LoSARE: Low-Rank Optimization for Training LLMs with Adaptive Subspace Tracking

Anonymous authors
Paper under double-blind review

Abstract

Low-rank optimization has emerged as an effective approach to reducing the memory cost for training Large Language Models (LLMs). Existing methods are still insufficient in adapting to evolving gradient subspaces and recovering gradient information discarded by low-rank projection. We present LoSARE, a memory-efficient low-rank optimization method that addresses both issues through adaptive subspace tracking and energy-aware residual enhancement. LoSARE incrementally updates the projection subspace according to the current gradient geometry. The gradient subspace is continually tracked with low-complexity QR decomposition, avoiding the need for costly SVD re-computation in GaLore. It further enhances the discarded gradient residual according to its relative energy, while low-rank optimizer states and stable norm-growth control are still retained. Extensive experiments on LLaMA models demonstrate that LoSARE consistently outperforms GaLore and Fira in memory-efficient pre-training and fine-tuning, under the constraints of preserving their low memory footprint. Notably, on the LLaMA-7B pre-training task, LoSARE with projection rank 64 achieves comparable or better performance than GaLore with ranks 512 and 1024, using $8\times$ and $16\times$ smaller ranks, respectively. At the same time, LoSARE achieves $1.12\times$ and $1.22\times$ faster training speed than GaLore with rank 1024 and Fira with rank 64, respectively. Furthermore, LoSARE maintains strong performance under substantially lower rank constraints and exhibits more effective late-stage optimization. These results demonstrate that our method provides an effective balance among performance, memory efficiency, and training cost for LLM training.

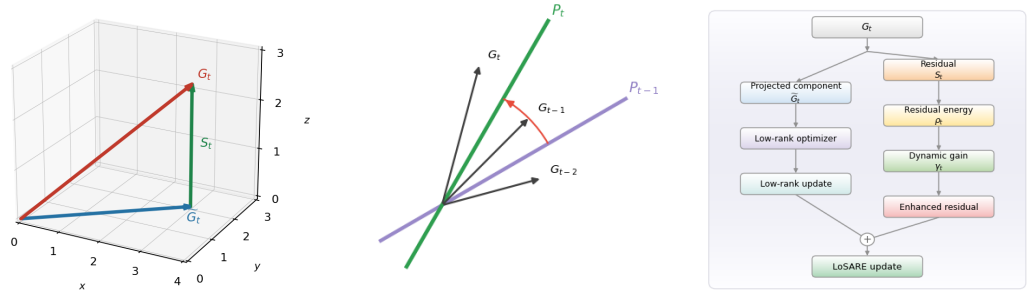
1 Introduction

The rapid growth of large language models (LLMs) has made training increasingly memory-intensive. In particular, AdamW maintains first- and second-order moment states for every parameter, creating substantial memory overhead beyond model parameters, gradients, and activations. For example, pre-training LLaMA-7B requires at least 58 GB of memory under BF16 (Zhao et al., 2024). This memory bottleneck motivates optimization methods that reduce optimizer-state memory without restricting the model parameters themselves.

Low-rank optimization provides a promising solution by exploiting the low-dimensional structure of neural network updates. LoRA (Hu et al., 2022) restricts parameter updates to low-rank adaptors, while ReLoRA (Lialin et al., 2024) extends low-rank updates to pre-training through periodic merging. GaLore (Zhao et al., 2024) instead preserves full-rank parameters while projecting gradients into a low-rank subspace, substantially reducing optimizer-state memory. As illustrated in Fig. 1a, a full-rank gradient G_t is decomposed into a projected component $\widetilde{G}_t$ within the tracked subspace and an out-of-subspace residual S_t. The projected component is processed by the low-rank optimizer, while the residual is discarded by standard low-rank projection of GaLore. Fira (Chen et al., 2025) further recovers information from this residual through norm-based scaling, improving the utilization of full-rank gradient information.

However, two challenges remain. First, the gradient directions can evolve throughout training, as illustrated in Fig. 1b. A fixed or periodically updated subspace may not fully track these changes, while frequent SVD-based basis updates introduce additional computational cost and abrupt changes. This motivates an optimizer that simultaneously tracks the evolving gradient sub-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) **Gradient projection.** A full-rank gradient G_t is decomposed into the projected component $\widetilde{G}_t$ and the out-of-subspace residual S_t.

(b) **Evolving gradient subspace.** Continual subspace adaptation updates the tracked basis from P_{t-1} to P_t in response to the current gradient.

(c) **Energy-aware residual enhancement.** The residual energy ρ_t determines a dynamic gain γ_t for enhancing informative out-of-subspace components.

Figure 1: **Motivation and overview of LoSARE.** (a) Gradient projection decomposes the full-rank gradient into a low-rank projected component and an out-of-subspace residual. (b) Continual subspace adaptation tracks the evolving gradient geometry using the previous basis and a newly estimated basis. (c) Energy-aware residual enhancement adaptively strengthens the residual according to its relative energy before combining it with the low-rank update.

space. Second, in existing methods the recovery strength do not explicitly depend on the amount of discarded gradient energy. This motivates an energy-aware mechanism that adjusts the residual contribution using an online estimate of its relative energy.

We propose **LoSARE** (**Lo**w-rank **S**ubspace **A**daptation with **R**esidual **E**nhancement), a memory-efficient optimizer that combines *adaptive subspace tracking* with *energy-aware residual enhancement*. LoSARE replaces periodic SVD-based projection updates with gradient-driven QR decomposition, continually adapting the tracked subspace from the previous basis P_{t-1} toward a newly estimated basis. The adapted basis is then re-orthogonalized, enabling efficient tracking of evolving gradient directions without repeated full SVD computations. LoSARE further measures the relative energy of the out-of-subspace residual, as shown in Fig. 1c. It is used to compute a bounded, dynamically adjusted scaling factor that strengthens informative residual components, while retaining the column-wise scaling and norm-growth control of Fira. Together, these two components improve the utilization of full-rank gradient information while maintaining low-rank optimizer states.

Our contributions are summarized as follows:

- We propose **LoSARE**, a memory-efficient low-rank optimizer that combines gradient-driven QR-based subspace adaptation with energy-aware residual enhancement, enabling full-rank parameter updates while maintaining low-rank optimizer states.
- Extensive experiments on LLaMA pre-training and fine-tuning demonstrate comparable or better performance than GaLore and Fira, with substantially lower ranks and reduced computational cost in the 7B setting.
- We conduct ablation studies on the residual enhancement coefficient and subspace adaptation rate, characterizing their effects on optimization performance and identifying stable operating regimes for LoSARE.

2 Related Work

Low-Rank Adaptation and Training. Low-rank adaptation (LoRA) (Hu et al., 2022) reduces fine-tuning memory by restricting updates to low-rank subspaces. Subsequent works improve its flexibility or efficiency through adaptive ranks (Zhang et al., 2023b), quantization (Dettmers et al., 2023), and alternative parameterizations (Zhang et al., 2023a). ReLoRA (Lialin et al., 2024) extends low-rank updates to pre-training by periodically merging them into full-rank weights, while Flora

(Hao et al., 2024) uses random projections to obtain high-rank updates. LoQT (Loeschcke et al., 2024) further combines low-rank adaptation with quantized pre-training.

Gradient Low-Rank Optimization. Another direction exploits the low-rank structure of neural network gradients (Gur-Ari et al., 2018; Cosson et al., 2023; Gooneratne et al., 2020) and related compression techniques (Wang et al., 2018; Vogels et al., 2020; Modoranu et al., 2024). GaLore (Zhao et al., 2024) projects full-rank gradients into a low-rank subspace and maintains optimizer states only in that space, enabling full-parameter training with substantially lower memory. Fira (Chen et al., 2025) further recovers discarded components through norm-based scaling and a norm-growth limiter. LDAdam (Robert et al., 2025) additionally addresses optimizer-state transitions across changing projection spaces. Subsequent methods explore alternative low-rank optimization strategies, including GoLore (He et al., 2025), Q-GaLore (Zhang et al., 2024), and GaLore-mini (Huang et al., 2024).

Geometric Subspace Updates. Recent works study how low-dimensional subspaces evolve during training. InGaLore (Meng et al., 2026) uses incremental SVD for continuous gradient basis tracking, while SubTrack++ (Rajabi et al., 2025) formulates subspace updates on the Grassmannian and introduces projection-aware optimization for changing subspaces. LDAdam (Robert et al., 2025) also considers changing low-rank spaces by adapting optimizer statistics across projection transitions. LORO (Mo et al., 2025) applies Riemannian optimization to fixed-rank representations. In contrast, LoSARE adopts a warm-started Gram–QR update with basis interpolation, avoiding repeated full SVD reconstruction.

3 PRELIMINARIES

We briefly introduce the notation and residual-recovery mechanism needed for LoSARE. We consider a weight matrix $W_t \in \mathbb{R}^{m\times n}$ and its full-rank gradient $G_t = \nabla_W \mathcal{L}_t(W_t) \in \mathbb{R}^{m\times n}$. Following GaLore (Zhao et al., 2024), we assume $m \le n$ and use a rank-r orthonormal basis $P_t \in \mathbb{R}^{m\times r}$. [1]

The gradient is projected into the low-rank space as

$$R_t = P_t^\top G_t, \tag{1}$$

where $R_t \in \mathbb{R}^{r\times n}$. The first- and second-order optimizer states are maintained in this low-rank space. With low-rank moments M_t and V_t, we denote the normalized update by

$$N_t = \frac{M_t}{\sqrt{V_t} + \epsilon}. \tag{2}$$

The projection separates the gradient into a retained component and a discarded residual. The residual used throughout this paper is

$$S_t = G_t - P_t R_t. \tag{3}$$

Following Fira (Chen et al., 2025), the residual is recovered using column-wise norm-based scaling:

$$\phi_t^{(i)} = \frac{\|N_{t,:,i}\|}{\|R_{t,:,i}\| + \epsilon}, \qquad \widehat{S}_{t,:,i} = \phi_t^{(i)} S_{t,:,i}, \qquad i = 1, \dots, n. \tag{4}$$

This mechanism recovers discarded gradient information without maintaining full-rank optimizer states. We also retain the norm-growth limiter of Fira (Chen et al., 2025), which restricts the step-to-step growth of the recovered residual:

$$q_t = \frac{\|\widehat{S}_t\|}{\|\widehat{S}_{t-1}\| + \epsilon}, \qquad \widehat{S}_t \leftarrow \frac{\widehat{S}_t}{\max(q_t, c)/c}, \qquad c = 1.01. \tag{5}$$

The above formulation provides the basic low-rank projection and residual-recovery components used by LoSARE. Additional background on GaLore projection, gradient decomposition, and Fira's residual recovery is provided in Appendix C.

[1] Following the convention of Zhao et al. (2024), we assume $m \le n$; the complementary case $m > n$ can be handled by an analogous projection along the row dimension, yielding $R_t = G_t Q_t$ where $Q_t \in \mathbb{R}^{n\times r}$.

4 METHOD

We propose **LoSARE** (**Lo**w-rank **S**ubspace **A**daptation with **R**esidual **E**nhancement), a memory-efficient optimizer that improves low-rank gradient optimization in two aspects. First, it continually adapts the projection basis using the current gradient, avoiding frequent full SVD recomputation. Second, it measures the energy of the discarded residual and dynamically adjusts its recovery strength, allowing more informative out-of-subspace components to contribute to the update. The detailed theoretical analysis is provided in Appendix G.

4.1 ADAPTIVE SUBSPACE TRACKING

The projection basis in conventional low-rank optimizers is typically obtained from an SVD of the gradient and kept fixed until the next basis update (Zhao et al., 2024; Chen et al., 2025). Since the dominant gradient subspace can evolve during training, a stale basis may become increasingly misaligned with the current gradient. LoSARE therefore updates the basis incrementally using the current gradient.

For the current gradient $G_t \in \mathbb{R}^{m\times n}$, let

$$G_t = U_t \Sigma_t V_t^\top. \tag{6}$$

For the left projection with $m \le n$, the desired rank-r basis is the subspace spanned by the leading r left singular vectors of G_t, which correspond to its largest singular values. Rather than explicitly computing them by SVD, LoSARE exploits

$$G_t G_t^\top = U_t \Sigma_t^2 U_t^\top, \tag{7}$$

which weights the contribution of each singular direction according to its squared singular value. This is the key observation underlying our incremental subspace update and is closely related to subspace iteration methods that use repeated applications of $A^\top A$ or $AA^\top$ to approach dominant singular subspaces in Kaloorazi et al. (2024).

Let $P_{t-1} \in \mathbb{R}^{m\times r}$ be the orthonormal basis from the previous iteration. We apply the Gram operator to this warm-started basis:

$$T_t = G_t^\top P_{t-1}, \qquad Y_t = G_t T_t = G_t G_t^\top P_{t-1}. \tag{8}$$

The Gram operator emphasizes components aligned with the dominant left singular subspace of the current gradient, while using P_{t-1} as a warm start to preserve continuity with the previous subspace. The resulting basis is obtained by reduced QR decomposition:

$$P_t^{\text{new}} = \mathrm{qf}(Y_t), \tag{9}$$

where $\mathrm{qf}(\cdot)$ denotes the orthonormal basis from reduced QR decomposition. The two matrix–basis products require $O(mnr)$ operations and the QR step requires $O(mr^2)$, giving a total basis-update cost of $O(mnr + mr^2)$, compared with $O(m^2 n)$ for a full SVD when $m \le n$. Since $r \ll m$, the proposed update avoids repeated full SVD computation, which is costly.

To prevent abrupt changes in the projection basis, LoSARE smoothly interpolates the previous and newly estimated bases:

$$\widetilde{P}_t = (1-\eta_a) P_{t-1} + \eta_a P_t^{\text{new}}, \qquad P_t = \mathrm{qf}(\widetilde{P}_t), \tag{10}$$

where η_a is the adaptation rate. This warm-started update enables the basis to evolve continuously with the gradient geometry. Here we adopt a simple basis-level damping step followed by QR re-orthogonalization, which is easier to implement than a geodesic interpolation on the Grassmann manifold. As a consequence, the update depends on the chosen basis representatives and does not explicitly address within-subspace rotation or sign ambiguities. Procrustes-aligned or manifold-based variants are left for future work. For $m > n$, the analogous procedure is applied to the right singular subspace using $G_t^\top G_t$.

4.2 ENERGY-AWARE RESIDUAL ENHANCEMENT

Even with an adaptive basis, rank-r projection discards gradient components outside the retained subspace. LoSARE recovers this information while keeping optimizer states in the low-rank space.

Algorithm 1 LoSARE Optimization

Input: A layer weight matrix $W_t \in \mathbb{R}^{m\times n}$ with $m \le n$, step size η, scale factor α, decay rates β_1, β_2, rank r, adaptation rate η_a, and residual coefficient α_r.

Compute the initial gradient $G_0 \leftarrow \nabla_W \mathcal{L}_0(W_0)$
Initialize $P_0 \in \mathbb{R}^{m\times r}$ using one SVD of G_0
$M_0 \in \mathbb{R}^{r\times n} \leftarrow 0,\ V_0 \in \mathbb{R}^{r\times n} \leftarrow 0,\ t \leftarrow 1$
repeat
  Compute gradient $G_t \leftarrow \nabla_W \mathcal{L}_t(W_t)$
  Compute the projected gradient basis update: $T_t \leftarrow G_t^\top P_{t-1}$
  $Y_t \leftarrow G_t T_t = G_t G_t^\top P_{t-1}$
  $P_t \leftarrow \mathrm{qf}((1-\eta_a)P_{t-1} + \eta_a \,\mathrm{qf}(Y_t))$ ▷ $\mathrm{qf}(\cdot)$: orthonormal basis from QR
  Project the gradient: $R_t \leftarrow P_t^\top G_t$
  Update the first-order moment: $M_t \leftarrow \beta_1 M_{t-1} + (1-\beta_1)R_t$
  Update the second-order moment: $V_t \leftarrow \beta_2 V_{t-1} + (1-\beta_2)R_t^2$
  Apply bias correction: $\widehat{M}_t \leftarrow M_t/(1-\beta_1^t)$, $\widehat{V}_t \leftarrow V_t/(1-\beta_2^t)$
  Compute the normalized low-rank update: $N_t \leftarrow \widehat{M}_t/(\sqrt{\widehat{V}_t} + \epsilon)$
  Reconstruct the low-rank gradient component: $\widetilde{G}_t \leftarrow P_t N_t$
  Compute the residual: $S_t \leftarrow G_t - P_t R_t$
  Compute the residual energy ratio: $\rho_t^{(i)} \leftarrow \|S_{t,:,i}\|_d^2/(\|R_{t,:,i}\|_d^2 + \epsilon)$
  Compute the dynamic boost: $\gamma_t^{(i)} \leftarrow 1 + \alpha_r \rho_t^{(i)}/(1+\rho_t^{(i)})$
  Compute column-wise scaling factors: $\phi_t^{(i)} \leftarrow \gamma_t^{(i)} \|N_{t,:,i}\|/(\|R_{t,:,i}\| + \epsilon)$ for $i = 1, \ldots, n$
  Scale the residual: $\widehat{S}_{t,:,i} \leftarrow \phi_t^{(i)} S_{t,:,i}$ for $i = 1, \ldots, n$
  Apply the norm-growth limiter to obtain $\widehat{S}_t$
  Combine the low-rank update and residual: $\widehat{G}_t \leftarrow \alpha \widetilde{G}_t + \widehat{S}_t$
  Update the weight matrix with decoupled weight decay: $W_{t+1} \leftarrow W_t - \eta \widehat{G}_t$
  $t \leftarrow t+1$
until the stopping criterion is satisfied
return W_t

Given the projected gradient R_t and the normalized low-rank update N_t, we reconstruct its full-rank counterpart and compute

$$\widetilde{G}_t = P_t N_t, \qquad S_t = G_t - P_t R_t. \tag{11}$$

Following the residual-recovery principle of Fira, LoSARE further adapts the residual strength according to the amount of information discarded by the current subspace. Specifically, using the dimension-wise norm adopted in our implementation, we define

$$\rho_t^{(i)} = \frac{\|S_{t,:,i}\|_d^2}{\|R_{t,:,i}\|_d^2 + \epsilon}, \qquad \gamma_t^{(i)} = 1 + \alpha_r \frac{\rho_t^{(i)}}{1+\rho_t^{(i)}}, \tag{12}$$

where α_r controls the maximum enhancement strength. The bounded form ensures $\gamma_t \in [1, 1+\alpha_r]$, increasing the residual contribution when its relative energy is large.

We incorporate this coefficient into the column-wise scaling mechanism of Fira. While Fira determines the residual scaling from the low-rank update, $\gamma_t^{(i)}$ further adjusts the recovery strength based on the relative energy of the discarded residual. Let $R_{t,:,i}$ and $N_{t,:,i}$ denote the i-th columns of the projected gradient and normalized low-rank Adam update, respectively. The column-wise scaling factor is modified with $\gamma_t^{(i)}$:

$$\phi_t^{(i)} = \gamma_t^{(i)} \frac{\|N_{t,:,i}\|}{\|R_{t,:,i}\| + \epsilon}, \qquad i = 1, \ldots, n. \tag{13}$$

The enhanced residual is then

$$\widehat{S}_{t,:,i} = \phi_t^{(i)} S_{t,:,i}, \qquad i = 1, \ldots, n. \tag{14}$$

Table 1: Comparison of memory and optimization properties among different methods. We denote $W_t \in \mathbb{R}^{m\times n}$ with $m \leq n$ and rank r ($r \ll m$).

	LoSARE	Fira	GaLore	LoRA	Full-Rank
Weights	mn	mn	mn	$mn + mr + nr$	mn
Optimizer States	$mr + 2nr + 1$	$mr + 2nr + 1$	$mr + 2nr$	$2mr + 2nr$	$2mn$
Full-Rank Gradients	✓	✓	✗	✓	✓
Full-Rank Weights	✓	✓	✓	✗	✓
Pre-Training	✓	✓	✓	✗	✓
Fine-Tuning	✓	✓	✓	✓	✓
w.o. Periodic SVD	✓	✗	✗	✓	✓

This operation requires no additional full-rank optimizer states. We retain the norm-growth limiter described in Appendix C.2 to stabilize the enhanced residual, restricting its Frobenius norm to grow by at most 1% between consecutive iterations. The final update is

$$\widetilde{G}_t^{\mathrm{LoSARE}} = \alpha P_t N_t + \widehat{S}_t, \tag{15}$$

where α is the scale factor proposed in GaLore. Thus, LoSARE combines the memory-efficient low-rank Adam update with an adaptively enhanced residual while retaining stable norm growth.

4.3 Optimization Algorithm

Algorithm 1 summarizes the overall LoSARE optimization procedure. At each iteration, LoSARE first updates the projection basis using the current gradient, then maintains Adam moments in the updated low-rank space. The discarded residual is subsequently enhanced based on its relative energy and combined with the low-rank update to obtain the full-rank parameter update.

As the basis evolves, LoSARE directly reuses the low-rank Adam moments across consecutive bases. Although this introduces an approximation due to the changing coordinate system, direct moment reuse avoids the overhead of state transport while preserving the $O(rn)$ optimizer-state requirement, providing a practical simplicity–efficiency trade-off. The memory and optimization properties of LoSARE are summarized in Table 1.

LoSARE introduces only marginal persistent state beyond Fira: the Adam moments require $2nr$ entries, while the norm-growth limiter requires one additional scalar per weight matrix. The intermediate quantities for subspace adaptation are temporary and can be discarded after each update. Thus, LoSARE has the same asymptotic optimizer-state memory complexity as Fira, $O(mr + nr)$, while retaining full-rank weights and gradients. Unlike Fira and GaLore, LoSARE avoids periodic full SVD recomputation, requiring only the gradient-driven subspace update and QR reorthogonalization.

5 Experiments

In this section, we evaluate LoSARE on both memory-efficient pre-training and fine-tuning of large language models. We further conduct hyperparameter ablations to analyze the sensitivity of the proposed adaptive subspace tracking and residual enhancement components, as well as the robustness under different rank budgets. All experiments run on NVIDIA L20Y 80GB GPUs.

5.1 Memory-Efficient Pre-Training

Experimental Setup. Following GaLore (Zhao et al., 2024) and Fira (Chen et al., 2025), we evaluate LoSARE on LLaMA-based models of 60M, 130M, 350M, and 1B parameters using the C4 dataset (Raffel et al., 2020). All models are trained in BF16 without data repetition, using the same rank for low-rank methods at each scale. We follow the LLaMA architecture with RMSNorm and SwiGLU activations (Zhang & Sennrich, 2019; Shazeer, 2020; Touvron et al., 2023); detailed settings are provided in Appendix A.1.

Table 2: Comparison of memory-efficient pre-training methods on LLaMA models of different sizes using the C4 dataset. Validation perplexity (↓) and estimated memory consumption of model parameters and optimizer states are reported. Results marked with † are reproduced by us, while results marked with ⋆ are directly taken from GaLore (Zhao et al., 2024). r refers to the rank and d_{model} is the full-rank dimension of models.

	60M	**130M**	**350M**	**1B**
Full-Rank⋆	34.06 (0.36G)	25.08 (0.76G)	18.80 (2.06G)	15.56 (7.80G)
LoSARE	**30.19** (0.24G)	**22.64** (0.52G)	**16.79** (1.22G)	**14.62** (4.38G)
Fira†	30.30 (0.24G)	22.73 (0.52G)	17.02 (1.22G)	14.96 (4.38G)
GaLore†	34.09 (0.24G)	24.90 (0.52G)	19.37 (1.22G)	15.58 (4.38G)
Low-Rank⋆	78.18 (0.26G)	45.51 (0.54G)	37.41 (1.08G)	142.53 (3.57G)
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)
r/d_{model}	128 / 512	256 / 768	256 / 1024	512 / 2048
Training Tokens	1.3B	2.6B	7.8B	13.1B

For a fair comparison, we reproduce GaLore and Fira under our setup, while the Full-Rank, Low-Rank (Kamalakara et al., 2022), LoRA (Hu et al., 2022), and ReLoRA (Lialin et al., 2024) results are taken from GaLore (Zhao et al., 2024). LoSARE uses $\eta_a = 0.05$ and $\alpha_r = 1.0$ for all model sizes. We report validation perplexity and estimated memory for model parameters and optimizer states, following the BF16 memory estimation procedure of GaLore.

Results. As shown in Table 2, LoSARE consistently outperforms GaLore and Fira across model scales while retaining the memory advantage of low-rank optimization through low-rank first- and second-order optimizer states, with negligible additional state for residual enhancement. Notably, LoSARE achieves comparable or even better perplexity than full-rank training. This may be partly attributed to the additional stochasticity introduced by adaptive subspace tracking and residual correction, an effect also observed in Fira (Chen et al., 2025), Adam-mini (Zhang et al., 2025), and APOLLO (Zhu et al., 2025). Such stochasticity may improve exploration and generalization while retaining the benefits of adaptive optimization (Zhou et al., 2020). We leave an investigation of the underlying mechanism to future work.

5.2 Scaling up to LLaMA 7B Pre-Training

Experimental Setup. To evaluate scalability, we pre-train LLaMA-7B on C4 and compare GaLore with $r \in \{64, 512, 1024\}$, Fira with $r = 64$, and LoSARE with $r = 64$. Due to computational constraints, we use a total batch size of 128 and train for 150K steps. Under the Linear Scaling Rule (Krizhevsky, 2014; Goyal et al., 2017), this matches the token budget of the 40K-step checkpoint in GaLore's original 7B pre-training (batch size 512). We therefore compare our validation perplexity against GaLore's result at that checkpoint (Zhao et al., 2024). Detailed training settings are provided in Appendix A.1.

Results. Figure 2 reports validation perplexity over training, with early (0–30K), middle (30K–100K), and late (100K–150K) stages. Increasing the rank improves the early-stage optimization of GaLore, but $r = 512$ and $r = 1024$ gradually plateau, while $r = 64$ remains substantially worse. In contrast, Fira and LoSARE continue to improve in the later stages, suggesting that larger subspace capacity alone is insufficient to track evolving gradient geometry and recover discarded information. This behavior is consistent with potential benefits from continual basis adaptation and residual recovery. We leave a direct measurement of tracking fidelity and a controlled isolation of the two mechanisms for future work. LoSARE-64 starts below Fira in the early stage, briefly falls behind in the middle stage, and recovers the lead near the end, achieving the best final perplexity. Notably, it uses only $1/8$ and $1/16$ of the rank of GaLore-512 and GaLore-1024, respectively, while being approximately $1.12\times$ faster than GaLore-1024 and $1.22\times$ faster than Fira-64. These results demonstrate the performance–efficiency trade-off enabled by LoSARE's adaptive subspace tracking and energy-aware residual enhancement.

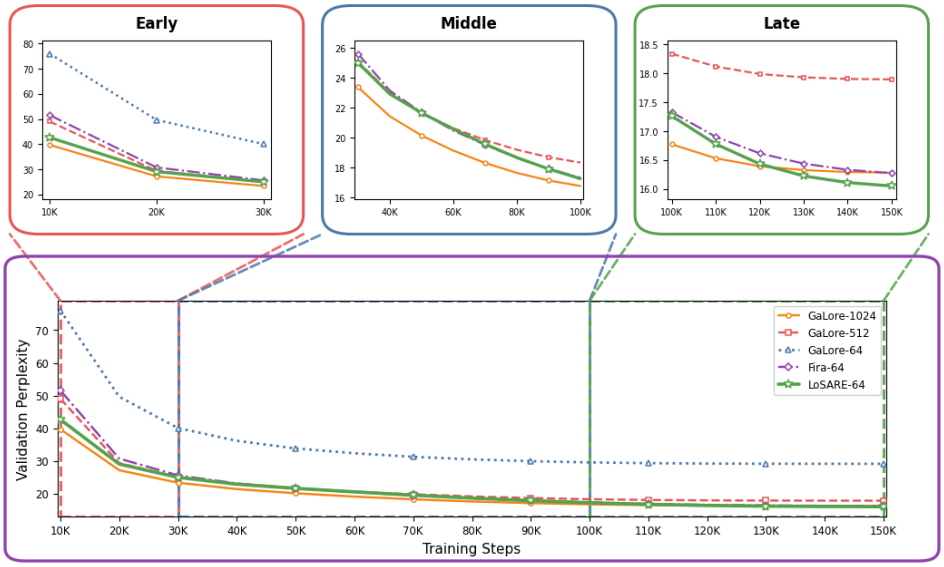


Figure 2: Validation perplexity (↓) of LLaMA-7B pre-training on the C4 dataset. The full training trajectory is shown at the bottom, with zoomed-in views of the early, middle, and late stages of training at the top. We denote each method as Method-r, where r is the rank.

5.3 Memory-Efficient Fine-Tuning

Experimental Setup. Following Fira (Chen et al., 2025), we evaluate LoSARE on memory-efficient fine-tuning of LLaMA-7B for commonsense reasoning. The benchmark consists of eight tasks: BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, ARC-e, ARC-c, and OBQA. Following Hu et al. (2023), we combine the training data from all tasks and evaluate each task individually. We use rank $r = 32$ and a learning rate of $1e-4$ for low-rank methods. The detailed settings of fine-tuning are provided in Appendix A.2.

For a fair comparison, we reproduce the Fira and GaLore results under our setup. Results for Prefix-tuning (Prefix) (Li & Liang, 2021), Series Adapter (Series) (Houlsby et al., 2019), Parallel Adapter (Parallel) (He et al., 2022), LoRA (Hu et al., 2022), ReLoRA (Lialin et al., 2024), and Flora (Hao et al., 2024) are taken from Fira (Chen et al., 2025) and LLM-Adapters (Hu et al., 2023). We report accuracy on each task.

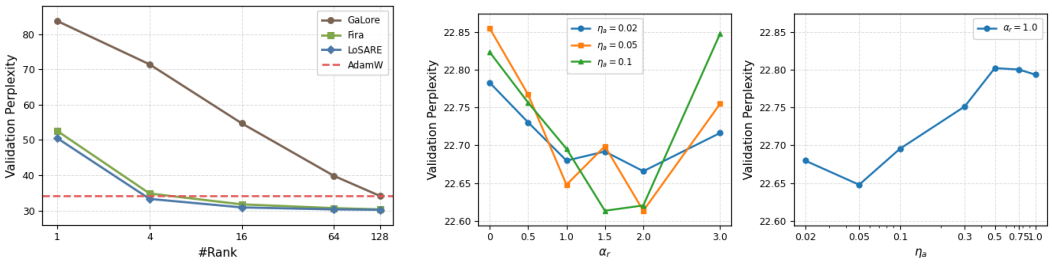
Results. As shown in Table 3, LoSARE achieves the highest performance on 4 of 8 datasets, demonstrating competitive or superior performance compared to the baselines across these tasks, while retaining the low memory footprint of low-rank optimization. Notably, LoSARE performs best on HellaSwag, which contains 10,042 test examples, corresponding to hundreds or even thousands of additional correct predictions over other methods. On OBQA, LoSARE is slightly below the best baseline, but the task contains only 500 examples, so the accuracy gap corresponds to fewer than ten examples. These results demonstrate that LoSARE effectively adapts to diverse commonsense reasoning tasks while maintaining the memory efficiency of low-rank optimization.

5.4 Ablation Study

Performance under Varying Ranks. Motivated by the results in Section 5.2, we evaluate LoSARE under ranks $r \in \{1, 4, 16, 64, 128\}$ on LLaMA-60M pre-training with C4, comparing it with GaLore and Fira. As shown in Figure 3a, LoSARE consistently outperforms both baselines and remains below the full-rank AdamW baseline at $r = 4$. Its performance improves sharply from $r = 1$ to $r = 4$ and then largely saturates, demonstrating that LoSARE remains effective under extremely low-rank constraints.

Table 3: Comparison of memory-efficient fine-tuning methods on LLaMA-7B across eight commonsense reasoning tasks. Accuracy (↑) is reported. Results marked with † are reproduced by us, while results marked with ⋆ are directly taken from Chen et al. (2025) and Hu et al. (2023).

Method	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg
LoSARE	**69.8**	82.3	77.4	**85.0**	**81.0**	81.5	**66.8**	78.0	**77.7**
Fira†	69.1	82.8	77.5	80.7	80.0	**81.8**	65.4	**79.8**	77.1
GaLore†	66.6	**83.0**	77.1	60.2	77.9	80.9	65.4	78.6	73.7
Prefix⋆	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
Series⋆	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
Parallel⋆	67.9	76.4	**78.8**	69.8	78.9	73.7	57.3	75.2	72.2
LoRA⋆	68.9	80.7	77.4	78.1	78.8	77.8	61.3	74.8	74.7
ReLoRA⋆	68.9	81.2	77.8	46.0	79.4	80.2	64.2	79.6	72.2
Flora⋆	50.1	77.5	74.2	53.8	45.5	79.0	64.6	74.8	64.9
Full-Rank⋆	64.2	68.1	68.0	42.3	66.5	55.6	43.9	60.0	58.6



(a) Validation perplexity of LLaMA-60M pre-training on C4 under different ranks. LoSARE consistently maintains strong performance at low ranks. The dashed line denotes the full-rank AdamW baseline.

(b) Hyperparameter sensitivity study of LoSARE on LLaMA-130M pre-training. **Left:** varying residual enhancement coefficient α_r under different η_a. **Right:** varying subspace adaptation rate η_a with $\alpha_r = 1.0$.

Figure 3: Ablation study of LoSARE under varying ranks and hyperparameter settings.

Hyperparameter Sensitivity. We study the residual enhancement coefficient α_r and subspace adaptation rate η_a on LLaMA-130M pre-training with C4. As shown in Figure 3b, moderate α_r values around 1.0–2.0 achieve the best performance, while too small or too large residual enhancement degrades performance. In particular, excessively large α_r (e.g., $\alpha_r = 5.0$) even leads to divergence (see Appendix D). For η_a, performance first improves and then degrades as the adaptation rate increases, consistent with the sensitivity to subspace update frequency T observed in GaLore (Zhao et al., 2024). We therefore use $\alpha_r = 1.0$ and $\eta_a = 0.05$ in the main experiments.

6 Conclusion

In this paper, we presented LoSARE, a memory-efficient low-rank optimization method for large language model training. LoSARE combines gradient-driven adaptive subspace tracking with energy-aware residual enhancement to better preserve useful gradient information while maintaining low-rank optimizer states. Extensive experiments on pre-training and fine-tuning demonstrate that LoSARE consistently outperforms GaLore and Fira, while retaining their memory efficiency. In particular, LoSARE achieves strong late-stage optimization and remains effective at substantially lower ranks, providing a favorable trade-off between performance, memory consumption, and training cost. Overall, LoSARE demonstrates the potential of adaptive low-rank optimization with energy-aware residual enhancement for training large language models under constrained computational resources. We hope LoSARE will encourage further research on adaptive low-rank optimization for scaling language model training under limited computational resources.

AI USE STATEMENT

Large language models (LLMs) were used to assist with the writing and literature discovery for this paper. Specifically, LLMs were used to polish and refine the language, improve clarity and conciseness, and assist with reorganizing and shortening parts of the manuscript. LLMs were also used as a retrieval and discovery aid to identify potentially relevant prior work and references. All suggested content and references were reviewed and verified by the authors. The LLMs were not used to generate experimental results or fabricate scientific claims. The authors retain full responsibility for the content, scientific contributions, and accuracy of the paper.

REFERENCES

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Xi Chen, Kaituo Feng, Changsheng Li, Xunhao Lai, Xiangyu Yue, Ye Yuan, and Guoren Wang. Fira: Can we achieve full-rank training of LLMs under low-rank constraint? In *Advances in Neural Information Processing Systems*, 2025. NeurIPS 2025.

Romain Cosson, Ali Jadbabaie, Anuran Makur, Amirhossein Reisizadeh, and Devavrat Shah. Low-rank gradient descent. *IEEE Open Journal of Control Systems*, 2:380–395, 2023.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations*, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, 2023.

Mary Gooneratne, Khe Chai Sim, Petr Zadrazil, Andreas Kabel, Françoise Beaufays, and Giovanni Motta. Low-rank gradient approximation for memory-efficient on-device training of deep neural network. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022.

Yutong He, Pengrui Li, Yipeng Hu, Chuyan Chen, and Kun Yuan. Subspace optimization for large language models with convergence guarantees. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. LLM-Adapters: An adapter family for parameter-efficient finetuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Weihao Huang, Zhenyu Zhang, Yushun Zhang, Zhi-Quan Luo, Ruoyu Sun, and Zhangyang Wang. Galore-mini: Low rank gradient learning with fewer learning rates. In *NeurIPS 2024 Workshop on Fine-Tuning in Modern Machine Learning: Principles and Scalability*, 2024.

M.F. Kaloorazi, S. Ahmadi-Asl, and S. Rahardja. Randomized two-sided subspace iteration for low-rank matrix and tensor decomposition. *Digital Signal Processing*, 149:104472, 2024.

Siddhartha Rao Kamalakara, Acyr F. Locatelli, Bharat Venkitesh, Jimmy Ba, Yarin Gal, and Aidan N. Gomez. Exploring low rank training of deep neural networks. *arXiv preprint arXiv:2209.13569*, 2022.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. ReLoRA: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*, 2024.

Sebastian Loeschcke, Mads Toftrup, Michael J. Kastoryano, Serge Belongie, and Vésteinn Snæbjarnarson. Loqt: Low-rank adapters for quantized pretraining. In *The Thirty-eighth Advances in Neural Information Processing Systems*, 2024.

Kai Lv, Hang Yan, Qipeng Guo, Haijun Lv, and Xipeng Qiu. Adalomo: Low-memory optimization with adaptive learning rate. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 12486–12502, 2024a.

Kai Lv, Yuqing Yang, Tengxiao Liu, Qipeng Guo, and Xipeng Qiu. Full parameter fine-tuning for large language models with limited resources. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 8187–8198, 2024b.

Xinmin Meng, Muhammad A.A. Abdelgawad, Peng Jing, Ray C.C. Cheung, and Hong Yan. Memory-efficient neural network training via gradient compression through continuous basis tracking. *Neurocomputing*, 680:133278, 2026.

Zhanfeng Mo, Long-Kai Huang, and Sinno Jialin Pan. Parameter and memory efficient pretraining via low-rank riemannian optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ionut-Vlad Modoranu, Aleksei Kalinov, Eldar Kurtic, Elias Frantar, and Dan Alistarh. Error feedback can accurately compress preconditioners. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 35910–35933, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Sahar Rajabi, Nayeema Nonta, and Sirisha Rambhatla. SubTrack++: Gradient subspace tracking for scalable LLM training. In *Advances in Neural Information Processing Systems*, 2025.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. Zero-offload: Democratizing billion-scale model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pp. 551–564, 2021.

Thomas Robert, Mher Safaryan, Ionut-Vlad Modoranu, and Dan Alistarh. Ldadam: Adaptive optimization from low-dimensional gradient statistics. In *The Thirteenth International Conference on Learning Representations*, 2025.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 4596–4604, 2018.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Practical low-rank communication compression in decentralized deep learning. In *Advances in Neural Information Processing Systems*, 2020.

Hongyi Wang, Scott Sievert, Shengchao Liu, Zachary Charles, Dimitris Papailiopoulos, and Stephen Wright. Atomo: Communication-efficient learning via atomic sparsification. In *Advances in Neural Information Processing Systems*, 2018.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, 2019.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*, 2023a.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023b.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik P. Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zhenyu Zhang, Ajay Jaiswal, Lu Yin, Shiwei Liu, Jiawei Zhao, Yuandong Tian, and Zhangyang Wang. Q-GaLore: Quantized galore with int4 projection and layer-adaptive low-rank gradients. *arXiv preprint arXiv:2407.08296*, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *International Conference on Machine Learning*, 2024. ICML 2024.

Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, and Steven Chu Hong Hoi. Towards theoretically understanding why SGD generalizes better than Adam in deep learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21285–21296, 2020.

Hanqing Zhu, Zhenyu Zhang, Wenyan Cong, Xi Liu, Sem Park, Vikas Chandra, Bo Long, David Z. Pan, Zhangyang Wang, and Jinwon Lee. APOLLO: SGD-like memory, AdamW-level performance. In *Proceedings of Machine Learning and Systems*, 2025.

A Detailed Experimental Settings

A.1 Pre-training Settings

We follow the pre-training settings of GaLore (Zhao et al., 2024) and Fira (Chen et al., 2025) for fair comparison. We train LLaMA-based architectures with RMSNorm and SwiGLU activations on C4 (Raffel et al., 2020). All models are trained in BF16 without data repetition, using a maximum sequence length of 256 and a batch size of 131K tokens. The learning rate is warmed up during the first 10% of training steps, followed by cosine annealing to 10% of the initial learning rate. Table 4 summarizes the model architectures and training budgets.

Table 4: Model architectures and training settings for LLaMA pre-training on C4.

Params	Hidden	Intermediate	Heads	Layers	Steps	Data Amount (Tokens)
60M	512	1376	8	8	10K	1.3B
130M	768	2048	12	12	20K	2.6B
350M	1024	2736	16	24	60K	7.8B
1B	2048	5461	32	24	100K	13.1B
7B	4096	11008	32	32	150K	4.9B

For methods other than Fira, GaLore, and LoSARE, we select the learning rate from $\{0.01, 0.005, 0.001, 0.0005, 0.0001\}$ for models from 60M to 1B based on validation perplexity. For Fira, GaLore, and LoSARE, the learning rates and scale factors α are summarized in Table 5a. The 60M–350M models use the same settings as Fira and GaLore, while the 1B and 7B models use the model-specific settings shown in the table. Following Fira and GaLore, α controls the strength of the low-rank update (Zhao et al., 2024; Chen et al., 2025). GaLore and Fira use a subspace change frequency of $T = 200$, while LoSARE uses $\eta_a = 0.05$ and $\alpha_r = 1.0$.

Due to computational constraints, we train LoSARE on LLaMA-7B with a total batch size of 128 for 150K steps, as described in Section 5.2. This amounts to 4.9B tokens, one quarter of the 19.7B tokens used in GaLore's original 7B pre-training setting (Zhao et al., 2024).

A.2 Fine-tuning Settings

We follow the fine-tuning settings of Fira (Chen et al., 2025) and fine-tune the pre-trained LLaMA-7B model on eight commonsense reasoning tasks: BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, ARC-e, ARC-c, and OBQA. The rank is $r = 32$ and the scale factor is $\alpha = 64$. We use Adam with a learning rate of 1×10^{-4} and a linear scheduler, a batch size of 16, 100 warm-up steps, 3 epochs, and dropout of 0.05. The complete configuration is summarized in Table 5b.

Table 5: Hyperparameter configurations for LLaMA pre-training and fine-tuning.

(a) Pre-training

Model	Method	LR	α
60M/130M/350M	**LoSARE**	0.01	0.25
60M/130M/350M	Fira	0.01	0.25
60M/130M/350M	GaLore	0.01	0.25
1B	**LoSARE**	0.0015	0.05
1B	Fira	0.0015	0.05
1B	GaLore	0.01	0.25
7B	**LoSARE**	0.002	0.05
7B	Fira	0.002	0.05
7B	GaLore	0.01	0.0625

(b) Fine-tuning

Hyper-parameter	Setting
Rank r	32
Scale factor α	64
Dropout	0.05
Base optimizer	Adam
Learning rate	1×10^{-4}
LR scheduler	Linear
Batch size	16
Warm-up steps	100
Epochs	3
Where	Q, K, V, Up, Down

B Additional Related Work

Memory-Efficient Optimization. Beyond low-rank gradient optimization, another line of work reduces optimizer-state memory through structured or compressed statistics. Adafactor (Shazeer & Stern, 2018) factorizes second-order statistics, while 8-bit optimizers (Dettmers et al., 2022) quantize optimizer states. LOMO (Lv et al., 2024b) and AdaLomo (Lv et al., 2024a) reduce memory by fusing gradient computation with parameter updates, while Adam-mini (Zhang et al., 2025) exploits redundancy in Adam's learning-rate adaptation. APOLLO (Zhu et al., 2025) further approximates AdamW's element-wise scaling using low-rank auxiliary states and random projections. These approaches are largely orthogonal to LoSARE: rather than compressing optimizer statistics alone, LoSARE focuses on preserving useful gradient information through adaptive subspace tracking and residual enhancement. System-level techniques such as gradient checkpointing and memory offloading (Chen et al., 2016; Ren et al., 2021) can also be combined with these algorithmic approaches.

C Additional Preliminaries

This appendix provides additional background on the low-rank gradient projection and residual recovery mechanisms underlying LoSARE. These details are omitted from the main text to focus the presentation on the proposed adaptive subspace tracking and energy-aware residual enhancement.

C.1 Gradient Low-Rank Projection

GaLore (Zhao et al., 2024) reduces optimizer-state memory by projecting a full-rank gradient into a low-rank subspace. For a weight matrix $W_t \in \mathbb{R}^{m\times n}$ with $m \le n$, let $P_t \in \mathbb{R}^{m\times r}$ be an orthonormal projection basis with $r \ll m$. The projected gradient is

$$R_t = P_t^\top G_t, \tag{16}$$

where $R_t \in \mathbb{R}^{r\times n}$. The optimizer maintains its first- and second-order states only in this low-rank space. For AdamW, the normalized low-rank update is

$$N_t = \frac{M_t}{\sqrt{V_t} + \epsilon}, \tag{17}$$

and the corresponding update in the original parameter space is

$$\widetilde{G}_t^{\mathrm{GaLore}} = \alpha P_t N_t, \tag{18}$$

where α is the scale factor. Thus, the optimizer states require $O(rn)$ rather than $O(mn)$ memory.

The projection also decomposes the full-rank gradient into its component inside the retained subspace and its orthogonal residual:

$$G_t = \underbrace{P_t P_t^\top G_t}_{\text{subspace component}} + \underbrace{(I - P_t P_t^\top) G_t}_{\text{residual component}}. \tag{19}$$

Since $R_t = P_t^\top G_t$, the residual can equivalently be written as

$$S_t = (I - P_t P_t^\top) G_t = G_t - P_t R_t. \tag{20}$$

GaLore optimizes the projected component while discarding the residual, which motivates residual recovery methods such as Fira.

C.2 Residual Recovery with Norm-Based Scaling

Fira (Chen et al., 2025) recovers the discarded residual while retaining low-rank optimizer states. Given the projected gradient R_t and normalized low-rank update N_t, Fira computes a separate scaling factor for each column:

$$\phi_t^{(i)} = \frac{\|N_{t,:,i}\|}{\|R_{t,:,i}\| + \epsilon}, \qquad i = 1, \dots, n. \tag{21}$$

The residual is then scaled column-wise as

$$\widehat{S}^{\text{Fira}}_{t,:,i} = \phi^{(i)}_t S_{t,:,i}, \qquad i = 1, \ldots, n, \tag{22}$$

and combined with the low-rank update:

$$\widetilde{G}^{\text{Fira}}_t = \alpha P_t N_t + \widehat{S}^{\text{Fira}}_t. \tag{23}$$

This allows residual information to contribute to the full-rank parameter update without introducing full-rank optimizer states.

To stabilize the residual correction, Fira further employs a norm-growth limiter (Chen et al., 2025). Letting $c = 1.01$, the step-to-step growth ratio of the scaled residual is

$$q_t = \frac{\|\widehat{S}_t\|}{\|\widehat{S}_{t-1}\| + \epsilon}. \tag{24}$$

The residual is then rescaled according to

$$\widehat{S}_t \leftarrow \frac{\widehat{S}_t}{\max(q_t, c)/c}, \tag{25}$$

which limits its step-to-step norm growth to the factor c.

The above residual-recovery mechanism forms the basis for the energy-aware enhancement introduced by LoSARE. Rather than using only the norm-based scaling in Eq. equation 21, LoSARE additionally measures the relative energy of the discarded residual and uses this signal to adapt its recovery strength.

D Training Instability with Excessive Residual Enhancement

We further evaluate a larger residual enhancement coefficient $\alpha_r = 5.0$ with $\eta_a = 0.05$. As shown in Figure 4, excessive residual enhancement causes the validation perplexity to diverge, supporting our conservative choice of $\alpha_r = 1.0$ in the main experiments. A theoretical analysis of α_r is provided in Appendix G.3.

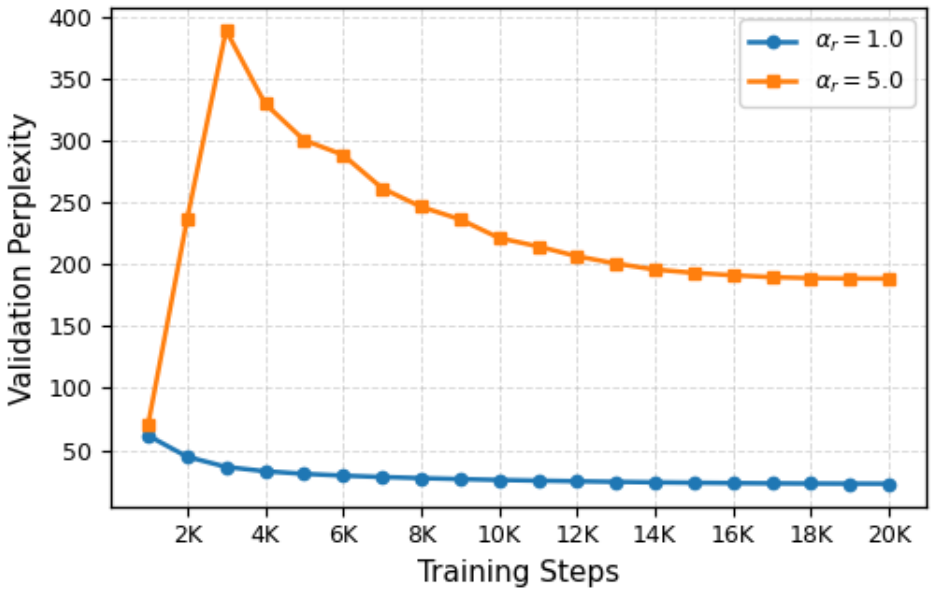


Figure 4: Training dynamics with different residual enhancement coefficients α_r under $\eta_a = 0.05$. Excessive residual enhancement with $\alpha_r = 5.0$ leads to divergence, while $\alpha_r = 1.0$ remains stable.

E Pre-training Perplexity Curves

We show the validation perplexity progression of the 60M, 130M, 350M, and 1B models in Figure 5. LoSARE and Fira achieve substantially lower perplexity than GaLore throughout training. For the smaller models (60M and 130M), Fira and LoSARE exhibit similar training trajectories, while for the larger models (350M and 1B), LoSARE converges slightly faster than Fira.

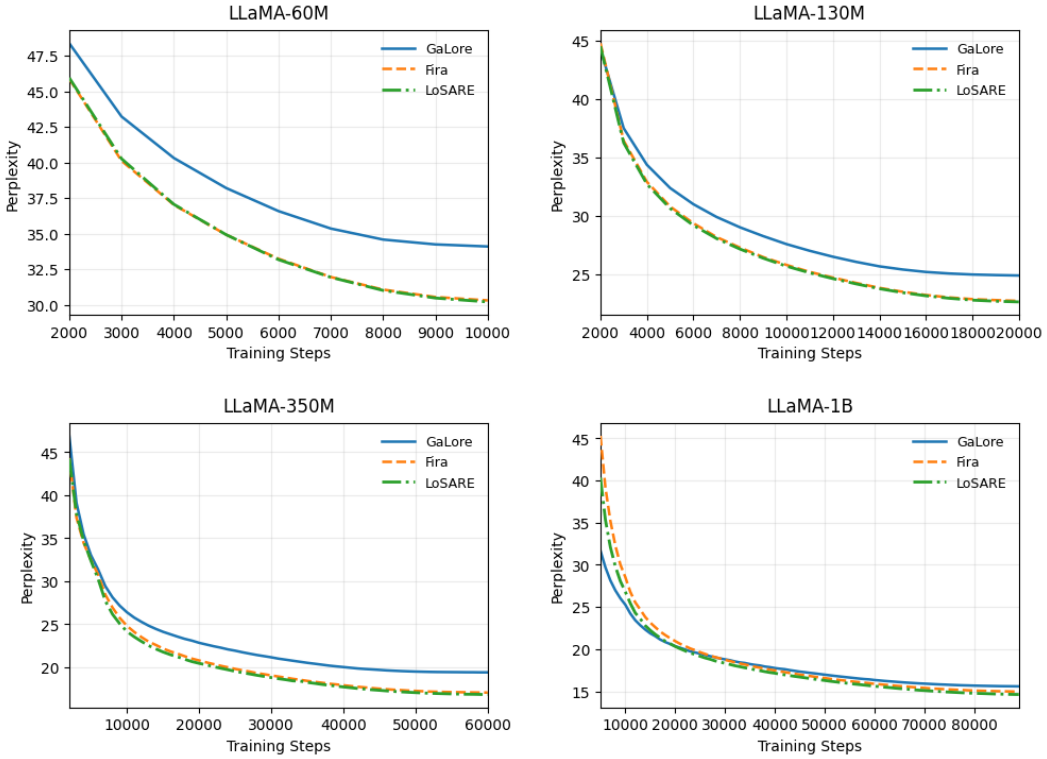


Figure 5: Validation perplexity curves of GaLore, Fira, and LoSARE during pre-training on C4 across different model scales.

F Memory Estimates

Following GaLore (Zhao et al., 2024), we estimate the memory requirements of the weight parameters and optimizer states under BF16. Since Fira and LoSARE retain the same low-rank optimizer-state structure as GaLore, their estimated memory usage is identical under this calculation.

Table 6: Memory estimates for LLaMA models under BF16, following the estimation procedure of GaLore (Zhao et al., 2024). Fira and LoSARE have the same estimates as GaLore under this calculation.

(a) Weight Parameters

	60M	130M	350M	1B
Full-Rank	0.12G	0.25G	0.68G	2.60G
LoSARE	0.12G	0.25G	0.68G	2.60G
Fira	0.12G	0.25G	0.68G	2.60G
GaLore	0.12G	0.25G	0.68G	2.60G
LoRA	0.20G	0.44G	1.04G	3.79G

(b) Optimizer States

	60M	130M	350M	1B
Full-Rank	0.23G	0.51G	1.37G	5.20G
LoSARE	0.13G	0.28G	0.54G	1.78G
Fira	0.13G	0.28G	0.54G	1.78G
GaLore	0.13G	0.28G	0.54G	1.78G
LoRA	0.17G	0.37G	0.72G	2.38G

G THEORETICAL ANALYSIS

This section provides theoretical analysis of the two components introduced by LoSARE. We first characterize the adaptive subspace tracking procedure and show that, under a spectral-gap condition, the QR-based update contracts the distance to the dominant gradient subspace. We then analyze the energy-aware residual enhancement and establish that its gain is bounded and monotonically increases with the discarded residual energy. Finally, we show that increasing the residual enhancement coefficient α_r necessarily increases the relative contribution of the residual correction, which provides a theoretical explanation for the training instability observed with excessively large α_r in Appendix D.

G.1 ADAPTIVE SUBSPACE TRACKING

We first analyze the subspace update used by LoSARE. At iteration t, let the singular value decomposition of the gradient be

$$G_t = U_t \Sigma_t V_t^\top, \tag{26}$$

where the singular values satisfy

$$\sigma_1(G_t) \geq \cdots \geq \sigma_r(G_t) > \sigma_{r+1}(G_t) \geq \cdots. \tag{27}$$

Let $U_{t,1} \in \mathbb{R}^{m\times r}$ denote the dominant left singular subspace associated with the largest r singular values. Since

$$G_t G_t^\top = U_t \Sigma_t^2 U_t^\top, \tag{28}$$

the intermediate update of LoSARE,

$$Y_t = G_t G_t^\top P_{t-1}, \tag{29}$$

is an orthogonal subspace iteration driven by the squared singular values of the current gradient.

For two r-dimensional subspaces represented by orthonormal matrices P and U, define their largest principal-angle distance as

$$d(P, U) = \left\|(I - PP^\top)U\right\|_2 = \sin\Theta(P, U). \tag{30}$$

Note that $\tan\Theta(P, U) = d(P, U)/\sqrt{1 - d(P, U)^2}$, which coincides with $d(P, U)$ in the small-angle regime.

Lemma 1 (Contraction of the QR Subspace Iteration)**.** *Let $A_t = G_t G_t^\top$ and suppose $\sigma_r(G_t) > \sigma_{r+1}(G_t)$. For an orthonormal matrix P_{t-1} such that $U_{t,1}^\top P_{t-1}$ is nonsingular, define*

$$P_t^{\text{new}} = \text{qf}(A_t P_{t-1}). \tag{31}$$

Then

$$\tan\Theta(P_t^{\text{new}}, U_{t,1}) \leq \mu_t \tan\Theta(P_{t-1}, U_{t,1}), \qquad \mu_t = \frac{\sigma_{r+1}^2(G_t)}{\sigma_r^2(G_t)} < 1, \tag{32}$$

where $\Theta(\cdot,\cdot)$ denotes the largest principal angle between the two subspaces.

Proof. Partition $U_t = [U_{t,1}, U_{t,2}]$ such that

$$A_t = U_{t,1}\Lambda_{t,1}U_{t,1}^\top + U_{t,2}\Lambda_{t,2}U_{t,2}^\top,$$

where the diagonal entries of $\Lambda_{t,1}$ are $\sigma_1^2(G_t), \ldots, \sigma_r^2(G_t)$ and those of $\Lambda_{t,2}$ are at most $\sigma_{r+1}^2(G_t)$. Write

$$P_{t-1} = U_{t,1}C_1 + U_{t,2}C_2.$$

Since $U_{t,1}^\top P_{t-1} = C_1$ is nonsingular, the tangent of the largest principal angle satisfies

$$\tan\Theta(P_{t-1}, U_{t,1}) = \|C_2 C_1^{-1}\|_2.$$

After multiplication by A_t,

$$A_t P_{t-1} = U_{t,1}\Lambda_{t,1}C_1 + U_{t,2}\Lambda_{t,2}C_2.$$

The QR factorization changes the basis but not the represented column space. Therefore,

$$\tan\Theta(P_t^{\text{new}}, U_{t,1}) = \|\Lambda_{t,2} C_2 (\Lambda_{t,1} C_1)^{-1}\|_2.$$

Using submultiplicativity,

$$\begin{aligned}\tan\Theta(P_t^{\text{new}}, U_{t,1}) &\le \|\Lambda_{t,2}\|_2 \|\Lambda_{t,1}^{-1}\|_2 \|C_2 C_1^{-1}\|_2 \\ &\le \frac{\sigma_{r+1}^2(G_t)}{\sigma_r^2(G_t)} \tan\Theta(P_{t-1}, U_{t,1}).\end{aligned}$$

The spectral-gap assumption gives $\mu_t < 1$, completing the proof. □

Lemma 1 shows that the Gram–QR update is not merely a computational alternative to full SVD reconstruction. Under a spectral gap, it performs a contractive subspace iteration toward the dominant gradient subspace. In particular, the contraction factor μ_t decreases as the spectral gap becomes larger, indicating faster idealized subspace alignment. This result provides an idealized motivation for the adaptive subspace update, but does not establish a tracking guarantee for the damped stochastic algorithm used in our experiments.

LoSARE further introduces the adaptation rate η_a to avoid replacing the previous basis abruptly. Let

$$\widetilde{P}_t = (1-\eta_a) P_{t-1} + \eta_a P_t^{\text{new}}, \qquad P_t = \text{qf}(\widetilde{P}_t). \tag{33}$$

The parameter η_a controls the amount of the newly estimated subspace incorporated at each iteration. Smaller η_a produces a smoother but slower adaptation, whereas larger η_a reacts more aggressively to changes in the current gradient geometry. The formal analysis above concerns the undamped Gram–QR step and therefore does not directly characterize the tracking error of this damped update. We leave a rigorous analysis of the stochastic, time-varying setting to future work.

G.2 Energy-Aware Residual Enhancement

We analyze the residual enhancement mechanism per gradient column, where $R_{t,:,i}$, $S_{t,:,i}$, and $N_{t,:,i}$ denote the projected low-rank component, residual component, and normalized low-rank Adam update as defined in Sec. 4. For each column i, LoSARE defines

$$\rho_t^{(i)} = \frac{\|S_{t,:,i}\|_d^2}{\|R_{t,:,i}\|_d^2 + \epsilon}, \tag{34}$$

where $\|\cdot\|_d$ denotes the dimension-normalized ℓ_2 norm, and

$$\gamma_t^{(i)} = 1 + \alpha_r \frac{\rho_t^{(i)}}{1+\rho_t^{(i)}}. \tag{35}$$

Here $\rho_t^{(i)}$ measures the relative energy of the discarded residual with respect to the projected low-rank component.

Proposition 1 (Bounded and Monotone Residual Enhancement)**.** *For $\alpha_r \ge 0$ and $\rho_t^{(i)} \ge 0$, the residual enhancement gain satisfies*

$$1 \le \gamma_t^{(i)} < 1 + \alpha_r. \tag{36}$$

Moreover,

$$\frac{\partial \gamma_t^{(i)}}{\partial \rho_t^{(i)}} = \frac{\alpha_r}{(1+\rho_t^{(i)})^2} \ge 0. \tag{37}$$

Therefore, the residual enhancement is monotonically increasing with the discarded residual energy while remaining bounded for any finite α_r.

Proof. Since $\rho_t^{(i)} \ge 0$,

$$0 \le \frac{\rho_t^{(i)}}{1+\rho_t^{(i)}} < 1.$$

Multiplying by $\alpha_r \geq 0$ and adding one gives

$$1 \leq \gamma_t^{(i)} < 1 + \alpha_r.$$

Furthermore,

$$\frac{\partial \gamma_t^{(i)}}{\partial \rho_t^{(i)}} = \alpha_r \frac{(1+\rho_t^{(i)}) - \rho_t^{(i)}}{(1+\rho_t^{(i)})^2} = \frac{\alpha_r}{(1+\rho_t^{(i)})^2} \geq 0.$$

Hence the gain increases monotonically with residual energy and approaches $1+\alpha_r$ as $\rho_t^{(i)}$ becomes large. □

The proposition establishes an important property of the proposed energy-aware mechanism. When the discarded residual contains little energy, $\rho_t^{(i)}$ is small and the enhancement remains close to its baseline value. When the residual becomes relatively more informative, the gain increases smoothly rather than changing discontinuously. In particular, the saturation term $\rho/(1+\rho)$ prevents the gain from growing unboundedly with the residual energy.

G.3 Effect of Excessive Residual Enhancement

Although Proposition 1 guarantees that the gain is bounded for any fixed α_r, it also shows that the upper bound itself grows linearly with α_r. We next characterize how this affects the relative contribution of the residual correction.

For simplicity, consider one column and define

$$\kappa_t^{(i)} = \frac{\|S_{t,:,i}\|}{\|R_{t,:,i}\| + \epsilon}. \tag{38}$$

Because P_t has orthonormal columns,

$$\|P_t N_{t,:,i}\| = \|N_{t,:,i}\|.$$

The enhanced residual is

$$\widehat{S}_{t,:,i} = \gamma_t^{(i)} \frac{\|N_{t,:,i}\|}{\|R_{t,:,i}\| + \epsilon} S_{t,:,i}. \tag{39}$$

Consequently, its norm relative to the projected low-rank update is

$$\begin{aligned} \frac{\|\widehat{S}_{t,:,i}\|}{\|\alpha P_t N_{t,:,i}\|} &= \frac{\gamma_t^{(i)}}{\alpha} \frac{\|S_{t,:,i}\|}{\|R_{t,:,i}\| + \epsilon} \\ &= \frac{\gamma_t^{(i)}}{\alpha} \kappa_t^{(i)}. \end{aligned} \tag{40}$$

Using the definition of $\gamma_t^{(i)}$, we obtain

$$\frac{\|\widehat{S}_{t,:,i}\|}{\|\alpha P_t N_{t,:,i}\|} = \frac{\kappa_t^{(i)}}{\alpha} \left(1 + \alpha_r \frac{\rho_t^{(i)}}{1+\rho_t^{(i)}}\right). \tag{41}$$

Proposition 2 (Residual Dominance under Excessive Enhancement)**.** *For a fixed gradient state and $\rho_t^{(i)} > 0$, the ratio*

$$\xi_t^{(i)} = \frac{\|\widehat{S}_{t,:,i}\|}{\|\alpha P_t N_{t,:,i}\|} \tag{42}$$

is strictly increasing and affine in α_r. In particular,

$$\frac{\partial \xi_t^{(i)}}{\partial \alpha_r} = \frac{\kappa_t^{(i)}}{\alpha} \frac{\rho_t^{(i)}}{1+\rho_t^{(i)}} > 0. \tag{43}$$

Hence, for sufficiently large α_r, the enhanced residual can become larger than the projected low-rank update and dominate the resulting parameter update.

Proof. Substituting the definition of $\gamma_t^{(i)}$ into $\xi_t^{(i)}$ gives

$$\xi_t^{(i)} = \frac{\kappa_t^{(i)}}{\alpha}\left(1 + \alpha_r \frac{\rho_t^{(i)}}{1+\rho_t^{(i)}}\right).$$

For fixed $\rho_t^{(i)} > 0$ and $\kappa_t^{(i)} > 0$, differentiating with respect to α_r yields

$$\frac{\partial \xi_t^{(i)}}{\partial \alpha_r} = \frac{\kappa_t^{(i)}}{\alpha}\frac{\rho_t^{(i)}}{1+\rho_t^{(i)}} > 0.$$

Thus the residual-to-low-rank update ratio increases strictly with α_r. Since this ratio grows without bound as α_r increases, the residual correction eventually becomes dominant whenever the residual is nonzero. □

Proposition 2 provides a direct mathematical reason why α_r should not be chosen excessively large. The purpose of LoSARE is not to replace the low-rank adaptive update with the discarded residual, but to recover useful information discarded by the projection. Increasing α_r therefore has two competing effects: it strengthens useful residual information, but it also increases the relative contribution of the residual independently of the low-rank optimizer update. When α_r becomes sufficiently large, the residual correction can dominate the update direction and substantially alter the optimization dynamics.

A useful sufficient condition for preventing the residual correction from exceeding a prescribed fraction $\tau > 0$ of the low-rank update is

$$\xi_t^{(i)} \le \tau. \tag{44}$$

For $\rho_t^{(i)} > 0$, this gives

$$\alpha_r \le \frac{1+\rho_t^{(i)}}{\rho_t^{(i)}}\left(\frac{\alpha\tau}{\kappa_t^{(i)}} - 1\right), \tag{45}$$

whenever the right-hand side is positive. This bound illustrates that the appropriate magnitude of α_r depends on the current residual geometry and the relative scale of the low-rank update; therefore, there is no universal numerical upper bound on α_r that is independent of the training state.

In our implementation, the norm-growth limiter inherited from Fira further constrains the overall update magnitude. However, this limiter does not eliminate the effect described above: even when the total update norm is clipped, an excessively large α_r can change the relative direction of the update by allowing the residual component to dominate. Thus, the role of the limiter is complementary to the energy-aware gain: the former controls update magnitude, while the latter determines how much discarded information is recovered.

Finally, the theoretical analysis does not claim that a particular value of α_r must cause divergence under all training conditions. Actual training instability depends on the evolving gradient, optimizer state, and model dynamics. Instead, the analysis establishes that increasing α_r monotonically increases the residual contribution and removes any intrinsic upper bound on its relative dominance. The divergence observed for excessively large α_r in Appendix D is therefore consistent with this mechanism-level analysis. In our experiments, we use the moderate value $\alpha_r = 1.0$, which provides residual enhancement without allowing the correction to dominate the low-rank adaptive update.