BEYOND LOW-RANK PARAMETERIZATION: NARROWING THE GAP BETWEEN LORA AND FULL FINE-TUNING VIA GRADIENT DECOMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-Rank Adaptation (LoRA) is a widely used approach to parameter-efficient fine-tuning (PEFT), yet a performance gap can remain relative to full fine-tuning (FFT). Many LoRA variants improve the initialization or optimization of low-rank factors. At each training step, however, their first-order weight-space directions are constrained by the current parameterization. We characterize the corresponding **LoRA-accessible gradient space** and show that it coincides with the tangent space induced by the current LoRA parameterization. This characterization yields an orthogonal decomposition of the full weight gradient at the current model parameters. We term the component orthogonal to this space the **normal gradient**. Based on this decomposition, we propose **GDLoRA (Gradient-Decomposed Low-Rank Adaptation)**. GDLoRA reconstructs the full weight gradient from forward activations and backward signals, extracts its normal component, and directly updates the base weights with this component, while retaining standard AdamW optimization for the LoRA factors. GDLoRA incorporates complementary normal gradients without increasing standard LoRA's optimizer-state memory budget under matched adapter and optimizer configurations. Experiments on natural language understanding, mathematical reasoning, commonsense reasoning, and image classification show that GDLoRA consistently improves over LoRA and narrows the performance gap to FFT. The code is available at `https://anonymous.4open.science/r/GDLoRA`.

1 INTRODUCTION

Large foundation models achieve strong performance across diverse applications (Devlin et al., 2019; Brown et al., 2020; Chowdhery et al., 2023), but adapting them through full fine-tuning (FFT) incurs substantial computation and memory costs for gradients and optimizer states (Rajbhandari et al., 2020). Parameter-efficient fine-tuning (PEFT) reduces these costs by optimizing a small subset of parameters while keeping most pretrained weights frozen (Houlsby et al., 2019; Li & Liang, 2021; Lester et al., 2021). Among these methods, Low-Rank Adaptation (LoRA) (Hu et al., 2021) introduces trainable low-rank factors into pretrained layers, reducing optimizer-state memory while maintaining competitive downstream performance.

Despite these advantages, a performance gap can remain between LoRA and FFT. A related line of research seeks to enable full-rank training with reduced optimizer-state memory. Fira (Chen et al., 2024) pursues this goal by combining adaptive optimization of low-rank projected gradients with scaled residual updates. Its projection bases are obtained through SVD and reused across multiple optimization steps to amortize decomposition costs. However, as the leading gradient subspace drifts during training, reused bases may become less aligned with important directions of subsequent gradients, potentially limiting the effectiveness of adaptive optimization, while more frequent basis refreshes incur additional computational costs. A detailed analysis is provided in Appendix A. Another line of research improves LoRA optimization through adapter initialization and gradient alignment. LoRA-GA (Wang et al., 2024) uses full-gradient information to initialize the factors so that their initial induced weight-space gradient better approximates that of full fine-tuning. LoRA-Pro (Wang et al., 2025) studies the equivalent weight-space gradient induced by the adapter gradients and adjusts them to improve its approximation to the full weight gradient.

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.

Although these adapter-based methods do not require periodic full-gradient SVD refreshes, their first-order weight-space updates remain restricted to directions accessible through the current LoRA parameterization. This motivates us to investigate gradient information beyond these accessible directions. Although the accessible space evolves as the factors change, gradient components orthogonal to it cannot be expressed through first-order factor perturbations at the current iterate. We therefore ask: ***Can we explicitly recover and exploit this complementary gradient information while retaining standard LoRA's optimizer-state memory budget?***

To address this question, we characterize the **LoRA-accessible gradient space**, consisting of all equivalent weight-space gradients induced by the Euclidean factor gradients, and show that it coincides with the tangent space induced by the current LoRA parameterization. This characterization yields an orthogonal decomposition of the full weight gradient evaluated at the current model parameters. We call the component orthogonal to the accessible space the **normal gradient**. When nonzero, its negative provides a local descent direction for the current batch loss with the adapters and other weights held fixed. It is also orthogonal to every first-order weight-space direction available to the current adapters. These properties motivate using the normal component to supplement standard LoRA optimization.

Based on this decomposition, we propose **GDLoRA (Gradient-Decomposed Low-Rank Adaptation)**. GDLoRA reconstructs the full weight gradient of LoRA-injected layers from forward activations and backward signals, normalizes the recovered gradient by the number of input positions, and extracts its normal component using the current LoRA tangent space. This additional token-count normalization applies only to the normal branch and preserves the gradient direction. The adapter parameters follow standard AdamW optimization, while the token-normalized normal gradient directly updates the base weights initialized from the pretrained model. Both branches use quantities computed from the same pre-update parameters. The normal branch introduces no additional momentum or variance buffers for the base weights. In this way, GDLoRA approaches or matches the performance of full fine-tuning in the evaluated settings, without introducing additional optimizer-state memory overhead relative to standard LoRA under matched adapter and optimizer configurations.

Our contributions are summarized as follows:

- We characterize the **LoRA-accessible gradient space** through its equivalence to the tangent space induced by the current parameterization, and identify the **normal gradient** as a complementary component with a local descent interpretation.
- We propose **GDLoRA**, which retains standard AdamW optimization of the LoRA factors and applies the token-normalized normal gradient directly to the base weights, without introducing additional base-weight momentum or variance buffers.
- Across natural language understanding, mathematical and commonsense reasoning, and image classification, GDLoRA improves average scores by 2-4% relative to LoRA, approaching FFT without additional optimizer-state memory under matched configurations.

2 Related work

Parameter-Efficient Fine-Tuning. Given the increasing scale of foundation models, parameter-efficient fine-tuning (PEFT) methods have been extensively studied to reduce the computational and memory costs of model adaptation (Houlsby et al., 2019; Li & Liang, 2021; Lester et al., 2021; Hu et al., 2021). Instead of updating all parameters during fine-tuning, PEFT methods optimize only a small subset of additional parameters while keeping the pretrained model largely frozen. Existing PEFT methods can be broadly categorized into several groups. One line of research explores adapter-based tuning (Houlsby et al., 2019; Pfeiffer et al., 2020), which introduces lightweight trainable modules into pretrained networks and updates only these additional parameters during adaptation. Another line focuses on prompt-based tuning (Li & Liang, 2021; Lester et al., 2021), which adapts pretrained models by optimizing learnable prompts or virtual tokens without modifying the model parameters. In a related direction, Low-Rank Adaptation (LoRA) (Hu et al., 2021) introduces trainable low-rank matrices into pretrained layers while keeping the base weights frozen, reducing the number of optimized parameters and the associated optimizer-state memory. In this work, we focus on improving the optimization capability of LoRA-based adaptation.

Low-Rank Adaptation. Low-Rank Adaptation (LoRA) (Hu et al., 2021) adapts pretrained models through trainable low-rank updates, balancing efficiency and performance. Subsequent variants improve initialization, optimization, and adaptation capability. PiSSA (Meng et al., 2024) initializes LoRA using principal singular components of pretrained weights. LoRA-GA (Wang et al., 2024) aligns initial LoRA-induced gradients with full fine-tuning gradients, while LoRA-One (Zhang et al., 2025) studies the relationship between LoRA adapters and full-gradient subspaces. LoRA+ (Hayou et al., 2024) assigns different learning rates to the two factors, and rsLoRA (Kalajdzievski, 2023) uses rank-dependent scaling to stabilize training. AdaLoRA (Zhang et al., 2023) dynamically allocates rank budgets across layers. DoRA (Liu et al., 2024) decomposes weights into magnitude and direction, while LoRA-Pro (Wang et al., 2025) adjusts adapter gradients to better approximate full weight gradients. GaLore (Zhao et al., 2024) reduces optimizer-state memory via low-rank gradient projection; Fira (Chen et al., 2024) adds scaled residual updates for full-rank training. Both reuse SVD bases, trading subspace alignment against refresh costs. GDLoRA instead retains standard LoRA optimization and derives projection bases from the current adapters each step, updating base weights along the normal gradient without full-gradient SVD.

3 METHOD

3.1 FULL GRADIENT RECOVERY

GDLoRA supplements standard LoRA optimization with gradient information orthogonal to the first-order directions accessible through the current LoRA parameterization. We first describe full weight-gradient recovery and the token-count normalization used by the normal branch.

For a LoRA-injected linear layer, let $W_{\mathrm{pt}} \in \mathbb{R}^{d_o \times d_i}$ denote the pretrained weight, and let $W_{\mathrm{base},t}$ denote the base weight at iteration t, initialized as $W_{\mathrm{base},0} = W_{\mathrm{pt}}$. The effective weight is

$$W'_t = W_{\mathrm{base},t} + sB_tA_t, \tag{1}$$

where $B_t \in \mathbb{R}^{d_o \times r}$ and $A_t \in \mathbb{R}^{r \times d_i}$ are the LoRA factors, and $s \neq 0$ is a fixed scaling factor (e.g., $s = \alpha/r$).

For input activations $X \in \mathbb{R}^{N \times d_i}$, the batch and token dimensions are flattened into N rows. The layer output is

$$Y = X(W')^T \in \mathbb{R}^{N \times d_o}. \tag{2}$$

Let

$$\delta = \frac{\partial \mathcal{L}}{\partial Y} \in \mathbb{R}^{N \times d_o} \tag{3}$$

denote the backward signal. By the chain rule, the full weight gradient is

$$G_{\mathrm{raw}} = \frac{\partial \mathcal{L}}{\partial W'} = \delta^T X \in \mathbb{R}^{d_o \times d_i}. \tag{4}$$

Here, G_{raw} is the unconstrained weight gradient at the current model parameters, and any loss scaling applied before backpropagation is already reflected in δ.

For the normal branch, GDLoRA further normalizes each recovered gradient by the number of input positions:

$$G = \frac{G_{\mathrm{raw}}}{N} = \frac{\delta^T X}{N}. \tag{5}$$

In preliminary runs, we observed unstable training with unnormalized normal-gradient updates, and selecting a suitable learning rate was difficult within the settings explored. This motivated scaling the normal gradient by $1/N$, where N counts the input positions. This preserves its direction and controls the effective step size, equivalently rescaling the learning rate for fixed N. Adapter gradients remain unchanged. Appendix H provides a magnitude bound and local descent analysis. For the following local analysis, let $\overline{\mathcal{L}} = \mathcal{L}/N$, with N fixed for the current batch. Adapter optimization continues to use $\mathcal{L}$.

Gradient recovery adds a matrix multiplication relative to standard LoRA but requires no momentum or variance buffers for the base weights. Full-weight gradient computation is also required by Fira (Chen et al., 2024). GDLoRA constructs its projection bases from thin adapter factors, avoiding SVD of the full gradient. The computational comparison is provided in Appendix B.

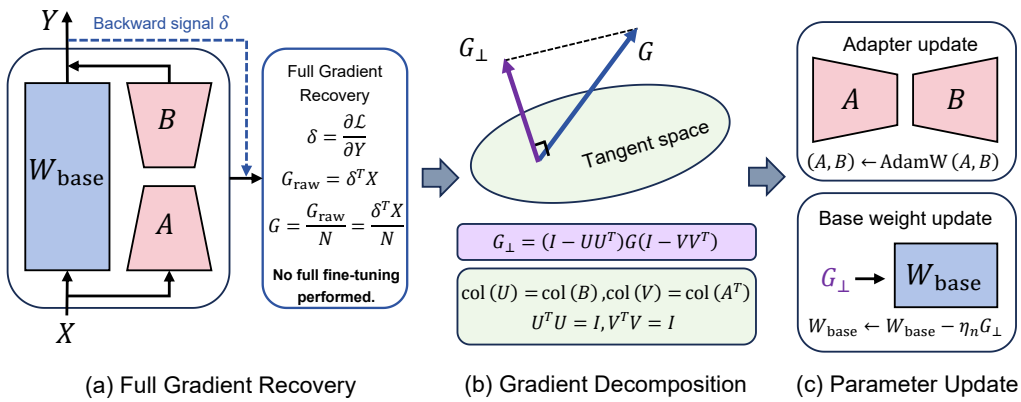


Figure 1: **Overview of GDLoRA. (a)** GDLoRA reconstructs the full weight gradient at the current model parameters from forward activations and backward signals, and obtains G as defined in Section 3.1. **(b)** The gradient is decomposed with respect to the current parameterization-induced LoRA tangent space, and its orthogonal component $G_\perp$ is the *normal gradient*. **(c)** The adapters follow standard AdamW optimization, while $G_\perp$ updates W_{base} without additional base-weight momentum or variance buffers. The two branches are complementary at first order at the current iterate.

3.2 Gradient Decomposition

Our goal is to add a gradient component that is orthogonal to all first-order weight-space directions available to the current adapters, while retaining their standard optimization rule. This provides a precise notion of complementary information; overlap with adapter directions is not, by itself, assumed to be harmful.

First-order accessible directions. For the parameterization $F_s(A, B) = sBA$, its differential maps factor perturbations to weight-space directions as

$$J_s(H_A, H_B) = DF_s(A, B)[H_A, H_B] = s(BH_A + H_B A). \tag{6}$$

We define the parameterization-induced tangent space by

$$\mathcal{T}_{\text{LoRA}} = \text{Range}(J_s) = \left\{ s(BH_A + H_B A) \;\middle|\; H_A \in \mathbb{R}^{r \times d_i},\, H_B \in \mathbb{R}^{d_o \times r} \right\}. \tag{7}$$

Throughout, "accessible" refers to these first-order directions at the current (A, B). This definition remains valid for rank-deficient factors. When both factors have rank r, it coincides with the usual tangent space of the rank-r matrix manifold at sBA, as used in geometric approaches to LoRA optimization.

By the chain rule, the factor gradients are

$$\nabla_A \mathcal{L} = sB^T G_{\text{raw}}, \qquad \nabla_B \mathcal{L} = sG_{\text{raw}} A^T. \tag{8}$$

Mapping these Euclidean gradients through J_s and dividing by N gives the first-order weight-space gradient direction

$$\begin{aligned} G_{\text{LoRA}}(G) &= \frac{s}{N}\left(B\nabla_A \mathcal{L} + \nabla_B \mathcal{L} A\right) \\ &= s^2\left(BB^T G + GA^T A\right). \end{aligned} \tag{9}$$

This is the Euclidean-gradient-induced direction underlying the equivalent-gradient view of LoRA; it is not the actual AdamW update. Define the corresponding accessible gradient space as

$$\mathcal{G}_{\text{LoRA}} = \text{Range}(\Phi_s), \qquad \Phi_s(H) = s^2\left(BB^T H + HA^T A\right). \tag{10}$$

Let $U \in \mathbb{R}^{d_o \times r_B}$ and $V \in \mathbb{R}^{d_i \times r_A}$ be orthonormal bases satisfying

$$\begin{aligned} U^T U = I, \qquad V^T V = I, \\ \text{col}(U) = \text{col}(B), \qquad \text{col}(V) = \text{col}(A^T), \end{aligned} \tag{11}$$

where $r_B = \mathrm{rank}(B)$ and $r_A = \mathrm{rank}(A)$. A zero-rank factor has an empty basis and a zero associated projector.

Theorem 1 (LoRA-Accessible Gradient Space Characterization)**.** *For any A, B and fixed $s \neq 0$,*

$$\mathcal{G}_{\mathrm{LoRA}} = \mathcal{T}_{\mathrm{LoRA}} = \left\{ UK + LV^T \;\middle|\; K \in \mathbb{R}^{r_B \times d_i},\, L \in \mathbb{R}^{d_o \times r_A} \right\}. \tag{12}$$

The proof is provided in Appendix C. This characterization identifies the orthogonal complement of the first-order LoRA-accessible space used in the following decomposition.

Normal-gradient projection. The operator Φ_s is generally not an orthogonal projector. Consequently, subtracting $G_{\mathrm{LoRA}}(G)$ from G does not generally produce a residual orthogonal to $\mathcal{G}_{\mathrm{LoRA}}$. The matrices BB^T and A^TA are also generally not orthogonal projectors and therefore cannot directly serve as the column- and row-space projectors required for this decomposition.

By Theorem 1, the LoRA-accessible gradient space is exactly the tangent space induced by the current LoRA parameterization. Thus, projecting G onto $\mathcal{G}_{\mathrm{LoRA}}$ and its orthogonal complement is equivalent to projecting it onto $\mathcal{T}_{\mathrm{LoRA}}$ and $\mathcal{T}^{\perp}_{\mathrm{LoRA}}$, respectively. Using the orthonormal bases U and V, the corresponding column- and row-space projectors are UU^T and VV^T, yielding the orthogonal decomposition

$$G = G_{\parallel} + G_{\perp}, \tag{13}$$

$$G_{\parallel} = UU^TG + GVV^T - UU^TGVV^T, \tag{14}$$

$$G_{\perp} = (I - UU^T)G(I - VV^T). \tag{15}$$

These are the standard two-sided tangent and normal projectors in the full-rank-factor case; the basis formulation also covers rank-deficient factors under our first-order-space definition. In particular,

$$G_{\parallel} \in \mathcal{T}_{\mathrm{LoRA}}, \qquad G_{\perp} \in \mathcal{T}^{\perp}_{\mathrm{LoRA}}, \qquad \langle G_{\perp}, Z \rangle_F = 0 \quad \forall Z \in \mathcal{T}_{\mathrm{LoRA}}. \tag{16}$$

We call $G_{\perp}$ the **normal gradient**. A direct verification of the projection is provided in Appendix D.

Why use the normal component? The normal update has a simple local optimization interpretation. Let D denote an additive update to W_{base}. With the adapters and all other weights fixed, the first-order change in $\overline{\mathcal{L}}$ is $\langle G, D \rangle_F$. We seek an update orthogonal to the current first-order adapter space while penalizing its magnitude:

$$\underset{D \in \mathcal{T}^{\perp}_{\mathrm{LoRA}}}{\arg\min} \left\{ \langle G, D \rangle_F + \frac{1}{2\eta_n} \|D\|_F^2 \right\} = -\eta_n G_{\perp}, \qquad \eta_n > 0. \tag{17}$$

To see this, orthogonality gives $\langle G_{\parallel}, D \rangle_F = 0$ and hence $\langle G, D \rangle_F = \langle G_{\perp}, D \rangle_F$. Completing the square yields

$$\langle G, D \rangle_F + \frac{1}{2\eta_n} \|D\|_F^2 = \frac{1}{2\eta_n} \|D + \eta_n G_{\perp}\|_F^2 - \frac{\eta_n}{2} \|G_{\perp}\|_F^2.$$

The last term is independent of D, and the squared term is minimized at $D = -\eta_n G_{\perp}$.

Furthermore, treating $G_{\perp}$ as a fixed direction computed at the current parameters, differentiability gives

$$\frac{d}{d\eta} \overline{\mathcal{L}}(W_{\mathrm{base}} - \eta G_{\perp}, A, B) \bigg|_{\eta=0} = -\langle G, G_{\perp} \rangle_F = -\|G_{\perp}\|_F^2. \tag{18}$$

Thus, whenever $G_{\perp} \neq 0$, a sufficiently small step along $-G_{\perp}$ decreases the current batch loss while remaining orthogonal to the current first-order adapter space. This local property motivates the decomposition.

3.3 Parameter Update

At iteration t, GDLoRA computes G_t and the projection bases from the same pre-update parameters. The adapters follow standard AdamW using the factor gradients in Eq. (8):

$$(A_{t+1}, B_{t+1}) = \mathrm{AdamW}(A_t, B_t; \eta_{a,t}), \tag{19}$$

Table 1: Results on six GLUE tasks with RoBERTa-Base and RoBERTa-Large. All LoRA-based methods use rank 8. For each model, the best and second-best scores among all methods, including FFT, are shown in **bold** and underlined, respectively.

Model	Method	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
RoBERTa-Base	FFT	**94.42**$_{\pm 0.18}$	**89.24**$_{\pm 0.61}$	64.53$_{\pm 1.08}$	92.81$_{\pm 0.22}$	78.96$_{\pm 0.84}$	**91.24**$_{\pm 0.19}$	85.20
	LoRA	93.04$_{\pm 0.27}$	87.12$_{\pm 0.74}$	62.89$_{\pm 1.31}$	92.31$_{\pm 0.33}$	75.81$_{\pm 1.12}$	89.49$_{\pm 0.24}$	83.44
	PiSSA	93.15$_{\pm 0.41}$	87.09$_{\pm 0.58}$	60.98$_{\pm 0.73}$	92.02$_{\pm 0.68}$	77.98$_{\pm 0.97}$	90.32$_{\pm 0.17}$	83.59
	DoRA	93.03$_{\pm 0.34}$	87.25$_{\pm 0.81}$	62.77$_{\pm 1.46}$	92.38$_{\pm 0.19}$	76.53$_{\pm 1.28}$	89.54$_{\pm 0.31}$	83.58
	rsLoRA	93.06$_{\pm 0.16}$	87.73$_{\pm 0.49}$	61.62$_{\pm 1.12}$	92.77$_{\pm 0.37}$	77.62$_{\pm 0.88}$	89.87$_{\pm 0.22}$	83.78
	Fira	92.51$_{\pm 0.29}$	86.92$_{\pm 0.67}$	60.64$_{\pm 1.24}$	92.62$_{\pm 0.26}$	77.34$_{\pm 1.05}$	89.99$_{\pm 0.18}$	83.34
	LoRA-GA	94.15$_{\pm 0.23}$	87.01$_{\pm 0.72}$	62.16$_{\pm 1.37}$	92.26$_{\pm 0.31}$	76.47$_{\pm 0.91}$	89.98$_{\pm 0.26}$	83.67
	LoRA-Pro	94.38$_{\pm 0.31}$	87.99$_{\pm 0.86}$	60.79$_{\pm 1.58}$	91.93$_{\pm 0.42}$	73.29$_{\pm 1.34}$	89.75$_{\pm 0.29}$	83.02
	GDLoRA	94.19$_{\pm 0.14}$	88.97$_{\pm 0.64}$	**65.49**$_{\pm 1.05}$	**92.93**$_{\pm 0.18}$	**79.42**$_{\pm 0.96}$	90.57$_{\pm 0.25}$	**85.26**
RoBERTa-Large	FFT	96.24$_{\pm 0.17}$	**91.52**$_{\pm 0.95}$	68.45$_{\pm 1.21}$	**94.97**$_{\pm 0.24}$	87.93$_{\pm 0.93}$	**92.67**$_{\pm 0.32}$	88.63
	LoRA	95.72$_{\pm 0.41}$	89.76$_{\pm 0.92}$	65.74$_{\pm 1.63}$	94.03$_{\pm 0.36}$	83.15$_{\pm 1.27}$	91.46$_{\pm 0.47}$	86.64
	PiSSA	96.33$_{\pm 0.22}$	90.69$_{\pm 0.63}$	66.44$_{\pm 0.97}$	94.44$_{\pm 0.29}$	86.28$_{\pm 0.88}$	91.95$_{\pm 0.18}$	87.69
	DoRA	95.89$_{\pm 0.35}$	89.95$_{\pm 0.84}$	65.92$_{\pm 1.42}$	94.27$_{\pm 0.21}$	85.21$_{\pm 1.14}$	92.12$_{\pm 0.39}$	87.23
	rsLoRA	95.87$_{\pm 0.26}$	89.28$_{\pm 0.71}$	66.56$_{\pm 1.18}$	94.42$_{\pm 0.33}$	85.64$_{\pm 1.02}$	92.01$_{\pm 0.27}$	87.30
	Fira	95.76$_{\pm 0.19}$	89.05$_{\pm 0.58}$	65.78$_{\pm 1.05}$	94.51$_{\pm 0.27}$	85.28$_{\pm 0.91}$	91.95$_{\pm 0.22}$	87.06
	LoRA-GA	96.08$_{\pm 0.28}$	90.69$_{\pm 0.76}$	65.78$_{\pm 1.36}$	94.25$_{\pm 0.31}$	85.56$_{\pm 1.09}$	92.29$_{\pm 0.34}$	87.44
	LoRA-Pro	96.22$_{\pm 0.32}$	89.71$_{\pm 0.88}$	66.77$_{\pm 1.49}$	93.85$_{\pm 0.43}$	85.57$_{\pm 1.21}$	91.79$_{\pm 0.41}$	87.32
	GDLoRA	**96.56**$_{\pm 0.25}$	90.93$_{\pm 0.49}$	**68.73**$_{\pm 1.08}$	94.76$_{\pm 0.23}$	**88.95**$_{\pm 0.89}$	92.31$_{\pm 0.36}$	**88.71**

where $\eta_{a,t}$ is the adapter learning rate and the optimizer state is implicit. The normal branch updates the base weight directly:

$$W_{\text{base},t+1} = W_{\text{base},t} - \eta_{n,t} G_{\perp,t}, \tag{20}$$

where $\eta_{n,t} \geq 0$ is the normal-gradient learning rate. Both updates are applied after backward propagation. Setting $\eta_{n,t} = 0$ throughout recovers standard LoRA under the same training configuration.

The normal-gradient branch introduces no additional momentum or variance buffers for the base weights, preserving standard LoRA's optimizer-state budget under matched adapter and optimizer configurations. At the current iterate, the normal update is orthogonal to all first-order weight-space directions induced by the LoRA factors. The finite-step relationship between the two branches is discussed in Appendix E.

4 Experiments

We evaluate GDLoRA on natural language understanding, mathematical reasoning, commonsense reasoning, and image classification, using RoBERTa, Llama-3-8B, Gemma-7B, and ViT models, respectively. These experiments span GLUE, mathematical and commonsense reasoning benchmarks, and eight image classification datasets. Dataset details, adapter configurations, and hyperparameters are provided in Appendix G.

4.1 Natural Language Understanding

As shown in Table 1, GDLoRA achieves the highest average scores among all evaluated methods on RoBERTa-Base and RoBERTa-Large, reaching 85.26 and 88.71, respectively. Compared with standard LoRA at rank 8, the average improvements are 1.82 and 2.07 points. GDLoRA also surpasses the strongest non-FFT baseline for each backbone, namely rsLoRA on RoBERTa-Base and PiSSA on RoBERTa-Large, by 1.48 and 1.02 points, respectively. Against Fira, the corresponding

Table 2: Results on six mathematical reasoning tasks with Llama-3-8B. All LoRA-based methods use rank 8. The best and second-best scores among all methods, including FFT, are shown in **bold** and underlined, respectively.

Method	AddSub	MultiArith	SingleEq	SVAMP	GSM8K	AQuA	Avg.
FFT	**86.78**$_{\pm 1.37}$	90.21$_{\pm 1.83}$	**93.21**$_{\pm 0.84}$	**70.87**$_{\pm 1.26}$	**56.72**$_{\pm 0.75}$	25.96$_{\pm 1.54}$	**70.63**
LoRA	81.52$_{\pm 1.54}$	87.28$_{\pm 1.11}$	91.54$_{\pm 0.69}$	66.93$_{\pm 1.59}$	55.34$_{\pm 0.66}$	22.44$_{\pm 0.60}$	67.51
PiSSA	82.78$_{\pm 1.58}$	86.89$_{\pm 1.44}$	91.27$_{\pm 0.16}$	66.40$_{\pm 0.79}$	54.11$_{\pm 0.57}$	25.33$_{\pm 0.68}$	67.80
DoRA	80.51$_{\pm 1.98}$	88.78$_{\pm 0.86}$	91.08$_{\pm 1.40}$	67.00$_{\pm 1.61}$	55.45$_{\pm 0.80}$	23.75$_{\pm 1.27}$	67.76
rsLoRA	81.34$_{\pm 1.23}$	86.15$_{\pm 1.09}$	90.23$_{\pm 0.78}$	67.24$_{\pm 1.24}$	54.21$_{\pm 0.68}$	24.72$_{\pm 0.95}$	67.32
Fira	80.64$_{\pm 1.47}$	86.69$_{\pm 1.16}$	90.39$_{\pm 0.96}$	67.64$_{\pm 1.12}$	54.18$_{\pm 0.77}$	24.57$_{\pm 1.25}$	67.35
LoRA-GA	82.05$_{\pm 1.05}$	89.36$_{\pm 0.98}$	91.59$_{\pm 0.28}$	68.56$_{\pm 1.08}$	55.78$_{\pm 0.54}$	23.19$_{\pm 0.89}$	68.42
LoRA-Pro	83.26$_{\pm 1.68}$	89.24$_{\pm 1.21}$	91.95$_{\pm 1.06}$	67.83$_{\pm 0.86}$	55.42$_{\pm 0.77}$	25.45$_{\pm 0.94}$	68.86
GDLoRA	84.81$_{\pm 1.24}$	**90.69**$_{\pm 0.97}$	92.52$_{\pm 0.59}$	69.61$_{\pm 0.95}$	56.03$_{\pm 0.61}$	**26.38**$_{\pm 0.88}$	70.01

improvements are 1.92 and 1.65 points. This advantage extends to individual tasks: GDLoRA outperforms both standard LoRA and Fira in all 12 model–task combinations and achieves the highest non-FFT score in 11 of them. The largest gains over standard LoRA occur on RTE, reaching 3.61 and 5.80 points. Its average scores closely match FFT's 85.20 and 88.63, while the normal branch requires no additional base-weight momentum or variance buffers.

4.2 Mathematical Reasoning

Table 2 shows that GDLoRA achieves an average accuracy of 70.01, outperforming standard LoRA at rank 8 by 2.50 percentage points. It also exceeds LoRA-Pro, the strongest non-FFT baseline, by 1.15 points and Fira by 2.66 points. GDLoRA obtains the highest non-FFT score on all six tasks, showing consistent improvements over both adapter-based baselines and Fira in this setting. The gains over standard LoRA are particularly pronounced on AQuA (+3.94 points), MultiArith (+3.41), and AddSub (+3.29). Relative to FFT, GDLoRA reduces the average gap from 3.12 points for standard LoRA to 0.62 points, closing approximately 80.1% of the original LoRA–FFT gap. It exceeds FFT on MultiArith and AQuA, while ranking second on the remaining four tasks. These results demonstrate broad improvements across the evaluated mathematical reasoning tasks and substantially narrow the performance gap to FFT.

4.3 Commonsense Reasoning

As reported in Table 3, GDLoRA achieves an average accuracy of 85.15, surpassing standard LoRA by 1.75 percentage points and the strongest non-FFT baseline, LoRA-Pro, by 1.16 points. It also outperforms Fira by 2.26 points on average and achieves higher scores on all eight tasks. GDLoRA obtains the highest non-FFT score on six tasks, with particularly large gains over standard LoRA on OpenBookQA (+4.00 points), Social IQa (+3.07), and ARC-Challenge (+2.41). Compared with FFT's average accuracy of 85.70, GDLoRA reduces the remaining difference to 0.55 points and achieves higher scores on ARC-Challenge, Social IQa, and ARC-Easy. LoRA-Pro and LoRA-GA retain small advantages on PIQA and BoolQ, respectively, but GDLoRA delivers the strongest aggregate performance among the non-FFT methods. These results extend the reasoning improvements observed on Llama-3-8B to Gemma-7B.

4.4 Image Classification

Table 4 shows that GDLoRA achieves average accuracies of 86.79 and 89.20 on ViT-Base and ViT-Large, respectively, obtaining the highest average scores among all evaluated methods on both backbones. These results exceed standard LoRA at rank 8 by 2.07 and 2.46 percentage points. Compared with Fira, GDLoRA improves average accuracy by 1.64 and 1.87 points and achieves higher

Table 3: Results on eight commonsense reasoning tasks with Gemma-7B. All LoRA-based methods use rank 8. The best and second-best scores among all methods, including FFT, are shown in **bold** and underlined, respectively.

Method	OBQA	ARC-c	WinoGrande	PIQA	SIQA	ARC-e	BoolQ	HellaSwag	Avg.
FFT	**86.77**$_{\pm 1.46}$	84.93$_{\pm 1.66}$	**88.05**$_{\pm 1.25}$	**89.11**$_{\pm 1.07}$	80.65$_{\pm 1.43}$	93.06$_{\pm 1.28}$	**70.12**$_{\pm 0.24}$	**92.89**$_{\pm 0.84}$	**85.70**
LoRA	81.62$_{\pm 0.82}$	82.74$_{\pm 0.89}$	85.08$_{\pm 0.88}$	87.54$_{\pm 0.65}$	77.79$_{\pm 1.14}$	92.67$_{\pm 1.01}$	68.84$_{\pm 0.25}$	90.95$_{\pm 0.65}$	83.40
PiSSA	81.24$_{\pm 0.57}$	81.06$_{\pm 1.06}$	81.69$_{\pm 0.73}$	86.24$_{\pm 0.48}$	75.74$_{\pm 1.28}$	91.12$_{\pm 0.84}$	67.43$_{\pm 0.31}$	89.13$_{\pm 0.79}$	81.71
DoRA	82.53$_{\pm 0.94}$	81.95$_{\pm 0.68}$	83.58$_{\pm 1.02}$	86.82$_{\pm 0.72}$	76.93$_{\pm 0.96}$	91.16$_{\pm 1.12}$	68.45$_{\pm 0.22}$	83.56$_{\pm 0.58}$	81.87
rsLoRA	82.86$_{\pm 0.71}$	81.83$_{\pm 1.14}$	84.50$_{\pm 0.79}$	87.63$_{\pm 0.54}$	77.12$_{\pm 1.21}$	92.13$_{\pm 0.93}$	68.09$_{\pm 0.37}$	92.23$_{\pm 0.83}$	83.30
Fira	83.12$_{\pm 1.03}$	81.66$_{\pm 0.74}$	84.16$_{\pm 0.91}$	87.92$_{\pm 0.60}$	75.84$_{\pm 1.35}$	91.26$_{\pm 0.76}$	68.71$_{\pm 0.29}$	90.42$_{\pm 0.52}$	82.89
LoRA-GA	82.68$_{\pm 0.63}$	82.72$_{\pm 0.97}$	85.95$_{\pm 1.08}$	87.19$_{\pm 0.69}$	78.84$_{\pm 0.88}$	91.69$_{\pm 1.15}$	68.93$_{\pm 0.21}$	91.34$_{\pm 0.74}$	83.67
LoRA-Pro	83.66$_{\pm 0.89}$	82.89$_{\pm 0.59}$	85.75$_{\pm 0.95}$	88.65$_{\pm 0.77}$	78.69$_{\pm 1.18}$	91.86$_{\pm 0.81}$	68.56$_{\pm 0.34}$	91.89$_{\pm 0.61}$	83.99
GDLoRA	85.62$_{\pm 0.76}$	**85.15**$_{\pm 1.09}$	86.27$_{\pm 0.84}$	88.36$_{\pm 0.57}$	**80.86**$_{\pm 1.26}$	**93.77**$_{\pm 0.98}$	68.89$_{\pm 0.27}$	92.28$_{\pm 0.71}$	85.15

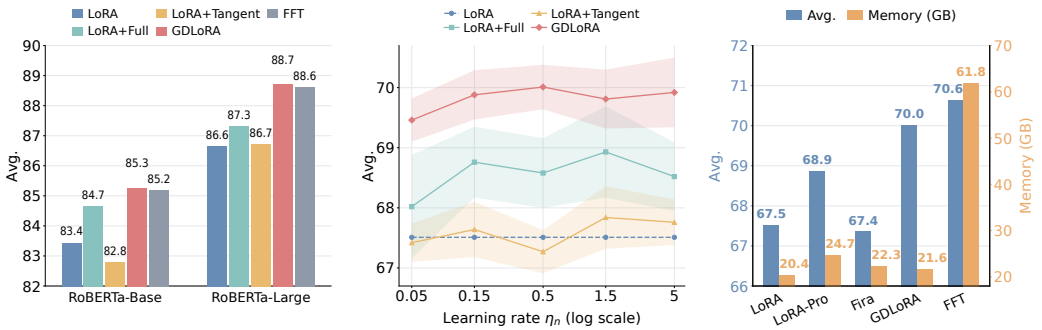
Table 4: Image classification results with ViT-Base and ViT-Large. For each model, the best and second-best scores among all methods are shown in **bold** and underlined, respectively.

Model	Method	OxfordPets	StanfordCars	CIFAR-10	DTD	EuroSAT	FGVC	RESISC45	CIFAR-100	Avg.
ViT-Base	FFT	96.13$_{\pm 0.21}$	76.81$_{\pm 0.94}$	**98.96**$_{\pm 0.04}$	77.75$_{\pm 0.88}$	98.92$_{\pm 0.19}$	**54.64**$_{\pm 0.97}$	**96.12**$_{\pm 0.34}$	**92.11**$_{\pm 0.18}$	86.43
	LoRA	95.92$_{\pm 0.37}$	74.93$_{\pm 0.58}$	97.26$_{\pm 0.06}$	75.03$_{\pm 0.54}$	98.57$_{\pm 0.12}$	49.17$_{\pm 0.71}$	95.43$_{\pm 0.29}$	91.44$_{\pm 0.22}$	84.72
	PiSSA	95.46$_{\pm 0.18}$	74.24$_{\pm 0.83}$	97.92$_{\pm 0.03}$	74.82$_{\pm 0.97}$	98.58$_{\pm 0.05}$	51.07$_{\pm 0.52}$	95.12$_{\pm 0.17}$	90.14$_{\pm 0.13}$	84.67
	DoRA	95.62$_{\pm 0.29}$	74.69$_{\pm 0.41}$	97.64$_{\pm 0.07}$	75.77$_{\pm 0.63}$	98.76$_{\pm 0.11}$	49.81$_{\pm 0.86}$	95.59$_{\pm 0.23}$	91.32$_{\pm 0.20}$	84.90
	rsLoRA	95.51$_{\pm 0.33}$	74.12$_{\pm 0.69}$	97.66$_{\pm 0.05}$	75.87$_{\pm 0.45}$	98.49$_{\pm 0.08}$	50.78$_{\pm 0.61}$	95.48$_{\pm 0.12}$	91.08$_{\pm 0.16}$	84.87
	Fira	95.06$_{\pm 0.16}$	75.28$_{\pm 1.07}$	98.54$_{\pm 0.02}$	76.19$_{\pm 0.76}$	98.87$_{\pm 0.06}$	51.22$_{\pm 0.74}$	95.08$_{\pm 0.31}$	90.95$_{\pm 0.24}$	85.15
	LoRA-GA	95.74$_{\pm 0.42}$	74.71$_{\pm 0.35}$	98.08$_{\pm 0.07}$	76.45$_{\pm 0.59}$	98.48$_{\pm 0.13}$	50.21$_{\pm 0.49}$	95.33$_{\pm 0.20}$	91.31$_{\pm 0.19}$	85.04
	LoRA-Pro	95.92$_{\pm 0.24}$	74.23$_{\pm 0.92}$	98.18$_{\pm 0.04}$	77.66$_{\pm 1.14}$	98.33$_{\pm 0.10}$	50.81$_{\pm 1.08}$	95.17$_{\pm 0.27}$	90.52$_{\pm 0.15}$	85.10
	GDLoRA	**97.01**$_{\pm 0.15}$	**78.65**$_{\pm 0.47}$	98.86$_{\pm 0.03}$	**78.78**$_{\pm 0.69}$	**98.96**$_{\pm 0.07}$	54.27$_{\pm 0.65}$	95.73$_{\pm 0.26}$	92.02$_{\pm 0.21}$	**86.79**
ViT-Large	FFT	**97.44**$_{\pm 0.41}$	85.87$_{\pm 0.84}$	99.15$_{\pm 0.04}$	**81.17**$_{\pm 0.72}$	99.04$_{\pm 0.08}$	60.17$_{\pm 1.12}$	**96.21**$_{\pm 0.09}$	93.55$_{\pm 0.17}$	89.08
	LoRA	95.92$_{\pm 0.19}$	78.91$_{\pm 0.58}$	99.16$_{\pm 0.03}$	79.12$_{\pm 0.44}$	98.35$_{\pm 0.05}$	53.59$_{\pm 0.63}$	95.56$_{\pm 0.13}$	93.34$_{\pm 0.23}$	86.74
	PiSSA	96.24$_{\pm 0.33}$	82.21$_{\pm 1.29}$	99.19$_{\pm 0.13}$	80.43$_{\pm 0.91}$	98.78$_{\pm 0.07}$	56.62$_{\pm 1.27}$	95.73$_{\pm 0.08}$	93.17$_{\pm 0.14}$	87.80
	DoRA	95.91$_{\pm 0.27}$	78.28$_{\pm 0.66}$	99.11$_{\pm 0.07}$	78.88$_{\pm 0.35}$	98.48$_{\pm 0.09}$	53.92$_{\pm 0.52}$	95.76$_{\pm 0.12}$	93.24$_{\pm 0.18}$	86.70
	rsLoRA	96.25$_{\pm 0.52}$	81.66$_{\pm 0.47}$	99.15$_{\pm 0.13}$	80.05$_{\pm 0.63}$	98.65$_{\pm 0.04}$	55.42$_{\pm 0.88}$	96.08$_{\pm 0.07}$	93.15$_{\pm 0.21}$	87.55
	Fira	94.89$_{\pm 0.14}$	82.21$_{\pm 0.61}$	99.23$_{\pm 0.05}$	80.21$_{\pm 0.49}$	98.77$_{\pm 0.08}$	55.52$_{\pm 0.41}$	95.24$_{\pm 0.14}$	92.56$_{\pm 0.09}$	87.33
	LoRA-GA	96.47$_{\pm 0.24}$	78.77$_{\pm 0.38}$	99.22$_{\pm 0.08}$	79.04$_{\pm 0.83}$	98.48$_{\pm 0.03}$	54.79$_{\pm 0.74}$	95.49$_{\pm 0.11}$	93.24$_{\pm 0.16}$	86.94
	LoRA-Pro	96.65$_{\pm 0.08}$	78.55$_{\pm 0.54}$	99.12$_{\pm 0.09}$	79.31$_{\pm 0.57}$	98.41$_{\pm 0.12}$	55.77$_{\pm 0.97}$	95.08$_{\pm 0.10}$	**93.76**$_{\pm 0.12}$	87.08
	GDLoRA	97.27$_{\pm 0.36}$	**86.14**$_{\pm 0.31}$	**99.28**$_{\pm 0.03}$	80.74$_{\pm 0.68}$	**99.16**$_{\pm 0.07}$	**61.66**$_{\pm 1.35}$	95.92$_{\pm 0.13}$	93.41$_{\pm 0.15}$	**89.20**

scores in all 16 model–dataset combinations. It also surpasses PiSSA, the strongest non-FFT baseline on ViT-Large, by 1.40 points. The largest gains over standard LoRA occur on FGVC-Aircraft, reaching 5.10 and 8.07 points. StanfordCars also improves substantially, by 3.72 and 7.23 points, with both fine-grained recognition tasks showing larger gains on ViT-Large. The improvements on these two datasets exceed the corresponding average gains for both backbones, highlighting fine-grained recognition as a strong setting for GDLoRA within the evaluated image classification suite. GDLoRA's average accuracies are also comparable to FFT's 86.43 and 89.08, extending its competitive performance to visual adaptation.

4.5 Ablation Studies and Analysis

Gradient component ablation. Figure 2(a) evaluates the effect of the gradient component used to update the base weights. We compare GDLoRA with standard LoRA, LoRA+Full, and LoRA+Tangent, where the latter two apply the full gradient and its tangent component to the base weights, respectively. GDLoRA improves the average GLUE score over LoRA by 1.82 and 2.07 points on RoBERTa-Base and RoBERTa-Large, respectively. It also outperforms LoRA+Full by 0.59 and 1.40 points. LoRA+Tangent reduces performance on RoBERTa-Base and yields only a marginal improvement on RoBERTa-Large. Tangent updates introduce no directions outside the adapters' current first-order accessible space. LoRA+Full also modifies updates within this space,



(a) Gradient component ablation (b) Learning-rate sensitivity (c) Accuracy and memory

Figure 2: **Ablation studies and analysis of GDLoRA.** (a) Comparison of gradient components on six GLUE tasks with RoBERTa-Base and RoBERTa-Large. (b) Sensitivity to the base-update learning rate when fine-tuning Llama-3-8B on Math10K. Shaded bands indicate ± 1 standard deviation. (c) Average mathematical reasoning accuracy and memory usage with Llama-3-8B on Math10K.

potentially interacting with the adapter optimizer. GDLoRA isolates the normal component, supplying complementary gradient information that the current adapters cannot represent at first order. The observed advantage is consistent with the benefit of this complementary update, although overlap alone does not imply that tangent updates are inherently harmful. GDLoRA also achieves average scores numerically comparable to FFT on both backbones.

Learning-rate sensitivity. Figure 2(b) examines sensitivity to the base-update learning rate on Llama-3-8B fine-tuned on Math10K. We vary $\eta_n \in \{0.05, 0.15, 0.5, 1.5, 5\}$ for the methods with additional base-weight updates, while standard LoRA serves as a fixed baseline. GDLoRA achieves mean accuracies between 69.46 and 70.01, with a variation of 0.55 percentage points across the tested range. Its improvements over LoRA range from 1.95 to 2.50 percentage points, with the highest accuracy obtained at $\eta_n = 0.5$. Moreover, its lowest mean score exceeds the best LoRA+Full mean score of 68.93. These results suggest that GDLoRA's gains persist across the evaluated learning-rate range and do not depend on a single narrowly selected setting.

Accuracy and memory. Figure 2(c) compares GPU memory usage during fine-tuning of Llama-3-8B on Math10K and the resulting average accuracy across six mathematical reasoning tasks. GDLoRA achieves an average accuracy of 70.01 with 21.6 GB of memory, improving over LoRA by 2.50 percentage points while requiring only 1.2 GB (5.9%) additional memory. Compared with LoRA-Pro and Fira, GDLoRA improves accuracy by 1.15 and 2.66 percentage points while reducing memory usage by 12.6% and 3.1%, respectively. Relative to FFT, GDLoRA reduces memory usage from 61.79 GB to 21.6 GB, a reduction of 65.0%, while trailing by 0.62 percentage points in accuracy. Thus, GDLoRA narrows the accuracy gap to FFT from 3.12 to 0.62 percentage points while incurring a modest memory increase over standard LoRA.

5 Conclusion

We presented GDLoRA, which complements standard LoRA with gradient information beyond the first-order directions accessible through its current parameterization. By characterizing the LoRA-accessible gradient space as the induced tangent space, GDLoRA isolates the normal component of the full weight gradient and applies it directly to the base weights, while retaining standard AdamW optimization for the adapters. Under matched adapter and optimizer configurations, GDLoRA preserves LoRA's optimizer-state memory budget. Experiments on natural language understanding, mathematical reasoning, commonsense reasoning, and image classification demonstrate consistent improvements over LoRA and a reduced performance gap to FFT. Ablation studies further support the effectiveness of normal-gradient updates in complementing low-rank adaptation.

REFERENCES

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Xi Chen, Kaituo Feng, Chang-Sheng Li, Xunhao Lai, Xiangyu Yue, Ye Yuan, and Guoren Wang. Fira: Can we achieve full-rank training of llms under low-rank constraint? *ArXiv*, abs/2410.01623, 2024. URL https://api.semanticscholar.org/CorpusID:273026172.

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

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of machine learning research*, 24(240):1–113, 2023.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020. URL https://api.semanticscholar.org/CorpusID:225039882.

Abhimanyu Dubey et al. The llama 3 herd of models. 2024. URL https://api.semanticscholar.org/CorpusID:271571434.

Ziqi Gao, Qi-Chao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform. *ArXiv*, abs/2405.03003, 2024. URL https://api.semanticscholar.org/CorpusID:269605083.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. 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, 2019.

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 *International conference on machine learning*, pp. 2790–2799. PMLR, 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. *arXiv preprint arXiv:2106.09685*, 2021.

Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *ArXiv*, abs/2304.01933, 2023. URL https://api.semanticscholar.org/CorpusID:257921386.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*, 2023.

Jiale Kang and Qingyu Yin. Miss: Revisiting the trade-off in lora with an efficient shard-sharing structure. 2024. URL https://api.semanticscholar.org/CorpusID:272832329.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

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.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *ArXiv*, abs/2205.05638, 2022. URL https://api.semanticscholar.org/CorpusID:248693283.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019. URL https://api.semanticscholar.org/CorpusID:198953378.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072, 2024.

Gemma Team Thomas Mesnard et al. Gemma: Open models based on gemini research and technology. *ArXiv*, abs/2403.08295, 2024. URL https://api.semanticscholar.org/CorpusID:268379206.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 46–54, 2020.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pp. 1–16. IEEE, 2020.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*, 2018. URL https://api.semanticscholar.org/CorpusID:5034059.

Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. *Advances in Neural Information Processing Systems*, 37:54905–54931, 2024.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. Lora-pro: Are low-rank adapters properly optimized? In *International Conference on Learning Representations*, volume 2025, pp. 93787–93808, 2025.

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. *arXiv preprint arXiv:2303.10512*, 2023.

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

Jiawei Zhao, Zhenyu (Allen) Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *ArXiv*, abs/2403.03507, 2024. URL https://api.semanticscholar.org/CorpusID:268253596.

A Gradient Subspace Drift and Projection Refresh Costs

We investigate the trade-off between the temporal relevance of gradient projection bases and the computational cost of refreshing them. Our comparison uses native Fira (Chen et al., 2024), which directly updates the base weights using low-rank adaptive optimization and scaled residual updates. We examine gradient subspace variation across training steps and network modules, and evaluate how the projection refresh interval affects accuracy and training time.

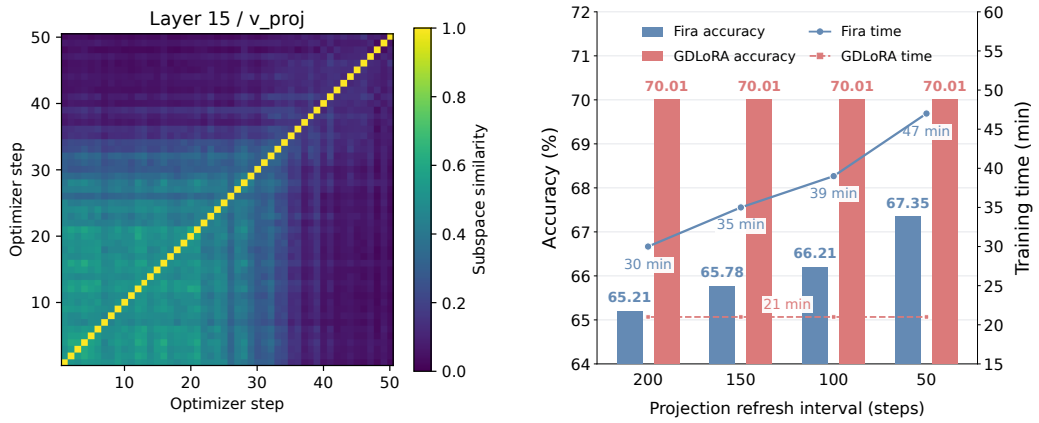
Gradient subspace drift and projection refresh. SVD-based gradient projection methods, including GaLore (Zhao et al., 2024) and Fira, periodically construct projection bases from full weight gradients and reuse them between refreshes. Let G_t denote the full weight gradient at step t, and let $\tau(t)$ denote the most recent projection refresh step. For a left-sided projection with orthonormal basis $P_{\tau(t)}$, the projected gradient and unscaled residual are

$$R_t = P_{\tau(t)}^T G_t, \qquad S_t = (I - P_{\tau(t)} P_{\tau(t)}^T) G_t. \tag{21}$$

The projected gradient, residual, and low-dimensional optimizer moments continue to update at every step. However, the subspace receiving adaptive optimization remains fixed until the next basis refresh.

To examine the temporal relevance of this subspace, we record the full weight gradient of `model.layers.15.self_attn.v_proj` during Llama-3-8B fine-tuning on Math10K. At each of 50 optimizer steps, we perform an SVD, retain the leading $r = 8$ singular directions, and compute pairwise subspace similarities. As shown in Figure 3(a), similarities are relatively higher among early steps and substantially lower for many pairs involving later steps, indicating drift in the leading gradient subspace along the recorded trajectory. Figure 4 provides additional results for six selected attention projections spanning layers 0, 15, and 31 and collectively covering query, key, and value projections, showing that subspace variation occurs across the selected modules and network depths. Consequently, reusing an earlier projection basis can leave the adaptive branch operating in a subspace that no longer adequately represents the current leading gradient directions. Although Fira still incorporates components outside this basis, its residual branch does not maintain separate Adam first- and second-moment estimates for each residual entry. Instead, it applies norm-based scaling derived from the projected branch. Thus, delayed refreshes can make the allocation of low-dimensional optimizer states less aligned with the evolving leading gradient subspace. The variation observed within the 50-step window also indicates that refreshing every 50 steps need not eliminate subspace mismatch between refreshes. These observations motivate an explicit evaluation of the accuracy and runtime effects of changing the projection refresh interval.

More frequent refreshes incur substantial training costs. Figure 3(b) evaluates native Fira with refresh intervals $T \in \{200, 150, 100, 50\}$. Its corresponding accuracies are 65.21%, 65.78%, 66.21%, and 67.35%, with training times of 30, 35, 39, and 47 minutes, respectively. Reducing



(a) Gradient subspace drift

(b) Accuracy and training time

Figure 3: **Gradient subspace drift and projection refresh costs. (a)** Pairwise similarity between the leading rank-8 gradient subspaces of `model.layers.15.self_attn.v_proj` over 50 optimizer steps. **(b)** Accuracy and training time of native Fira with projection refresh intervals of 200, 150, 100, and 50 steps on Math10K with Llama-3-8B. GDLoRA is shown as a fixed reference; its repeated values correspond to the same configuration.

the interval from 200 to 50 steps improves accuracy by 2.14 percentage points, but increases training time by 56.7%. Thus, more frequent refreshes improve performance in this experiment while imposing a substantial runtime penalty.

GDLoRA achieves 70.01% accuracy in 21 minutes. Even at $T = 50$, the highest-accuracy Fira configuration in this sweep remains 2.66 percentage points below GDLoRA and requires $2.24\times$ its training time. More frequent refreshes therefore do not close the observed performance gap, despite their increased computational cost. Such a runtime penalty can substantially increase the compute requirements of longer training workloads and may limit practicality under constrained training budgets. The repeated GDLoRA values are a fixed reference, rather than separate experiments with different refresh intervals.

These observations motivate placing adaptive optimization in the LoRA factor coordinates rather than in a periodically refreshed gradient-SVD subspace. GDLoRA maintains AdamW moments for the trainable factors A and B, while deriving the normal component from the current adapter space at each step. This design avoids repeated full-gradient SVD and supplements adapter optimization with a direct normal-gradient update.

GDLoRA recomputes the normal component from current factors. GDLoRA avoids periodically estimating a gradient-SVD subspace by constructing its projection bases directly from the current LoRA factors:

$$U_t = \mathrm{orth}(B_t), \qquad V_t = \mathrm{orth}(A_t^T), \tag{22}$$

where $\mathrm{orth}(\cdot)$ returns an orthonormal basis for the corresponding column space in the exact formulation. Appendix H.2 specifies the numerical thresholding procedure and its limitations for rank-deficient factors. The normal gradient is then

$$G_{\perp,t} = (I - U_t U_t^T) G_t (I - V_t V_t^T). \tag{23}$$

Both the gradient and projection bases are computed from the same pre-update parameters. With exact column-space bases, each normal update is orthogonal to the first-order weight-space directions accessible through the current adapters.

Basis construction operates on thin factors rather than the full gradient matrix. For full-rank factors, thin QR factorizations require $O((d_o + d_i)r^2)$ operations. Meanwhile, the adapter optimizer states remain in the parameter coordinates of A and B and follow standard AdamW updates. Updating the projection bases therefore does not require re-expressing the adapter moments in a new SVD basis. The normal branch directly updates the base weights without maintaining additional momentum or variance buffers for them.

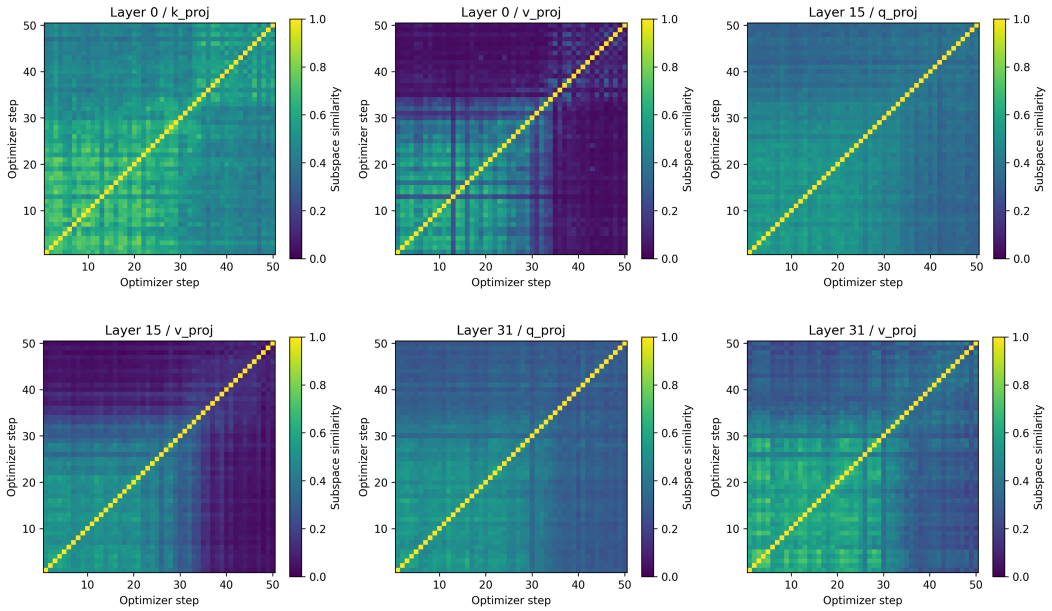


Figure 4: **Gradient subspace variation across network depths and attention projections.** Each heatmap shows pairwise similarity between leading rank-8 gradient subspaces over 50 optimizer steps during Llama-3-8B fine-tuning on Math10K. From left to right, the top row shows the layer-0 key, layer-0 value, and layer-15 query projections; the bottom row shows the layer-15 value, layer-31 query, and layer-31 value projections. Layer numbers follow the model's module indices. The layer-15 value projection is also shown in Figure 3(a). All panels use the same color scale.

GDLoRA thus recomputes its projection bases from the current LoRA factors at every step without repeatedly computing an SVD of the full weight gradient. The cross-layer diagnostics demonstrate variation in leading gradient subspaces across multiple attention projections, while the refresh-interval experiment establishes the accuracy–runtime trade-off in the evaluated Fira configuration.

B Computational Comparison with Fira

We compare the leading FLOPs of GDLoRA and native Fira (Chen et al., 2024) for a linear layer with weight dimensions $d_o \times d_i$, rank r, and N input positions. A multiplication and an addition count as two FLOPs. Shared base-layer forward and input-gradient computations, as well as lower-order elementwise operations, are excluded.

Both methods compute the full weight gradient, costing $2Nd_od_i$ FLOPs. Their main difference in basis construction is the matrix being decomposed. GDLoRA orthonormalizes the thin factors B and A^T. Reduced Householder QR, including explicit basis formation, has leading cost

$$C_{\mathrm{QR}} \approx 4(d_o + d_i)r^2. \tag{24}$$

Fira instead obtains its basis through SVD of the full gradient, with dense-decomposition cost

$$C_{\mathrm{SVD}} = O\big(d_o d_i \min(d_o, d_i)\big). \tag{25}$$

Refreshing every T steps amortizes this cost to C_{SVD}/T per step.

GDLoRA evaluates its two-sided projection through thin-basis multiplications without forming dense projection matrices. Although this requires more projection FLOPs than Fira's one-sided formulation, its basis construction is inexpensive: for square layers of width d, thin-factor QR scales as $O(dr^2)$, compared with $O(d^3)$ for a dense gradient SVD per refresh. For $r \ll d$, this makes recomputing the adapter-derived bases at every step practical. The corresponding end-to-end training-time comparison is reported in Figure 3(b) and Table 6.

Table 5: Leading FLOPs per layer and optimization step. Fira's SVD cost is amortized over its refresh interval T. GDLoRA recomputes both bases every step. Projection counts assume rank-r bases and efficient matrix-product ordering.

Operation	Fira	GDLoRA
Full-weight gradient	$2Nd_od_i$	$2Nd_od_i$
Gradient projection and reconstruction	$6d_od_ir$	$8d_od_ir$
Basis construction	C_{SVD}/T	C_{QR}
Adapter forward and backward	0	$6Nr(d_o + d_i)$

Table 6: Training time (hours) on six GLUE tasks with RoBERTa-Base.

Method	SST-2	MRPC	CoLA	QNLI	RTE	STS-B
FFT	10.16	1.64	5.78	19.56	2.21	3.64
LoRA	6.92	0.58	3.54	13.24	1.06	2.50
PiSSA	13.06	1.08	4.92	16.78	1.47	2.98
DoRA	9.84	0.65	4.03	13.78	1.19	2.52
rsLoRA	9.98	0.57	3.51	14.23	1.03	2.50
Fira	16.18	1.53	6.31	19.65	2.34	4.35
LoRA-GA	6.62	0.57	3.51	14.26	1.03	2.51
LoRA-Pro	7.07	0.61	5.62	15.77	1.11	4.06
GDLoRA	7.22	0.62	3.83	14.45	1.12	2.75

C Proof of the LoRA-Accessible Gradient Space Characterization

Proof of Theorem 1. We characterize the parameter-induced directions and their orthogonal complement, then relate this complement to the kernel of the equivalent gradient operator.

For fixed $s \neq 0$, the differential of $F_s(A, B) = sBA$ is

$$DF_s(A, B)[H_A, H_B] = s(BH_A + H_BA). \tag{26}$$

Since a nonzero scalar does not change a linear subspace,

$$\begin{aligned} \mathcal{T}_{\mathrm{LoRA}} &= \{BH_A + H_BA\} \\ &= \left\{UK + LV^T \;\middle|\; K \in \mathbb{R}^{r_B \times d_i},\, L \in \mathbb{R}^{d_o \times r_A}\right\}. \end{aligned} \tag{27}$$

Here $H_A \in \mathbb{R}^{r \times d_i}$ and $H_B \in \mathbb{R}^{d_o \times r}$ vary freely. For the second equality, write $B = UR_B$ and $A^T = VR_A$, where R_B and R_A have full row rank. Every parameter-induced direction then has the stated form. Conversely, the maps $H_A \mapsto R_BH_A$ and $H_B \mapsto H_BR_A^T$ are surjective, so every pair (K, L) can be attained. When either factor has rank zero, its basis is empty and the corresponding term vanishes.

For any $Z \in \mathbb{R}^{d_o \times d_i}$,

$$\langle Z, UK + LV^T\rangle_F = \langle U^TZ, K\rangle_F + \langle ZV, L\rangle_F. \tag{28}$$

Because K and L vary independently, this gives

$$\mathcal{T}_{\mathrm{LoRA}}^{\perp} = \{Z \in \mathbb{R}^{d_o \times d_i} : U^TZ = 0,\; ZV = 0\}. \tag{29}$$

Next, consider $\Phi_s(H) = s^2(BB^TH + HA^TA)$. The symmetry of BB^T and A^TA implies

$$\langle Z, \Phi_s(H)\rangle_F = \langle \Phi_s(Z), H\rangle_F, \tag{30}$$

so Φ_s is self-adjoint. Moreover,

$$\langle H, \Phi_s(H)\rangle_F = s^2\left(\|B^TH\|_F^2 + \|HA^T\|_F^2\right). \tag{31}$$

Thus $\Phi_s(H) = 0$ implies $B^T H = 0$ and $HA^T = 0$; the converse follows directly from the definition of Φ_s. Using the column spaces spanned by U and V, we obtain

$$\begin{aligned} \ker(\Phi_s) &= \{H : B^T H = 0,\ HA^T = 0\} \\ &= \{H : U^T H = 0,\ HV = 0\} = \mathcal{T}_{\mathrm{LoRA}}^{\perp}. \end{aligned} \tag{32}$$

Finally, the range of a self-adjoint operator on a finite-dimensional inner-product space is the orthogonal complement of its kernel. Hence

$$\begin{aligned} \mathcal{G}_{\mathrm{LoRA}} &= \mathrm{Range}(\Phi_s) = \ker(\Phi_s)^{\perp} \\ &= (\mathcal{T}_{\mathrm{LoRA}}^{\perp})^{\perp} = \mathcal{T}_{\mathrm{LoRA}}. \end{aligned} \tag{33}$$

Together with the basis representation above, this proves the theorem. □

D Orthogonal Projection onto the Normal Space

Let $P_U = UU^T$ and $P_V = VV^T$. For the tangent space $\mathcal{T}_{\mathrm{LoRA}} = \{UK + LV^T\}$, the identity $\langle Z, UK + LV^T\rangle_F = \langle U^T Z, K\rangle_F + \langle ZV, L\rangle_F$ implies

$$\mathcal{T}_{\mathrm{LoRA}}^{\perp} = \{Z \in \mathbb{R}^{d_o \times d_i} : U^T Z = 0,\ ZV = 0\}. \tag{34}$$

Define $G_\perp = (I - P_U)G(I - P_V)$. Since $U^T(I - P_U) = 0$ and $(I - P_V)V = 0$, we have $G_\perp \in \mathcal{T}_{\mathrm{LoRA}}^{\perp}$. The residual satisfies

$$\begin{aligned} G_\parallel = G - G_\perp &= P_U G + G P_V - P_U G P_V \\ &= U(U^T G) + \big((I - P_U)GV\big)V^T \in \mathcal{T}_{\mathrm{LoRA}}. \end{aligned} \tag{35}$$

Thus $G = G_\parallel + G_\perp$ is the unique orthogonal decomposition into the tangent and normal spaces, establishing that $G_\perp$ is the orthogonal projection of G onto $\mathcal{T}_{\mathrm{LoRA}}^{\perp}$. By Theorem 1, $\mathcal{G}_{\mathrm{LoRA}} = \mathcal{T}_{\mathrm{LoRA}}$; hence $\langle G_\perp, Z\rangle_F = 0$ for every $Z \in \mathcal{G}_{\mathrm{LoRA}}$.

E Finite-Step Analysis of the Two Update Branches

Section 3.2 extracts the normal gradient relative to the current LoRA-accessible space. This appendix examines whether finite adapter updates can themselves supply the missing normal displacement. Although bilinear factor updates can produce such a displacement, its reachable set is constrained by the factor-step size and internal rank. We characterize these constraints and give a sufficient condition under which direct normal compensation improves matching to a full-gradient reference step. All statements use the current, fixed space and exact column-space bases; the space may change at subsequent iterations.

E.1 Normal displacement of a finite adapter update

Let

$$\Delta A_t = A_{t+1} - A_t, \qquad \Delta B_t = B_{t+1} - B_t \tag{36}$$

denote the factor increments produced by the adapter optimizer. The corresponding finite-step change in the adapter weight is

$$\begin{aligned} D_t := \Delta W_{\mathrm{adapter},t} &= s(B_{t+1}A_{t+1} - B_t A_t) \\ &= s(B_t \Delta A_t + \Delta B_t A_t) + s\Delta B_t \Delta A_t. \end{aligned} \tag{37}$$

The first-order term belongs to the tangent space induced by the current factors:

$$D_{\tan,t} = s(B_t \Delta A_t + \Delta B_t A_t) \in \mathcal{T}_{\mathrm{LoRA},t}. \tag{38}$$

Since the normal gradient is computed using the same pre-update factors, it satisfies

$$\langle G_{\perp,t}, D_{\tan,t}\rangle_F = 0. \tag{39}$$

This property holds for arbitrary factor increments, including those generated by AdamW.

Let $P_{\perp,t}(Z) = (I - U_t U_t^T) Z (I - V_t V_t^T)$. The actual normal displacement of the adapter update is therefore

$$Q_t := P_{\perp,t}(D_t) = s(I - U_t U_t^T)\Delta B_t \Delta A_t (I - V_t V_t^T). \tag{40}$$

Thus, a finite LoRA update can have a normal component, but that component arises entirely from its bilinear remainder. In particular,

$$\langle G_{\perp,t}, D_t \rangle_F = \langle G_{\perp,t}, Q_t \rangle_F = s \langle G_{\perp,t}, \Delta B_t \Delta A_t \rangle_F \tag{41}$$

need not be zero. Including the base-weight update, the total effective-weight displacement is

$$\begin{aligned} W'_{t+1} - W'_t = & -\eta_{n,t} G_{\perp,t} + D_{\tan,t} \\ & + s \Delta B_t \Delta A_t. \end{aligned} \tag{42}$$

E.2 Reachable normal displacements under a factor-step budget

Fix an internal rank $r \geq 1$, $A \in \mathbb{R}^{r \times d_i}$, $B \in \mathbb{R}^{d_o \times r}$, and $s \neq 0$, and suppress the time index. The factors need not have full rank. Write $\mathcal{T} = \mathcal{T}_{\mathrm{LoRA}}$ and $P_\perp(Z) = (I - UU^T)Z(I - VV^T)$, using the zero-rank convention of Section 3.2. For $\varepsilon \geq 0$, define

$$\mathcal{R}_\varepsilon = \left\{ P_\perp(s[(B + \Delta B)(A + \Delta A) - BA]) \;\middle|\; \|\Delta A\|_F^2 + \|\Delta B\|_F^2 \leq \varepsilon^2 \right\}. \tag{43}$$

Here $\|\cdot\|_*$ denotes the nuclear norm, the sum of singular values.

Theorem E.1 (Exact normal reachability)**.** *For the fixed factor coordinates above,*

$$\mathcal{R}_\varepsilon = \left\{ Z \in \mathcal{T}^\perp : \operatorname{rank}(Z) \leq r, \quad \|Z\|_* \leq \frac{|s|\varepsilon^2}{2} \right\}. \tag{44}$$

Proof. Let $\widetilde{B} = (I - UU^T)\Delta B$ and $\widetilde{A} = \Delta A(I - VV^T)$. Equation (40) gives $Q = s\widetilde{B}\widetilde{A} \in \mathcal{T}^\perp$, with $\operatorname{rank}(Q) \leq r$. The product inequality for the nuclear norm, nonexpansiveness of orthogonal projections, and $2ab \leq a^2 + b^2$ yield

$$\begin{aligned} \|Q\|_* &\leq |s| \|\widetilde{B}\|_F \|\widetilde{A}\|_F \leq |s| \|\Delta B\|_F \|\Delta A\|_F \\ &\leq \frac{|s|}{2}\left(\|\Delta B\|_F^2 + \|\Delta A\|_F^2\right) \leq \frac{|s|\varepsilon^2}{2}. \end{aligned} \tag{45}$$

This proves one inclusion.

Conversely, let a nonzero $Z \in \mathcal{T}^\perp$ have compact SVD $Z = L\Sigma R^T$ and rank $k \leq r$. Because $U^T Z = 0$ and $ZV = 0$, its nonzero singular vectors satisfy $U^T L = 0$ and $R^T V = 0$. Let $J_k = [I_k \; 0] \in \mathbb{R}^{k \times r}$ and choose

$$\Delta B = \frac{1}{\sqrt{|s|}} L \Sigma^{1/2} J_k, \qquad \Delta A = \frac{\operatorname{sgn}(s)}{\sqrt{|s|}} J_k^T \Sigma^{1/2} R^T. \tag{46}$$

These increments satisfy $s\Delta B \Delta A = Z$ and are unchanged by the corresponding left and right normal projectors. Hence the normal projection of their finite adapter displacement is exactly Z. Moreover,

$$\|\Delta A\|_F^2 + \|\Delta B\|_F^2 = \frac{2 \operatorname{tr}(\Sigma)}{|s|} = \frac{2\|Z\|_*}{|s|} \leq \varepsilon^2. \tag{47}$$

The zero matrix is realized by zero increments, completing the proof. □

The theorem describes the normal projection, not the whole weight displacement: the construction can also produce a tangent component. It characterizes all admissible factor increments, not the subset selected by a particular optimizer. The budget is a Euclidean step budget in the fixed factor coordinates, rather than a memory or compute budget; it need not be invariant to rescaling the factors. Within these coordinates, finite-step normal displacement has rank at most r and nuclear norm at most quadratic in ε.

Corollary E.2 (Optimal normal matching error). *Let $Y \in \mathcal{T}^{\perp}$ be a target normal displacement, with singular values $\sigma_1 \geq \cdots \geq \sigma_p \geq 0$, where $p = \min(d_o, d_i)$. Set $k = \min(r, p)$ and $c = |s|\varepsilon^2/2$. Define $a_i = (\sigma_i - \tau)_+$ for $i \leq k$, where $(x)_+ = \max\{x, 0\}$. Take $\tau = 0$ if $\sum_{i=1}^{k} \sigma_i \leq c$; otherwise take $\tau \in [0, \sigma_1]$ satisfying $\sum_{i=1}^{k}(\sigma_i - \tau)_+ = c$. Then*

$$\min_{Q \in \mathcal{R}_\varepsilon} \|Y - Q\|_F^2 = \sum_{i=1}^{k}(\sigma_i - a_i)^2 + \sum_{i=k+1}^{p} \sigma_i^2 \geq \sum_{i=k+1}^{p} \sigma_i^2. \tag{48}$$

Proof. By Theorem E.1, feasible matrices have rank at most r and nuclear norm at most c. First relax the normal-space constraint. For fixed singular values of Q, the trace inequality $\langle Y, Q\rangle_F \leq \sum_i \sigma_i(Y)\sigma_i(Q)$ shows that aligning its singular vectors with those of Y minimizes the Frobenius error. The relaxed problem reduces to

$$\min_{a_i \geq 0,\, \sum_{i=1}^{k} a_i \leq c} \sum_{i=1}^{k}(\sigma_i - a_i)^2 + \sum_{i=k+1}^{p} \sigma_i^2. \tag{49}$$

The optimality conditions give the stated common-threshold solution, which preserves the ordering and has $a_i = 0$ whenever $\sigma_i = 0$. Since the nonzero singular vectors of Y lie in the corresponding normal subspaces, the aligned solution also lies in $\mathcal{T}^{\perp}$. It is reachable by Theorem E.1, so the relaxation is exact. The nonnegative first sum gives the spectral-tail lower bound. □

For $Y = -\eta_n G_\perp$, the spectral-tail term is $\eta_n^2 \sum_{i>r} \sigma_i(G_\perp)^2$, with an empty sum interpreted as zero. Thus a nonzero tail cannot be matched by an adapter's normal displacement even without a step-size constraint. The first term in Eq. (48) additionally quantifies the error imposed by the factor-step budget. These are best attainable errors over admissible increments, not predictions of optimizer behavior.

E.3 Effect of direct normal compensation

Fix the same current state and the same candidate factor increments for both updates. Let $D = s[(B + \Delta B)(A + \Delta A) - BA]$ and $Q = P_\perp(D)$. Use the recovered full gradient G with the normalization of Section 3.1, and $G_\perp = P_\perp(G)$. For $\eta_n > 0$, take $-\eta_n G$ as the reference weight step. The normal-component matching errors of the adapter-only update D and the compensated update $D - \eta_n G_\perp$ are, respectively,

$$\begin{aligned} E_{\mathrm{LoRA}} &= \|P_\perp(D + \eta_n G)\|_F = \|Q + \eta_n G_\perp\|_F, \\ E_{\mathrm{GD}} &= \|P_\perp(D - \eta_n G_\perp + \eta_n G)\|_F = \|Q\|_F. \end{aligned} \tag{50}$$

Proposition E.3 (Improvement from direct normal compensation). *For the fixed candidate update above,*

$$E_{\mathrm{LoRA}}^2 - E_{\mathrm{GD}}^2 = \eta_n^2 \|G_\perp\|_F^2 + 2\eta_n \langle Q, G_\perp\rangle_F. \tag{51}$$

If $\|\Delta A\|_F^2 + \|\Delta B\|_F^2 \leq \varepsilon^2$ and $c = |s|\varepsilon^2/2$, then

$$E_{\mathrm{LoRA}}^2 - E_{\mathrm{GD}}^2 \geq \eta_n \|G_\perp\|_F \big(\eta_n \|G_\perp\|_F - 2c\big). \tag{52}$$

Consequently, $\eta_n \|G_\perp\|_F > 2c$ guarantees $E_{\mathrm{GD}} < E_{\mathrm{LoRA}}$.

Proof. Expanding the squared norms in Eq. (50) gives the identity. Theorem E.1 implies $\|Q\|_F \leq \|Q\|_* \leq c$. Cauchy–Schwarz then gives $\langle Q, G_\perp\rangle_F \geq -c\|G_\perp\|_F$, which proves the bound and the strict-improvement condition. □

The condition is sufficient, not necessary: the exact effect also depends on the alignment of Q with $G_\perp$. It identifies a regime where the desired normal step exceeds what the bounded bilinear remainder can compensate, so the direct branch improves normal matching for every candidate factor update within that budget. The reference is a current full-gradient step under the paper's normalization, not an FFT AdamW update. This comparison holds at the same state with identical candidate factor increments; it does not order the losses or complete training trajectories of the two methods. Together, the results explain both the finite-step normal displacement already available to LoRA and the additional role of direct normal compensation, while retaining the first-order interpretation in the main text.

Table 7: Hyperparameters used for GDLoRA in the natural language understanding experiments on the GLUE benchmark.

Model	Hyperparameter	SST-2	MRPC	CoLA	QNLI	RTE	STS-B
Optimizer				AdamW			
Warmup ratio				0.06			
Learning-rate schedule				Linear			
RoBERTa-Base	# GPUs			1			
	Epochs	60	30	80	25	160	80
	Learning rate (LoRA)			1e-4			
	Normal-gradient learning rate			1			
	LoRA rank			8			
	Maximum sequence length			512			
	Batch size per GPU			32			
RoBERTa-Large	# GPUs			1			
	Epochs	20	40	40	20	40	40
	Learning rate (LoRA)			1e-4			
	Normal-gradient learning rate			1			
	LoRA rank			8			
	Maximum sequence length			128			
	Batch size per GPU			32			

F More Experimental Results

F.1 Comparison with Additional PEFT Methods

Table 8 compares GDLoRA with standard LoRA ($r = 8$), the established LoRA variant AdaLoRA (Zhang et al., 2023), the additional PEFT methods IA^3(Liu et al., 2022) and FourierFT(Gao et al., 2024), and the recent LoRA variant MiSS(Kang & Yin, 2024), using Llama-3-8B on six mathematical reasoning tasks. GDLoRA achieves the highest average accuracy of 70.01%, exceeding LoRA by 2.50 percentage points and MiSS by 2.43 points. It obtains the best scores on MultiArith, SingleEq, SVAMP, and GSM8K, while IA^3 and FourierFT lead on AddSub and AQuA, respectively.

Table 8: Additional comparison on six mathematical reasoning tasks with Llama-3-8B. Scores are accuracy (%); Avg. is the arithmetic mean across the six tasks. The best mean score in each column is shown in **bold**.

Method	AddSub	MultiArith	SingleEq	SVAMP	GSM8K	AQuA	Avg.
LoRA ($r = 8$)	81.52	87.28	91.54	66.93	55.34	22.44	67.51
AdaLoRA	80.57	84.22	89.47	64.80	52.12	25.44	66.10
IA^3	**86.84**	83.50	86.42	64.50	50.64	26.38	66.38
FourierFT	85.06	88.67	88.98	64.30	45.40	**27.95**	66.73
MiSS	84.67	86.93	88.71	64.62	55.21	25.35	67.58
GDLoRA	84.81	**90.69**	**92.52**	**69.61**	**56.03**	26.38	**70.01**

F.2 Effect of Adapter Rank

Table 9 compares LoRA and GDLoRA at ranks $r \in \{8, 16, 32\}$ on Llama-3-8B. GDLoRA achieves average accuracies of 70.01%, 71.96%, and 72.81%, exceeding LoRA at the same ranks by 2.50, 2.29, and 2.01 percentage points, respectively. GDLoRA at rank 8 also outperforms LoRA at rank 16, while GDLoRA at rank 16 outperforms LoRA at rank 32. Rank-32 GDLoRA achieves the highest scores on five tasks; AQuA peaks at rank 16. These results show that GDLoRA's advantage persists across the evaluated ranks, although individual task scores do not always improve monotonically.

Table 9: LoRA and GDLoRA across adapter ranks on six mathematical reasoning tasks with Llama-3-8B. Scores are accuracy (%); Avg. is the arithmetic mean across tasks. GDLoRA rows are shaded blue, and the best score in each column is shown in **bold**.

Method	AddSub	MultiArith	SingleEq	SVAMP	GSM8K	AQuA	Avg.
LoRA ($r = 8$)	81.52	87.28	91.54	66.93	55.34	22.44	67.51
LoRA ($r = 16$)	83.56	91.34	92.43	68.81	56.97	24.89	69.67
LoRA ($r = 32$)	85.14	91.87	93.45	69.72	57.66	26.95	70.80
GDLoRA ($r = 8$)	84.81	90.69	92.52	69.61	56.03	26.38	70.01
GDLoRA ($r = 16$)	88.15	92.16	93.78	70.85	57.94	**28.89**	71.96
GDLoRA ($r = 32$)	**90.36**	**94.48**	**94.23**	**71.35**	**58.47**	27.94	**72.81**

G Experimental Details

Natural Language Understanding. For natural language understanding, we use RoBERTa-Base and RoBERTa-Large (Liu et al., 2019) on six tasks from the GLUE benchmark (Wang et al., 2018): SST-2, MRPC, CoLA, QNLI, RTE, and STS-B. We apply rank-8 LoRA adapters to the query and value projections in each Transformer layer.

Mathematical and Commonsense Reasoning. For mathematical reasoning, we fine-tune Llama-3-8B (Dubey et al., 2024) on Math10K (Hu et al., 2023) and evaluate it on six tasks: AddSub, MultiArith, SingleEq, SVAMP, GSM8K, and AQuA. For commonsense reasoning, we fine-tune Gemma-7B (Mesnard et al., 2024) on Commonsense170K (Hu et al., 2023) and evaluate it on eight tasks: OpenBookQA, ARC-Challenge, WinoGrande, PIQA, Social IQa, ARC-Easy, BoolQ, and HellaSwag. The two reasoning training collections contain approximately 10K and 170K examples, respectively. In both reasoning settings, we use rank-8 adapters for the query, key, and value projections.

Image Classification. For image classification, we evaluate ViT-Base and ViT-Large (Dosovitskiy et al., 2020) on eight datasets: OxfordPets (Parkhi et al., 2012), StanfordCars (Krause et al., 2013), CIFAR-10 (Krizhevsky et al., 2009), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), FGVC-Aircraft (Maji et al., 2013), RESISC45 (Cheng et al., 2017), and CIFAR-100 (Krizhevsky et al., 2009).

Baseline Methods. We compare GDLoRA with full fine-tuning (FFT), standard LoRA (Hu et al., 2021), Fira (Chen et al., 2024), and five LoRA variants: PiSSA (Meng et al., 2024), DoRA (Liu et al., 2024), rsLoRA (Kalajdzievski, 2023), LoRA-GA (Wang et al., 2024), and LoRA-Pro (Wang et al., 2025). FFT updates all model parameters during fine-tuning. The LoRA variants seek to narrow the performance gap between LoRA and FFT through improved initialization, weight parameterization, scaling, or optimization. During fine-tuning, their updates remain governed by their respective adapter parameterizations. Fira instead combines low-rank gradient projection with scaled residual updates to enable full-rank weight adaptation. GDLoRA supplements standard LoRA with normal-gradient updates applied directly to the base weights, extending adaptation beyond the first-order weight-space directions accessible through the current LoRA parameterization.

Table 10: Training and evaluation settings for mathematical reasoning with Llama-3-8B fine-tuned on Math10K.

Hyperparameter	Setting
Optimizer	AdamW
Learning rate (LoRA)	1e-4
Normal-gradient learning rate	0.5
Weight decay	0.0
Warmup ratio	0.06
Learning-rate schedule	Linear
Epochs	1
LoRA rank	8
Target modules	Q, K, V
Training precision	BF16
# GPUs per run	1
Micro-batch size per GPU	2
Gradient accumulation steps	8
Effective training batch size	16
Maximum sequence length	256

Experimental Hyperparameters The hyperparameters used by GDLoRA for natural language understanding, mathematical reasoning, commonsense reasoning, and image classification are provided in Tables 7, 10, 11, and 12, respectively. Within each comparison, all LoRA-based methods use the same adapter rank and target modules as GDLoRA. Apart from the main learning rate, the other training and optimization settings are also held consistent. For each language model–dataset setting (GLUE, Math10K, and Commonsense170K), we independently search for the best main-optimizer learning rate for every method, including FFT, Fira, GDLoRA, and all LoRA variants, using the same candidate set $\{1, 3, 5, 7, 9, 10\} \times 10^{-5}$. This grid starts at 10^{-5} with increments of 2×10^{-5}, and additionally includes the upper endpoint 10^{-4}. The selected rate is used for full-parameter optimization in FFT and adapter optimization in LoRA-based methods. The "Learning rate (LoRA)" entries in the language-task tables report the selected rates for GDLoRA; its normal-gradient learning rate is reported separately. Image-classification experiments retain the shared settings in Table 12.

For Fira, the SVD-based gradient projection bases are refreshed every 50 optimizer steps, except in the refresh-interval study in Appendix A, where the interval is varied explicitly.

All experiments were conducted on a server equipped with 8 NVIDIA A100 GPUs. The GPU counts reported in the tables refer to individual training runs. Each experiment was repeated with three different random seeds, and we report the mean and standard deviation across the three runs.

H Magnitude Control and Local Descent of the Normal Update

We analyze one normal-branch update with the adapters and all other weights fixed. Let $N > 0$ count the input positions, and write the recovered gradient as

$$H = \nabla_{W_{\text{base}}} \mathcal{L} = \delta^T X = \sum_{i=1}^{N} h_i, \qquad h_i = \delta_i x_i^T, \tag{53}$$

where x_i and δ_i are column vectors for position i. At the current parameters, define the fixed orthogonal projector $P_\perp(Z) = (I - UU^T)Z(I - VV^T)$ and set $G = H/N$, $G_\perp = P_\perp(G)$, and $\Delta W = -\eta_n G_\perp$ for $\eta_n > 0$. This analysis, like Eq. equation 18, assumes one common positive divisor, an exact orthogonal projector, and an update applied without rounding. The fixed-count accumulation convention is specified in Section H.1.

Proposition 1 (Magnitude control and local descent)**.** *The normal update satisfies*

$$\|\Delta W\|_F \le \frac{\eta_n}{N} \sum_{i=1}^{N} \|h_i\|_F. \tag{54}$$

Table 11: Training and evaluation settings for commonsense reasoning with Gemma-7B fine-tuned on Commonsense170K.

Hyperparameter	Setting
Optimizer	AdamW
Learning rate (LoRA)	3e-5
Normal-gradient learning rate	0.5
Weight decay	0.0
Warmup ratio	0.06
Learning-rate schedule	Linear
Epochs	1
LoRA rank	8
Target modules	Q, K, V
Training precision	BF16
# GPUs per run	1
Micro-batch size per GPU	2
Gradient accumulation steps	8
Effective training batch size	16
Maximum sequence length	256

Table 12: Hyperparameters used for GDLoRA in the image classification experiments with ViT-Base and ViT-Large. The normal-gradient learning rate controls the direct updates to the base weights.

Model	Hyperparameter	Oxford Pets	Stanford Cars	CIFAR-10	DTD	EuroSAT	FGVC	RESISC45	CIFAR-100
Optimizer						AdamW			
Weight decay						0.01			
Learning-rate schedule						Linear			
Epochs						20			
ViT-Base	# GPUs					1			
	Learning rate (head)	5e-3	5e-2	5e-2	5e-2	5e-2	5e-2	5e-2	1e-2
	Learning rate (LoRA)	5e-3	1e-2	1e-2	1e-2	1e-2	1e-2	1e-2	1e-2
	Normal-gradient learning rate					1e-3			
	LoRA rank					8			
	Batch size per GPU					128			
ViT-Large	# GPUs					1			
	Learning rate (head)	2e-2	1e-2	5e-3	1e-2	2e-2	1e-2	1e-2	5e-3
	Learning rate (LoRA)	1e-2	1e-2	1e-2	1e-2	1e-2	1e-2	1e-2	1e-2
	Normal-gradient learning rate					5e-2			
	LoRA rank					8			
	Batch size per GPU					32			

In particular, if $\|h_i\|_F \le C$ for all positions, then $\|\Delta W\|_F \le \eta_n C$. If $\mathcal{L}$ has a β-Lipschitz gradient with respect to W_{base} on a neighborhood containing the update segment, where $\beta > 0$, then

$$\mathcal{L}(W_{\text{base}} - \eta_n G_\perp) \le \mathcal{L}(W_{\text{base}}) - \eta_n \left(N - \frac{\beta \eta_n}{2} \right) \|G_\perp\|_F^2. \tag{55}$$

Consequently, for $G_\perp \neq 0$ and $0 < \eta_n < 2N/\beta$, the normal step strictly decreases the current loss.

Proof. An orthogonal projector is nonexpansive in the Frobenius norm. The triangle inequality therefore gives

$$\|\Delta W\|_F = \frac{\eta_n}{N} \|P_\perp(H)\|_F \le \frac{\eta_n}{N} \|H\|_F \le \frac{\eta_n}{N} \sum_{i=1}^{N} \|h_i\|_F. \tag{56}$$

The uniform bound follows immediately. Self-adjointness and idempotence of $P_\perp$ also imply

$$\langle H, G_\perp \rangle_F = N \langle G, P_\perp(G) \rangle_F = N \|G_\perp\|_F^2. \tag{57}$$

Applying the smoothness inequality to $\Delta W = -\eta_n G_\perp$ yields

$$\mathcal{L}(W_{\text{base}} + \Delta W) \le \mathcal{L}(W_{\text{base}}) + \langle H, \Delta W \rangle_F + \frac{\beta}{2}\|\Delta W\|_F^2$$
$$= \mathcal{L}(W_{\text{base}}) - \eta_n \left(N - \frac{\beta \eta_n}{2}\right) \|G_\perp\|_F^2. \tag{58}$$

This proves the descent claim. □

With a uniform per-position bound C, the corresponding unnormalized update has bound $\eta_n NC$. Dividing by N removes this explicit linear dependence from the magnitude bound, without requiring independent token contributions. This common positive scaling preserves the normal direction and its orthogonality to the current adapter tangent space.

These results support magnitude control of the normal-gradient branch. Any loss reduction is already included in δ; division by N is an additional step-size scaling, equivalent to using η_n/N for the unnormalized normal gradient when N is fixed. This equivalence does not establish invariance to sequence length. The descent result concerns a single normal step with fixed adapters and does not guarantee descent of the joint AdamW and base-weight update or convergence of the full training procedure.

H.1 Microbatch accumulation with a fixed input count

In the reported experiments, the normalization count is held fixed across microbatches within each configuration. Let an optimizer step contain M microbatches, each with input activations $X_j \in \mathbb{R}^{N_\mu \times d_i}$, where $N_\mu = bL$ is determined by the fixed micro-batch size b and input sequence dimension L. Let δ_j denote the backward signal, including the loss reduction and microbatch weighting applied by the trainer, but with any AMP loss scaling removed. Writing $H_j = \delta_j^T X_j$ and $H = \sum_j H_j$, the normal branch uses

$$\widetilde{G}_t = \sum_{j=1}^{M} \frac{H_j}{N_\mu} = \frac{H}{N_\mu}, \qquad G_{\perp,t} = \frac{1}{N_\mu} P_{\perp,t}(H). \tag{59}$$

Thus the additional normalization applies one common positive scale to the accumulated gradient, preserving its direction and the relative weighting of the microbatches. If ℓ_j denotes the trainer-weighted loss contribution, then $H = \nabla \mathcal{L}_{\text{acc}}$ for $\mathcal{L}_{\text{acc}} = \sum_j \ell_j$; both branches therefore use derivatives of the same accumulated objective under this convention. The normal-gradient controller performs no further division by M; the trainer's accumulation weighting is already contained in δ_j.

The effective learning rate for the accumulated unnormalized normal gradient is η_n/N_μ. For a different fixed input count N'_μ, the same ideal update at fixed H and projection bases is obtained by setting $\eta'_n = \eta_n N'_\mu / N_\mu$. This specifies the scaling convention when changing the microbatch shape; the reported GDLoRA comparisons retain their stated fixed microbatch and accumulation configurations. N_μ counts rows of the collated input tensor, not unmasked target tokens. With padding to the same fixed tensor shape, differences in unpadded sequence lengths therefore do not change this divisor.

All microbatches use the same pre-update parameters. At the optimizer boundary, we construct U_t, V_t from A_t, B_t and project the accumulated gradient before the adapter optimizer step. We cache this residual, update the adapters with AdamW, and then apply the cached normal update to the base weights. If AMP skips the optimizer step, the base update is also skipped; the accumulation and residual buffers are then cleared. For both reasoning settings in Tables 10 and 11, each run uses one GPU, micro-batch size 2, and 8 accumulation steps, yielding an effective batch size of 16. The maximum sequence length is 256, and the fixed count N_μ is determined by the input tensor shape used in each run.

H.2 Numerical bases and zero-rank factors

For each factor matrix $F \in \{B, A^T\}$, the implementation uses the absolute tolerance $\epsilon_{\text{orth}} = 10^{-6}$. It removes columns whose Euclidean norms do not exceed this tolerance. If no columns remain,

it returns an empty basis with shape $\mathrm{rows}(F) \times 0$. Otherwise, it computes an unpivoted reduced QR factorization of the retained columns and keeps only those columns of Q whose corresponding diagonal entries of R satisfy $|R_{ii}| > \epsilon_{\mathrm{orth}}$.

An empty basis has a zero associated projector, and its projection subtraction is skipped. In particular, $B_0 = 0$ gives an empty U_0 and

$$G_{\perp,0} = \widetilde{G}_0(I - V_0 V_0^T). \tag{60}$$

If both bases are empty, $G_{\perp,0} = \widetilde{G}_0$. Thus, arbitrary columns returned by QR on an all-zero matrix are not used as a nonzero projection basis.

This thresholded QR procedure explicitly handles zero initialization, but it is not a general rank-revealing factorization: for nonzero rank-deficient factors, unpivoted QR with diagonal thresholding need not recover the complete column space. The exact tangent-space statements assume bases of the true column spaces; the numerical implementation projects using the retained bases described above.