arXiv:2602.01233v1 [cs.LG] 1 Feb 2026

Lotus：通过自适应子空间切换实现高效大语言模型训练的随机低秩梯度投影

Tianhao Miao[1,*], *Zhongyuan Bao*[2,*], *Lejun Zhang*[3,*]

[1]香港浸会大学，理学院，香港，中国[2]复旦大学，数据科学学院，上海，中国[3]纽约大学，坦登工程学院，纽约，美国

摘要

大规模模型的训练效率通常通过内存消耗、训练时间和模型性能来评估。当前方法往往需要在这些指标之间权衡，因为优化其中一项通常会导致至少一项其他指标变差。如何解决这种权衡仍是算法设计中的核心挑战。GaLore 通过在低秩子空间中更新梯度，实现了内存高效训练，但对梯度进行奇异值分解（SVD）会带来相当的额外训练时间成本。本文提出**Lotus**，通过简单修改投影过程来解决这一权衡。我们提出一种衡量单位梯度位移的准则，从而能够在低秩梯度子空间之间高效切换。实验结果表明，Lotus 是效率最高的方法，使梯度和优化器状态的**训练时间减少 30%**，并**内存消耗降低 40%**。此外，它在预训练和微调任务中的表现均优于基线方法。

关键词— 高效训练、预训练、微调、大语言模型

1. 引言

随着大型语言模型（大语言模型）的迅速发展，训练所需的 GPU 内存需求大幅增加。具体而言，采用更大的批次大小和更长的序列长度等策略虽能提升模型稳定性和泛化能力，却显著增加了内存消耗。大语言模型训练期间的内存占用通常由四部分构成：模型权重、梯度、优化器状态和激活值。研究人员针对上述不同方面提出了多种节省内存的技术，包括激活重计算 [1]，混合精度训练、节省内存的优化器 [2]，以及参数高效微调（PEFT）。在这些方法中，模型权重的低秩适配（例如 Lora [3] 及其变体 [4, 5]，）因内存使用量较低且性能相当而受到关注

全参数微调。近期研究日益关注利用奇异值分解等低秩矩阵分解方法对模型权重进行处理，以提升性能 [6, 7]。

从另一个角度来看，模型权重可能本身并不具备低秩结构，学习到的新特征的表示同样如此，从而导致较差的模型性能 [8]。与分解权重矩阵相比，近期研究针对激活值 [8]，梯度以及优化器状态提出了创新方法

*同等贡献。

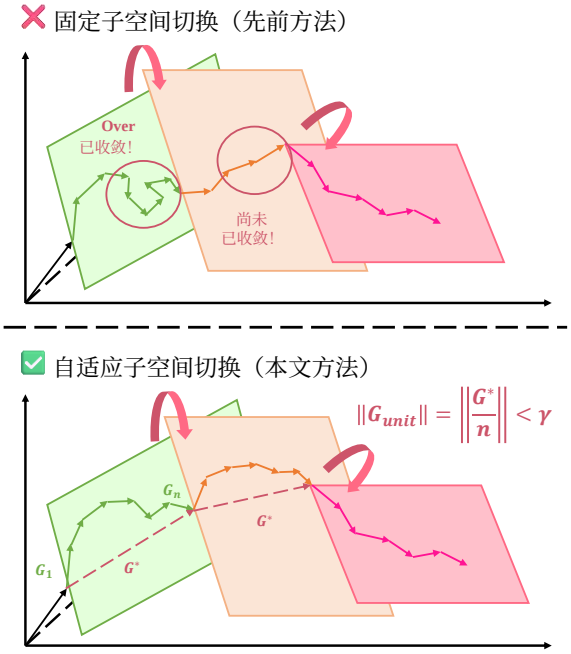


图 1。对比了采用固定切换频率的先前方法（例如 GaLore）与我们的贪心搜索策略，后者会自适应地更新子空间。 G^* 表示子空间中单位梯度的位移。当单位梯度向量 G_{unit} 的平均位移低于 γ时，将切换子空间。

采用低秩范式。 [9] 首先发现，LoRA更新可被视为对梯度进行随机投影。 [10] 改进了随机投影矩阵，并基于梯度信息初始化权重。GaLore[11] 利用低秩梯度进行子空间更新，并通过投影优化权重。若干GaLore变体 [12, 13] 在进一步缓解内存占用方面取得了进展。

然而，先前的方法通常存在固有的取舍：**内存、计算时间和性能**难以兼顾，这主要是因为它们需要频繁地在全秩空间和低秩空间之间投影梯度。GaLore 显著提升了内存效率，但由于需要频繁进行代价高昂的基于奇异值分解的子空间更新，其训练时间约为 LoRA 的 2× 倍。指定秩和更新间隔也带来了调优歧义，并增加了实际部署的难度。当

梯度因鞍点或极小值而在子空间内振荡时，就会出现问题；而固定间隔可能会过早或过晚地触发子空间变化。这些缺陷如图1所示。相比之下，近期的自适应方法（例如 [14]，提出的缩减秩策略）需要复杂的计算。

我们的方法 为了解决上述权衡与局限，我们对训练过程中的梯度矩阵分解进行了理论分析。我们发现，现有大多数方法依赖 SVD，而其计算时间和内存需求会随矩阵规模呈超线性增长。此外，固定间隔的子空间切换可能导致梯度过度旋转，因而在当前子空间尚未得到充分利用之前就发生切换。初步实验得到了一项关键观察：尽管梯度幅值在优化过程中会发生波动，**单位范数梯度往往遵循稳定的优化轨迹**。基于此，我们假设，单位范数梯度与子空间几何结构之间的对齐程度，是影响更新有效性和效率的主要因素。

因此，我们提出了更快且更节省内存的算法 **Lotus**。Lotus 通过随机投影将随机性融入矩阵分解，以近似实现分解，从而大幅降低训练内存峰值。基于受物理学启发的梯度下降观点，我们设计了一种自适应更新策略，由低秩单位梯度之间的欧几里得距离控制；当方向一致性表明收益正在减弱时，动态切换子空间。我们的贡献概括如下：

1. 我们提出了 Lotus，这是一种贪心搜索策略，基于单位梯度向量的平均位移自适应地更新子空间，以获得更好的性能。2. 与 GaLore 相比，Lotus 还能进一步节省约 40% 的梯度和优化器状态内存消耗，并减少 30% 的时间开销。在性能方面，Lotus 也超越了相关算法。

2. 相关工作

权重低秩化。 LoRA 展现了利用内在低秩结构实现大语言模型高效内存学习的潜力 [3]。[15] 通过约束权重矩阵的秩，在稠密网络中动态寻找“中奖彩票”。ReLoRA [4] 利用局部低秩更新来训练高秩网络。RandLoRA [16] 通过学习低秩、不可训练随机矩阵的线性组合来进行更新。Dora [17] 将权重分解为方向矩阵和幅值向量，用于微调。后续工作借助奇异值分解改进低秩适配器：在适配器初始化时对权重进行处理 [6], , 仅训练前 r 个奇异值 [18], 或相应的系数 [7]。Lora-Null [19] 寻找代表性激活的零空间，并使用该零空间中的预训练权重初始化 LoRA 适配器。 [20] 通过从预训练权重中选择骨架并学习一个较小的矩阵，以较低成本实现高秩更新。 [21] 将权重压缩与高效内存微调统一起来，从而实现自适应低秩权重投影。

优化器状态中的低秩。 除了将优化器状态投影到与梯度结构相对应的低秩空间外，FiRA[13] 还通过利用基于范数的缩放和梯度范数增长限制器，在低秩优化器约束下实现全秩训练。Adapprox [22] 发现了最大奇异值对 Adam 二阶动量的关键影响，并通过随机投影对其进行近似。Apollo [23] 采用低秩缩放因子，通过随机投影梯度，并在 Adam 中采用按通道更新而非逐元素更新，以降低优化器内存开销。Alice [24] 利用费舍尔信息矩阵近似 Adam，以节省内存。COSMOS [25] 结合了主导梯度子空间上的完整二阶方法，以及残差方向上的 Newton-Schulz近似。

梯度低秩化。 Flora 使用随机矩阵对梯度进行下投影，并对低秩梯度进行上投影以用于优化；GoRA 则通过自适应随机投影矩阵和梯度相关初始化 [9, 10]改进了 Flora。GaLore [11]基于奇异值分解对梯度进行分解，并将低秩更新投影回原空间后优化全秩权重；其变体尝试通过随机投影或对低秩梯度进行逐层自适应来减少内存占用并降低复杂度 [12]。Adarankgrad[14] 在训练过程中基于较低的内在秩采用自适应子空间。 [26] 利用满足克罗内克分解等价条件的线性梯度变换，将其作为线性适配器。 [27] 在子空间学习中使用在线 PCA 更新投影矩阵。 [28] 根据梯度得出的重要性分数动态剪枝和扩展秩。 [8]仅使用关键行和列来近似矩阵乘法，并改进反向传播。GaLore 2 和 Galore+ 加速了投影过程，但未提供证明和时间复杂度分析 [29]。

3. 方法

3.1. 自适应子空间切换

使用最新的全秩梯度更新正交投影算子，可使低秩基重新对齐其当前的主导方向，让前 r 个奇异向量重新获得此前漂移到陈旧子空间之外的能量；因此，压缩梯度的弗罗贝尼乌斯范数会“跃升回去”。然而，由于梯度的不同谱分量漂移速度各异，固定的更新频率既会在已稳定的方向上浪费计算，又会让快速变化的方向在下次更新前流失能量。自适应切换计划根据子空间漂移调节这一过程的速度，从而解决这种失衡。

为量化这种计划能够保留多少位移，考虑这样一种理想情形：每一步投影梯度都恰好指向同一方向；那么，经过 k 步后的累积位移为

$$D_{\text{ideal}} = \left\| \sum_{i=0}^{k-1} -\alpha \hat{g}_{t-i} \right\|_2 = \alpha \left\| \sum_{i=0}^{k-1} \hat{g}_{t-i} \right\|_2 \tag{1}$$

在“对齐最佳”的情形下，所有$\hat{g}$ 都相互平行且范数为 1，$D_{\text{ideal}} \approx k\alpha$，其中 α 为学习率。那么，实际位移为：

$$D_{\text{actual}} = \left\| \sum_{i=0}^{k-1} \Delta w_{t-i} \right\|_2 = \left\| \sum_{i=0}^{k-1} -\alpha P_k \hat{g}_{t-i} \right\|_2 \tag{2}$$

接下来，我们定义路径效率比：

$$\rho_t = \frac{D_{\text{actual}}}{D_{\text{ideal}}} = \frac{\left\| \sum_{i=0}^{k-1} P_k \hat{g}_{t-i} \right\|_2}{\left\| \sum_{i=0}^{k-1} \hat{g}_{t-i} \right\|_2} \in [0, 1] \tag{3}$$

当 $\rho_t \approx 1$时，梯度仍局限于狭窄的方向锥内，这表明当前子空间 P_k 足以代表优化过程。若 $\rho_t \ll 1$，连续步骤之间会发生显著抵消，这意味着梯度

表 1。我们通过在 C4 数据集上对不同规模的 LLaMA 模型进行预训练，比较若干低秩训练算法与 Lotus 的性能。此处，r 表示低秩分解的秩，d_{model} 表示每种模型规模下的隐藏状态维度。

方法	60M	130M	350M	1B
全秩	34.06(0.36G)	25.08(0.76G)	18.80(2.06G)	15.56(7.80G)
GaLore	34.88(0.24G)	25.36(0.52G)	18.95(1.22G)	15.64(4.38G)
低秩	78.18(0.26G)	45.51(0.54G)	37.41(1.11G)	142.53(3.62G)
LoRA	34.99(0.36G)	33.92(0.80G)	25.58(1.76G)	19.21(6.17G)
ReLoRA	37.04(0.36G)	29.37(0.80G)	29.08(1.76G)	18.33(6.17G)
AdaRankGrad	34.24(0.21G)	25.22(0.50G)	18.91(1.11G)	**14.71(3.62G)**
Lotus	**33.75(0.23G)**	**24.87(0.51G)**	**18.91(1.19G)**	15.33(4.20G)
r / d_{model}	128 / 256	256 / 768	256 / 1024	512 / 2048
训练词元	1.1B	2.2B	6.4B	13.1B

具有显著的方向变化，或超出子空间 P_k的张成范围。当$\rho_t < \gamma$ 且 $t - t_{\text{last}} \geq T_{\min}$时，Lotus 会自适应地切换子空间，阈值为 $\gamma \in (0,1)$。需要注意的是，我们设置了最小间隔条件，施加约束 $t - t_{\text{last}} \geq T_{\min}$，以防止在优化初期的噪声阶段过于频繁地切换子空间。特别是，$k = 1$ 意味着 ρ_t 简化为单步位移-梯度比。

引理 3.1（单步投影下降）如果 $\rho_t \geq \rho$，且损失函数具有标准的 L- 光滑性。我们可以将 L- 光滑函数的标准上界应用于子空间投影更新规则，得到：

$$\mathcal{L}(w_{t+1}) \leq \mathcal{L}(w_t) - \alpha\rho^2 \|g_t\|_2^2 + \frac{1}{2}\alpha^2 L \|g_t\|_2^2 \tag{4}$$

其中，$w_t \in \mathbb{R}^d$ 是第 t 次迭代中的参数向量，$g_t = \nabla\mathcal{L}(w_t)$ 是第 t 步损失函数的梯度。

算法 1 Lotus 算法

输入： 全秩梯度 $G_F \in \mathbb{R}^{m\times n}$；平均单位梯度位移阈值 γ；验证间隔 η
初始化： 投影计数 $T \leftarrow 0$
如果 初始化或子空间切换**则**
  $O_G \leftarrow$ E高效 L低秩-R P投影(G_F)
  $G_{\text{init}} \leftarrow O_G \cdot G_F$
  $d_{\text{init}} \leftarrow$N归一化$(G_{\text{init}})$
  $T \leftarrow 1$
结束如果
$G_{\text{cur}} \leftarrow O_G \cdot G_F$
$d_{\text{cur}} \leftarrow$正规化$(G_{\text{cur}})$
$T \leftarrow T + 1$
如果 T模 $\eta = 0$ **那么**
  $\Delta d \leftarrow d_{\text{cur}} - d_{\text{init}}$
  $\|\bar{d}\| \leftarrow \|\Delta d\|/T$ （平均位移）
  如果 $\|d\| < \gamma$ **那么**
    触发子空间更新
  结束如果
结束如果

定理 3.2（采用自适应策略可加快收敛）令 N_{fix} 和N_{ada} 分别表示固定步长策略和自适应步长策略达到梯度容差条件 $\sum_{t=0}^{N-1} G_t \leq \epsilon$所需的迭代次数，其中 $G_t = \|g_t\|_2^2$ 且步长约束为 $\alpha < 2\rho_{fix}^2/L$。则有以下不等式成立：

$$N_{\text{ada}} \leq \frac{c_{\text{fix}}}{c_{\text{ada}}}\frac{k}{T}N_{\text{fix}} < N_{\text{fix}} \tag{5}$$

该结果表明，Lotus 的自适应子空间切换在达到相同收敛准则的同时，所需迭代次数严格少于固定策略，凸显了其高效性。

3.2. Lotus 算法

本节介绍 Lotus，这是一种旨在同时加速计算并减少内存使用量的训练策略。Lotus 使用基于幂迭代的随机奇异值分解，显著加速梯度投影步骤。此外，它还采用了一种新颖且更灵活的路径效率切换策略：我们定义累积梯度位移的路径效率 ρ_t；当 ρ_t 低于预设阈值，且距离上次切换的时间超过 $T_{\min}$时，算法便会触发子空间重新计算。具体细节见算法 1。与固定间隔子空间切换相比，该机制可确保自适应策略以更少的迭代达到相同的梯度阈值，从而实现更快的收敛。验证间隔 η应设为 25–100 步，阈值 γ 应设为 0.005–0.02，以避免更新过于频繁或过少。

4. 实验

实现细节 我们使用 GaLore [11]作为模型训练与评估的代码库。实验中涉及的所有模型架构均与 GaLore 保持一致。训练和验证所用的数据格式为 BF16。我们对实验所需的超参数进行调优，以获得最佳性能。

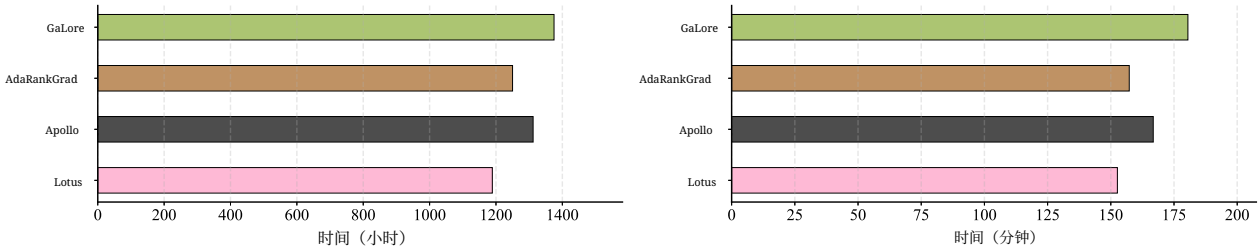
4.1. 预训练与微调

为评估 Lotus 的有效性，我们在 C4 数据集上对不同规模的 LLaMA 模型进行预训练。C4 数据集是 Common Crawl 语料库经过广泛使用的清洗版本。实验设置遵循 GaLore，并以困惑度作为主要指标。相应的实验结果见表 1。此处所有预训练实验均使用 NVIDIA H100 GPU。

我们在 8 个 GLUE 任务上对 RoBERTa-Base 模型进行微调，并将结果与全秩微调、LoRA、GaLore 和 AdaRankGrad 进行比较，结果见表2。我们报告 MNLI 的总体（匹配和不匹配）准确率、CoLA 的 Matthew 相关系数、STS-B 的皮尔逊相关系数、MRPC 的 F1 分数，以及其他任务的准确率。Lotus 在大多数任务上优于以往方法，在降低内存开销的同时取得了更高的平均分。所有微调实验均使用 NVIDIA RTX 4090 GPU。

表 2。在 GLUE 基准上评估 Lotus。我们将 Lotus 与各种内存高效训练方法进行比较，并报告平均指标。在所有微调任务中，我们将阈值设为 γ=0.01，验证间隔设为 η=50，作为基线。

方法	内存	CoLA	STS-B	MRPC	RTE	SST2	MNLI	QNLI	QQP	Avg
全量微调	747M	62.24	90.92	91.30	79.42	94.57	87.18	86.28	92.28	86.28
LoRA（秩=4）	257M	61.38	90.57	91.07	78.70	92.89	86.82	92.18	**91.29**	85.61
GaLore（秩=4）	253M	60.35	90.73	92.25	79.42	94.04	87.00	92.24	91.06	85.89
Apollo（秩=4）	251M	59.75	89.89	90.74	74.00	93.11	85.62	91.70	89.38	83.87
AdaRankGrad（秩=4）	202M	61.40	**90.97**	92.60	81.23	**94.80**	86.60	92.50	90.40	86.31
Lotus（秩=4）	251M	**64.67**	90.79	**93.14**	**83.39**	94.72	**87.47**	**93.00**	91.06	**87.28**
LoRA（秩=8）	264M	61.83	90.80	91.90	79.06	93.46	86.94	92.25	**91.22**	85.93
GaLore（秩=8）	257M	60.06	90.82	92.01	79.78	94.38	87.17	92.20	91.11	85.94
Apollo（秩=8）	251M	60.63	90.08	90.51	74.36	93.34	85.90	92.20	88.98	84.50
AdaRankGrad（秩=8）	237M	62.00	90.89	93.20	81.23	94.80	86.50	92.60	89.70	86.36
Lotus（秩=8）	254M	**63.44**	**91.06**	**93.35**	**81.58**	**94.95**	**87.32**	**93.11**	91.15	**86.99**



（a）各方法在 C4 上预训练的预计时间　　（b）各方法在 GLUE 上微调的实测时间

图 2。我们展示了 GaLore、AdaRankGrad、Apollo 和 Lotus 在预训练及微调任务中的训练时间消耗对比。(a) 显示了在 C4 上预训练 LLaMA 类型 3B 模型的预计完成时间。(b) 显示了 8 个 GLUE 任务的平均耗时。就计算时间效率而言，Lotus 是最有效的方法。（**越少越好**）

4.2. 训练时间效率与性能比较

我们按照 GaLore 的方法，并参考图2所示微调任务的平均耗时，比较了在单块 NVIDIA RTX 4090 GPU 上使用 8 位优化器和逐层权重更新对 LLaMA 类 3B 模型进行预训练的预计完成时间（ETA）。与 GaLore、AdaRankGrad 和 Apollo 相比，我们的方法显著节省了时间。此外，如表3所示，Lotus 的子空间更新频率高于 GaLore。这些结果表明，就计算效率而言，Lotus 是最简单且最有效的方法。我们还量化了随机 SVD（rSVD）和自适应子空间切换（AdaSS）的贡献，如表4所示。结果表明，在相同秩下，rSVD 的表现与精确 SVD 变体十分接近，而大部分收益来自我们的自适应子空间更新。

表 3。GaLore 与 Lotus 在 GLUE 基准上的更新频率对比。

方法	子空间账户	子空间切换频率
GaLore（秩=4）	3536	1.6
Lotus（秩=4）	**11614**	**6.5** ↑306%
GaLore（秩=8）	3544	1.6
Lotus（秩=8）	**11736**	**6.3** ↑320%

表4。对奇异值分解、rSVD 以及采用 AdaSS 的 rSVD 进行组件级评估，展示各组件如何影响结果。

Rank	rSVD	AdaSS	Avg
4			85.89
4	✓		85.89
4	✓	✓	**87.28**
8			85.94
8	✓		86.07
8	✓	✓	**86.99**

5. 结论

我们提出 Lotus，这是一种自适应低秩更新算法，可实现**30%**的训练加速和**40%** 的内存节省，同时比全秩预训练、微调和近期的内存高效方法收敛更好。Lotus 基于这样一个假设：单位梯度位移与子空间几何之间的对齐决定了更新效率。受此假设启发，Lotus 以物理学为灵感看待梯度下降：追踪低秩单位梯度之间的欧几里得距离，并在方向一致性下降时动态切换子空间。我们通过理论分析和大量实验表明，Lotus 能在大语言模型训练中有效平衡内存、计算时间与性能，为高效的大模型优化提供了实用工具。

6. 参考文献

[1]Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro, “减少大型 Transformer 模型中的激活重计算”，机器学习与系统论文集，第 5 卷，第 341–353 页，2023 年。[2] Noam Shazeer 和 Mitchell Stern，“Adafactor：具有次线性内存开销的自适应学习率”，载于国际机器学习会议。PMLR，2018 年，第 4596–4604 页。[3] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen 等，“Lora：大型语言模型的低秩适配。”，*ICLR*，第 1 卷，第 2 期，第 3 页，2022 年。[4] Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira 和 Anna Rumshisky，“Relora：通过低秩更新进行高秩训练”，*arXiv* 预印本 *arXiv:2307.05695*，2023 年。[5] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen 和 Tuo Zhao，“Adalora：参数高效微调的自适应预算分配”，*arXiv* 预印本 *arXiv:2303.10512*，2023 年。[6] Fanxu Meng, Zhaohui Wang 和 Muhan Zhang，“Pissa：大型语言模型的主奇异值与奇异向量适配”，神经信息处理系统进展，第 37 卷，第 121038–121072 页，2024年。[7] Vijay Chandra Lingam, Atula Neerkaje, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Eunsol Choi, Alex Dimakis, Aleksandar Bojchevski 和 Sujay Sanghavi，“Svft：利用奇异向量进行参数高效微调”，神经信息处理系统进展，第 37 卷，第 41425–41446 页，2024年。[8] Guanduo Chen, Yutong He, Yipeng Hu, Kun Yuan 和 Binhang Yuan，“Ce-lora：面向语言模型的计算高效 LoRA 微调”，*arXiv* 预印本 *arXiv:2502.01378*，2025年。[9]Yongchang Hao, Yanshuai Cao 和 Lili Mou，“Flora：低秩适配器其实是梯度压缩器”，*arXiv* 预印本 *arXiv:2402.03293*，2024年。[10] Haonan He, Peng Ye, Yuchen Ren, Yuan Yuan 和 Lei Chen，“Gora：梯度驱动的自适应低秩适配”，*arXiv* 预印本 *arXiv:2502.12171*，2025年。[11] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar 和 Yuandong Tian，“Galore：通过梯度低秩投影实现内存高效的大语言模型训练”，*arXiv* 预印本 *arXiv:2403.03507*，2024年。[12] Weihao Huang, Zhenyu Zhang, Yushun Zhang, Zhi-Quan Luo, Ruoyu Sun 和 Zhangyang Wang，“Galore-mini：以更少的学习率进行低秩梯度学习”，载于*NeurIPS 2024* 现代机器学习微调研讨会：原理与可扩展性。[13] Xi Chen, Kaituo Feng, Changsheng Li, Xunhao Lai, Xiangyu Yue, Ye Yuan 和 Guoren Wang，“Fira：我们能否在低秩约束下实现大语言模型的全秩训练？”，*arXiv* 预印本 *arXiv:2410.01623*，2024年。[14] Yehonathan Refael, Jonathan Svirsky, Boris Shustin, Wasim Huleihel 和 Ofir Lindenbaum，“Adarankgrad：面向内存高效大语言模型训练与微调的自适应梯度秩与矩”，*arXiv* 预印本 *arXiv:2410.17881*，2024年。

[15] Steffen Schotth¨ofer、Emanuele Zangrando、Jonas Kusch、Gian- luca Ceruti 和 Francesco Tudisco，“低秩彩票票据：通过矩阵微分方程寻找高效的低秩神经网络”，神经信息处理系统进展，第 35 卷，第 20051–20063 页，2022年。[16] Paul Albert、Frederic Z Zhang、Hemanth Saratchandran、Cris- tian Rodriguez-Opazo、Anton van den Hengel 和 Ehsan Ab- basnejad，“RandLoRA：大型模型的全秩参数高效微调”，*arXiv* 预印本 *arXiv:2502.00987*，2025年。[17]Shih-Yang Liu、Chien-Yi Wang、Hongxu Yin、Pavlo Molchanov、Yu-Chiang Frank Wang、Kwang-Ting Cheng 和 Min-Hung Chen，“Dora：权重分解的低秩适配”，*arXiv* 预印本 *arXiv:2402.09353*，2024年。[18] Chengwei Sun、Jiwei Wei、Yujia Wu、Yiming Shi、Shiyuan He、Zeyu Ma、Ning Xie 和 Yang Yang，“Svfit：使用奇异值对大型预训练模型进行参数高效微调”，*arXiv* 预印本 *arXiv:2409.05926*，2024年。[19] Pengwei Tang、Yong Liu、Dongjie Zhang、Xing Wu 和 Debing Zhang，“Lora-Null：通过零空间实现大型语言模型的低秩适配”，*arXiv* 预印本 *arXiv:2503 02659*，2025。[20] Qibin Wang、Xiaolin Hu、Weikai Xu、Wei Liu、Jian Luan 和 Bin Wang，“Pmss：用于大语言模型微调的预训练矩阵骨架选择”，*arXiv* 预印本 *arXiv:2409.16722*，2024年。[21] Ajay Jaiswal、Lu Yin、Zhenyu Zhang、Shiwei Liu、Jiawei Zhao、Yuandong Tian 和 Zhangyang Wang，“从 GaLore 到 Welore：低秩权重如何从低秩梯度中非均匀地产生”，*arXiv* 预印本 *arXiv:2407.11239*，2024年。[22] Pengxiang Zhao、Ping Li、Yingjie Gu、Yi Zheng、Stephan Ludger K¨olker、Zhefeng Wang 和 Xiaoming Yuan，“Adapprox：通过随机低秩矩阵实现 Adam 优化中的自适应近似”，*arXiv* 预印本 *arXiv:2403.14958*，2024年。[23] Hanqing Zhu、Zhenyu Zhang、Wenyan Cong、Xi Liu、Sem Park、Vikas Chandra、Bo Long、David P Pan、Zhangyang Wang 和 Jin won Lee，“Apollo：类似 SGD 的内存占用，达到 AdamW 级别的性能”，*arXiv* 预印本 *arXiv:2412.05270*，2024年。[24] Wenbo Gong、Meyer Scetbon、Chao Ma 和 Edward Meeds，“通过带有低秩扩展的结构化费舍尔近似实现大语言模型的高效优化器设计”，*arXiv* 预印本 *arXiv:2502.07752*，2025年。[25] Liming Liu、Zhenghao Xu、Zixuan Zhang、Hao Kang、Zichong Li、Chen Liang、Weizhu Chen 和 Tuo Zhao，“COSMOS：用于大语言模型内存高效训练的混合自适应优化器”，*arXiv* 预印本 *arXiv:2502.17410*，2025年。[26] Lucas Torroba-Hennigen、Hunter Lang、Han Guo 和 Yoon Kim，“论梯度变换与适配器之间的对偶性”，*arXiv* 预印本 *arXiv:2502.13811*，2025年。[27]Kaizhao Liang、Bo Liu、Lizhang Chen 和 Qiang Liu，“通过在线子空间下降实现内存高效的大语言模型训练”，*arXiv* 预印本 *arXiv:2408.12857*，2024年。[28] Huandong Chang、Zicheng Ma、Mingyuan Ma、Zhenting Qi、Andrew Sabot、Hong Jiang 和 HT Kung，“Elalora：用于高效模型微调的弹性且可学习的低秩适配”，*arXiv* 预印本 *arXiv:2504.00254*，2025年。[29] DiJia Su、Andrew Gu、Jane Xu、Yuandong Tian 和 Jiawei Zhao，“GaLore 2：通过梯度低秩投影进行大规模大语言模型预训练”，*arXiv* 预印本 *arXiv:2504.20437*，2025年。