LoRA-GA^2: Low Rank Adaptation with Multi-step Gradient Adaptive Alignment

Haonan He[1]*†, Xinyue Fan[2]*

[1]University of Science and Technology of China, [2]Independent Researcher
hehn@mail.ustc.edu.cn

arXiv:2608.19800v1 [cs.CL] 20 Aug 2026

Abstract

Low-Rank Adaptation (LoRA) is a prominent fine-tuning method for large models, achieving competitive performance with reduced memory overhead. However, a persistent performance gap remains between LoRA and full fine-tuning. Recent studies have sought to narrow this gap by employing one-step gradient approximations of pretrained weights to align LoRA updates with the principal directions or intrinsic dimensionalities of full fine-tuning updates. Nevertheless, these approaches fail to capture the full dynamics of the gradients. In this paper, we propose LoRA-GA^2, an effective fine-tuning algorithm that fully leverages multi-step gradient information. Specifically, we introduce a lightweight probe for multi-step gradients of pretrained weights that incurs no additional GPU memory cost and only marginal time overhead. We further employ a spectrum-aware, importance-based rank allocation and optimal initialization derived from multi-step gradients. Extensive experimental results demonstrate that LoRA-GA^2 consistently outperforms existing LoRA variants while preserving the efficiency advantages of vanilla LoRA. For instance, LoRA-GA^2 surpasses the leading baseline by an average of 0.66 points on the GLUE benchmark, and outperforms the strongest baseline by 1.03 points on GSM8K and 0.87 points on HumanEval, respectively.

1 Introduction

The advent of open-source pre-trained large language models (LLMs), such as the Llama series (Touvron et al. 2023; Dubey et al. 2024), has revolutionized natural language processing by enabling practitioners to adapt these powerful models to a wide range of downstream tasks through fine-tuning. However, full fine-tuning of LLMs entails substantial memory requirements, as modern optimizers like Adam (Kingma and Ba 2014) demand significant GPU memory to store 32-bit precision optimizer states for the entire model, often leading to out-of-memory issues under constrained resources. This memory bottleneck has spurred the development of Parameter-Efficient Fine-Tuning (PEFT) methods, among which Low-Rank Adaptation (LoRA) (Hu et al. 2021) has gained widespread adoption due to its strong empirical performance, zero inference latency, and straightforward implementation. LoRA is grounded in the hypothesis that weight updates during fine-tuning reside in low-rank subspaces. As shown in Figure 1(a), LoRA decomposes the weight update $\Delta \boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ into low-rank matrices $\boldsymbol{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\boldsymbol{B} \in \mathbb{R}^{d_{\text{out}} \times r}$ such that $\Delta \boldsymbol{W} = \frac{\alpha}{r} \boldsymbol{B}\boldsymbol{A}$, where $r \ll \min(d_{\text{out}}, d_{\text{in}})$ and α is a scaling factor. By freezing the pre-trained weights $\boldsymbol{W}_0$ and updating only $\boldsymbol{A}$ and $\boldsymbol{B}$, LoRA significantly reduces the memory footprint of optimizer states. Despite its efficiency and superior performance relative to other PEFT methods, LoRA still underperforms full fine-tuning in many scenarios (Biderman et al. 2024; Shuttleworth et al. 2025).

Over the past three years, a diverse array of LoRA variants has emerged to bridge the performance gap between LoRA and full fine-tuning. For instance, AdaLoRA (Zhang et al. 2023b) introduces an adaptive pruning strategy to eliminate less important ranks during training; LoRA+ (Hayou, Ghosh, and Yu 2024) proposes a learning rate decoupling strategy to enhance training stability by mitigating the inherent imbalance between the down-projection and up-projection low-rank matrices. Among the various LoRA-based approaches, aligning low-rank updates with full fine-tuning updates via gradient information from pre-trained weights is considered a particularly promising direction for closing the gap, as LoRA adapters can be essentially interpreted as gradient compressors (Hao et al. 2024; He et al. 2025). For example, LoRA-GA (Wang et al. 2024) presents a leading and easily implementable paradigm that computes accumulated one-step gradients prior to training and leverages the principal components to initialize the low-rank weights, effectively reducing the discrepancy between low-rank and full fine-tuning updates at the initial training step. LoRA-One (Zhang, Liu, and Chen 2025) further substantiates the theoretical and empirical validity of this paradigm.

However, existing gradient-guided methods suffer from critical limitations in both the scope of gradient information and the metrics used for rank allocation. First, single-step gradient approaches shown in the Figure 1(b) are inherently myopic; the gradient at the initial checkpoint fails to capture the complex optimization dynamics of the actual fine-tuning trajectory. Conversely, as shown in Figure 1(c), while multi-step alignment methods like LoRA-Pro (Wang et al. 2025) directly minimize the discrepancy between low-rank and full fine-tuning updates at each step, they do so at a severe cost. Modifying the low-rank gradients during training necessitates changes to the optimizer, drastically increases GPU memory consumption for optimizer states, prolongs train-

*These authors contributed equally.
†Corresponding author.

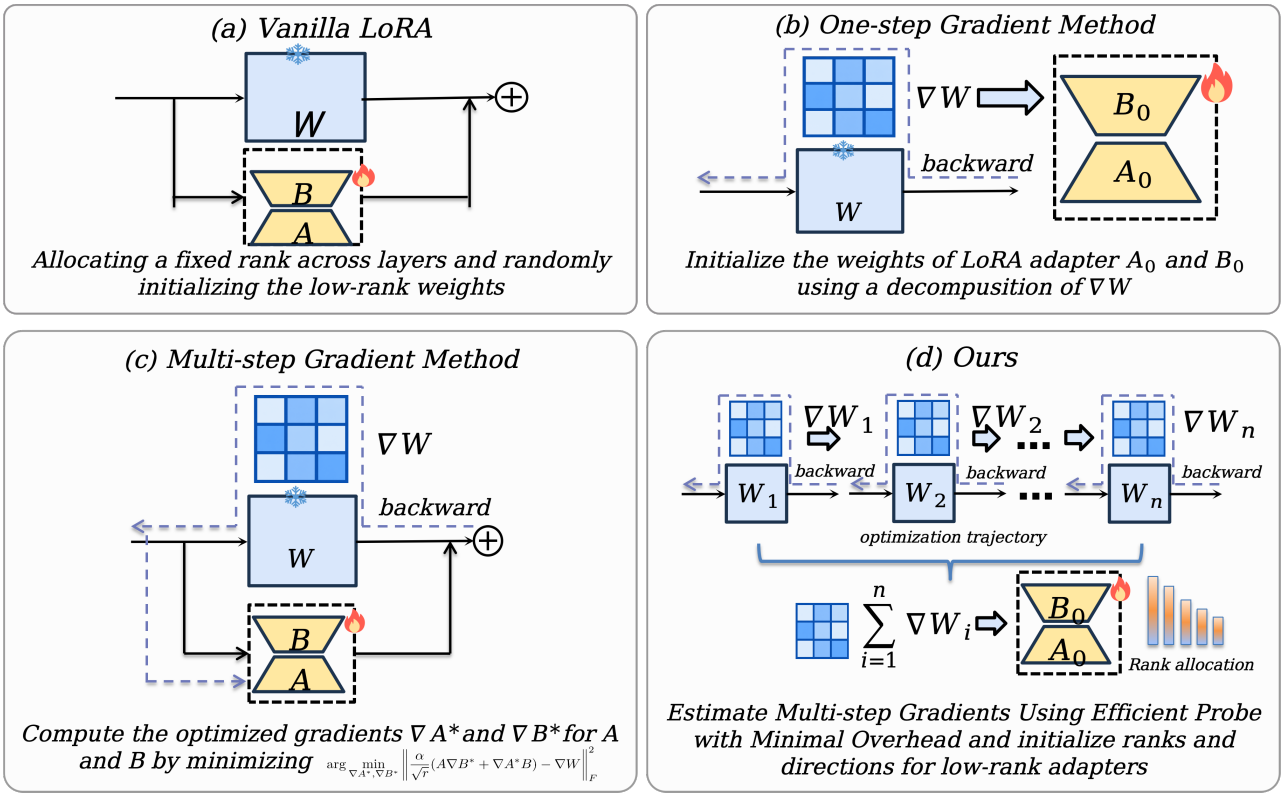

Figure 1: Illustration of (a) LoRA; (b) One-step gradients based LoRA Variants; (c) Multi-step gradients based LoRA Variants; and (d) Ours (LoRA-GA^2), which introduces an efficient probe for estimating multi-step gradients of pre-trained weights and utilizing the directions and intrinsic dimensionalities of the gradients to initialize low-rank weights.

ing time, and suffers from poor compatibility with standard training pipelines. Second, current methods rely on one-sided metrics for rank allocation that fail to comprehensively evaluate the gradient's structure. GoRA (He et al. 2025) allocates rank based solely on sensitivity-based importance scores, while RaLoRA (Ye et al. 2026) relies exclusively on intrinsic dimensionality derived from effective rank. Both metrics are suboptimal in isolation: a layer might be highly sensitive to the loss but have gradients concentrated in a single direction (requiring minimal rank capacity), or conversely, it might exhibit a high effective rank (widely dispersed gradients) but possess low overall sensitivity, making a high rank allocation wasteful. Relying on either metric alone leads to inefficient parameter distribution. This naturally raises the following question: *How can we efficiently align low-rank updates with multi-step gradients' directions, importances, and intrinsic dimensionalities without incurring unacceptable training overhead?*

To address this challenge, we employ AdaLomo (Lv et al. 2023), a memory-efficient optimizer that uses fused backward operations and in-training gradient accumulation, to serve as a lightweight probe for multi-step gradients. Unlike naive gradient averaging, we accumulate per-step gradients while updating weights along the training trajectory utilizing this lightweight optimizer, ensuring the accumulated gradient integrates information along the actual optimization path. After a few steps of accumulation, we restore the pre-trained weights and use this robust gradient for rank allocation and low-rank weight initialization. The subsequent training then proceeds seamlessly with common optimizers like Adam. This look-ahead strategy provides stable and representative gradient information that faithfully reflects the true fine-tuning dynamics, yet requires negligible time and zero permanent memory overhead. To this end, we propose **LoRA-GA**^{2} (Figure 1(d)) (**L**ow-**R**ank **A**daptation with **G**radient **A**daptive **A**lignment), a unified algorithm leveraging these multi-step accumulated gradients to simultaneously address rank allocation and initialization. Our approach introduces a novel gradient intrinsic dimensionality guided sensitivity-based importance score. The key insight is that the required rank should grow with sensitivity (more important layers need more capacity), but must be modulated by effective rank (concentrated gradients need less capacity than dispersed ones). This synergistic combination enables principled rank allocation that respects both the magnitude and the geometric structure of the gradients. For initialization, we design an SVD-based strategy that extracts principal directions from the accumulated multi-step gradients, providing a highly stable approximation of effective update directions without manipulating pre-trained weights. In summary, our core contributions are:

- We systematically identify the informational deficiency

of existing single-step gradient methods and the crippling computational overhead of continuous multi-step alignment methods. Furthermore, we reveal the theoretical blind spots of using solely sensitivity (e.g., GoRA) or effective rank (e.g., RaLoRA) for rank allocation.

- We introduce a lightweight, multi-step gradient probing and initialization method, LoRA-GA^2. By simulating the initial training trajectory via AdaLomo without permanent weight modifications, LoRA-GA^2 extracts stable multi-step gradient information for rank allocation and initialization. This approach achieves superior empirical performance with a minimal time cost, no extra memory overhead, and seamless compatibility with modern distributed training frameworks.
- We comprehensively evaluate LoRA-GA^2 across diverse modalities, model architectures, and task complexities. Experimental results demonstrate that our method achieves the best average performance among the compared methods across natural language understanding (GLUE benchmark via T5), mathematical reasoning and code generation (GSM8K and HumanEval), and computer vision (image classification via CLIP), outperforming a wide array of recent initialization-optimized, convergence-optimized, and adaptive LoRA variants.

2 Related Work

2.1 LoRA Variants

LoRA (Hu et al. 2021) has inspired numerous variants, which can be broadly categorized into three directions (Mao et al. 2024). Rank augmentation methods improve parameter utilization by dynamically reallocating ranks based on importance scores (Zhang et al. 2023b,a; He et al. 2025) or by stacking low-rank subspaces to increase effective rank (Ren et al. 2024; Lialin, Deshpande, and Rumshisky 2023). Training dynamics optimization methods focus on stabilizing the learning process through improved scaling factors (Kalajdzievski 2023) or asymmetric learning rates (Hayou, Ghosh, and Yu 2024). Initialization-based methods (Meng, Wang, and Mu 2024; Büyükakyüz 2024; Wang et al. 2024) leverage singular value decomposition or QR decomposition to initialize low-rank matrices, accelerating convergence and improving adaptation.

2.2 LoRA and Gradients of Pre-trained Weights

Beyond this simple taxonomy, the connection between LoRA and gradient dynamics of pretrained weights has garnered increasing attention. FLoRA (Hao et al. 2024) interprets LoRA as a gradient compressor, demonstrating that low-rank matrices accumulate compressed gradient information during training. GaLore (Zhao et al. 2024) projects gradients onto low-rank subspaces for memory-efficient training. LoRA-GA (Wang et al. 2024) initializes LoRA matrices using gradient singular features to minimize the difference between LoRA and full fine-tuning. GoRA (He et al. 2025) employs gradient sensitivity for adaptive rank allocation and pseudo-inverse initialization. However, these methods rely on single-step gradient estimates, which exhibit high variance and may not capture the persistent update directions essential for effective fine-tuning.

3 Method

3.1 LoRA as Gradient Compression

LoRA can be analyzed as a structured gradient-compression mechanism. For a pretrained matrix $\boldsymbol{W}_0$ and a rank-r adapter, the effective update after T steps is constrained to the product space spanned by $\boldsymbol{BA}$. Under the standard small-step approximation, or in the common setting where one low-rank factor is initialized to zero and the other determines the initial subspace, the accumulated LoRA update can be written as

$$\Delta \boldsymbol{W}_T \approx -\eta(\frac{\alpha}{r})^2 \sum_{t=1}^{T} \boldsymbol{G}_t \boldsymbol{A}_0^\top \boldsymbol{A}_0, \tag{1}$$

where $\boldsymbol{G}_t = \partial \boldsymbol{L}_t / \partial \boldsymbol{W}_0$ is the full-weight gradient. Thus, LoRA does not simply reduce parameter count; it projects the full gradient sequence through the adapter subspace. Vanilla random initialization makes this projection generic, so the approximation quality depends on whether $\boldsymbol{A}_0^\top \boldsymbol{A}_0$ happens to preserve the dominant update directions.

This view exposes two controllable error sources. First, the adapter subspace should align with the persistent gradient directions that appear during full fine-tuning process, not only with the directions of one-step gradients that computed without optimization steps. Second, the rank assigned to each layer should reflect how much gradient energy must be preserved in that layer. If $\boldsymbol{G}^l$ has singular values $\sigma_1 \geq \sigma_2 \geq \cdots$, then the best rank-$r$ approximation preserves energy $\sum_{i=1}^{r} \sigma_i^2 / \sum_j \sigma_j^2$. A layer whose gradient spectrum is concentrated can be represented with a small rank, while a layer with dispersed singular values needs more capacity. LoRA-GA^2 directly targets these two errors by estimating trajectory gradients and using their spectra for both allocation and initialization.

3.2 Trajectory-Aware Gradient Probe

For a layer with frozen pretrained weight $\boldsymbol{W}_0$, LoRA computes

$$\boldsymbol{h} = \boldsymbol{W}_0 \boldsymbol{x} + \frac{\alpha}{r} \boldsymbol{BAx}. \tag{2}$$

The LoRA gradients are $\partial \boldsymbol{L} / \partial \boldsymbol{A} = \frac{\alpha}{r} \boldsymbol{B}^\top \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{W}_0}$ and $\partial \boldsymbol{L} / \partial \boldsymbol{B} = \frac{\alpha}{r} \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{W}_0} \boldsymbol{A}^\top$, so adapter learning is controlled by how well the low-rank factors represent the full gradient subspace. Instead of estimating this subspace from a single batch, LoRA-GA^2 collects

$$\boldsymbol{G}_{\text{avg}}^l = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \boldsymbol{L}_i}{\partial \boldsymbol{W}^l} \Big|_{\boldsymbol{W}^l = \boldsymbol{W}^{(i),l}}, \tag{3}$$

$$\boldsymbol{W}^{(i+1),l} = \boldsymbol{W}^{(i),l} - \eta \mathcal{U}(\boldsymbol{G}_i^l). \tag{4}$$

where $\mathcal{U}$ is the AdaLomo update (Lv et al. 2023). Gradients are accumulated on CPU and the pretrained weights are restored after probing, thereby eliminating the parameter changes introduced during the probe. The accumulated gradients are retained only for rank allocation and adapter initialization.

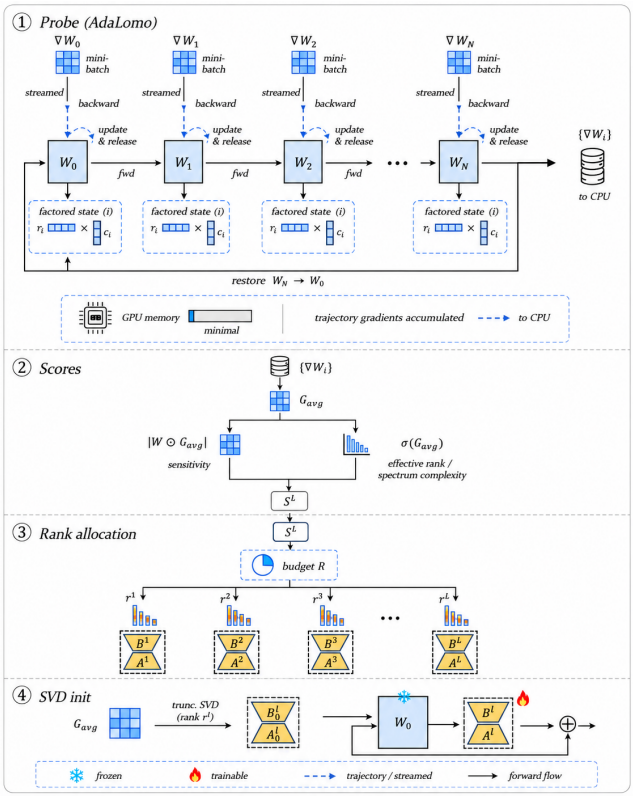

Figure 2: Framework of LoRA-GA^2. A temporary multi-step probe collects trajectory gradients, restores the pretrained weights, and uses the accumulated signal for both rank allocation and SVD initialization before standard LoRA training.

3.3 Spectrum-Aware Rank Allocation

Sensitivity measures task importance:

$$I^l_{\text{sens}} = \text{avg}(|\boldsymbol{W}^l_0 \odot \boldsymbol{G}^l_{\text{avg}}|). \tag{5}$$

This quantity approximates the first-order loss variation induced by perturbing the pretrained weights, and therefore identifies layers where downstream adaptation has large effect. However, sensitivity alone is not a rank requirement. Two layers can have the same gradient magnitude but very different singular-value spectra: if one gradient is nearly rank-one, a small adapter can capture most of its useful update; if another distributes energy across many directions, the same rank will lose much more information.

Effective rank measures how many directions are needed to represent the accumulated gradient:

$$\text{erank}(\boldsymbol{G}^l_{\text{avg}}) = \exp\left(-\sum_i p_i \log p_i\right), \qquad p_i = \frac{\sigma_i}{\sum_j \sigma_j}. \tag{6}$$

Effective rank alone is also insufficient because it ignores task importance. A layer may have a broad gradient spectrum but contribute little to the downstream loss, in which case assigning a large rank wastes the fixed LoRA budget. The two metrics therefore capture orthogonal properties: sensitivity measures *how much* the layer matters, while effective rank measures *how many directions* are needed to represent the useful update.

We normalize both scores across target layers and define

$$S^l = \bar{I}^l_{\text{sens}}(\bar{I}^l_{\text{erank}})^\lambda, \qquad \bar{S}^l = \frac{S^l}{\sum_k S^k}. \tag{7}$$

Given reference rank r_{ref}, the total budget is $P_{\text{total}} = \sum_l \sqrt{d^l_{\text{in}} + d^l_{\text{out}}}\, r_{\text{ref}}$, and the allocated rank is

$$r^l = \text{clip}\left(\left\lfloor \frac{P_{\text{total}} \bar{S}^l}{\sqrt{d^l_{\text{in}} + d^l_{\text{out}}}} \right\rfloor, r_{\min}, r_{\max}\right). \tag{8}$$

The square-root shape factor prevents wide layers from consuming the entire budget.

3.4 SVD Initialization

For each layer, we compute a truncated SVD of the negative accumulated gradient $\boldsymbol{D}^l_{\text{avg}} = -\boldsymbol{G}^l_{\text{avg}} \approx \boldsymbol{U}^l \boldsymbol{\Sigma}^l (\boldsymbol{V}^l)^\top$ and initialize the LoRA factors as

$$\boldsymbol{B}^l_0 = \boldsymbol{U}^l \sqrt{\boldsymbol{\Sigma}^l} / \sqrt{\sigma^l_1 \gamma}, \qquad \boldsymbol{A}^l_0 = \sqrt{\boldsymbol{\Sigma}^l} (\boldsymbol{V}^l)^\top / \sqrt{\sigma^l_1 \gamma}, \tag{9}$$

where σ^l_1 is the largest singular value of $\boldsymbol{D}^l_{\text{avg}}$. This places the initial adapter update along dominant descent directions. In the main experiments, we use $N = 64$, $r_{\text{ref}} = 8$, and tune γ in a small stability range. Formal proofs and the approximation-error bound are provided in the supplement.

3.5 Why Multi-Step Alignment Helps

The one-step estimator averages gradients only at $\boldsymbol{W}_0$, whereas LoRA training immediately moves through a sequence of nearby weights. For the quadratic loss $\mathcal{L}(\boldsymbol{W}) = \frac{1}{2}\|\boldsymbol{W} - \boldsymbol{W}^\star\|^2_F$, a trajectory estimator has error bounded by a variance term plus a controllable drift term, $\sigma^2 d/N + \mathcal{O}(\eta^2 N^2)$, while a static estimator retains the irreducible bias $\|\boldsymbol{W}_0 - \boldsymbol{W}^\star\|^2_F$. This stylized result explains why using many mini-batches at a frozen point is not equivalent to probing the early optimization path.

The dual rank score is motivated by a complementary observation. Sensitivity alone can over-allocate rank to layers whose gradients are large but nearly one-dimensional; effective rank alone can over-allocate rank to layers whose gradients are dispersed but irrelevant to the downstream loss. Multiplying normalized sensitivity with normalized effective rank gives each layer capacity only when both conditions hold. The supplement provides the full compression-error and allocation derivations.

4 Experiments

We evaluate LoRA-GA^2 in three settings: T5-Base (Raffel et al. 2019) on five tasks from GLUE (Wang et al. 2019), Llama-3.1-8B-Base (Dubey et al. 2024) on GSM8K (Cobbe et al. 2021) and HumanEval (Chen et al. 2021) (trained on MetaMathQA-100K (Yu et al. 2024) and CodeFeedback-100K (Weyssow et al. 2024), respectively), and CLIP-ViT-B/16 (Radford et al. 2021) on seven image classification tasks (the classifier is constructed using text prompts such as "a photo of a {class}.") (Krause et al. 2013; Cimpoi et al.

Algorithm 1 LoRA-GA^2: Low Rank Adaptation with Multi-step Gradient Adaptive Alignment

Require: Model f, weights $\{\boldsymbol{W}_0^l\}_{l=1}^L$, steps N, r_{ref}, γ, $r_{\min}, r_{\max}$, λ.
Ensure: Initialized $\{\boldsymbol{A}_0^l, \boldsymbol{B}_0^l\}_{l=1}^L$.

Phase 1: Gradient Accumulation (AdaLomo)
Save pretrained weights $\boldsymbol{W}_0^l$ to CPU, and set $\boldsymbol{G}_{\text{avg}}^l \leftarrow \boldsymbol{0}$ for all l.
for $i = 1$ to N **do**
  Sample batch, compute loss $\boldsymbol{L}$, obtain $\boldsymbol{G}_i^l = \text{FusedBackward}(\boldsymbol{L}, \boldsymbol{W}^l)$ ▷ AdaLomo fused kernel
  Update $\boldsymbol{W}^l \leftarrow \boldsymbol{W}^l - \eta \mathcal{U}(\boldsymbol{G}_i^l)$ and accumulate $\boldsymbol{G}_{\text{avg}}^l \mathrel{+}= \boldsymbol{G}_i^l / N$ ▷ Adaptive LR & CPU accumulation
end for
Restore pretrained weights: $\boldsymbol{W}^l \leftarrow \boldsymbol{W}_{\text{CPU}}^l$.
Phase 2: Importance Scores
Compute sensitivity $I_{\text{sens}}^l = \text{avg}(|\boldsymbol{W}_0^l \odot \boldsymbol{G}_{\text{avg}}^l|)$ and effective rank $I_{\text{erank}}^l = \text{erank}(\boldsymbol{G}_{\text{avg}}^l)$.
Normalize: $\bar{S}^l = \dfrac{\bar{I}_{\text{sens}}^l \cdot (\bar{I}_{\text{erank}}^l)^\lambda}{\sum_k \bar{I}_{\text{sens}}^k \cdot (\bar{I}_{\text{erank}}^k)^\lambda}$ ▷ $\bar{I}$ denotes min-max normalization across layers
Phase 3: Rank Allocation
Total budget $P_{\text{total}} \leftarrow \sum_l \sqrt{d_{\text{in}}^l + d_{\text{out}}^l} \cdot r_{\text{ref}}$.
for $l = 1$ to L **do**
  $r^l \leftarrow \text{clip}\left(\left\lfloor \dfrac{P_{\text{total}} \cdot \bar{S}^l}{\sqrt{d_{\text{in}}^l + d_{\text{out}}^l}} \right\rfloor,\ r_{\min},\ r_{\max}\right)$
end for
Phase 4: SVD-based Initialization
for $l = 1$ to L **do**
  $(\boldsymbol{U}^l, \boldsymbol{\Sigma}^l, \boldsymbol{V}^l) \leftarrow \text{SVD}_{r^l}(\boldsymbol{G}_{\text{avg}}^l)$ ▷ Truncated SVD
  $\boldsymbol{B}_0^l \leftarrow \boldsymbol{U}^l \sqrt{\boldsymbol{\Sigma}^l} / \sqrt{\sigma_1^l \gamma}, \quad \boldsymbol{A}_0^l \leftarrow \sqrt{\boldsymbol{\Sigma}^l} (\boldsymbol{V}^l)^\top / \sqrt{\sigma_1^l \gamma}$ ▷ σ_1^l = largest singular value
end for
return $\{\boldsymbol{A}_0^l, \boldsymbol{B}_0^l\}_{l=1}^L$

2014; Helber et al. 2019; Houben et al. 2013; Cheng, Han, and Lu 2017; Xiao et al. 2010; Netzer et al. 2011). Unless otherwise specified, fixed-rank LoRA variants use rank 8, while adaptive-rank variants use a reference rank of 8 with a minimum rank of 4 and a maximum rank of 32. We use the same optimizer settings as the corresponding baselines to ensure a fair comparison. All experiments are conducted on a single node equipped with four NVIDIA H200 GPUs. Each experiment is repeated using three different random seeds.

4.1 Experimental Settings

For GLUE, we follow the settings of LoRA-GA (Wang et al. 2024) and GoRA (He et al. 2025): the Adam optimizer with a peak learning rate of 1×10^{-4}, a batch size of 32, one training epoch, zero weight decay, cosine learning-rate decay with a warmup ratio of 0.03, a maximum sequence length of 128, and FP32 training. LoRA adapters are applied to all linear layers except the language head. For GSM8K (evaluated using accuracy) and HumanEval (evaluated using pass@1), we follow the setup of GoRA: AdamW with a peak learning rate of 5×10^{-5}, a batch size of 64, one training epoch, zero weight decay, a maximum sequence length of 1024, BF16 model weights, and FP32 low-rank weights, with LoRA applied to the attention projection matrices. For CLIP-ViT-B/16, we follow RaLoRA (Ye et al. 2026) and fine-tune all linear layers using Adam with a peak learning rate of 1×10^{-4} and a batch size of 64. The number of training epochs for each image classification dataset is computed as $\max\left(1, \text{round}\left(\frac{1000}{N_{\text{batch}}}\right)\right)$, where N_{batch} denotes the corresponding number of batches. Unless otherwise stated, we use a LoRA rank of 8, a LoRA scaling parameter α of 16, and a LoRA dropout rate of 0 for LoRA and its variants.

4.2 Experimental Results

Language understanding. On GLUE, LoRA-GA^2 improves the average score to 88.62, outperforming GoRA by 0.66 points, LoRA-GA by 0.85 points, and full fine-tuning by 0.71 points under the reported setting. The largest gain appears on CoLA, where LoRA-GA^2 reaches 82.39, improving over LoRA-GA by 1.82 points and over GoRA by 2.53 points. This task is sensitive to linguistic acceptability patterns, so the result is consistent with our claim that a short trajectory probe captures more stable early update directions than a one-step gradient. LoRA-GA^2 also obtains the best MRPC score, suggesting that the same alignment helps on datasets with small scales.

Reasoning and code. On Llama3.1-8B-Base, LoRA-GA^2 improves over the strongest baseline GoRA by 1.03 points on GSM8K and 0.87 points on HumanEval. It also exceeds full fine-tuning on GSM8K by 0.25 points with reference rank 8, while HumanEval remains below full fine-tuning but substantially narrows the gap from LoRA's 8.54-point deficit to 1.78 points. This pattern matches the motivation of the method: reasoning and code tasks expose the weakness of a randomly initialized low-rank subspace, and aligning the adapter with multi-step full-gradient directions helps recover part of the full fine-tuning trajectory without changing the formal LoRA training loop. As shown in Figure 3, LoRA-GA^2 shows clear optimization and convergence advantages over LoRA for fine-tuning Llama3.1-8B-Base on MetamathQA, demonstrating the effectiveness of our method.

Table 1: Performance of fine-tuning T5-Base on 5 sub-tasks of the GLUE benchmark. **Bold** and underline indicate the highest and second-highest scores of low-rank methods with $r = 8$ or $r^{\text{ref}} = 8$.

Method	MNLI	SST-2	CoLA	QNLI	MRPC	Average
Full	86.33±0.00	94.75±0.21	80.70±0.24	93.19±0.22	84.56±0.73	87.91
LoRA (Hu et al. 2021)	85.30±0.04	94.04±0.11	69.35±0.05	93.19±0.22	84.56±0.73	85.29
Convergence Optimization Methods for LoRA						
rsLoRA (Kalajdzievski 2023)	85.73±0.10	94.19±0.23	72.32±1.12	93.12±0.09	52.86±2.27	79.64
DoRA (Liu et al. 2024)	85.67±0.09	94.04±0.53	72.04±0.94	93.04±0.06	68.08±0.51	82.57
LoRA+ (Hayou, Ghosh, and Yu 2024)	85.81±0.09	93.85±0.24	77.53±0.20	93.14±0.03	74.43±1.39	84.95
Initialization Optimization Methods for LoRA						
PiSSA (Meng, Wang, and Mu 2024)	85.75±0.07	94.07±0.06	74.27±0.39	93.15±0.14	76.31±0.51	84.71
LoRA-GA (Wang et al. 2024)	85.70±0.09	94.11±0.18	80.57±0.20	93.18±0.06	85.29±0.24	87.77
Adaptive Methods for LoRA						
AdaLoRA (Zhang et al. 2023b)	85.45±0.11	93.69±0.20	69.16±0.24	91.66±0.05	68.14±0.28	81.62
RaLoRA (Ye et al. 2026)	85.76±0.03	94.22±0.29	78.11±0.45	**93.36±0.14**	84.74±0.27	87.24
GoRA (He et al. 2025)	85.91±0.22	94.68±0.43	79.86±0.35	93.27±0.08	86.10±0.20	87.96
LoRA-GA^2 (Ours)	**85.91±0.01**	**94.72±0.37**	**82.39±0.24**	93.19±0.05	**86.88±0.14**	**88.62**

Table 2: Performance of fine-tuning Llama3.1-8B-Base on GSM8K and HumanEval.

Method	GSM8K	HumanEval
Full	73.69±0.28	51.63±1.27
LoRA (Hu et al. 2021)	67.78±1.25	43.09±0.35
rsLoRA (Kalajdzievski 2023)	68.36±0.74	45.78±2.80
DoRA (Liu et al. 2024)	69.17±1.00	43.70±1.54
LoRA+ (Hayou, Ghosh, and Yu 2024)	71.29±0.93	44.51±2.11
OLoRA (Büyükakyüz 2024)	68.54±0.42	43.29±2.44
PiSSA (Meng, Wang, and Mu 2024)	68.56±1.03	44.10±1.54
LoRA-GA (Wang et al. 2024)	71.39±0.90	43.29±0.61
AdaLoRA (Zhang et al. 2023b)	70.63±0.77	41.46±3.66
RaLoRA (Ye et al. 2026)	72.25±0.59	48.78±1.61
GoRA (He et al. 2025)	72.91±0.76	48.98±2.14
LoRA-GA^2 (Ours)	**73.94±0.48**	**49.85±0.33**

Vision transfer. On CLIP-ViT-B/16, LoRA-GA^2 obtains a 91.16 average over seven image classification tasks, improving over RaLoRA by 0.63 points. It achieves the best result on six of seven datasets and is second best on RESISC45. The gains are not concentrated in a single dataset: Cars, DTD, SUN397, and SVHN all improve over the strongest baselines. This supports the claim that the combined sensitivity/effective-rank score is not tied to a single model family or modality. Since the CLIP experiments use the same reference-rank budget as the adaptive-rank baselines, the improvement is better interpreted as more effective allocation and initialization rather than a larger trainable parameter count.

5 Ablation and Efficiency

As shown in Table 4, the main ablations isolate the two design choices: trajectory gradients and dual rank scoring. Multi-step gradients improve over a fixed-weight one-step variant on both GSM8K and HumanEval (73.94 vs. 72.13 on GSM8K; 49.85 vs. 49.65 on HumanEval). Removing SVD initialization causes the largest drop (68.92 on GSM8K), while replacing the dual score with sensitivity-only or effective-rank-only allocation also degrades performance. The supplement provides more ablation results on hyperparameters.

The one-step comparison follows the same data budget as

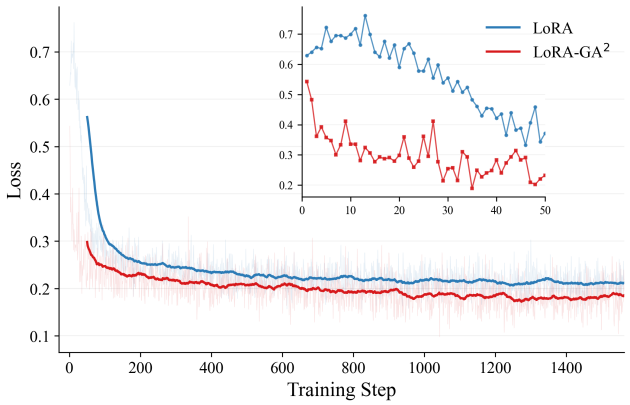

Figure 3: Loss comparison of fine-tuning Llama3.1-8B-Base on MetamathQA using LoRA and LoRA-GA^2

the trajectory probe: gradients are computed over 64 mini-batches, but the weights are kept fixed at the pretrained checkpoint. This controls for the number of gradient evaluations and isolates the effect of following the early optimization path. The 1.81-point GSM8K gap between one-step and multi-step variants indicates that the improvement is not merely due to averaging more minibatches; the location at which gradients are collected matters. HumanEval shows a smaller but consistent gain, suggesting that code generation is less sensitive to this particular trajectory signal but still benefits from more stable alignment.

The component ablations separate two roles of the accumulated gradient. Without SVD initialization, the method loses the direct alignment between the initial adapter update and the dominant full-gradient directions, causing the largest GSM8K drop. Without adaptive rank allocation, the initialization remains useful but the fixed rank budget is not redistributed toward layers that are both important and spectrally complex. The score ablations further support the dual-signal design: sensitivity-only allocation ignores directional dispersion, while effective-rank-only allocation ignores task relevance. Their weaker performance explains why LoRA-GA^2 combines both signals rather than treating either as a

Table 3: Performance of fine-tuning CLIP-ViT-B/16 on seven image classification tasks.

Method	**Cars**	**DTD**	**EuroSAT**	**GTSRB**	**RESISC45**	**SUN397**	**SVHN**	**Average**
Zero-shot	63.75	44.39	42.22	35.22	56.46	62.56	15.53	45.73
LoRA (Hu et al. 2021)	82.31±0.08	76.97±0.51	98.38±0.20	97.10±0.06	94.99±0.11	77.19±0.19	96.62±0.06	89.08±0.10
MELoRA (Ren et al. 2024)	82.65±0.38	75.16±0.59	98.64±0.05	98.88±0.05	95.78±0.16	74.69±0.22	96.95±0.09	88.96±0.15
MoRA (Jiang et al. 2024)	84.61±0.21	77.34±0.14	98.65±0.16	98.68±0.18	96.33±0.19	78.12±0.06	97.17±0.15	90.13±0.16
AdaLoRA (Zhang et al. 2023b)	73.58±0.09	73.79±0.48	96.96±0.12	58.87±0.38	89.07±0.60	72.00±0.10	94.26±0.13	79.79±0.27
DoRA (Liu et al. 2024)	82.44±0.26	76.86±0.84	98.43±0.17	97.25±0.12	95.10±0.16	77.30±0.17	96.63±0.04	89.14±0.07
rsLoRA (Kalajdzievski 2023)	83.94±0.22	77.64±0.33	98.51±0.17	98.69±0.17	95.90±0.20	77.96±0.21	96.94±0.06	89.94±0.06
LoRA+ (Hayou, Ghosh, and Yu 2024)	86.61±0.36	73.33±1.30	98.54±0.14	98.99±0.20	96.06±0.38	76.80±0.34	96.98±0.08	89.62±0.19
PiSSA (Meng, Wang, and Mu 2024)	83.36±0.38	77.38±0.57	98.54±0.09	98.32±0.09	95.92±0.40	77.46±0.13	97.00±0.09	89.71±0.25
OLoRA (Büyükakyüz 2024)	83.85±0.13	78.60±0.25	98.62±0.03	98.49±0.14	96.01±0.28	77.30±0.08	97.15±0.14	90.00±0.15
RaLoRA (Ye et al. 2026)	86.63±0.30	77.75±0.20	98.66±0.27	98.98±0.11	**96.62±0.28**	77.86±0.05	97.24±0.11	90.53±0.03
LoRA-GA^2 (Ours)	**87.82±0.13**	**79.49±0.11**	**98.83±0.11**	**99.01±0.08**	96.48±0.24	**79.05±0.14**	**97.42±0.07**	**91.16±0.05**

Table 4: Key ablations with standard deviations on Llama3.1-8B-Base.

Variant	GSM8K	HumanEval
One-step gradient	72.13±0.34	49.65±1.01
No SVD init.	68.92±0.12	48.37±2.50
No rank alloc.	72.40±1.32	48.88±0.63
Sensitivity only	72.98±0.65	48.17±2.28
Effective rank only	73.51±0.51	48.78±1.00
Full LoRA-GA^2	**73.94±0.48**	**49.85±0.33**

sufficient criterion.

The probing overhead is small relative to formal training. On Llama3.1-8B-Base, the multi-step gradient probe takes 5 minutes 55.83 seconds with 108,019 MB peak GPU memory, while the subsequent LoRA training takes 31 minutes 14.4 seconds with 137,695 MB peak GPU memory. Because all probe weights are restored before training, LoRA-GA^2 does not change inference cost or checkpoint format.

6 Discussion

The empirical pattern suggests that LoRA-GA^2 is most useful when the initial gradient is not a sufficient proxy for the first phase of training. This is visible on GSM8K and CoLA, where the gains over LoRA-GA and GoRA are larger than on already-saturated classification tasks. The method also clarifies the roles of sensitivity and effective rank. Sensitivity answers whether a layer matters for the downstream loss; effective rank answers how many independent directions are required to represent its gradient. A layer with high sensitivity but low effective rank should receive enough rank to capture the dominant direction, but not a large budget. A layer with high effective rank but low sensitivity should also be restrained because its broad spectrum may not correspond to useful task adaptation. The multiplicative score enforces this interaction in a simple form. An example of rank allocation results of our method is shown in Figure 4. There are two limitations of our method. First, the probe introduces a small extra probing phase. Second, the current method uses a global exponent λ for all layers and tasks. A future version could adapt λ based on gradient-spectrum concentration or use a small validation set to choose it automatically.

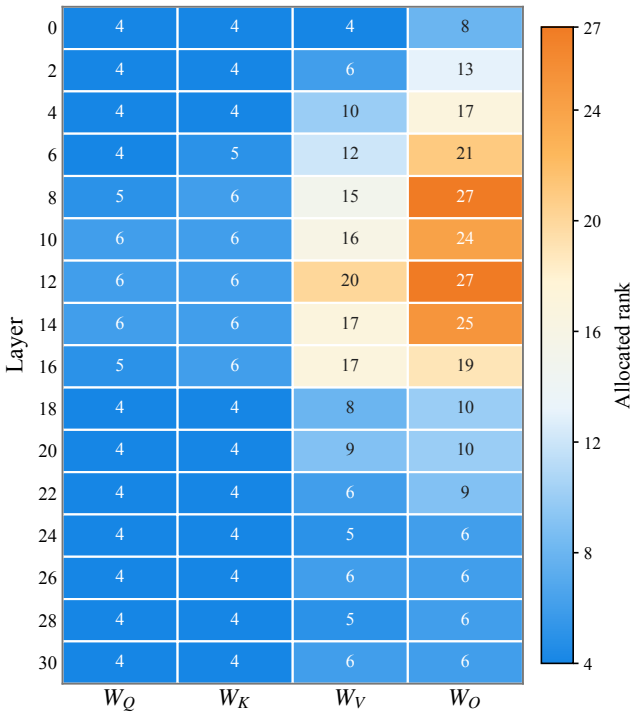

Figure 4: Rank allocation results of fine-tuning Llama3.1-8B-Base-base on MetamathQA.

7 Conclusion

We presented LoRA-GA^2, a LoRA variant with adaptive rank allocation and initialization based on multi-step gradient information. Specifically, we propose a lightweight probe for multi-step gradient approximation with no extra GPU memory cost and negligible extra computing time. Furthermore, we propose a spectrum-aware importance-based rank allocation and an SVD-based initialization for multi-step gradient alignment for improved alignment between low-rank adapters and directions, importances, and intrinsic dimensionalities of multi-step gradients. Extensive experiments demonstrate the effectiveness of our approach across diverse tasks and model architectures.

References

Biderman, D.; Portes, J.; Jacobson, J.; Baker, D.; Ramezani-Moghadam, A.; and Greene, S. 2024. LoRA Learns Less and Forgets Less. *arXiv preprint arXiv:2405.09673*.

Büyükakyüz, K. 2024. OLoRA: Orthonormal Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2406.01775*.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; de Oliveira Pinto, H. P.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; Ray, A.; Puri, R.; Krueger, G.; Petrov, M.; Khlaaf, H.; Sastry, G.; Mishkin, P.; Chan, B.; Gray, S.; Ryder, N.; Pavlov, M.; Power, A.; Kaiser, L.; Bavarian, M.; Winter, C.; Tillet, P.; Such, F. P.; Cummings, D.; Plappert, M.; Chantzis, F.; Barnes, E.; Herbert-Voss, A.; Guss, W. H.; Nichol, A.; Paino, A.; Tezak, N.; Tang, J.; Babuschkin, I.; Balaji, S.; Jain, S.; Saunders, W.; Hesse, C.; Carr, A. N.; Leike, J.; Achiam, J.; Misra, V.; Morikawa, E.; Radford, A.; Knight, M.; Brundage, M.; Murati, M.; Mayer, K.; Welinder, P.; McGrew, B.; Amodei, D.; McCandlish, S.; Sutskever, I.; and Zaremba, W. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374.

Cheng, G.; Han, J.; and Lu, X. 2017. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10): 1865–1883.

Cimpoi, M.; Maji, S.; Kokkinos, I.; Mohamed, S.; and Vedaldi, A. 2014. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3606–3613.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Hao, Y.; Liu, Y.; Li, X.; Dong, Y.; and Zhang, L. 2024. Flora: Low-Rank Adapters Are Secretly Gradient Compressors. *arXiv preprint arXiv:2402.03293*.

Hayou, S.; Ghosh, N.; and Yu, B. 2024. LoRA+: Efficient Low Rank Adaptation of Large Models. *arXiv preprint arXiv:2402.12354*.

He, H.; et al. 2025. GoRA: Gradient-driven Adaptive Low-Rank Adaptation for Large Language Models. *arXiv preprint arXiv:2502.12171*.

Helber, P.; Bischke, B.; Dengel, A.; and Borth, D. 2019. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7): 2217–2226.

Houben, S.; Stallkamp, J.; Salmen, J.; Schlipsing, M.; and Igel, C. 2013. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *The 2013 international joint conference on neural networks (IJCNN)*, 1–8. Ieee.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*.

Jiang, T.; Huang, S.; Luo, S.; Zhang, Z.; Huang, H.; Wei, F.; Deng, W.; Sun, F.; Zhang, Q.; Wang, D.; et al. 2024. Mora: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*.

Kalajdzievski, D. 2023. A Rank Stabilization Scaling Factor for Fine-Tuning with LoRA. *arXiv preprint arXiv:2312.03732*.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, 554–561.

Lialin, V.; Deshpande, N.; and Rumshisky, A. 2023. ReLoRA: High-Rank Training Through Low-Rank Updates. *arXiv preprint arXiv:2307.05695*.

Liu, S.-Y.; Wang, C.-Y.; Liu, F.; Chen, W.-Y.; et al. 2024. DoRA: Weight-Decomposed Low-Rank Adaptation. *arXiv preprint arXiv:2402.09353*.

Lv, K.; et al. 2023. AdaLomo: Low-Memory Optimization with Adaptive Learning Rate. *arXiv preprint arXiv:2310.10195*.

Mao, Y.; et al. 2024. A Survey on LoRA of Large Language Models. *arXiv preprint arXiv:2407.11046*.

Meng, F.; Wang, Z.; and Mu, C. 2024. PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models. *arXiv preprint arXiv:2404.02948*.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A. Y.; et al. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, 4. Granada.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *CoRR*, abs/1910.10683.

Ren, R.; Wang, Y.; Liu, K.; Zhao, Z.; Zhang, R.; Wei, H.; et al. 2024. MELoRA: Mini-Ensemble Low-Rank Adapters for Parameter-Efficient Fine-Tuning. *arXiv preprint arXiv:2402.17295*.

Shuttleworth, R. S.; Andreas, J.; Torralba, A.; and Sharma, P. 2025. LoRA vs Full Fine-tuning: An Illusion of Equivalence.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. arXiv:1804.07461.

Wang, S.; Li, X.; Ye, Q.; Dong, Y.; and Zhang, L. 2024. LoRA-GA: Low-Rank Adaptation with Gradient Approximation. *arXiv preprint arXiv:2407.05000*.

Wang, Z.; Liang, J.; He, R.; Wang, Z.; and Tan, T. 2025. LoRA-Pro: Are Low-Rank Adapters Properly Optimized? In *The Thirteenth International Conference on Learning Representations*.

Weyssow, M.; Kamanda, A.; Zhou, X.; and Sahraoui, H. 2024. CodeUltraFeedback: An LLM-as-a-Judge Dataset for Aligning Large Language Models to Coding Preferences. arXiv:2403.09032.

Xiao, J.; Hays, J.; Ehinger, K. A.; Oliva, A.; and Torralba, A. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, 3485–3492. IEEE.

Ye, J.; He, H.; Li, M.; Han, F.; Chen, T.; and Ye, P. 2026. Gradient Intrinsic Dimensionality Alignment: Narrowing The Gap Between Low-Rank Adaptation and Full Fine-Tuning. In *The Fourteenth International Conference on Learning Representations*.

Yu, L.; Jiang, W.; Shi, H.; Yu, J.; Liu, Z.; Zhang, Y.; Kwok, J. T.; Li, Z.; Weller, A.; and Liu, W. 2024. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. arXiv:2309.12284.

Zhang, L.; et al. 2023a. IncreLoRA: Incremental Parameter Allocation Strategy for Parameter-Efficient Fine-tuning. *arXiv preprint arXiv:2308.12043*.

Zhang, Q.; Chen, M.; Bukharin, A.; He, P.; Cheng, Y.; Chen, W.; and Zhao, T. 2023b. AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *Proceedings of the International Conference on Learning Representations*.

Zhang, Y.; Liu, F.; and Chen, Y. 2025. Lora-one: One-step full gradient could suffice for fine-tuning large language models, provably and efficiently. *arXiv preprint arXiv:2502.01235*.

Zhao, J.; Zhang, Z.; Chen, B.; Wang, Z.; and Kyrillidis, A. 2024. GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection. *arXiv preprint arXiv:2403.03507*.

A Implementation Details

A.1 Hyperparameter Selection

The key hyperparameters in LoRA-GA[2] are as follows. (1) Gradient accumulation steps N: we use $N = 64$ for stable trajectory-gradient estimation. (2) Stability scaling γ: we increase γ over a power-of-two grid until the initialized adapter satisfies the initial-loss stability criterion described below. (3) Combined-score exponent λ: the default value 1.0 balances sensitivity and effective rank. (4) Rank bounds $r_{\min}$ and $r_{\max}$: these are set relative to the reference rank r_{ref}, typically as $r_{\min} = r_{\text{ref}}/2$ and $r_{\max} = 4r_{\text{ref}}$.

As shown in Figure 5, we additionally conduct an ablation study on N and γ by training Llama-3.1-8B-Base (Dubey et al. 2024) on MetaMathQA-100K (Yu et al. 2024) and testing it on GSM8K (Cobbe et al. 2021) with a fixed random seed. Moderate values of N, such as 32 and 64, significantly improve performance. An excessively small N yields a high-variance estimate, whereas an excessively large N may include gradients from a probe trajectory that has drifted too far from the restored initialization; both effects can degrade downstream performance. Based on these observations, we use $N = 64$ in the main experiments.

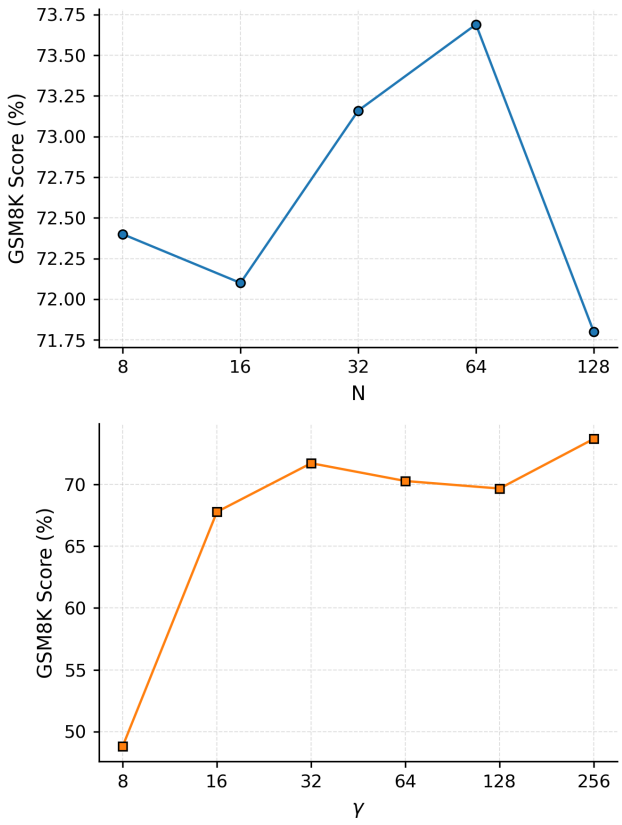

Figure 5: Ablation of the hyperparameters N and γ.

For γ, we evaluate candidates on a power-of-two grid. For each candidate, the pretrained weights are restored and the spectrally initialized LoRA branch is inserted. We accept the first candidate for which the resulting initial loss is lower than the corresponding starting loss of the restored model without a LoRA branch. This criterion directly prevents the nonzero initialization from producing an adverse initial-loss increase and requires only an initial-loss comparison for each candidate. In our search grid, small positive values up to 128 can cause a sharp loss spike; increasing γ reduces the spectral magnitude of the initialization.

A.2 Probe Protocol and Accounting

The temporary AdaLomo probe uses a fixed learning rate of 5×10^{-5}, with no learning-rate schedule during its N steps. After the probe, all pretrained weights are restored before rank allocation, adapter initialization, and formal LoRA training. The formal-training data loader is restarted rather than advanced past the probe samples; consequently, mini-

Algorithm 2 Effective-rank computation

Require: accumulated gradient $\boldsymbol{G}$, $\varepsilon = 10^{-10}$
if $\boldsymbol{G}$ is not a two-dimensional matrix **then**
  return 1
end if
cast $\boldsymbol{G}$ to FP32 on its current device
$\boldsymbol{\sigma} \leftarrow \mathrm{svdvals}(\boldsymbol{G})$ ▷ full spectrum
if the SVD fails or $|\boldsymbol{\sigma}| = 0$ **then**
  return 1
end if
$z \leftarrow \sum_i \sigma_i$
if $z < \varepsilon$ **then**
  return 1
end if
$p_i \leftarrow \sigma_i / z$
$q \leftarrow \exp(-\sum_i p_i \log(p_i + \varepsilon))$
return $\max(1, q)$

batches observed during probing may be used again during formal training.

The reported probe time covers the complete preprocessing stage: the N forward/backward passes, AdaLomo updates, CPU gradient transfer and accumulation, restoration of the pretrained weights, sensitivity and effective-rank computation, rank allocation, truncated SVD, and construction of the initialized adapters. Thus, the reported timing is not restricted to gradient collection alone.

A.3 Effective-Rank Computation

Effective rank is computed from the full singular-value spectrum in FP32 on the device holding the accumulated gradient. Algorithm 2 gives implementation-equivalent pseudocode. We use $\varepsilon = 10^{-10}$. A non-matrix input, an empty spectrum, a numerically zero nuclear norm, or an SVD failure returns the conservative fallback value 1. Temporary tensors are released after the computation.

Sensitivity and effective rank are min–max normalized across target layers before they are combined. If all target layers have identical values and this normalization is degenerate, adaptive allocation is skipped and every target layer is assigned r_{ref}. This fallback is therefore identical to the fixed-rank allocation of standard LoRA.

A.4 Negative-Gradient Spectral Initialization

Throughout the implementation, $\boldsymbol{G}^l_{\mathrm{avg}}$ denotes the accumulated loss gradient stored during the probe, while the spectral initialization is constructed from its descent direction. Explicitly, we define

$$\boldsymbol{D}^l_{\mathrm{avg}} = -\boldsymbol{G}^l_{\mathrm{avg}} = -\frac{1}{N}\sum_{i=1}^{N} \left.\frac{\partial \mathcal{L}_i}{\partial \boldsymbol{W}^l}\right|_{\boldsymbol{W}^l = \boldsymbol{W}^{(i),l}}, \tag{10}$$

and compute its rank-r^l truncated SVD:

$$\boldsymbol{D}^l_{\mathrm{avg}} \approx \boldsymbol{U}^l_r \boldsymbol{\Sigma}^l_r (\boldsymbol{V}^l_r)^\top. \tag{11}$$

Accordingly, SVD factors written compactly in terms of the accumulated gradient signal refer to the factors of $\boldsymbol{D}^l_{\mathrm{avg}} = -\boldsymbol{G}^l_{\mathrm{avg}}$. Equivalently, an implementation that decomposes $\boldsymbol{G}^l_{\mathrm{avg}}$ directly may absorb a minus sign into either one of the two LoRA factors.

The LoRA factors are initialized as

$$\boldsymbol{B}^l_0 = \frac{\boldsymbol{U}^l_r \sqrt{\boldsymbol{\Sigma}^l_r}}{\sqrt{\sigma^l_1 \gamma}}, \tag{12}$$

$$\boldsymbol{A}^l_0 = \frac{\sqrt{\boldsymbol{\Sigma}^l_r}(\boldsymbol{V}^l_r)^\top}{\sqrt{\sigma^l_1 \gamma}}, \tag{13}$$

where σ^l_1 is the largest singular value of $\boldsymbol{D}^l_{\mathrm{avg}}$. Consequently,

$$\boldsymbol{B}^l_0 \boldsymbol{A}^l_0 = \frac{(\boldsymbol{D}^l_{\mathrm{avg}})_{r^l}}{\sigma^l_1 \gamma}. \tag{14}$$

Since $\boldsymbol{D}^l_{\mathrm{avg}} = -\boldsymbol{G}^l_{\mathrm{avg}}$, the initial effective LoRA perturbation is

$$\Delta \boldsymbol{W}^l_0 = \frac{\alpha}{r^l} \boldsymbol{B}^l_0 \boldsymbol{A}^l_0 = -\frac{\alpha}{r^l \sigma^l_1 \gamma} (\boldsymbol{G}^l_{\mathrm{avg}})_{r^l}. \tag{15}$$

Thus, the nonzero adapter initialization is a controlled low-rank warm start aligned with the dominant accumulated descent directions.

After gradient probing, we restore the original pretrained weights $\boldsymbol{W}^l_0$ and keep them frozen during subsequent LoRA training. Since the adapter branch itself introduces a descent-aligned low-rank update, we do not apply an additional compensation or re-centering modification to $\boldsymbol{W}^l_0$; sufficient local descent conditions are given in Proposition 10. The adapter therefore begins training in the estimated task-relevant gradient subspace.

The normalization by $\sigma^l_1 \gamma$ controls the magnitude of the initial perturbation:

$$\left\|\boldsymbol{B}^l_0 \boldsymbol{A}^l_0\right\|_2 = \frac{1}{\gamma}, \qquad \left\|\frac{\alpha}{r^l} \boldsymbol{B}^l_0 \boldsymbol{A}^l_0\right\|_2 = \frac{\alpha}{r^l \gamma}. \tag{16}$$

Therefore, γ provides direct control over the spectral magnitude of the initialization.

Using the negative gradient does not change layer-wise sensitivity, effective rank, or rank allocation. Indeed,

$$\mathrm{avg}\left(\left|\boldsymbol{W}^l_0 \odot \boldsymbol{D}^l_{\mathrm{avg}}\right|\right) = \mathrm{avg}\left(\left|\boldsymbol{W}^l_0 \odot \boldsymbol{G}^l_{\mathrm{avg}}\right|\right), \tag{17}$$

and $\boldsymbol{D}^l_{\mathrm{avg}}$ and $\boldsymbol{G}^l_{\mathrm{avg}}$ have identical singular values. The sign convention affects only the orientation of the initialized adapter update.

B Additional Theoretical Analysis

Notation. For one adapted matrix, let

$$\widetilde{\boldsymbol{W}}_t = \boldsymbol{W}_0 + c\boldsymbol{B}_t \boldsymbol{A}_t, \qquad c := \frac{\alpha}{r}, \tag{18}$$

where $\boldsymbol{A}_t \in \mathbb{R}^{r \times d_{\mathrm{in}}}$ and $\boldsymbol{B}_t \in \mathbb{R}^{d_{\mathrm{out}} \times r}$. We write $\boldsymbol{G}_t := \nabla_{\widetilde{\boldsymbol{W}}} \mathcal{L}_t$ for the full-weight gradient evaluated at $\widetilde{\boldsymbol{W}}_t$. Unless stated otherwise, the factor optimizer is SGD with step size η.

B.1 LoRA as Gradient Compression

There are two complementary ways to derive the gradient-compression interpretation of LoRA. The first freezes one low-rank factor, $\boldsymbol{A}_t \equiv \boldsymbol{A}_0$, and trains only $\boldsymbol{B}$. This setting

yields an exact and transparent expression, but imposes a stronger assumption than standard LoRA. The second considers the usual coupled optimization of both $\boldsymbol{A}$ and $\boldsymbol{B}$ and shows that the same compressed-gradient expression remains the leading-order term over a short horizon.

Proposition 1 (Exact compression with a frozen factor)**.** *Let $\widetilde{\boldsymbol{W}}_t = \boldsymbol{W}_0 + c\boldsymbol{B}_t\boldsymbol{A}_t$, where $c = \alpha/r$. Suppose $\boldsymbol{A}_t \equiv \boldsymbol{A}_0$ and only $\boldsymbol{B}$ is trained by gradient descent. Then*

$$\widetilde{\boldsymbol{W}}_T - \widetilde{\boldsymbol{W}}_0 = -\eta c^2 \sum_{t=0}^{T-1} \boldsymbol{G}_t \boldsymbol{A}_0^\top \boldsymbol{A}_0. \tag{19}$$

Thus, LoRA applies the fixed right-side compression operator $c^2\boldsymbol{A}_0^\top\boldsymbol{A}_0$ to every full-weight gradient.

Proof. Since $\widetilde{\boldsymbol{W}}_t = \boldsymbol{W}_0 + c\boldsymbol{B}_t\boldsymbol{A}_0$, the chain rule gives $\nabla_{\boldsymbol{B}}\mathcal{L}_t = c\boldsymbol{G}_t\boldsymbol{A}_0^\top$. Hence

$$\boldsymbol{B}_T - \boldsymbol{B}_0 = -\eta c \sum_{t=0}^{T-1} \boldsymbol{G}_t \boldsymbol{A}_0^\top. \tag{20}$$

Multiplying by $c\boldsymbol{A}_0$ on the right proves equation 19. □

The preceding result is exact but assumes that the initial adapter subspace remains fixed. Standard LoRA updates both factors. The next proposition shows that, when $\boldsymbol{B}_0 = \boldsymbol{0}$ and the step is small, the frozen-subspace expression is the first nonvanishing term of the coupled dynamics.

Proposition 2 (Coupled factors: leading-order dynamics)**.** *Let $\boldsymbol{B}_0 = \boldsymbol{0}$ and update*

$$\begin{aligned} \boldsymbol{A}_{t+1} &= \boldsymbol{A}_t - \eta c \boldsymbol{B}_t^\top \boldsymbol{G}_t, \\ \boldsymbol{B}_{t+1} &= \boldsymbol{B}_t - \eta c \boldsymbol{G}_t \boldsymbol{A}_t^\top. \end{aligned} \tag{21}$$

Assume, for $0 \le t < T$, that

$$\|\boldsymbol{G}_t\|_2 \le M, \quad \|\boldsymbol{G}_t\|_F \le M_F, \quad \|\boldsymbol{A}_0\|_2 \le a, \quad \eta c M T \le 1. \tag{22}$$

Then

$$\widetilde{\boldsymbol{W}}_T - \widetilde{\boldsymbol{W}}_0 = -\eta c^2 \sum_{t=0}^{T-1} \boldsymbol{G}_t \boldsymbol{A}_0^\top \boldsymbol{A}_0 + \boldsymbol{R}_T, \tag{23}$$

where, for an absolute constant C,

$$\|\boldsymbol{R}_T\|_F \le C\eta^3 c^4 M_F M^2 a^2 T^3. \tag{24}$$

Consequently, for fixed T, the frozen-subspace expression is the leading-order term as $\eta \to 0$.

Proof. Expanding one product update gives

$$\begin{aligned} \boldsymbol{B}_{t+1}\boldsymbol{A}_{t+1} - \boldsymbol{B}_t\boldsymbol{A}_t = &-\eta c \boldsymbol{G}_t \boldsymbol{A}_t^\top \boldsymbol{A}_t - \eta c \boldsymbol{B}_t \boldsymbol{B}_t^\top \boldsymbol{G}_t \\ &+ \eta^2 c^2 \boldsymbol{G}_t \boldsymbol{A}_t^\top \boldsymbol{B}_t^\top \boldsymbol{G}_t. \end{aligned} \tag{25}$$

Under equation 22, induction yields

$$\|\boldsymbol{B}_t\|_2 \le 2\eta c M a t, \quad \|\boldsymbol{A}_t - \boldsymbol{A}_0\|_2 \le \eta^2 c^2 M^2 a\, t(t-1), \quad \|\boldsymbol{A}_t\|_2 \le 2a. \tag{26}$$

Insert these bounds into equation 25, sum over t, and multiply by the outer scale c. The difference between $\boldsymbol{A}_t^\top\boldsymbol{A}_t$ and $\boldsymbol{A}_0^\top\boldsymbol{A}_0$, the term containing $\boldsymbol{B}_t\boldsymbol{B}_t^\top$, and the quadratic term are each bounded by a constant times $\eta^3 c^4 M_F M^2 a^2 T^3$. Absorbing constants into C proves the claim. □

Remark. The assumption $\boldsymbol{B}_0 = \boldsymbol{0}$ is used only to recover the one-sided expression emphasized in the main text. Equation equation 25 holds for arbitrary initial factors. In particular, when both factors are nonzero,

$$\begin{aligned} \widetilde{\boldsymbol{W}}_1 - \widetilde{\boldsymbol{W}}_0 = &-\eta c^2 \boldsymbol{G}_0 \boldsymbol{A}_0^\top \boldsymbol{A}_0 - \eta c^2 \boldsymbol{B}_0 \boldsymbol{B}_0^\top \boldsymbol{G}_0 \\ &+ \eta^2 c^3 \boldsymbol{G}_0 \boldsymbol{A}_0^\top \boldsymbol{B}_0^\top \boldsymbol{G}_0. \end{aligned} \tag{27}$$

Thus the nonzero spectral initialization used by LoRA-GA[2] induces bidirectional compression to first order: $\boldsymbol{A}_0$ selects a right singular subspace and $\boldsymbol{B}_0$ selects a left singular subspace.

B.2 Why a Task-Aligned Subspace Matters

For a fixed gradient sequence $\{\boldsymbol{G}_t\}_{t=0}^{T-1}$, let $\boldsymbol{H}_T := \sum_{t=0}^{T-1} \boldsymbol{G}_t$. The corresponding full-SGD update is $-\eta\boldsymbol{H}_T$, whereas the frozen-factor LoRA update is $-\eta\boldsymbol{H}_T\boldsymbol{P}$, with $\boldsymbol{P} = c^2\boldsymbol{A}_0^\top\boldsymbol{A}_0$. Their discrepancy is

$$\eta\boldsymbol{H}_T(\boldsymbol{I} - \boldsymbol{P}). \tag{28}$$

Proposition 3 (Optimal rank-r compression subspace)**.** *Let $\boldsymbol{H}_T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ with singular values $\sigma_1 \ge \cdots \ge \sigma_p$. Among all rank-$r$ orthogonal projectors $\boldsymbol{P}$,*

$$\min_{\boldsymbol{P}} \|\boldsymbol{H}_T(\boldsymbol{I} - \boldsymbol{P})\|_F^2 = \sum_{i>r} \sigma_i^2. \tag{29}$$

The minimum is attained by the projector onto the top-r right singular subspace of $\boldsymbol{H}_T$.

Proof. For an orthogonal projector,

$$\|\boldsymbol{H}_T(\boldsymbol{I} - \boldsymbol{P})\|_F^2 = \sum_i \sigma_i^2 \|(\boldsymbol{I} - \boldsymbol{P})\boldsymbol{v}_i\|_2^2. \tag{30}$$

The minimum retains the r singular directions with the largest squared singular values, by the Eckart–Young–Mirsky theorem. □

Proposition 4 (Gaussian reconstruction error)**.** *Let $\boldsymbol{R} \in \mathbb{R}^{r\times d}$ have independent entries $R_{ij} \sim \mathcal{N}(0, 1/r)$. Then*

$$\mathbb{E}[\boldsymbol{R}^\top\boldsymbol{R}] = \boldsymbol{I}. \tag{31}$$

However, for every fixed matrix $\boldsymbol{G}$ with d columns,

$$\mathbb{E}\|\boldsymbol{G}\boldsymbol{R}^\top\boldsymbol{R} - \boldsymbol{G}\|_F^2 = \frac{d+1}{r}\|\boldsymbol{G}\|_F^2. \tag{32}$$

Thus Gaussian compression–decompression is unbiased but need not have small reconstruction error when $r \ll d$.

Proof. Equation equation 31 follows from $\mathbb{E}[R_{ki}R_{kj}] = \delta_{ij}/r$. For a fixed vector $\boldsymbol{x}$, rotational invariance allows $\boldsymbol{x} = \|\boldsymbol{x}\|_2\boldsymbol{e}_1$. Writing $R_{ki} = Z_{ki}/\sqrt{r}$ gives

$$\mathbb{E}\|(\boldsymbol{R}^\top\boldsymbol{R} - \boldsymbol{I})\boldsymbol{e}_1\|_2^2 = \frac{2}{r} + \frac{d-1}{r} = \frac{d+1}{r}. \tag{33}$$

Scaling by $\|\boldsymbol{x}\|_2^2$ and summing over rows proves the claim. □

B.3 Why the Multi-Step Probe Helps

The common estimation target below is the average population gradient along the early optimization trajectory. Under this target, the trajectory estimator directly samples the target and its mean-squared error contains only the sampling-variance term. The $O(\eta^2 N^2)$ discrepancy discussed in the compact main-text account corresponds to the frozen-point bias of a static estimator that replaces the trajectory average by the gradient at the restored initialization; it is not an additional error term of the trajectory estimator under the common target used here.

Proposition 5 (Trajectory versus static estimation)**.** *Vectorize the target weights and consider*

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}^\star\|_2^2, \qquad 0 < \eta < 1. \tag{34}$$

The population-gradient trajectory satisfies $\boldsymbol{g}_i = (1-\eta)^i(\boldsymbol{w}_0 - \boldsymbol{w}^\star)$*. Define*

$$\bar{\boldsymbol{g}}_N = \frac{1}{N}\sum_{i=0}^{N-1} \boldsymbol{g}_i = a_N(\boldsymbol{w}_0 - \boldsymbol{w}^\star), \qquad a_N := \frac{1-(1-\eta)^N}{\eta N}. \tag{35}$$

Assume noisy observations $\widetilde{\boldsymbol{g}}_i = \boldsymbol{g}_i + \boldsymbol{\xi}_i$ *are evaluated along this deterministic trajectory, where the* $\boldsymbol{\xi}_i$ *are independent, zero mean, and satisfy* $\mathbb{E}\|\boldsymbol{\xi}_i\|_2^2 = d\sigma^2$*. Then an* N*-sample trajectory estimator satisfies*

$$\mathbb{E}\|\widehat{\boldsymbol{g}}_{\mathrm{traj}} - \bar{\boldsymbol{g}}_N\|_2^2 = \frac{d\sigma^2}{N}. \tag{36}$$

A static estimator using M *samples at* $\boldsymbol{w}_0$ *satisfies*

$$\mathbb{E}\|\widehat{\boldsymbol{g}}_{\mathrm{static}} - \bar{\boldsymbol{g}}_N\|_2^2 = \frac{d\sigma^2}{M} + (1-a_N)^2\|\boldsymbol{w}_0 - \boldsymbol{w}^\star\|_2^2. \tag{37}$$

Thus additional static batches remove sampling variance but not the bias induced by evaluating every gradient at the initial point.

Proof. The expression for a_N follows from a geometric series. The trajectory estimator is unbiased for $\bar{\boldsymbol{g}}_N$, and independence gives equation 36. The static estimator has bias $(1-a_N)(\boldsymbol{w}_0 - \boldsymbol{w}^\star)$. The cross term between this deterministic bias and zero-mean noise vanishes, proving equation 37. □

Remark. For a short trajectory,

$$0 \le 1 - a_N \le \frac{\eta(N-1)}{2}. \tag{38}$$

Consequently,

$$\|\nabla\mathcal{L}(\boldsymbol{W}_0) - \bar{\boldsymbol{G}}_N\|_F^2 \le \frac{\eta^2(N-1)^2}{4}\|\boldsymbol{W}_0 - \boldsymbol{W}^\star\|_F^2. \tag{39}$$

For fixed N and η, this term cannot be removed by increasing the number of mini-batches evaluated at $\boldsymbol{W}_0$. Thus static batch averaging and trajectory probing remain statistically distinct even with arbitrarily many static samples. This stylized result isolates the frozen-point bias; stochastic probe dynamics would introduce additional terms.

B.4 Spectrum-Aware Rank Allocation

Proposition 6 (Rank required for energy preservation)**.** *Let* $\boldsymbol{G}$ *have singular values* $\sigma_1 \ge \cdots \ge \sigma_p \ge 0$*. For* $\rho \in (0,1]$*, define*

$$r_\rho := \min\left\{ r : \sum_{i=1}^{r}\sigma_i^2 \ge \rho\sum_{i=1}^{p}\sigma_i^2 \right\}. \tag{40}$$

There exists a rank-at-most-r *matrix* $\boldsymbol{X}$ *satisfying*

$$\|\boldsymbol{G} - \boldsymbol{X}\|_F^2 \le (1-\rho)\|\boldsymbol{G}\|_F^2 \tag{41}$$

if and only if $r \ge r_\rho$*.*

Proof. The Eckart–Young–Mirsky theorem gives

$$\min_{\mathrm{rank}(\boldsymbol{X})\le r}\|\boldsymbol{G} - \boldsymbol{X}\|_F^2 = \sum_{i>r}\sigma_i^2, \tag{42}$$

which is equivalent to equation 40. □

Proposition 7 (Need for sensitivity and spectral complexity)**.** *Let*

$$I_{\mathrm{sens}}(\boldsymbol{W}, \boldsymbol{G}) := \mathrm{avg}(|\boldsymbol{W} \odot \boldsymbol{G}|), \tag{43}$$

$$q(\boldsymbol{G}) := \exp\left(-\sum_i p_i \log p_i\right), \tag{44}$$

where $p_i = \sigma_i / \sum_j \sigma_j$ *for a nonzero spectrum. Consistent with Algorithm 2, the implementation sets* $q(\boldsymbol{G}) = 1$ *when* $\sum_j \sigma_j < \varepsilon$*. Then:*

1. $q(a\boldsymbol{G}) = q(\boldsymbol{G})$ *for* $a \ne 0$*, while* $I_{\mathrm{sens}}(\boldsymbol{W}, a\boldsymbol{G}) = |a| I_{\mathrm{sens}}(\boldsymbol{W}, \boldsymbol{G})$*.*
2. *Sensitivity alone does not determine rank demand. For* $k \ge 2$*, let* $\boldsymbol{W} = \boldsymbol{I}_k$*,* $\boldsymbol{G}_1 = \mathrm{diag}(k, 0, \ldots, 0)$*, and* $\boldsymbol{G}_2 = \boldsymbol{I}_k$*. Both sensitivities equal* $1/k$*, but* $\boldsymbol{G}_1$ *has rank one, whereas preserving a fraction* ρ *of the energy of* $\boldsymbol{G}_2$ *requires rank at least* $\lceil \rho k \rceil$*.*

Therefore a useful allocation score must depend on both task importance and spectral spread.

Proof. Scaling all singular values by $|a|$ leaves p_i unchanged. For the second claim, both elementwise sensitivities equal the sum of diagonal magnitudes divided by k^2; the spectral claims follow from Proposition 6. □

Remark. The score

$$S^l = \bar{I}^l_{\mathrm{sens}}(\bar{I}^l_{\mathrm{erank}})^\lambda \tag{45}$$

is a monotone combination of two necessary signals. Effective rank is only a scalar spectrum summary, so the product is not claimed to be the unique optimizer of the exact spectral tail.

Proposition 8 (Proportional shape-adjusted allocation)**.** *Let* $s_l > 0$ *be layer scores and* $\kappa_l > 0$ *per-rank surrogate costs. With* $b_l := \kappa_l r_l$*, consider*

$$\max_{b_l > 0}\sum_{l=1}^{L} s_l \log b_l, \qquad \textit{s.t.}\ \sum_{l=1}^{L} b_l = P. \tag{46}$$

Its unique solution is

$$r_l^* = \frac{P}{\kappa_l} \frac{s_l}{\sum_k s_k}. \tag{47}$$

Taking $s_l = S^l$ *and* $\kappa_l = \sqrt{d_{\text{in}}^l + d_{\text{out}}^l}$ *recovers the continuous, unclipped allocation used by LoRA-GA*[2].

Proof. Strict concavity gives the stationarity condition $s_l/b_l = \mu$, so $b_l = s_l/\mu$. Enforcing $\sum_l b_l = P$ yields $\mu = (\sum_k s_k)/P$ and proves equation 47. □

Remark. The square-root shape factor defines a shape-adjusted capacity budget, not the exact number of LoRA parameters. The exact per-rank parameter cost is $d_{\text{in}}^l + d_{\text{out}}^l$. After integer rounding and clipping, any residual surrogate budget can be reassigned according to the largest marginal utility gain. If the layerwise normalization is degenerate, the implementation instead uses the fixed $r_l = r_{\text{ref}}$ fallback described in Section A.3.

B.5 Theory of Negative-Gradient Initialization

The method computes the SVD of the negative trajectory gradient, making the nonzero adapter an intentional low-rank warm start rather than an arbitrary change to the restored model.

Proposition 9 (Exact scale of the spectral warm start)**.** *Let* $-\boldsymbol{G}_{\text{avg}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$*, with largest singular value* $\sigma_1 > 0$*, and write* $\boldsymbol{D}_r := \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^\top$*. Initialize*

$$\boldsymbol{B}_0 = \frac{\boldsymbol{U}_r\boldsymbol{\Sigma}_r^{1/2}}{\sqrt{\sigma_1\gamma}}, \qquad \boldsymbol{A}_0 = \frac{\boldsymbol{\Sigma}_r^{1/2}\boldsymbol{V}_r^\top}{\sqrt{\sigma_1\gamma}}. \tag{48}$$

Then

$$c\boldsymbol{B}_0\boldsymbol{A}_0 = \beta\boldsymbol{D}_r, \qquad \beta := \frac{c}{\sigma_1\gamma}, \tag{49}$$

and

$$\|c\boldsymbol{B}_0\boldsymbol{A}_0\|_2 = \frac{c}{\gamma}. \tag{50}$$

Thus γ *controls the spectral norm independently of the absolute gradient scale. If* $\sigma_1 = 0$*, the accumulated gradient vanishes and we use the zero adapter.*

Proof. Multiplying the factors gives $\boldsymbol{B}_0\boldsymbol{A}_0 = \boldsymbol{D}_r/(\sigma_1\gamma)$. Since the largest singular value of $\boldsymbol{D}_r$ is σ_1, equation 50 follows. □

Proposition 10 (Descent guarantee without modifying $\boldsymbol{W}_0$)**.** *Let* $\boldsymbol{G}_0 := \nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W}_0)$*, and suppose* $\mathcal{L}$ *is* L_{sm}*-smooth on the line segment between* $\boldsymbol{W}_0$ *and* $\boldsymbol{W}_0 + \beta\boldsymbol{D}_r$*. Define*

$$a := -\frac{\langle \boldsymbol{G}_0, \boldsymbol{D}_r\rangle_F}{\|\boldsymbol{D}_r\|_F^2}. \tag{51}$$

If $a > 0$*, then* $\widetilde{\boldsymbol{W}}_0 = \boldsymbol{W}_0 + \beta\boldsymbol{D}_r$ *satisfies*

$$\mathcal{L}(\widetilde{\boldsymbol{W}}_0) - \mathcal{L}(\boldsymbol{W}_0) \le \left(-\beta a + \frac{L_{\text{sm}}}{2}\beta^2\right)\|\boldsymbol{D}_r\|_F^2. \tag{52}$$

Hence the initialization is a strict descent step whenever

$$0 < \beta < \frac{2a}{L_{\text{sm}}}. \tag{53}$$

No compensating change to the pretrained backbone is required: the additive LoRA branch implements the controlled low-rank warm start.

Proof. Local smoothness gives

$$\mathcal{L}(\boldsymbol{W}_0 + \beta\boldsymbol{D}_r) \le \mathcal{L}(\boldsymbol{W}_0) + \beta\langle \boldsymbol{G}_0, \boldsymbol{D}_r\rangle_F + \frac{L_{\text{sm}}}{2}\beta^2\|\boldsymbol{D}_r\|_F^2. \tag{54}$$

Substituting equation 51 proves the bound, whose right-hand side is negative under equation 53. □

Corollary 11 (Exact-gradient case)**.** *If* $\boldsymbol{G}_{\text{avg}} = \boldsymbol{G}_0$*, then* $\boldsymbol{D}_r = -(\boldsymbol{G}_0)_r$ *and* $a = 1$*. Therefore it is sufficient that*

$$0 < \frac{c}{\sigma_1\gamma} < \frac{2}{L_{\text{sm}}}. \tag{55}$$

Proof. Orthogonality of singular components gives $-\langle \boldsymbol{G}_0, \boldsymbol{D}_r\rangle_F = \|(\boldsymbol{G}_0)_r\|_F^2 = \|\boldsymbol{D}_r\|_F^2$. □

Corollary 12 (Trajectory-gradient alignment)**.** *Let* $\boldsymbol{E} := \boldsymbol{G}_0 - \boldsymbol{G}_{\text{avg}}$*. Then*

$$a \ge 1 - \frac{\|\boldsymbol{E}\|_F}{\|\boldsymbol{D}_r\|_F}. \tag{56}$$

In particular, if $\|\boldsymbol{E}\|_F < \|\boldsymbol{D}_r\|_F$*, then* $a > 0$*, and choosing* γ *large enough to satisfy equation 53 guarantees descent.*

Proof. Since $\boldsymbol{D}_r = -(\boldsymbol{G}_{\text{avg}})_r$,

$$-\langle \boldsymbol{G}_0, \boldsymbol{D}_r\rangle_F = \|\boldsymbol{D}_r\|_F^2 + \langle \boldsymbol{E}, (\boldsymbol{G}_{\text{avg}})_r\rangle_F. \tag{57}$$

Apply Cauchy–Schwarz and divide by $\|\boldsymbol{D}_r\|_F^2$. □

Proposition 13 (Optimal local rank-r descent direction)**.** *Let* $\boldsymbol{G}_0$ *be the gradient at* $\boldsymbol{W}_0$ *and assume* $(\boldsymbol{G}_0)_r \neq \boldsymbol{0}$*. Among all perturbations* $\boldsymbol{\Delta}$ *with* $\operatorname{rank}(\boldsymbol{\Delta}) \le r$ *and* $\|\boldsymbol{\Delta}\|_F \le \tau$*,*

$$\min_{\boldsymbol{\Delta}} \langle \boldsymbol{G}_0, \boldsymbol{\Delta}\rangle_F = -\tau\|(\boldsymbol{G}_0)_r\|_F, \tag{58}$$

attained by

$$\boldsymbol{\Delta}^* = -\tau\frac{(\boldsymbol{G}_0)_r}{\|(\boldsymbol{G}_0)_r\|_F}. \tag{59}$$

Thus the negative-gradient singular subspace is locally optimal under rank and Frobenius-norm constraints.

Proof. Von Neumann's trace inequality and Cauchy–Schwarz give

$$|\langle \boldsymbol{G}_0, \boldsymbol{\Delta}\rangle_F| \le \sum_{i=1}^{r} \sigma_i(\boldsymbol{G}_0)\sigma_i(\boldsymbol{\Delta}) \le \|(\boldsymbol{G}_0)_r\|_F\|\boldsymbol{\Delta}\|_F. \tag{60}$$

The stated perturbation attains equality with negative sign. □

Interpretation. Early LoRA training compresses full gradients through its adapter subspace. A trajectory probe estimates directions encountered during early optimization rather than only at the restored point. The singular-value tail determines how much rank is needed in each layer, while sensitivity determines whether that layer matters to the task. Finally, the SVD of the negative trajectory gradient places the initialized adapter in a locally optimal low-rank descent subspace, and normalization by $\sigma_1\gamma$ makes the warm-start magnitude explicit and controllable.