的自适应学习率与动量累积机制，实现子空间内的加速收敛。旨在通过高精度的梯度估计与高效的优化策略相结合，显著提升零阶优化算法的收敛效率，使其最终性能逼近一阶优化水平。

（3）面向零阶优化特性的软硬协同优化。零阶优化以前向传播为主要计算模式，其算法特性与底层硬件可深度结合。未来可针对零阶优化特有的双前向传播模式和低秩子空间运算特征，优化数据访存与计算调度；结合激活重计算、CPU/GPU 异构卸载以及动态精度缩放等策略，形成算法与系统协同的显存管理方案。此外，可探索将 Polaris 的极坐标量化硬件加速单元与 SubZero 的扰动生成流程协同设计，实现端到端的零阶优化训练加速，为单卡微调百亿级参数模型等场景提供支持。

附录 A SubZero 算法补充内容

本附录是第 3 章 SubZero 算法的补充材料。内容包括 SubZero 算法的具体实现细节（如原地操作与逐层更新策略）、超参数搜索网格以及实验中使用的提示模板。这些细节旨在辅助读者复现本文提出的零阶微调方法，并理解其在不同模型规模与任务下的配置依据。

A.1 实现细节

SubZero 中的梯度估计适用于参数矩阵，而大语言模型主要由线性层构成。对于其他可训练参数（如偏置和层归一化参数），本文建议采用 MeZO[16] 中的梯度估计方法，因为这些层参数量较少。

本文引入两种有效策略以显存高效地实现 SubZero。

原地操作。如式 (A-1) 所示，直接计算损失差值 ρ 需要同时存储参数矩阵集 $\mathcal{W}$ 和扰动矩阵集 $\tilde{\mathcal{Z}}$，显存需求为推理的两倍。为解决此问题，本文从 MeZO 获得启发，采用原地操作。利用随机种子技巧，本文存储一个随机种子用于计算 ρ（见算法3.3第 10-13 行及算法3.2），并重新生成低维扰动矩阵 $\boldsymbol{Z}_1, \boldsymbol{Z}_2, \cdots, \boldsymbol{Z}_l$（见算法3.3第 16 行）。因此，SubZero 的微调显存开销逼近推理阶段水平，并与 MeZO 保持相当（见表3.9和表3.10）。

逐层参数更新。一阶优化器在反向传播后需存储完整梯度，然后更新所有模型参数。相比之下，SubZero 等零阶优化器先通过两次前向传播计算损失差值，再利用该差值与各层扰动逐层计算梯度估计。为降低训练过程中显存占用，本文可在每层计算完梯度估计后立即执行 `optimizer.step()` 完成参数更新。

借助上述两种实现策略，SubZero 显著降低了 GPU 显存消耗。需要注意的是，本文在所有实验中均对 MeZO 采用了逐层参数更新策略。

为简化超参数调优，本文采用范数对齐技巧，使得 SubZero 可以直接沿用 MeZO[16] 的超参数设置（如学习率）。对于随机扰动矩阵 $\boldsymbol{Z} \in \mathbb{R}^{m\times n}$，其低秩近似为 $\hat{\boldsymbol{Z}} = \boldsymbol{U}\boldsymbol{Z}'\boldsymbol{V}^{\top}$，其中 $\boldsymbol{U} \in \mathbb{R}^{m\times r}$，$\boldsymbol{V} \in \mathbb{R}^{n\times r}$，$\boldsymbol{Z}' \in \mathbb{R}^{r\times r}$。若 $\boldsymbol{Z}$ 和 $\boldsymbol{Z}'$ 为高斯随机矩阵，$\boldsymbol{U}$ 和 $\boldsymbol{V}$ 为列正交矩阵，则有：

$$\mathbb{E}[\|\boldsymbol{Z}\|_F] = \sqrt{\frac{m\times n}{r^2}}\mathbb{E}\left[\|\hat{\boldsymbol{Z}}\|_F\right]. \tag{A-1}$$

记 $\mu = \sqrt{\frac{m\times n}{r^2}}$。设 MeZO 的学习率为 η、扰动尺度为 ε，则 SubZero 有两种等价方式获得扰动。第一种方法是将随机低维扰动矩阵乘以 μ，SubZero 直接采用 MeZO 的超参数：$\eta' = \eta$，$\varepsilon' = \varepsilon$。第二种方法保持随机低维扰动矩阵不变，将

SubZero 的学习率和扰动尺度设置为：

$$\eta' = \eta\mu^2, \quad \varepsilon' = \varepsilon\mu.$$

本文认为范数对齐对 SubZero 至关重要，因为改变秩 r 会影响梯度估计的范数，从而使得相关学习率的调优复杂化。

A.2 超参数搜索

使用更大批量可一致地降低零阶优化的方差，从而提升微调性能[16,22,30]。然而，批量增大会增加前向传播时间并显著提升显存占用。本文专注于开发在小批量下仍能降低方差、提升性能的零阶方法，默认批量大小为 16。在部分 SGD 实验中（如 MultiRC 和 SQuAD），受限于 GPU 资源，批量大小降至 8。

与先前研究一致[16,18-19,30]，为保持显存效率，本文默认使用无动量 SGD。SGD 采用线性学习率调度，而所有结合 SGD 的零阶方法均采用常数学习率调度，权重衰减设为 0。

对于 RoBERTa，Adam 运行 1K 步，零阶方法运行 100K 步。其余实验中，Adam 运行 5 轮，SGD 及零阶方法运行 20K 步。

本文参照先前工作设置 PEFT 方案中的超参数[16,18]。对于 LoRA，秩设为 8，α 设为 16。对于前缀微调，前缀词元长度设为 5，本文从词汇表中随机采样词元，输入 LLM 以获得不同注意力层的键和值，以此初始化可调表示。对于提示微调，提示虚拟词元长度设为 10，提示词元使用模型嵌入中的实际词元值初始化。

本文在表 A.1 和表 A.2 中给出超参数搜索网格，以辅助结果复现。对于 OPT-1.3B，采用与表 A.2 相同的超参数设置。对于 RoBERTa-large，MeZO 和 SubZero 的学习率设为 {1e-6, 5e-6}，ε=1e-3，批量大小 64。SubZero 的秩设为 {8, 16, 24}，子空间更新频率调整为 {1000, 2000}。

A.3 提示模板

对于自回归大语言模型，本文处理三种任务类型：分类、多项选择与问答。本文采用文献[16] 中各类任务的提示模板，汇总于表 A.3。对于掩码大语言模型，本文同样采用文献[16] 中的提示模板，如表 A.4 所示。

表 A.1　OPT-13B 的超参数搜索网格。每个任务中，零阶方法（MeZO、S-MeZO 和 SubZero）及 SGD 均运行 20K 步。本文每 500 训练步根据验证损失记录最佳模型检查点。

实验	超参数	取值
MeZO(全参数微调)	批量大小	16
	学习率	{1e-7, 2e-7, 5e-7, 1e-6}
	ε	1e-3
MeZO(LoRA)	批量大小	16
	学习率	{1.5e-5, 3e-5, 5e-5}
	ε	1e-3
S-MeZO(全参数微调)	批量大小	16
	学习率	{1e-6, 5e-6}
	ε	1e-3
	稀疏率	0.75
S-MeZO(LoRA)	批量大小	16
	学习率	{5e-5, 1e-4, 1e-3}
	ε	1e-3
	稀疏率	0.75
SubZero(全参数微调)	批量大小	16
	学习率	{1e-7, 2e-7, 5e-7, 1e-6}
	ε	1e-3
	秩	{32, 64, 128, 256 }
	子空间更新频率	{500, 1000, 2000}
SubZero(LoRA)	批量大小	16
	学习率	{1.5e-5, 3e-5, 5e-5}
	ε	1e-3
	秩	{4, 8, 16}
	子空间更新频率	{500, 1000, 2000}
SGD(全参数微调)	批量大小	16
	学习率	{1e-4, 1e-3, 5e-3}

表 A.2　LLaMA2-7B 和 Mistral-7B 的超参数搜索网格。每个任务中，零阶方法（MeZO 和 SubZero）及 SGD 均运行 20K 步。本文每 500 训练步根据验证损失记录最佳模型检查点。

实验	超参数	取值
MeZO(全参数微调)	批量大小	16
	学习率	{1e-7, 5e-7, 1e-6}
	ε	1e-3
MeZO(LoRA)	批量大小	16
	学习率	{1e-6, 5e-6, 1e-5, 3e-5}
	ε	1e-3
MeZO(前缀微调)	批量大小	16
	学习率	{1e-3, 5e-3, 1e-2}
	ε	1e-1
MeZO(提示微调)	批量大小	16
	学习率	{1e-3, 5e-3, 1e-2}
	ε	1e-2
SubZero(全参数微调)	批量大小	16
	学习率	{1e-7, 5e-7, 1e-6}
	ε	1e-3
	秩	{24, 48}
	子空间更新频率	1000
SubZero(LoRA)	批量大小	16
	学习率	{1e-6, 5e-6, 1e-5, 3e-5}
	ε	1e-3
	秩	{4, 8}
	子空间更新频率	1000
SubZero(前缀微调)	批量大小	16
	学习率	{1e-3, 5e-3, 1e-2}
	ε	1e-1
	秩	{4, 8}
	子空间更新频率	1000
SubZero(提示微调)	批量大小	16
	学习率	{1e-3, 5e-3, 1e-2}
	ε	1e-2
	秩	{16, 24}
	子空间更新频率	1000
SGD(全参数微调)	批量大小	16
	学习率	{1e-5, 1e-4, 1e-3, 5e-3}

表 A.3　OPT-1.3B、OPT-13B、LLaMA2-7B 及 Mistral-7B 实验中使用的提示模板

任务	类型	提示模板
SST-2	分类	<text> It was terrible/great
RTE	分类	<premise> Does this mean that "<hypothesis>" is true? Yes or No? Yes/No
CB	分类	Does this mean that "<hypothesis>" is true? Yes or No? Yes/No/Maybe
BoolQ	分类	<passage> <question>? Yes/No
WSC	分类	<text> In the previous sentence, does the pronoun "<span2>" refer to <span1>? Yes or No? Yes/No
WIC	分类	Does the word "<word>" have the same meaning in these two sentences? Yes, No? <sentence1> <sentence2> Yes/No
MultiRC	分类	<paragraph> Question: <question> I found this answer "<answer>". Is that correct? Yes or No? Yes/No
COPA	多项选择	<premise> so/because <candidate>
ReCoRD	多项选择	<passage> <query>.replace("@placeholder", <candidate>)
SQuAD	问答	Title: <title> Context: <context> Question: <question> Answer:
DROP	问答	Passage: <context> Question: <question> Answer:

表 A.4　RoBERTa-large 实验中使用的提示模板。*C* 为分类类别数。

任务	*C*	类型	提示模板
SST-2	2	情感分类	<sentence1> It was great/terrible
SST-5	5	情感分类	<sentence1> It was great/good/okay/bad/terrible
MNLI	3	自然语言推理	<sentence1> ? Yes/Maybe/No , <sentence2>
SNLI	3	自然语言推理	<sentence1> ? Yes/Maybe/No , <sentence2>

附录 B Polaris 算法补充内容

本附录是第 4 章 Polaris 算法的补充材料。内容包括详细的实验数据集设置、量化算法的具体实现流程、核心理论引理的证明过程以及一维标量量化映射的构造细节。这些内容旨在为读者提供完整的复现依据，并深入理解二维极坐标量化背后的数学原理。

B.1 实验数据集及设置

本文实验基于 PyTorch 2.2.0 和 CUDA 12.1 框架，使用单张 A800 GPU 进行。为获取每张 GPU 的总峰值显存消耗，本文调用 `torch.cuda.max_memory_allocated` 函数。总显存开销包括数据、模型参数、激活值、梯度、优化器状态及显存碎片。优化器状态的显存开销通过计算使用目标优化器训练时的显存占用与使用无动量优化器训练时的显存占用之差得到。

对于 Adafactor，默认设置 eps $=(10^{-30},10^{-3})$，clip_threshold $=1.0$，decay_rate $=-0.8$ 且 $\beta_1=0.9$。对于 AdamW，设置 $\beta_1=0.9$ 且 $\beta_2=0.95$。在量化设置中，尺寸小于 4096 的矩阵不进行量化。

C4 上训练 LLaMA-2 的设置。训练 130M LLaMA-2 时，Adafactor/AdamW 设置 2000 步预热；训练 350M LLaMA-2 时，设置 4000 步预热。总批量大小设为 512。训练 130M LLaMA-2 时批量大小设为 256，训练 350M LLaMA-2 时设为 128。数据类型为 `bfloat16`。初始学习率为 0.001，权重衰减为 0.0。

OWT 上训练 GPT-2 的设置。 Adafactor/AdamW 设置 2000 步预热。总批量大小设为 480。训练 124M GPT-2 时批量大小设为 24。数据类型为 `bfloat16`。初始学习率为 0.0006，权重衰减为 0.1。

ImageNet-1k 上训练 ResNet50 的设置。使用 SGDM[48]/AdamW/Adafactor 训练 100 个 epoch，前 10 个 epoch 采用线性预热。小批量大小设为 512。对于 SGDM，动量衰减 β 设为 0.9，初始学习率设为 0.1，权重衰减设为 0.0005。对于 AdamW/Adafactor，初始学习率设为 0.001，权重衰减设为 0.05。采用余弦学习率调度。数据增强遵循[105] 中训练 ResNet50 的配置。训练利用 PyTorch 原生自动混合精度（AMP）功能（`torch.cuda.amp`）。

ImageNet-1k 上训练 ViT-Base/16 的设置。使用 Adafactor/AdamW 训练 150 个 epoch，前 10 个 epoch 采用线性预热。小批量大小设为 512。初始学习率为 0.001，权重衰减为 0.05。采用余弦学习率调度。数据增强遵循[105] 中训练 ViT-

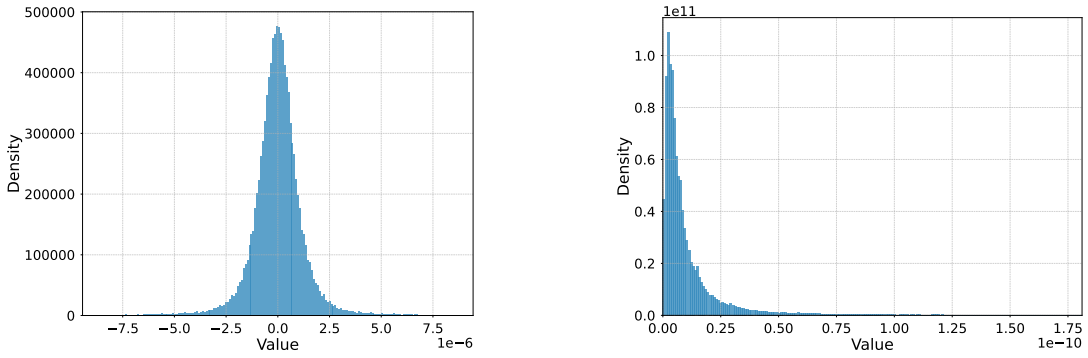
Base/16 的配置，不包括重复增强。训练利用 PyTorch 原生自动混合精度（AMP）功能（`torch.cuda.amp`）。

微调 7B 模型的设置。数据类型为 `bfloat16`。训练 epoch 数设为 3。批量大小设为 2，梯度累积步数设为 32。初始学习率为 0.00003，权重衰减为 0.0。设置 warmup_ratio = 0.3，并采用余弦学习率衰减。

运行 GLUE 基准的设置。设置 batch_size = auto 且 num_fewshot = 5。

B.2 优化器状态分布

如图 B.1 所示，本文绘制了一阶动量（图 B.1(a)）和二阶动量（图 B.1(b)）的分布，数据捕获自 LLaMA-130M 模型训练过程中第 8 层的 `q_proj` 权重。关键观察在于两者分布的显著差异：一阶动量呈现准高斯分布，具有明显的圆对称性；而二阶动量为非负且高度偏斜，值高度集中于零附近。



(a) 一阶动量　　(b) 二阶动量

图 B.1　LLaMA-130M 模型第 8 层 `q_proj` 权重的 AdamW 优化器状态的非均匀分布。

B.3 量化算法细节

低比特的 Adafactor 优化器详见算法 B.1。

算法 B.1 低比特 Adafactor 优化器

输入： 步数 T, 学习率 η, 超参数 β_1, β_2, 常数 ϵ_1, ϵ_2
输出： $\boldsymbol{\theta}_T$

初始化 $\boldsymbol{\theta}_0$, $\boldsymbol{m}_0^q \leftarrow 0$, $\boldsymbol{R}_0 \leftarrow 0$, $\boldsymbol{C}_0 \leftarrow 0$
for $t = 1, \cdots, T$ **do**
　$\boldsymbol{g}_t \leftarrow \nabla_\theta f(\boldsymbol{\theta}_{t-1})$　// 计算梯度
　$\boldsymbol{m}_{t-1} \leftarrow \texttt{dequantize}(\boldsymbol{m}_{t-1}^q)$　// 反量化一阶动量
　$\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$　// 更新一阶动量
　$\boldsymbol{R}_t \leftarrow \beta_2 \boldsymbol{R}_{t-1} + (1-\beta_2)\mathrm{E}_{\text{行}}[\boldsymbol{g}_t^2]$,　$\boldsymbol{C}_t \leftarrow \beta_2 \boldsymbol{C}_{t-1} + (1-\beta_2)\mathrm{E}_{\text{列}}[\boldsymbol{g}_t^2]$　// 分解二阶动量
　$\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t/(1-\beta_1^t)$,　$\boldsymbol{V}_t \leftarrow (\boldsymbol{R}_t \otimes \boldsymbol{C}_t)/\mathrm{mean}(\boldsymbol{R}_t)$　// 偏差校正与重构
　$\eta_t \leftarrow \max(\epsilon_2, \mathrm{RMS}(\boldsymbol{\theta}_{t-1}))^{-1}$　// 自适应学习率
　$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left(\frac{\alpha \cdot \hat{\boldsymbol{m}}_t}{\sqrt{\boldsymbol{V}_t} + \epsilon_1} \right)$　// 更新参数
　$\boldsymbol{m}_t^q \leftarrow \texttt{quantize}(\boldsymbol{m}_t)$　// 量化存储一阶动量
end for

本文的量化策略选择性地应用于模型内的特定参数组。定义 S_Q 为设计用于量化的参数组索引集合。在实验中，S_Q 主要包含线性层的核心权重张量（通常呈二维形状），因为这些是显存占用最大的部分。

相反，若干参数组被故意保持为全精度，因此被排除在 S_Q 之外。对于大语言模型（LLM），这些非量化部分主要包括嵌入层参数、线性层内的偏置向量以及归一化层（如 LayerNorm、RMSNorm）内的所有参数。

对于所有参数组 $\boldsymbol{\theta}_i$，引入逐层缩放因子 α_i，该因子在主参数更新前应用于归一化动量项。该因子正式定义为：

$$\alpha_i = \begin{cases} \alpha_{\text{scale}} & \text{若} i \in S_Q \\ 1.0 & \text{若} i \notin S_Q \end{cases} \tag{B-1}$$

相比之下，不使用此类机制的先前方法[17,37] 等价于对所有层设置 $\alpha_i = 1.0$。超参数 α_{scale} 根据量化级别设置（例如 2 比特设为 2.0）。

修改后的优化过程详见算法 B.2 和 B.3。

算法 B.2 Polaris 低比特 AdamW 优化器

输入： 步数 T, 学习率 η, 超参数 β_1, β_2, 权重衰减 λ, 常数 ϵ
输出： $\boldsymbol{\theta}_T$

设 S_Q 为量化层索引集合
for 每个参数组 i **do**
　初始化 $\boldsymbol{\theta}_{0,i}$, $\boldsymbol{m}^q_{0,i} \leftarrow 0$, $\boldsymbol{v}^q_{0,i} \leftarrow 0$
end for
for $t = 1, \cdots, T$ **do**
　for 每个参数组 i **do**
　　$\boldsymbol{g}_{t,i} \leftarrow \nabla_\theta f(\boldsymbol{\theta}_{t-1,i})$　// 计算梯度
　　if $i \in S_Q$ **then**
　　$\boldsymbol{m}_{t-1,i}, \boldsymbol{v}_{t-1,i} \leftarrow \texttt{dequantize}(\boldsymbol{m}^q_{t-1,i}), \texttt{dequantize}(\boldsymbol{v}^q_{t-1,i})$　// 反量化
　　else
　　$\boldsymbol{m}_{t-1,i}, \boldsymbol{v}_{t-1,i} \leftarrow \boldsymbol{m}^q_{t-1,i}, \boldsymbol{v}^q_{t-1,i}$　// 未量化
　　end if
　　$\boldsymbol{m}_{t,i} \leftarrow \beta_1 \boldsymbol{m}_{t-1,i} + (1-\beta_1)\boldsymbol{g}_{t,i}, \quad \boldsymbol{v}_{t,i} \leftarrow \beta_2 \boldsymbol{v}_{t-1,i} + (1-\beta_2)\boldsymbol{g}^2_{t,i}$　// 更新动量
　　$\hat{\boldsymbol{m}}_{t,i} \leftarrow \boldsymbol{m}_{t,i}/(1-\beta_1^t), \quad \hat{\boldsymbol{v}}_{t,i} \leftarrow \boldsymbol{v}_{t,i}/(1-\beta_2^t)$　// 偏差校正
　　按式 (B-1) 定义 α_i
　　$\boldsymbol{\theta}_{t,i} \leftarrow \boldsymbol{\theta}_{t-1,i} - \eta_t \left(\frac{\alpha_i \cdot \hat{\boldsymbol{m}}_{t,i}}{\sqrt{\hat{\boldsymbol{v}}_{t,i}} + \epsilon} + \lambda \boldsymbol{\theta}_{t-1,i} \right)$　// 更新参数
　　if $i \in S_Q$ **then**
　　$\boldsymbol{m}^q_{t,i}, \boldsymbol{v}^q_{t,i} \leftarrow \texttt{quantize}(\boldsymbol{m}_{t,i}), \texttt{quantize}(\boldsymbol{v}_{t,i})$　// 量化存储
　　else
　　$\boldsymbol{m}^q_{t,i}, \boldsymbol{v}^q_{t,i} \leftarrow \boldsymbol{m}_{t,i}, \boldsymbol{v}_{t,i}$　// 保持全精度
　　end if
　end for
end for

算法 B.3 Polaris 低比特 Adafactor 优化器

输入： 步数 T, 学习率 η, 超参数 β_1, β_2, 常数 ϵ_1, ϵ_2
输出： θ_T

设 S_Q 为量化层索引集合
for 每个参数组 i **do**
　初始化 $\theta_{0,i}, \boldsymbol{m}_{0,i}^q \leftarrow 0, \boldsymbol{R}_{0,i} \leftarrow 0, \boldsymbol{C}_{0,i} \leftarrow 0$
end for
for $t = 1, \cdots, T$ **do**
　for 每个参数组 i **do**
　　$\boldsymbol{g}_{t,i} \leftarrow \nabla_\theta f(\theta_{t-1,i})$　// 计算梯度
　　if $i \in S_Q$ **then**
　　$\boldsymbol{m}_{t-1,i} \leftarrow \texttt{dequantize}(\boldsymbol{m}_{t-1,i}^q)$　// 反量化一阶动量
　　else
　　$\boldsymbol{m}_{t-1,i} \leftarrow \boldsymbol{m}_{t-1,i}^q$　// 未量化
　　end if
　　$\boldsymbol{m}_{t,i} \leftarrow \beta_1 \boldsymbol{m}_{t-1,i} + (1-\beta_1)\boldsymbol{g}_{t,i}$　// 更新一阶动量
　　$\boldsymbol{R}_{t,i} \leftarrow \beta_2 \boldsymbol{R}_{t-1,i} + (1-\beta_2)\mathrm{E}_{\text{行}}[\boldsymbol{g}_{t,i}^2], \quad \boldsymbol{C}_{t,i} \leftarrow \beta_2 \boldsymbol{C}_{t-1,i} + (1-\beta_2)\mathrm{E}_{\text{列}}[\boldsymbol{g}_{t,i}^2]$　// 分解二阶动量
　　$\hat{\boldsymbol{m}}_{t,i} \leftarrow \boldsymbol{m}_{t,i}/(1-\beta_1^t), \quad \boldsymbol{V}_{t,i} \leftarrow (\boldsymbol{R}_{t,i} \otimes \boldsymbol{C}_{t,i})/\mathrm{mean}(\boldsymbol{R}_{t,i})$　// 偏差校正与重构
　　$\eta_t \leftarrow \max(\epsilon_2, \mathrm{RMS}(\theta_{t-1,i}))^{-1}$　// 自适应学习率
　　按式 (B-1) 定义 α_i
　　$\theta_{t,i} \leftarrow \theta_{t-1,i} - \eta_t \left(\frac{\alpha_i \cdot \hat{\boldsymbol{m}}_{t,i}}{\sqrt{\boldsymbol{V}_{t,i}} + \epsilon_1} \right)$　// 更新参数
　　if $i \in S_Q$ **then**
　　$\boldsymbol{m}_{t,i}^q \leftarrow \texttt{quantize}(\boldsymbol{m}_{t,i})$　// 量化存储一阶动量
　　else
　　$\boldsymbol{m}_{t,i}^q \leftarrow \boldsymbol{m}_{t,i}$　// 保持全精度
　　end if
　end for
end for

B.4 一维标量量化映射

本节介绍一维 b 比特量化器中不同量化映射的构造。见图 B.2 所示示意。注意 $\mathbb{T}_b = \{0, 1, \cdots, 2^b - 1\}$。

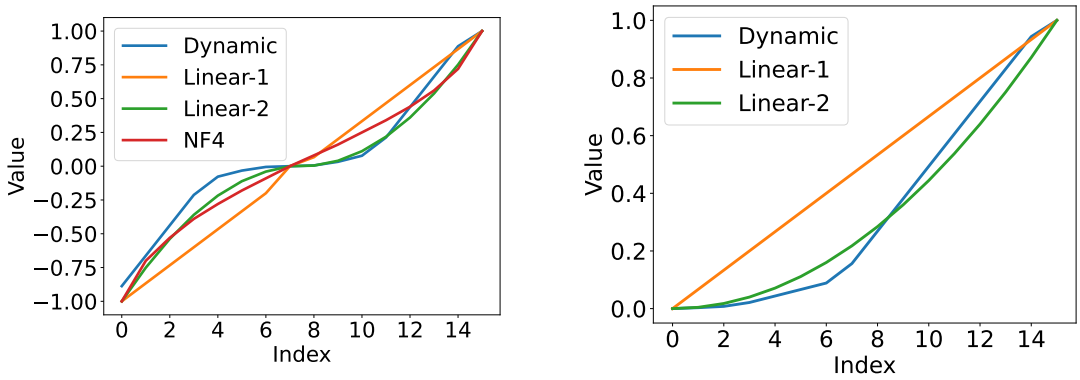
线性幂量化。针对有符号输入的线性幂（Linear-p）量化定义为

$$\mathcal{R}(j) = \begin{cases} -\left(-1 + 2j/(2^b-1)\right)^p, & j < 2^{b-1} - 1; \\ 0, & j = 2^{b-1} - 1; \\ \left(-1 + 2j/(2^b-1)\right)^p, & j > 2^{b-1} - 1, \end{cases}$$

其中 $j \in \mathbb{T}_b$ 且 $p > 0$。针对无符号输入，其定义为

$$\mathcal{R}(j) = \left(j/(2^b-1)\right)^p, \quad j \in \mathbb{T}_b, p > 0.$$

动态量化。 b 比特量化的动态量化 $\mathcal{R}$ 将 $\mathbb{T}_b$ 映射至 $\{0, 1\} \cup G$。对于有符号



(a) 有符号输入的映射　　(b) 无符号输入的映射

图 B.2　有符号和无符号输入的不同 4 比特量化映射可视化。

输入，G 是具有以下性质的数集：G 中的数形如 $\pm q_k \times 10^{-E}$，其中

$$\begin{cases} b = 2 + E + F, & E, F \in \mathbb{N}; \\ q_k = (p_k + p_{k+1})/2, & k \in \{0, \cdots, 2^F - 1\}; \\ p_j = 0.9j/2^F + 0.1, & j \in \{0, \cdots, 2^F\}. \end{cases}$$

对于无符号输入，G 是具有以下性质的数集：G 中的数形如 $q_k \times 10^{-E}$，其中

$$\begin{cases} b = 2 + E + F, & E, F \in \mathbb{N}; \\ q_k = (p_k + p_{k+1})/2, & k \in \{0, \cdots, 2^{F+1} - 1\}; \\ p_j = 0.9j/2^F + 0.1, & j \in \{0, \cdots, 2^{F+1}\}. \end{cases}$$

正态浮点。b 比特正态浮点（NormalFloat）$\mathcal{R}$ 基于分位数量化构建。range($\mathcal{R}$) 构造如下：首先对标准正态分布均匀采样 2^b 个分位数，然后将其归一化至 $[-1, 1]$。对于处理有符号输入的 4 比特正态浮点（NF4），range($\mathcal{R}$) 约为 {-1.00, -0.70, -0.53, -0.39, -0.28, -0.18, -0.09, 0.00, 0.08, 0.16, 0.25, 0.34, 0.44, 0.56, 0.72, 1.00}。

参考文献

[1] Singh A, Fry A, Perelman A, et al. OpenAI GPT-5 system card[J]. arXiv preprint arXiv:2601.03267, 2025.

[2] Yang A, Li A, Yang B, et al. Qwen3 technical report[J]. arXiv preprint arXiv:2505.09388, 2025.

[3] Liu A, Feng B, Xue B, et al. DeepSeek-V3 technical report[J]. arXiv preprint arXiv:2412.19437, 2024.

[4] 车万翔, 窦志成, 冯岩松, 等. 大模型时代的自然语言处理: 挑战、机遇与发展[J]. 中国科学: 信息科学, 2023, 53(9): 1645-1687.

[5] Wu T, Luo L, Li Y F, et al. Continual learning for large language models: A survey[J]. arXiv preprint arXiv:2402.01364, 2024.

[6] Hu E J, Shen Y, Wallis P, et al. LoRA: Low-rank adaptation of large language models[C]//Proceedings of the International Conference on Learning Representations. 2022: 12513-12525.

[7] Liu Z, Zhao C, Iandola F, et al. MobileLLM: Optimizing sub-billion parameter language models for on-device use cases[C]//Proceedings of the International Conference on Machine Learning. 2024: 32431-32454.

[8] Dettmers T, Pagnoni A, Holtzman A, et al. QLoRA: Efficient finetuning of quantized LLMs [C]//Advances in Neural Information Processing Systems. 2024: 10088-10115.

[9] Cui C, Ma Y, Cao X, et al. A survey on multimodal large language models for autonomous driving[C]//Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2024: 958-979.

[10] Chen L, Sinavski O, Hünermann J, et al. Driving with LLMs: Fusing object-level vector modality for explainable autonomous driving[C]//Proceedings of the IEEE International Conference on Robotics and Automation. 2024: 14093-14100.

[11] Xu M, Cai D, Yin W, et al. Resource-efficient algorithms and systems of foundation models: A survey[J]. ACM Computing Surveys, 2025, 57(5): 1-39.

[12] Loshchilov I, Hutter F. Decoupled weight decay regularization[C]//Proceedings of the International Conference on Learning Representations. 2019: 4061-4078.

[13] Micikevicius P, Narang S, Alben J, et al. Mixed precision training[C]//Proceedings of the International Conference on Learning Representations. 2018: 1086-1097.

[14] Wortsman M, Dettmers T, Zettlemoyer L, et al. Stable and low-precision training for large-scale vision-language models[C]//Advances in Neural Information Processing Systems. 2023: 10271-10298.

[15] Zhao J, Zhang Z, Chen B, et al. GaLore: Memory-efficient LLM training by gradient low-rank projection[C]//Proceedings of the International Conference on Machine Learning. 2024: 61121-61143.

[16] Malladi S, Gao T, Nichani E, et al. Fine-tuning language models with just forward passes[C]//Advances in Neural Information Processing Systems. 2023: 53038-53075.

[17] Li B, Chen J, Zhu J. Memory efficient optimizers with 4-bit states[C]//Advances in Neural Information Processing Systems. 2023: 15136-15171.

[18] Zhang Y, Li P, Hong J, et al. Revisiting zeroth-order optimization for memory-efficient LLM fine-tuning: A benchmark[C]//Proceedings of the International Conference on Machine Learning. 2024: 59173-59190.

[19] Liu Y, Zhu Z, Gong C, et al. Sparse MeZO: Less parameters for better performance in zeroth-order LLM fine-tuning[J]. arXiv preprint arXiv:2402.15751, 2024.

[20] Nozawa R, Poirion P L, Takeda A. Zeroth-order random subspace algorithm for non-smooth convex optimization[J]. Journal of Optimization Theory and Applications, 2025, 204(3): 53.

[21] Zhang Y, Chen C, Li Z, et al. Adam-mini: Use fewer learning rates to gain more[C]//Proceedings of the International Conference on Learning Representations. 2025: 44736-44766.

[22] Gautam T, Park Y, Zhou H, et al. Variance-reduced zeroth-order methods for fine-tuning language models[C]//Proceedings of the International Conference on Machine Learning. 2024: 15180-15208.

[23] Hao Y, Cao Y, Mou L. Flora: Low-rank adapters are secretly gradient compressors[C]//Proceedings of the International Conference on Machine Learning. 2024: 17554-17571.

[24] Zhang Z, Liu B, Shao J. Fine-tuning happens in tiny subspaces: Exploring intrinsic task-specific subspaces of pre-trained language models[C]//Proceedings of the Annual Meeting of the Association for Computational Linguistics. 2023: 1701-1713.

[25] Bucklew J, Gallagher N. Quantization schemes for bivariate Gaussian random variables[J]. IEEE Transactions on Information Theory, 1979, 25(5): 537-543.

[26] Swaszek P, Thomas J. Multidimensional spherical coordinates quantization[J]. IEEE Transactions on Information Theory, 1983, 29(4): 570-576.

[27] Nesterov Y, Spokoiny V. Random gradient-free minimization of convex functions[J]. Foundations of Computational Mathematics, 2017, 17(2): 527-566.

[28] Jiang S, Chen Q, Pan Y, et al. ZO-AdaMU optimizer: Adapting perturbation by the momentum and uncertainty in zeroth-order optimization[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2024: 18363-18371.

[29] Yue P, Yang L, Fang C, et al. Zeroth-order optimization with weak dimension dependency [C]//Proceedings of the Annual Conference on Learning Theory. 2023: 4429-4472.

[30] Yang Y, Zhen K, Banijamali E, et al. AdaZeta: Adaptive zeroth-order tensor-train adaption for memory-efficient large language models fine-tuning[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2024: 977-995.

[31] Roberts L, Royer C W. Direct search based on probabilistic descent in reduced spaces[J]. SIAM Journal on Optimization, 2023, 33(4): 3057-3082.

[32] 陈楚岩, 刘烨谞, 贾维宸, 等. 一种基于单比特通信压缩的大模型训练方法研究[J]. 自动化学报, 2025, 51(1): 1-18.

[33] Shazeer N, Stern M. Adafactor: Adaptive learning rates with sublinear memory cost[C]//Proceedings of the International Conference on Machine Learning. 2018: 4596-4604.

[34] Zhang Z, Jaiswal A, Yin L, et al. Q-GaLore: Quantized GaLore with int4 projection and layer-adaptive low-rank gradients[J]. arXiv preprint arXiv:2407.08296, 2024.

[35] Anil R, Gupta V, Koren T, et al. Memory efficient adaptive optimization[C]//Advances in Neural Information Processing Systems. 2019: 9749-9758.

[36] 王恩东, 闫瑞栋, 郭振华, 等. 分布式训练系统及其优化算法综述[J]. 计算机学报, 2024, 47(1): 1-28.

[37] Dettmers T, Lewis M, Shleifer S, et al. 8-bit optimizers via block-wise quantization[C]// Proceedings of the International Conference on Learning Representations. 2022: 7891-7909.

[38] Egiazarian V, Panferov A, Kuznedelev D, et al. Extreme compression of large language models via additive quantization[C]//Proceedings of the International Conference on Machine Learning. 2024: 12284-12303.

[39] Li J, Zhang Y, Hassan M Y, et al. CommVQ: Commutative vector quantization for KV cache compression[C]//Proceedings of the International Conference on Machine Learning. 2025: 36831-36845.

[40] Tian Z, Zhao W X, Wen J R. Irrational complex rotations empower low-bit optimizers[J]. arXiv preprint arXiv:2501.12896, 2025.

[41] 高赫然, 吴恒, 许源佳, 等. 面向深度学习训练的内存交换机制综述[J]. 软件学报, 2023, 34(12): 5862.

[42] 冯杨洋, 汪庆, 谢旻晖, 等. 从 BERT 到 ChatGPT：大模型训练中的存储系统挑战与技术发展[J]. 计算机研究与发展, 2024, 61(4): 809-823.

[43] Liu S, Kailkhura B, Chen P Y, et al. Zeroth-order stochastic variance reduction for nonconvex optimization[C]//Advances in Neural Information Processing Systems. 2018: 3731-3741.

[44] Ghadimi S, Lan G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming[J]. SIAM Journal on Optimization, 2013, 23(4): 2341-2368.

[45] Amari S i. Backpropagation and stochastic gradient descent method[J]. Neurocomputing, 1993, 5(4): 185-196.

[46] Kingma D P, Ba J. Adam: A method for stochastic optimization[C]//Proceedings of the International Conference on Learning Representations. 2015: 1-15.

[47] Xie X, Zhou P, Li H, et al. Adan: Adaptive Nesterov momentum algorithm for faster optimizing deep models[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024, 46(12): 9508-9520.

[48] Qian N. On the momentum term in gradient descent learning algorithms[J]. Neural Networks, 1999, 12(1): 145-151.

[49] Xi H, Li C, Chen J, et al. Training transformers with 4-bit integers[C]//Advances in Neural Information Processing Systems. 2023: 49146-49168.

[50] 杨春, 张睿尧, 黄泷, 等. 深度神经网络模型量化方法综述[J]. 工程科学学报, 2023, 45(10): 1613-1629.

[51] Lin H, Xu H, Wu Y, et al. DuQuant: Distributing outliers via dual transformation makes stronger quantized LLMs[C]//Advances in Neural Information Processing Systems. 2024: 87766-87800.

[52] Ashkboos S, Mohtashami A, Croci M L, et al. QuaRot: Outlier-free 4-bit inference in rotated LLMs[C]//Advances in Neural Information Processing Systems. 2024: 100213-100240.

[53] Sayood K. Introduction to data compression[M]. 5th. Morgan Kaufmann, 2017.

[54] 纪荣嵘, 林绍辉, 晁飞, 等. 深度神经网络压缩与加速综述[J]. 计算机研究与发展, 2018, 55(9): 1871-1888.

[55] Gish H, Pierce J. Asymptotically efficient quantizing[J]. IEEE Transactions on Information Theory, 1968, 14(5): 676-683.

[56] Cover T M, Thomas J A. Elements of Information Theory[M]. 2nd. Wiley-Interscience, 2006.

[57] Frantar E, Ashkboos S, Hoefler T, et al. OPTQ: Accurate quantization for generative pre-trained Transformers[C]//Proceedings of the International Conference on Learning Representations. 2023: 33657-33672.

[58] Lin J, Tang J, Tang H, et al. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration[C]//Proceedings of Machine Learning and Systems. 2024: 87-100.

[59] Liu Y, Wen J, Wang Y, et al. VPTQ: Extreme low-bit vector post-training quantization for large language models[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2024: 8181-8196.

[60] Van Baalen M, Kuzmin A, Koryakovskiy I, et al. GPTVQ: The blessing of dimensionality for LLM quantization[J]. arXiv preprint arXiv:2402.15319, 2024.

[61] Tseng A, Chee J, Sun Q, et al. Quip#: Even better LLM quantization with Hadamard incoherence and lattice codebooks[C]//Proceedings of the International Conference on Machine Learning. 2024: 48630-48656.

[62] 高晗, 田育龙, 许封元, 等. 深度学习模型压缩与加速综述[J]. 软件学报, 2021, 32(1): 68-92.

[63] Xiao G, Lin J, Seznec M, et al. Smoothquant: Accurate and efficient post-training quantization for large language models[C]//Proceedings of the International Conference on Machine Learning. 2023: 38087-38099.

[64] Wang H, Ma S, Dong L, et al. Bitnet: Scaling 1-bit transformers for large language models [J]. arXiv preprint arXiv:2310.11453, 2023.

[65] Ma S, Wang H, Ma L, et al. The era of 1-bit LLMs: All large language models are in 1.58 bits [J]. arXiv preprint arXiv:2402.17764, 2024.

[66] Wang H, Ma S, Wei F. Bitnet A4.8: 4-bit activations for 1-bit LLMs[J]. arXiv preprint arXiv:2411.04965, 2024.

[67] Horn R A, Johnson C R. Matrix Analysis[M]. 2nd. Cambridge University Press, 2012.

[68] Trefethen L N, Bau D, III. Numerical Linear Algebra[M]. 1st. Society for Industrial, 1997.

[69] Eckart C, Young G. The approximation of one matrix by another of lower rank[J]. Psychometrika, 1936, 1(3): 211-218.

[70] Kaushik P, Chaudhari S, Vaidya A, et al. The universal weight subspace hypothesis[J]. arXiv preprint arXiv:2512.05117, 2025.

[71] Lialin V, Muckatira S, Shivagunde N, et al. ReLoRA: High-rank training through low-rank updates[C]//Proceedings of the International Conference on Learning Representations. 2024: 26288-26304.

[72] Zhu H, Zhang Z, Cong W, et al. APOLLO: SGD-like memory, AdamW-level performance [C]//Proceedings of Machine Learning and Systems. 2025: 1-26.

[73] Chen Y, Zhang Y, Cao L, et al. Enhancing zeroth-order fine-tuning for language models with low-rank structures[C]//Proceedings of the International Conference on Learning Representations. 2025: 16042-16068.

[74] Aghajanyan A, Gupta S, Zettlemoyer L. Intrinsic dimensionality explains the effectiveness of language model fine-tuning[C]//Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing. 2021: 7319-7328.

[75] Zhang S, Roller S, Goyal N, et al. OPT: Open pre-trained transformer language models[J]. arXiv preprint arXiv:2205.01068, 2022.

[76] Li X L, Liang P. Prefix-tuning: Optimizing continuous prompts for generation[C]//Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing. 2021: 4582-4597.

[77] Lester B, Al-Rfou R, Constant N. The power of scale for parameter-efficient prompt tuning [C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2021: 3045-3059.

[78] Neyshabur B, Sedghi H, Zhang C. What is being transferred in transfer learning?[C]//Advances in Neural Information Processing Systems. 2020: 512-523.

[79] Liu Y, Ott M, Goyal N, et al. RoBERTa: A robustly optimized BERT pretraining approach [J]. arXiv preprint arXiv:1907.11692, 2019.

[80] Touvron H, Lavril T, Izacard G, et al. LLaMA: Open and efficient foundation language models[J]. arXiv preprint arXiv:2302.13971, 2023.

[81] Jiang A Q, Sablayrolles A, Mensch A, et al. Mistral 7B[J]. arXiv preprint arXiv:2310.06825, 2023.

[82] Wang A, Pruksachatkun Y, Nangia N, et al. SuperGLUE: A stickier benchmark for general-purpose language understanding systems[J]. arXiv preprint arXiv:1905.00537, 2019.

[83] Clark C, Lee K, Chang M W, et al. BoolQ: Exploring the surprising difficulty of natural yes/no questions[C]//Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2019: 2924-2936.

[84] De Marneffe M C, Simons M, Tonhauser J. The CommitmentBank: Investigating projection in naturally occurring discourse[C]//Proceedings of Sinn und Bedeutung. 2019: 107-124.

[85] Roemmele M, Bejan C A, Gordon A S. Choice of plausible alternatives: An evaluation of commonsense causal reasoning[C]//Proceedings of the AAAI Spring Symposium Series. 2011: 90-95.

[86] Khashabi D, Chaturvedi S, Roth M, et al. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences[C]//Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2018: 252-262.

[87] Zhang S, Liu X, Liu J, et al. ReCoRD: Bridging the gap between human and machine commonsense reading comprehension[J]. arXiv preprint arXiv:1810.12885, 2018.

[88] Dagan I, Glickman O, Magnini B. The PASCAL recognising textual entailment challenge[C]//Machine learning challenges workshop. 2005: 177-190.

[89] Pilehvar M T, Camacho-Collados J. WiC: The word-in-context dataset for evaluating context-sensitive meaning representations[C]//Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: vol. 1. 2019: 1267-1273.

[90] Levesque H, Davis E, Morgenstern L. The Winograd schema challenge[C]//Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning. 2012: 552-561.

[91] Socher R, Perelygin A, Wu J, et al. Recursive deep models for semantic compositionality over a sentiment treebank[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2013: 1631-1642.

[92] Rajpurkar P, Zhang J, Lopyrev K, et al. SQuAD: 100,000+ questions for machine comprehension of text[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2016: 2383-2392.

[93] Dua D, Wang Y, Dasigi P, et al. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs[C]//Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2019: 2368-2378.

[94] Williams A, Nangia N, Bowman S. A broad-coverage challenge corpus for sentence understanding through inference[C]//Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2018: 1112-1122.

[95] Bowman S R, Angeli G, Potts C, et al. A large annotated corpus for learning natural language inference[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2015: 632-642.

[96] Zhao Y, Dang S, Ye H, et al. Second-order fine-tuning without pain for LLMs: A Hessian informed zeroth-order optimizer[C]//Proceedings of the International Conference on Learning Representations. 2025: 22055-22079.

[97] Brown T B, Mann B, Ryder N, et al. Language models are few-shot learners[C]//Advances in Neural Information Processing Systems. 2020: 1877-1901.

[98] Kumar A, Raghunathan A, Jones R, et al. Fine-tuning can distort pretrained features and underperform out-of-distribution[C]//Proceedings of the International Conference on Learning Representations. 2022: 16358-16399.

[99] Wang S, Zhou P, Li J, et al. 4-bit shampoo for memory-efficient network training[J]. Advances in Neural Information Processing Systems, 2024: 126997-127029.

[100] Dettmers T. 8-bit approximations for parallelism in deep learning[C]//Proceedings of the International Conference on Learning Representations. 2016: 1-14.

[101] Gokaslan A, Cohen V, Pavlick E, et al. OpenWebText corpus[Z]. http://Skylion007.github.io/OpenWebTextCorpus. 2019.

[102] Raffel C, Shazeer N, Roberts A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer[J]. Journal of Machine Learning Research, 2020, 21(140): 1-67.

[103] Taori R, Gulrajani I, Zhang T, et al. Stanford Alpaca: An instruction-following LLaMA model [Z]. https://github.com/tatsu-lab/stanford_alpaca. 2023.

[104] Wang A, Singh A, Michael J, et al. GLUE: A multi-task benchmark and analysis platform for natural language understanding[C]//Proceedings of the International Conference on Learning Representations. 2019: 1786-1805.

[105] Zhou P, Xie X, Yan S. Win: Weight-decay-integrated Nesterov acceleration for adaptive gradient algorithms[C]//Proceedings of the International Conference on Learning Representations. 2023: 2694-2721.

[106] Russakovsky O, Deng J, Su H, et al. ImageNet large scale visual recognition challenge[J]. International Journal of Computer Vision, 2015, 115(3): 211-252.

[107] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[C]//Proceedings of the International Conference on Learning Representations. 2021: 611-631.

[108] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2016: 770-778.

在学期间完成的相关学术成果

研究项目

[1] 参研：小米 2025 揭榜挂帅项目，端侧大模型零阶训练，2025/07-2026/06。

学术论文

[2] **Ziming Yu**, Pan Zhou, Sike Wang, Jia Li, Mi Tian, and Hua Huang. Zeroth-order fine-tuning of LLMs in random subspaces[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2025: 4475-4485. (CCF-A 类会议)

致　谢

行文至此，落笔为终。回首数载求学之路，感慨良多。深夜实验室的灯火、论文被拒时的沮丧、灵感乍现时的欣喜，皆化作生命中最珍贵的印记。谨向所有给予支持、启发与温暖的人，致以最诚挚的谢意。

首先，感谢李嘉老师。李老师给予学生自由探索空间，让我们在试错中成长。从科研选题的斟酌，到论文写作的推敲；从投稿策略的规划，到审稿意见的解读——李老师在每一个细节中都倾注了心血。他教导我，人的进步是非线性的，需要长期积累，耐心等待质变。李老师对科研品味的执着，让我明白好的研究源于对问题更深刻的理解，这也将成为我学术生涯的永恒坐标。

衷心感谢黄华教授。黄老师为我们营造稳定的科研环境，在经费与硬件资源上给予了我们充分的支持，实验室规范有序，让我们心无旁骛投入研究。黄老师以开阔的视野引领我们提升认知，不仅是学术洞见，更是对人生方向的把握。他常告诫我们，做学术就要做有用的研究，脚踏实地解决真问题。黄老师注重逻辑培养，从论文结构的梳理到论证链条的严密，这种训练让我受益良多。

感谢周攀老师。周老师在学术前沿敏锐的洞察力令我启发良多，使我对领域动态有更清晰的认识。在科研探索、论文撰写及投稿过程中，周老师给予专业性指导，帮助我规避弯路，尤其在瓶颈期时，周老师的建议往往能帮我理清思路。这段经历收获的严谨与坚持，将成为我未来道路上的宝贵财富。

感谢实验室同门在技术攻关中的经验分享与实验调试时的协助。尤其要感谢王思科师兄，在组会研讨与日常交流中，师兄治学严谨、思维缜密，深厚的数学功底与卓越的代码能力令我钦佩。感谢师兄毫无保留的教导，教会我科研方法与处世道理，这段情谊我将铭记于心。此外，感谢乒乓球搭子们，科研压力重重，沮丧是常态，球台前的挥拍与奔跑，是解压良方。感谢朋友们，是你们让我们在埋头论文的日子里，依然能够抬头看见生活的诗意。

感谢家人。谨向我的父母和家人致以最诚挚的感谢。在我迷茫时，你们帮我分析利弊；在我焦虑时，你们不断支持鼓励。你们尊重我的每个选择，毫无保留的关爱与理解，始终是我追求理想的坚实后盾。

最后，感谢不曾放弃的自己。感谢在失败中依然选择相信，在等待中依然保持耐心，在浮躁中依然坚守初心。初心如磐，笃行致远。此去经年，愿携今日之所学，怀感恩之心，赴未来之征程。

余梓铭

2026 年 5 月

答辩委员会名单

	姓名	工作单位	职称	导师类型
主席	王立志	北京师范大学	教授	博士生导师
委员	黄石生	北京师范大学	副教授	博士生导师
	李健	北京师范大学	副教授	博士生导师
秘书	李国璋	北京师范大学	助理研究员	无