IMPROVING LOW-RANK ADAPTATION BY ADJUSTING ITS DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-Rank Adaptation (LoRA) efficiently fine-tunes pre-trained large language models with two trainable low-rank matrices B and A. Previous LoRA variants primarily improve performance by designing the initialization or update dynamics of $\mathbf{B}$ and $\mathbf{A}$ to preserve more information from the full parameter gradient matrix. Such information preservation depends on the alignment between the subspaces induced by $\mathbf{B}^t$, $\mathbf{A}^t$ and the current gradient, which is difficult to guarantee throughout training. Nevertheless, few works investigate the conditions on $\mathbf{B}^t$ and $\mathbf{A}^t$ that yield the most loss descent for a given current adapted weight $\mathbf{W}^t$ at an arbitrary training iterate. In this paper, we theoretically analyze the loss descent of LoRA from the t-th step to the $t+1$-th step and provide the conditions on $\mathbf{B}^t$ and $\mathbf{A}^t$ for the most loss descent: the largest and smallest singular values of $\mathbf{B}^t$ and $\mathbf{A}^t$ equal the square roots of the singular values of $\mathbf{B}^t\mathbf{A}^t$ with aligned internal singular subspaces. We reveal that the balance $(\mathbf{B}^t)^\top\mathbf{B}^t = \mathbf{A}^t(\mathbf{A}^t)^\top$ is a simple and easy-to-implement case of the conditions on $\mathbf{B}^t$ and $\mathbf{A}^t$, and propose the method Balance-Preserving Dynamics Adjustment for LoRA (BDA-LoRA) by adjusting the dynamics to make $\mathbf{B}^t$ and $\mathbf{A}^t$ balance in each step. Specifically, BDA-LoRA obtains a gradient-approximate balanced initialization via LoRA-based randomized SVD without computing or storing the full parameter gradient matrix, and modifies raw optimizer update directions slightly to make $\mathbf{B}^{t+1}$ and $\mathbf{A}^{t+1}$ balance from $\mathbf{B}^t$ and $\mathbf{A}^t$ for all t. Experiments on NLU, NLG and commonsense reasoning tasks show the effectiveness and superiority of BDA-LoRA.

1 INTRODUCTION

Low-Rank Adaptation (LoRA) (Hu et al., 2022) is widely used for parameter-efficient fine-tuning of large language models (LLMs) (Hu et al., 2023) and adapts a pre-trained weight matrix through

$$\mathbf{W} = \mathbf{W}^0 + s\mathbf{B}\mathbf{A},$$

where $\mathbf{W}^0 \in \mathbb{R}^{m\times n}$ is a frozen weight matrix of the pre-trained model, $\mathbf{B} \in \mathbb{R}^{m\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times n}$ are trainable, $r \ll \min\{m, n\}$, and s is a scaling factor. Existing LoRA variants primarily improve performance by designing the initialization or update dynamics of $\mathbf{B}$ and $\mathbf{A}$ to preserve more information from the full parameter gradient matrix. For example, LoRA-Pro (Wang et al., 2025b) adjusts the gradients of the two LoRA matrices so that their induced equivalent gradient better approximates the full parameter gradient matrix, while AltLoRA (Yu et al., 2025) alternates the updates of $\mathbf{B}$ and $\mathbf{A}$ for the same purpose. Such information preservation depends on the alignment between the subspaces induced by $\mathbf{B}^t$ and $\mathbf{A}^t$ and the current gradient, which is difficult to guarantee throughout training. However, relatively few existing works investigate, for a given current adapted weight $\mathbf{W}^t$ at an arbitrary training iterate, the conditions on $\mathbf{B}^t$ and $\mathbf{A}^t$ that yield the most loss descent.

Several works have tried to touch this problem. RefLoRA (Zhang et al., 2025a) directly studies it by deriving a one-step L-smooth loss upper bound for a given preconditioner $\mathbf{S}^t$ and minimizing that bound over $\mathbf{S}^t$, thereby obtaining a preconditioner associated with an exactly balanced refactoring. This establishes balance as an effective per-step configuration, but does not characterize all effective matrix pairs. BaLoRA (Castin et al., 2026) identifies a more general condition based on Hessian conditioning: near a limiting minimizer, better-conditioned factorizations yield faster asymptotic convergence, and the balance condition attains the optimal conditioning. However, the theory of BaLoRA is restrictive to the limiting solution and does not cover the full LoRA training trajectory.

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.

In this paper, we study the conditions on the two LoRA matrices throughout the optimization trajectory. Theoretical analysis of low-rank adaptation generally requires prior assumptions on the loss function. We therefore assume L-smoothness (Boyd & Vandenberghe, 2004) and identify the conditions on $\mathbf{B}^t$ and $\mathbf{A}^t$ that yield the most loss descent at any training step t: the largest and smallest singular values of both LoRA matrices equal the square roots of the corresponding singular values of $\mathbf{B}^t\mathbf{A}^t$, and their associated internal singular subspaces align. Balance is not necessary, but it provides a simple and easy-to-implement case that satisfies these conditions, so we maintain balance in our algorithm. Based on this choice, we propose BDA-LoRA (**B**alance-Preserving **D**ynamics **A**djustment for LoRA). BDA-LoRA first obtains a gradient-approximate balanced initialization through LoRA-based randomized SVD and multi-step accumulation, without computing or storing the full parameter gradient matrix and modifies raw optimizer update directions slightly through a balance-preserving matrix to make $\mathbf{B}^{t+1}$ and $\mathbf{A}^{t+1}$ balance from $\mathbf{B}^t$ and $\mathbf{A}^t$ for all t.

Our contributions are as follows:

- We prove the conditions on $\mathbf{B}^t$ and $\mathbf{A}^t$ that yield the most loss descent at any training iterate. The result applies throughout training, extends beyond the strictly balanced configuration identified by RefLoRA, and complements BaLoRA's local analysis near a limiting minimizer. See Section 3.1.
- We propose BDA-LoRA, which obtains a gradient-approximate balanced initialization through LoRA-based randomized SVD and multi-step accumulation without computing or storing the full parameter gradient matrix and maintains exact balance through a post-optimizer balance-preserving matrix while approximately minimizing the Frobenius-norm perturbation of the raw optimizer update directions.
- Across 17 NLU, NLG, and commonsense reasoning benchmarks, BDA-LoRA consistently achieves strong performance across tasks and model settings. These results provide empirical evidence consistent with our theoretical findings on the conditions that $\mathbf{B}^t$ and $\mathbf{A}^t$ should satisfy for a given current adapted weight $\mathbf{W}^t$.

2 Related Work

Initialization. Many recent studies have investigated LoRA initialization (Hayou et al., 2024a; Li et al., 2025b; Yang et al., 2024; Wang et al., 2025a; He et al., 2025; Li et al., 2025a; Tang et al., 2026; Lin et al., 2026; Meng et al., 2024). LoRA-GA (Wang et al., 2024) and LoRA-One (Zhang et al., 2025b) initialize $\mathbf{B}$ and $\mathbf{A}$ using principal singular directions of full parameter gradient matrices. Ding et al. (2026) initialize $\mathbf{A}$ as an equiangular tight frame so that the induced low-rank projection retains as much gradient information as possible when the full parameter gradient matrix is unavailable. These methods seek an informative starting point to preserve more information but do not specify which conditions $\mathbf{B}^t$ and $\mathbf{A}^t$ should satisfy throughout training. Moreover, LoRA-GA and LoRA-One require computing and storing full parameter gradient matrices, which incurs considerable additional computational and memory overhead compared with standard LoRA.

Update dynamics. Many recent studies have investigated LoRA update dynamics (Hayou et al., 2024b; Yaras et al., 2024; Lialin et al., 2024; Zhang & Pilanci, 2024; Zhang et al., 2026). Existing studies have shown that LoRA updates can be interpreted as projecting the full parameter gradient matrix onto a low-rank subspace. FLoRA (Hao et al., 2024) further interprets LoRA as gradient compression through Gaussian random projection: when the projection matrix is Gaussian, its Gram matrix recovers the identity in expectation, thereby preserving the gradient information in expectation. LoRA-Pro (Wang et al., 2025b) uses least squares to adjust the gradients of the LoRA matrices so that their induced equivalent gradient approximates the full parameter gradient matrix. AltLoRA (Yu et al., 2025) alternates the updates of $\mathbf{B}^t$ and $\mathbf{A}^t$ for the same purpose. GaLore (Zhao et al., 2024) projects full parameter gradient matrices into low-rank subspaces. These methods seek to better preserve information from the full parameter gradient matrix through their update dynamics. However, how much gradient information is retained depends on the relationship between the induced low-rank subspaces and the current gradient, which is difficult to guarantee throughout training. GaLore periodically updates its low-rank subspaces via gradient SVD, but may fail to track rapidly changing gradients and incurs additional decomposition overhead.

Conditions on $\mathbf{B}^t$ and $\mathbf{A}^t$. RefLoRA (Zhang et al., 2025a) derives a one-step loss upper bound under L-smoothness and minimizes it over a positive-definite preconditioner. This yields a closed-form refactoring, including a balance-inducing solution. Thus, RefLoRA establishes balance as an effective per-step configuration but does not characterize the more general conditions considered here. BaLoRA (Castin et al., 2026) studies the Hessian at a limiting minimizer. Its local asymptotic convergence rate is governed by the Hessian condition number, and this condition number depends on the singular values of $\mathbf{B}$ and $\mathbf{A}$. Balanced solutions attain the smallest value. This gives a condition more general than balance itself: a smaller Hessian condition number yields a tighter local asymptotic convergence bound, while balance provides an optimally conditioned construction. BaLoRA extends the analysis to higher-rank targets and interpolating nonlinear models through conditioning bounds whose best values are attained by balanced LoRA matrices. However, these results concern a limiting minimizer, and their best convergence bound uses a learning rate determined by the Hessian spectrum; they do not characterize arbitrary iterates along the LoRA training trajectory. Our theory instead characterizes effective conditions on the two LoRA matrices at arbitrary training iterates and shows that balance is sufficient but not necessary.

3 Our Work

3.1 Conditions for the most loss descent

LoRA (Hu et al., 2022) adapts a pre-trained weight matrix as $\mathbf{W} = \mathbf{W}^0 + s\mathbf{BA}$, where $\mathbf{W}^0 \in \mathbb{R}^{m\times n}$ is a frozen pre-trained weight matrix, $\mathbf{B} \in \mathbb{R}^{m\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times n}$ are trainable matrices, $1 \leqslant r < \min\{m, n\}$, and $s > 0$ is a fixed scaling factor. At step t, the adapted weight is $\mathbf{W}^t = \mathbf{W}^0 + s\mathbf{B}^t\mathbf{A}^t$. For simplicity, we set $s = 1$, which does not affect comparisons among matrix pairs with the same product $\mathbf{B}^t\mathbf{A}^t$. Let $\mathcal{L}$ denote the loss function, and let $\mathbf{G}^t := \nabla\mathcal{L}(\mathbf{W}^0 + \mathbf{B}^t\mathbf{A}^t) \in \mathbb{R}^{m\times n}$ denote the full parameter gradient matrix. Let $\mathbf{g}_B^t := \nabla_{\mathbf{B}^t}\mathcal{L}$ and $\mathbf{g}_A^t := \nabla_{\mathbf{A}^t}\mathcal{L}$ denote the gradients with respect to the two LoRA matrices. Theoretical analysis of low-rank adaptation generally requires prior assumptions on the loss function. A standard choice in optimization theory is L-smoothness (Boyd & Vandenberghe, 2004), which is also a common assumption in theoretical analyses of LoRA (Zhang et al., 2025a; Malinovsky et al., 2024; Ding et al., 2026).

Assumption 1 (L**-smoothness).** *The loss $\mathcal{L} : \mathbb{R}^{m\times n} \to \mathbb{R}$ is differentiable, and there exists $L > 0$ such that, for all $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{m\times n}$,*

$$\|\nabla\mathcal{L}(\mathbf{X}) - \nabla\mathcal{L}(\mathbf{Y})\| \leqslant L\,\|\mathbf{X} - \mathbf{Y}\|\,. \tag{1}$$

Consider a fine-tuning step $\mathbf{W}^{t+1} = \mathbf{W}^t + \Delta\mathbf{W}^t$, the following inequality holds for an L-smooth loss (Boyd & Vandenberghe, 2004): $\mathcal{L}(\mathbf{W}^{t+1}) - \mathcal{L}(\mathbf{W}^t) \leqslant \langle\nabla\mathcal{L}(\mathbf{W}^t), \Delta\mathbf{W}^t\rangle + \frac{L}{2}\|\Delta\mathbf{W}^t\|^2$. At an arbitrary iteration t, the matrices $\mathbf{B}^t$ and $\mathbf{A}^t$ determine the current LoRA update. For simultaneous gradient descent, substituting the product update $\mathbf{B}^{t+1}\mathbf{A}^{t+1} - \mathbf{B}^t\mathbf{A}^t = -\eta\big(\mathbf{B}^t(\mathbf{B}^t)^\top\mathbf{G}^t + \mathbf{G}^t(\mathbf{A}^t)^\top\mathbf{A}^t\big) + \eta^2\mathbf{G}^t(\mathbf{A}^t)^\top(\mathbf{B}^t)^\top\mathbf{G}^t$ into the L-smooth descent inequality gives

$$\mathcal{L}(\mathbf{W}^0 + \mathbf{B}^{t+1}\mathbf{A}^{t+1}) \leqslant \mathcal{L}(\mathbf{W}^0 + \mathbf{B}^t\mathbf{A}^t) - \eta\Big\langle\mathbf{G}^t, \mathbf{B}^t(\mathbf{B}^t)^\top\mathbf{G}^t + \mathbf{G}^t(\mathbf{A}^t)^\top\mathbf{A}^t\Big\rangle \tag{2}$$

$$+ \frac{L\eta^2}{2}\Big\|\mathbf{B}^t(\mathbf{B}^t)^\top\mathbf{G}^t + \mathbf{G}^t(\mathbf{A}^t)^\top\mathbf{A}^t\Big\|^2 + \varepsilon_\eta^t, \tag{3}$$

where the exact finite-step remainder is $\varepsilon_\eta^t := \eta^2\big\langle\mathbf{G}^t, \mathbf{G}^t(\mathbf{A}^t)^\top(\mathbf{B}^t)^\top\mathbf{G}^t\big\rangle - L\eta^3\big\langle\mathbf{B}^t(\mathbf{B}^t)^\top\mathbf{G}^t + \mathbf{G}^t(\mathbf{A}^t)^\top\mathbf{A}^t, \mathbf{G}^t(\mathbf{A}^t)^\top(\mathbf{B}^t)^\top\mathbf{G}^t\big\rangle + \frac{L\eta^4}{2}\big\|\mathbf{G}^t(\mathbf{A}^t)^\top(\mathbf{B}^t)^\top\mathbf{G}^t\big\|^2$. Assume that the candidate factors remain bounded, $\varepsilon_\eta^t = \mathcal{O}(\eta^2)$. Denote x_1 and x_r as the largest and smallest eigenvalues of $(\mathbf{B}^t)^\top\mathbf{B}^t$, and denote y_1 and y_r as the largest and smallest eigenvalues of $\mathbf{A}^t(\mathbf{A}^t)^\top$. Denote $d_-(\mathbf{B}^t, \mathbf{A}^t) := \min\{x_r, y_r\}$ and $d_+(\mathbf{B}^t, \mathbf{A}^t) := x_1 + y_1$. Let $q_\eta(\lambda) := -\eta\lambda + \frac{L\eta^2}{2}\lambda^2$ and $H_\eta(\mathbf{B}^t, \mathbf{A}^t) := \max\{q_\eta(d_-(\mathbf{B}^t, \mathbf{A}^t)), q_\eta(d_+(\mathbf{B}^t, \mathbf{A}^t))\}$. Write the compact SVD of the rank-r product as $\mathbf{B}^t\mathbf{A}^t = \mathbf{U}^t\mathbf{\Sigma}^t(\mathbf{V}^t)^\top$, and let $\mathbf{P}_{\mathbf{U}^t} = \mathbf{U}^t(\mathbf{U}^t)^\top$ and $\mathbf{P}_{\mathbf{V}^t} = \mathbf{V}^t(\mathbf{V}^t)^\top$. The effective gradient space is $\mathcal{T}_{\mathbf{B}^t\mathbf{A}^t} := \{\mathbf{X} : (\mathbf{I} - \mathbf{P}_{\mathbf{U}^t})\mathbf{X}(\mathbf{I} - \mathbf{P}_{\mathbf{V}^t}) = 0\}$, and the orthogonal projection of $\mathbf{G}^t$ onto this space is $\overline{\mathbf{G}}^t = \mathbf{P}_{\mathbf{U}^t}\mathbf{G}^t + \mathbf{G}^t\mathbf{P}_{\mathbf{V}^t} - \mathbf{P}_{\mathbf{U}^t}\mathbf{G}^t\mathbf{P}_{\mathbf{V}^t}$. The remaining component $\mathbf{G}_\perp^t = (\mathbf{I}-\mathbf{P}_{\mathbf{U}^t})\mathbf{G}^t(\mathbf{I}-\mathbf{P}_{\mathbf{V}^t})$ is orthogonal to $\mathcal{T}_{\mathbf{B}^t\mathbf{A}^t}$ and does not contribute to either LoRA matrix

gradient. Using the above notation, we obtain the following loss upper bound, whose derivation is provided in Appendix A.1 due to space limitations:

$$\mathcal{L}(\mathbf{W}^0 + \mathbf{B}^{t+1}\mathbf{A}^{t+1}) \leqslant \mathcal{L}(\mathbf{W}^0 + \mathbf{B}^t\mathbf{A}^t) + \|\overline{\mathbf{G}}^t\|^2 H_\eta(\mathbf{B}^t, \mathbf{A}^t) + \varepsilon_\eta^t. \tag{4}$$

Thus, the conditions that minimize H_η yield the most loss descent. We now compare different matrix pairs with the same current LoRA update, keeping the starting weight of the next optimization step fixed. Let $\mathbf{M} := \mathbf{B}^t\mathbf{A}^t$ denote this fixed product. We seek the conditions on $\mathbf{B}^t$ and $\mathbf{A}^t$ that minimize H_η subject to $\mathbf{B}^t\mathbf{A}^t = \mathbf{M}$. Denote the internal right singular subspaces of $\mathbf{B}^t$ associated with its largest (+) and smallest nonzero (−) singular values as $\mathcal{V}^+_{\mathbf{B}^t} := \{\mathbf{z} : (\mathbf{B}^t)^\top\mathbf{B}^t\mathbf{z} = \sigma_1^2(\mathbf{B}^t)\mathbf{z}\}$ and $\mathcal{V}^-_{\mathbf{B}^t} := \{\mathbf{z} : (\mathbf{B}^t)^\top\mathbf{B}^t\mathbf{z} = \sigma_r^2(\mathbf{B}^t)\mathbf{z}\}$, respectively. Analogously, let $\mathcal{U}^+_{\mathbf{A}^t}$ and $\mathcal{U}^-_{\mathbf{A}^t}$ denote the internal left singular subspaces of $\mathbf{A}^t$ associated with its largest and smallest nonzero singular values, respectively. Let k_+ and k_- denote the multiplicities of $\sigma_1(\mathbf{M})$ and $\sigma_r(\mathbf{M})$.

Theorem 1 (Conditions on $\mathbf{B}^t$ and $\mathbf{A}^t$ for the most loss descent). *Under Assumption 1, for $\eta > 0$, the pair $(\mathbf{B}^t_*, \mathbf{A}^t_*) \in \arg\min_{\mathbf{B}^t,\mathbf{A}^t:\,\mathbf{B}^t\mathbf{A}^t=\mathbf{M}} H_\eta(\mathbf{B}^t, \mathbf{A}^t)$ if*

$$\begin{aligned} \sigma_1(\mathbf{B}^t_*) = \sigma_1(\mathbf{A}^t_*) = \sqrt{\sigma_1(\mathbf{M})}, \qquad & \sigma_r(\mathbf{B}^t_*) = \sigma_r(\mathbf{A}^t_*) = \sqrt{\sigma_r(\mathbf{M})}, \\ \dim(\mathcal{V}^+_{\mathbf{B}^t_*} \cap \mathcal{U}^+_{\mathbf{A}^t_*}) \geq k_+, \qquad & \dim(\mathcal{V}^-_{\mathbf{B}^t_*} \cap \mathcal{U}^-_{\mathbf{A}^t_*}) \geq k_-. \end{aligned} \tag{5}$$

These conditions yield $d_-(\mathbf{B}^t_*, \mathbf{A}^t_*) = \sigma_r(\mathbf{M})$ and $d_+(\mathbf{B}^t_*, \mathbf{A}^t_*) = 2\sigma_1(\mathbf{M})$. Consequently, every such pair yields the most loss descent in Eq. (4).

We leave the proof of Theorem 1 to Appendix A.1 due to space limitations. Theorem 1 requires that the largest and smallest singular values of $\mathbf{B}^t$ and $\mathbf{A}^t$ equal the square roots of the corresponding singular values of $\mathbf{B}^t\mathbf{A}^t$, with aligned internal singular subspaces. These conditions are more general than RefLoRA's strict balance (Zhang et al., 2025a) and complement BaLoRA's local convergence analysis (Castin et al., 2026) by covering arbitrary training iterates.

The conditions above allow the two matrices to have different energies along their interior singular directions. To see this, suppose $r \geq 3$ and $\sigma_1(\mathbf{B}^t\mathbf{A}^t) > \sigma_i(\mathbf{B}^t\mathbf{A}^t) > \sigma_r(\mathbf{B}^t\mathbf{A}^t)$ for $i = 2, \ldots, r-1$. Let $\alpha = (\alpha_2, \ldots, \alpha_{r-1})$, where each $\alpha_i > 0$ is chosen close to one so that both $\alpha_i\sqrt{\sigma_i(\mathbf{B}^t\mathbf{A}^t)}$ and $\alpha_i^{-1}\sqrt{\sigma_i(\mathbf{B}^t\mathbf{A}^t)}$ remain strictly between $\sqrt{\sigma_r(\mathbf{B}^t\mathbf{A}^t)}$ and $\sqrt{\sigma_1(\mathbf{B}^t\mathbf{A}^t)}$, $\forall i = 2, \ldots, r-1$. Set $\mathbf{D}_\alpha := \mathrm{diag}(1, \alpha_2, \ldots, \alpha_{r-1}, 1)$, $\mathbf{B}_\alpha := \mathbf{U}^t(\mathbf{\Sigma}^t)^{\frac{1}{2}}\mathbf{D}_\alpha$, and $\mathbf{A}_\alpha := \mathbf{D}_\alpha^{-1}(\mathbf{\Sigma}^t)^{\frac{1}{2}}(\mathbf{V}^t)^\top$. Their product remains $\mathbf{B}^t\mathbf{A}^t$, and the rescaling leaves the extreme singular values and their internal subspaces unchanged, so Eq. (5) still holds. Hence each α_i controls the energy allocation along one interior direction. When all $\alpha_i = 1$, $\mathbf{D}_\alpha = \mathbf{I}_r$ and the resulting configuration is balanced. Other admissible choices can produce unbalanced configurations while still yielding the most loss descent. Therefore, Eq. (5) is more general than the strict balance used by RefLoRA (Zhang et al., 2025a). Since $\mathbf{A}^t$ and $\mathbf{B}^t$ play distinct roles (Zhu et al., 2024), the preferable energy allocation may be task-dependent. We leave such task-specific choices for future work and adopt the following balance condition:

$$(\mathbf{B}^t)^\top\mathbf{B}^t = \mathbf{A}^t(\mathbf{A}^t)^\top \tag{6}$$

as a simple and easy-to-implement case of the conditions on $\mathbf{B}^t$ and $\mathbf{A}^t$. We next propose a method for LoRA to maintain this balance throughout training.

3.2 METHOD

Preserving balance under optimizer updates. Suppose the LoRA matrices are balanced at iteration t, *i.e.* $(\mathbf{B}^t)^\top\mathbf{B}^t = \mathbf{A}^t(\mathbf{A}^t)^\top$. Under SGD, substituting the gradients of the LoRA matrices into the update gives $(\mathbf{B}^{t+1})^\top\mathbf{B}^{t+1} - \mathbf{A}^{t+1}(\mathbf{A}^{t+1})^\top = \eta^2(\mathbf{A}^t(\mathbf{G}^t)^\top\mathbf{G}^t(\mathbf{A}^t)^\top - (\mathbf{B}^t)^\top\mathbf{G}^t(\mathbf{G}^t)^\top\mathbf{B}^t)$. For a general base optimizer such as AdamW, let $\mathbf{U}^t_B \in \mathbb{R}^{m\times r}$ and $\mathbf{U}^t_A \in \mathbb{R}^{r\times n}$ denote the raw optimizer update directions. If these directions are applied directly, then $\mathbf{B}^{t+1} = \mathbf{B}^t - \eta\mathbf{U}^t_B$ and $\mathbf{A}^{t+1} = \mathbf{A}^t - \eta\mathbf{U}^t_A$. Expanding their next-step balance residual gives $(\mathbf{B}^{t+1})^\top\mathbf{B}^{t+1} - \mathbf{A}^{t+1}(\mathbf{A}^{t+1})^\top = \eta(\mathbf{A}^t(\mathbf{U}^t_A)^\top + \mathbf{U}^t_A(\mathbf{A}^t)^\top - (\mathbf{B}^t)^\top\mathbf{U}^t_B - (\mathbf{U}^t_B)^\top\mathbf{B}^t) + \eta^2((\mathbf{U}^t_B)^\top\mathbf{U}^t_B - \mathbf{U}^t_A(\mathbf{U}^t_A)^\top)$. Thus, neither SGD nor adaptive optimizers naturally preserve exact balance at finite learning rates. BaLoRA's SVD-based refactorization (Castin et al., 2026) may change the coordinates of the LoRA matrices since singular-vector signs and bases within the repeated singular subspaces are not unique (Papadopoulo & Lourakis, 2000), potentially requiring optimizer-state adjustments (Lialin et al., 2024).

RefLoRA's gradient preconditioning (Zhang et al., 2025a) may produce different updates from explicit balance preservation because adaptive optimizers such as Adam are not equivariant to general rotations (Xie et al., 2025). We therefore seek modified update directions that preserve $\mathbf{B}^{t+1}$ and $\mathbf{A}^{t+1}$ exactly balanced from balanced $\mathbf{B}^t$ and $\mathbf{A}^t$ while changing the raw optimizer update directions as little as possible. Specifically, we replace $\mathbf{U}_B^t$ and $\mathbf{U}_A^t$ with $\mathbf{Z}_B^t$ and $\mathbf{Z}_A^t$, giving $\mathbf{B}^{t+1} = \mathbf{B}^t - \eta\mathbf{Z}_B^t$ and $\mathbf{A}^{t+1} = \mathbf{A}^t - \eta\mathbf{Z}_A^t$. We formulate this goal as the following constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{Z}_B^t, \mathbf{Z}_A^t} \quad & \frac{1}{2}\|\mathbf{Z}_B^t - \mathbf{U}_B^t\|^2 + \frac{1}{2}\|\mathbf{Z}_A^t - \mathbf{U}_A^t\|^2, \\ \text{s.t.} \quad & (\mathbf{B}^t - \eta\mathbf{Z}_B^t)^\top(\mathbf{B}^t - \eta\mathbf{Z}_B^t) = (\mathbf{A}^t - \eta\mathbf{Z}_A^t)(\mathbf{A}^t - \eta\mathbf{Z}_A^t)^\top. \end{aligned} \tag{7}$$

The objective measures the change from the raw optimizer update directions, while the constraint requires the modified next-step matrices to be exactly balanced. When $(\mathbf{B}^t)^\top\mathbf{B}^t = \mathbf{A}^t(\mathbf{A}^t)^\top$, the constraint equivalently requires $(\mathbf{B}^t)^\top\mathbf{Z}_B^t + (\mathbf{Z}_B^t)^\top\mathbf{B}^t - \mathbf{Z}_A^t(\mathbf{A}^t)^\top - \mathbf{A}^t(\mathbf{Z}_A^t)^\top - \eta((\mathbf{Z}_B^t)^\top\mathbf{Z}_B^t - \mathbf{Z}_A^t(\mathbf{Z}_A^t)^\top) = 0$. Using the Lagrange multiplier method with the Lagrange multiplier $\mathbf{P}_\eta^t \in \mathbb{R}^{r\times r}$ gives $\mathbf{Z}_B^t = (\mathbf{U}_B^t - \mathbf{B}^t\mathbf{P}_\eta^t)(\mathbf{I}_r - \eta\mathbf{P}_\eta^t)^{-1}$ and $\mathbf{Z}_A^t = (\mathbf{I}_r + \eta\mathbf{P}_\eta^t)^{-1}(\mathbf{U}_A^t + \mathbf{P}_\eta^t\mathbf{A}^t)$. Computing these exact update directions requires solving a nonlinear matrix equation and evaluating two inverse factors, resulting in computational overhead. For bounded $\mathbf{P}_\eta^t$, both inverse factors equal $\mathbf{I}_r + \mathcal{O}(\eta)$, so we omit them and use the modified update directions:

$$\mathbf{Z}_B^t = \mathbf{U}_B^t - \mathbf{B}^t\mathbf{P}_\eta^t, \qquad \mathbf{Z}_A^t = \mathbf{U}_A^t + \mathbf{P}_\eta^t\mathbf{A}^t, \tag{8}$$

where $\mathbf{P}_\eta^t \in \mathbb{R}^{r\times r}$ is the balance-preserving matrix. We determine it by requiring these update directions to satisfy the finite-step constraint in Eq. (7), which makes the resulting LoRA matrices exactly balanced. Let $\mathbf{S}^t := (\mathbf{B}^t)^\top\mathbf{B}^t = \mathbf{A}^t(\mathbf{A}^t)^\top$. Substituting Eq. (8) into these updates and imposing $(\mathbf{B}^{t+1})^\top\mathbf{B}^{t+1} = \mathbf{A}^{t+1}(\mathbf{A}^{t+1})^\top$ yields the following equation:

$$\mathbf{P}_\eta^t(\mathbf{K}^t)^\top + \mathbf{K}^t\mathbf{P}_\eta^t = \mathbf{Q}^t, \tag{9}$$

where $\mathbf{K}^t := 2\mathbf{S}^t - \eta((\mathbf{U}_B^t)^\top\mathbf{B}^t + \mathbf{U}_A^t(\mathbf{A}^t)^\top)$ and $\mathbf{Q}^t := ((\mathbf{U}_B^t)^\top\mathbf{B}^t + (\mathbf{B}^t)^\top\mathbf{U}_B^t) - (\mathbf{U}_A^t(\mathbf{A}^t)^\top + \mathbf{A}^t(\mathbf{U}_A^t)^\top) - \eta((\mathbf{U}_B^t)^\top\mathbf{U}_B^t - \mathbf{U}_A^t(\mathbf{U}_A^t)^\top)$. This is a standard continuous Lyapunov equation (Simoncini, 2016) involving only $r\times r$ matrices, and $\mathbf{K}^t$ and $\mathbf{Q}^t$ are computable from the current LoRA matrices and raw optimizer update directions. When η is small enough that $\mathbf{K}^t$ has eigenvalues with positive real parts, the equation has a unique solution $\mathbf{P}_\eta^t$, which can be obtained with a standard numerical Lyapunov solver. Solving this $r \times r$ equation requires $\mathcal{O}(r^3)$ time. For bounded $\mathbf{P}_\eta^t$, the omitted inverse factors in the exact update directions obtained by the Lagrange multiplier method differ from $\mathbf{I}_r$ by $\mathcal{O}(\eta)$. Thus, the resulting update approximately retains the minimum-change form while satisfying the finite-step balance constraint. The detailed derivation of the above results is deferred to Appendix A.2 due to space limitations.

Gradient-approximate balanced initialization. The matrix-based adjustment described above preserves balance recursively: balanced LoRA matrices at step t remain balanced at step $t+1$. Therefore, we first construct balanced initial LoRA matrices $\mathbf{B}^1$ and $\mathbf{A}^1$. Let $\mathbf{g}_i := \nabla\mathcal{L}_i(\mathbf{W}^0) \in \mathbb{R}^{m\times n}$ denote the full parameter gradient matrix computed on the i-th minibatch, where $\mathcal{L}_i$ is its minibatch loss. Each $\mathbf{g}_i$ estimates the gradient of the full loss at the initial weight. Previous works construct $\mathbf{B}^1\mathbf{A}^1$ to approximate the first-step full parameter gradient matrix. For example, LoRA-One (Zhang et al., 2025b) initializes the two matrices using the dominant singular components obtained by SVD of a gradient estimate $\frac{1}{N}\sum_{i=1}^N \mathbf{g}_i$ aggregated over N minibatches. Even with randomized SVD, computing these full-size minibatch gradients and storing their aggregate still incurs greater computation and memory costs than standard LoRA. Randomized SVD uses random projections and QR factorizations to build an orthonormal basis, then performs a small projected SVD to approximate the dominant singular components with high probability (Halko et al., 2011). A QR factorization decomposes $\mathbf{X}$ as $\mathbf{X} = \mathbf{QR}$, where $\mathbf{Q}$ has orthonormal columns and $\mathbf{R}$ is upper triangular. We instead develop a LoRA-based randomized SVD that uses only LoRA factor gradients to obtain an approximate rank-r truncated SVD of each minibatch gradient matrix $\mathbf{g}_i$, without explicitly computing or storing the full parameter gradient matrix. The complete procedure is given in Algorithm 1.

Low-rank streaming gradient accumulation. We estimate the gradient using multiple minibatches. However, storing every estimate or materializing their full $m \times n$ sum would incur substantial memory cost. We update the accumulator through LowRankMerge, which adds the new

minibatch's low-rank gradient approximation to the current accumulator and truncates the sum to rank r. At minibatch i, denote the current low-rank accumulator and the approximation of the new minibatch gradient $\mathbf{g}_i$ as $\mathbf{H}^{i-1} = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^\top$ and $\widetilde{\mathbf{g}}_i = \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^\top$. We then construct the joint orthonormal bases $\mathbf{U}_c = \mathrm{QR}([\mathbf{U}_1, \mathbf{U}_2])$ and $\mathbf{V}_c = \mathrm{QR}([\mathbf{V}_1, \mathbf{V}_2])$, and form the small core matrix $\mathbf{K} = (\mathbf{U}_c^\top\mathbf{U}_1)\mathbf{\Sigma}_1(\mathbf{V}_1^\top\mathbf{V}_c) + (\mathbf{U}_c^\top\mathbf{U}_2)\mathbf{\Sigma}_2(\mathbf{V}_2^\top\mathbf{V}_c)$. After computing $\mathbf{K} = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}}^\top$, we retain its top r singular triplets by setting $\mathbf{U} = (\mathbf{U}_c\widetilde{\mathbf{U}})_{:,1:r}$, $\mathbf{\Sigma} = \widetilde{\mathbf{\Sigma}}_{1:r,1:r}$, and $\mathbf{V} = (\mathbf{V}_c\widetilde{\mathbf{V}})_{:,1:r}$. We then set $\mathbf{H}^i := \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ and repeat this merge across the N initialization minibatches. The detailed procedure is summarized in Algorithm 4.

Algorithm 1 Alternating RSVD via LoRA Gradients

Require: $\mathbf{y} = \mathbf{W}^0\mathbf{h}$, $\mathbf{h}$, $\mathrm{d}\mathbf{y} = \nabla_{\mathbf{y}}\mathcal{L}_j$, rank r, oversampling p, power iterations q
Ensure: Rank-r approximation $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ of $\mathbf{g}_j$

$\ell \leftarrow r + p$
$\mathbf{A} \in \mathbb{R}^{\ell\times n} \sim \mathcal{N}(0,1)$; $\mathbf{B} \leftarrow \mathbf{0}$
Compute the $\mathbf{B}$ gradient: $\nabla_{\mathbf{B}} \leftarrow (\mathrm{d}\mathbf{y})(\mathbf{A}\mathbf{h})^\top$
for $i = 1$ **to** q **do**
  $\mathbf{Q} \leftarrow \mathrm{QR}(\nabla_{\mathbf{B}})$; $\mathbf{B} \leftarrow \mathbf{Q}$; $\mathbf{A} \leftarrow \mathbf{0}$
  Compute the $\mathbf{A}$ gradient: $\nabla_{\mathbf{A}} \leftarrow (\mathbf{B}^\top\mathrm{d}\mathbf{y})\mathbf{h}^\top$
  $\mathbf{Q} \leftarrow \mathrm{QR}(\nabla_{\mathbf{A}}^\top)$; $\mathbf{A} \leftarrow \mathbf{Q}^\top$; $\mathbf{B} \leftarrow \mathbf{0}$
  Compute the $\mathbf{B}$ gradient: $\nabla_{\mathbf{B}} \leftarrow (\mathrm{d}\mathbf{y})(\mathbf{A}\mathbf{h})^\top$
end for
$\mathbf{Q} \leftarrow \mathrm{QR}(\nabla_{\mathbf{B}})$; $\mathbf{B} \leftarrow \mathbf{Q}$; $\mathbf{A} \leftarrow \mathbf{0}$
Compute the $\mathbf{A}$ gradient: $\nabla_{\mathbf{A}} \leftarrow (\mathbf{B}^\top\mathrm{d}\mathbf{y})\mathbf{h}^\top$
$\widetilde{\mathbf{U}}, \mathbf{\Sigma}, \mathbf{V} \leftarrow \mathrm{SVD}(\nabla_{\mathbf{A}})$; $\mathbf{U} \leftarrow \mathbf{Q}\widetilde{\mathbf{U}}$
return $\mathbf{U}_{:,1:r}$, $\mathbf{\Sigma}_{1:r,1:r}$, $\mathbf{V}_{:,1:r}$

Our Method. We propose BDA-LoRA (Balance-Preserving Dynamics Adjustment for LoRA), which combines a gradient-approximate balanced initialization with a post-optimizer balance-preserving adjustment. LoRA-based randomized SVD and low-rank streaming accumulation construct the initialization without computing or storing the full-size gradient matrices. During training, the balance-preserving matrix maintains exact balance while approximately minimizing the Frobenius-norm change to the raw optimizer update directions. The overall procedure is summarized in Algorithm 2. Compared with gradient-based initialization methods such as LoRA-One and LoRA-GA, our initialization has lower computational and memory complexity, while the training-time computation and memory complexity are of the same order as RefLoRA and BaLoRA. We provide the detailed complexity analysis in Appendix C.

Discussion with Existing Works. Existing studies often interpret LoRA updates as projecting the full parameter gradient matrix onto low-rank subspaces induced by the two LoRA matrices. Accordingly, many works improve LoRA initialization or update dynamics to better preserve information from the full parameter gradient matrix. FLoRA (Hao et al., 2024) initializes $\mathbf{A}$ as a Gaussian matrix so that its Gram matrix is the identity in expectation, thereby preserving gradient information. LoRA-Pro (Wang et al., 2025b) and AltLoRA (Yu et al., 2025) modify the updates of the LoRA matrices so that the induced update better approximates the full parameter gradient matrix, while Ding et al. (2026) use an equiangular tight frame initialization to maximize the retained gradient information in the worst case when the full gradient is unavailable. However, how much gradient information is retained depends on the relationship between the low-rank subspaces induced by $\mathbf{B}^t$ and $\mathbf{A}^t$ and the current gradient, which is difficult to guarantee throughout training. Our analysis is complementary: rather than directly seeking a better gradient-preserving subspace, we study, for the same current adapted weight $\mathbf{W}^t$, what properties of $\mathbf{B}^t$ and $\mathbf{A}^t$ lead to the most loss descent.

BaLoRA and RefLoRA incorporate balance differently from our method. BaLoRA uses SVD-based refactorization to construct balanced factors while preserving their product, but SVD non-uniqueness (Papadopoulo & Lourakis, 2000) may change factor coordinates and thus require optimizer-state adjustment (Lialin et al., 2024). RefLoRA instead preconditions the gradients of the LoRA matrices before optimization, which can differ from explicit balance preservation under adaptive optimizers such as Adam, which is not equivariant to general rotations (Xie et al., 2025). BDA-LoRA slightly modifies the raw update directions produced by the base optimizer through

Algorithm 2 BDA-LoRA

Require: Pre-trained weight $\mathbf{W}^0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, rank r, oversampling p, power iterations q, initialization batches N, training steps T, learning rate η, LoRA scale α, initialization scale $\gamma > 0$
Ensure: Fine-tuned $\mathbf{W}^{T+1}$

Phase 1: Build initial subspace via multi-step gradient accumulation
Initialize an empty low-rank gradient accumulator $\mathcal{A}$
for $j = 1$ **to** N **do**
  $(\mathbf{U}_j, \mathbf{\Sigma}_j, \mathbf{V}_j) \leftarrow$ LoRA-Based Randomized SVD$(\mathbf{W}^0, \mathcal{L}_j; r, p, q)$ ▷ *LoRA-Based RSVD*
  if $j = 1$ **then**
    $\mathcal{A} \leftarrow (\mathbf{U}_j,\ \mathbf{\Sigma}_j,\ \mathbf{V}_j)$
  else
    $\mathcal{A} \leftarrow$ LowRankMerge$(\mathcal{A},\ (\mathbf{U}_j,\ \mathbf{\Sigma}_j,\ \mathbf{V}_j))$ ▷ *Streaming Low Rank Accumulation*
  end if
end for
$(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}) \leftarrow \mathcal{A}$
Let ρ be the largest singular value of the accumulated gradient approximation
$\mathbf{B}^1 \leftarrow \sqrt{\gamma}\,\mathbf{U}(\mathbf{\Sigma}/\rho)^{1/2}$; $\mathbf{A}^1 \leftarrow \sqrt{\gamma}\,(\mathbf{\Sigma}/\rho)^{1/2}\mathbf{V}^\top$
$\mathbf{W}^0 \leftarrow \mathbf{W}^0 - (\alpha/\sqrt{r})\mathbf{B}^1\mathbf{A}^1$
Phase 2: Training with balance-preserving updates
for $t = 1$ **to** T **do**
  $\mathbf{U}_B^t, \mathbf{U}_A^t \leftarrow$ Optimizer$(\mathbf{G}_B^t, \mathbf{G}_A^t)$
  Solve $\mathbf{P}_\eta^t(\mathbf{K}^t)^\top + \mathbf{K}^t\mathbf{P}_\eta^t = \mathbf{Q}^t$ for $\mathbf{P}_\eta^t$ ▷ *Balance-Preserving Matrix*
  $\mathbf{B}^{t+1} \leftarrow \mathbf{B}^t - \eta(\mathbf{U}_B^t - \mathbf{B}^t\mathbf{P}_\eta^t)$ $\mathbf{A}^{t+1} \leftarrow \mathbf{A}^t - \eta(\mathbf{U}_A^t + \mathbf{P}_\eta^t\mathbf{A}^t)$
end for
$\mathbf{W}^{T+1} \leftarrow \mathbf{W}^0 + (\alpha/\sqrt{r})\mathbf{B}^{T+1}\mathbf{A}^{T+1}$
return $\mathbf{W}^{T+1}$

a balance-preserving matrix, so that $\mathbf{B}^{t+1}$ and $\mathbf{A}^{t+1}$ remain balanced given balanced $\mathbf{B}^t$ and $\mathbf{A}^t$, while approximately minimizing the Frobenius-norm change to the raw optimizer update directions.

4 Experiments

Datasets. We evaluate BDA-LoRA on natural language understanding (NLU), natural language generation (NLG), and commonsense reasoning tasks. For NLU, we use seven GLUE tasks (Wang et al., 2019): CoLA (Warstadt et al., 2019), MRPC (Dolan & Brockett, 2005), MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), QQP[1], RTE (Wang et al., 2019), and SST-2 (Socher et al., 2013). For NLG, we fine-tune on 100K samples from MetaMathQA (Yu et al., 2024) for GSM8K (Cobbe et al., 2021), and 100K samples from CodeFeedback-Filtered-Instruction (Zheng et al., 2024) for HumanEval (Chen et al., 2021). For commonsense reasoning, we use BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), Social IQA (SIQA) (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-Challenge and ARC-Easy (Clark et al., 2018), and OpenBookQA (OBQA) (Mihaylov et al., 2018). We report Matthews correlation coefficient (MCC) for CoLA (Wang et al., 2019), accuracy for the remaining GLUE tasks and commonsense reasoning and GSM8K, and pass@1 for HumanEval (Chen et al., 2021).

Baselines. We compare with LoRA (Hu et al., 2022), LoRA-One (Zhang et al., 2025b), LoRA-GA (Wang et al., 2024), PiSSA (Meng et al., 2024), DoRA (Liu et al., 2024), RefLoRA (Zhang et al., 2025a), BaLoRA (Castin et al., 2026), and AltLoRA (Yu et al., 2025). We additionally include BDA-LoRA*, an ablation variant of BDA-LoRA without balance preservation.

Experimental Setup. We use DeBERTa-v3-base (He et al., 2023) for NLU, Meta-Llama-3-8B (Llama Team, 2024) and Gemma-7B (Team, 2024) for NLG, and Meta-Llama-3-8B for commonsense reasoning. Each experiment is repeated over five random seeds, and we report the mean and standard deviation. For our BDA-LoRA, let $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be the rank-r SVD of the accumulated gradient approximation, and let ρ be its largest singular value. Following the normalized spectral initializa-

[1] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

tion of LoRA-One (Zhang et al., 2025b), we set $\mathbf{B}^1 = \sqrt{\gamma}\,\mathbf{U}(\mathbf{\Sigma}/\rho)^{1/2}$ and $\mathbf{A}^1 = \sqrt{\gamma}\,(\mathbf{\Sigma}/\rho)^{1/2}\mathbf{V}^\top$, with $\gamma = 1/128$. Since $\mathbf{U}$ and $\mathbf{V}$ have orthonormal columns, $(\mathbf{B}^1)^\top\mathbf{B}^1 = \gamma\mathbf{\Sigma}/\rho = \mathbf{A}^1(\mathbf{A}^1)^\top$, so the initialization is balanced. We use rank-stabilized LoRA (rsLoRA) scaling $s = \alpha/\sqrt{r}$ as in (Kalajdzievski, 2023) and offset the frozen weight by $\mathbf{W}^0 \leftarrow \mathbf{W}^0 - s\mathbf{B}^1\mathbf{A}^1$ to leave the initial effective weight unchanged. Detailed hyperparameters are provided in Appendix D.

Natural Language Understanding. We fine-tune DeBERTa-v3-base with all compared methods on the seven GLUE benchmark tasks and report the mean and standard deviation over five runs. The results are summarized in Table 1. As shown in Table 1, BDA-LoRA outperforms the baseline methods on the GLUE tasks. Its consistent gains over BDA-LoRA* with the same initialization support the benefit of preserving balance during training. The weaker results of RefLoRA and BaLoRA are consistent with the optimizer-related limitations discussed above: RefLoRA's gradient preconditioning may yield different updates from explicit balance preservation under Adam, while BaLoRA's SVD refactorization may misalign factor coordinates with optimizer states.

Table 1: Test score (%) of DeBERTa-v3-base with all compared methods on GLUE benchmark datasets. The best PEFT result is in **bold** and the second best is underlined.

Method	CoLA	MRPC	MNLI	QNLI	QQP	RTE	SST-2
LoRA	64.42 ± 0.39	81.94 ± 0.99	90.06 ± 0.04	93.84 ± 0.15	90.33 ± 0.13	69.13 ± 3.83	95.28 ± 0.10
LoRA-One	63.95 ± 0.63	86.11 ± 1.73	90.07 ± 0.07	93.99 ± 0.04	90.35 ± 0.04	68.11 ± 0.55	95.26 ± 0.12
LoRA-GA	66.12 ± 1.02	86.60 ± 2.33	90.05 ± 0.14	93.75 ± 0.10	91.04 ± 0.15	74.85 ± 1.27	95.24 ± 0.41
PiSSA	64.99 ± 1.50	84.31 ± 1.91	89.91 ± 0.13	93.42 ± 0.31	90.80 ± 0.06	69.92 ± 2.76	95.30 ± 0.40
DoRA	64.15 ± 0.38	81.86 ± 0.74	89.85 ± 0.07	93.64 ± 0.22	90.29 ± 0.19	68.59 ± 2.82	94.95 ± 0.34
RefLoRA	63.65 ± 0.59	81.21 ± 0.31	89.15 ± 0.03	93.28 ± 0.11	89.50 ± 0.07	66.67 ± 1.39	94.65 ± 0.19
BaLoRA	64.31 ± 0.59	80.47 ± 0.61	90.25 ± 0.05	94.11 ± 0.09	91.17 ± 0.04	66.06 ± 1.04	95.23 ± 0.09
AltLoRA	65.96 ± 0.23	86.59 ± 0.37	90.27 ± 0.07	94.14 ± 0.07	91.14 ± 0.05	64.86 ± 0.61	95.29 ± 0.09
BDA-LoRA*	66.18 ± 0.08	86.71 ± 1.53	90.28 ± 0.12	94.25 ± 0.09	91.33 ± 0.02	76.78 ± 1.36	95.52 ± 0.23
BDA-LoRA	**68.01 ± 0.88**	**87.82 ± 0.94**	**90.68 ± 0.06**	**94.67 ± 0.08**	**92.37 ± 0.14**	**78.09 ± 0.94**	**96.31 ± 0.20**

Natural Language Generation. We fine-tune Llama-3-8B and Gemma-7B with all compared methods on the NLG benchmark datasets and report the mean and standard deviation over five runs, together with the average score across GSM8K and HumanEval. The results are summarized in Table 2. As shown in Table 2, BDA-LoRA outperforms all compared methods across both tasks and model families. Even without balance preservation, BDA-LoRA* matches or exceeds LoRA-One, supporting the effectiveness of our gradient-approximate initialization. BDA-LoRA further improves on BDA-LoRA* across all settings, with particularly strong results on HumanEval, supporting the contribution of balance-preserving updates. The lower training losses in Figure 1 are consistent with the descent advantage predicted by our analysis.

Table 2: Test score (%) of Llama-3-8B and Gemma-7B with all compared methods on NLG benchmark datasets. The best PEFT result is in **bold** and the second best is underlined.

	Llama-3-8B			Gemma-7B		
Method	GSM8K	HumanEval	Avg	GSM8K	HumanEval	Avg
LoRA	73.84 ± 0.62	45.73 ± 1.22	59.79	73.58 ± 1.02	38.42 ± 1.22	56.00
LoRA-One	75.87 ± 0.27	46.14 ± 2.46	61.01	74.46 ± 0.61	44.51 ± 2.63	59.49
LoRA-GA	74.35 ± 1.23	44.92 ± 1.53	59.64	74.20 ± 0.35	40.65 ± 1.53	57.43
PiSSA	72.86 ± 0.32	45.12 ± 1.32	58.99	74.35 ± 0.37	45.33 ± 0.35	59.84
DoRA	74.17 ± 0.18	46.34 ± 1.52	60.26	74.60 ± 0.66	40.04 ± 2.88	57.32
RefLoRA	72.23 ± 0.36	45.53 ± 1.60	58.88	74.15 ± 0.55	40.65 ± 3.13	57.40
BaLoRA	74.88 ± 0.24	46.34 ± 1.83	60.61	74.35 ± 0.66	38.62 ± 2.14	56.49
AltLoRA	74.30 ± 0.59	45.73 ± 0.32	60.02	74.12 ± 0.71	41.67 ± 1.27	57.90
BDA-LoRA*	76.07 ± 0.29	46.75 ± 1.15	61.41	74.78 ± 0.52	45.73 ± 1.61	60.26
BDA-LoRA	**76.90 ± 0.38**	**49.16 ± 0.76**	**63.03**	**75.90 ± 0.48**	**46.54 ± 0.93**	**61.22**

Commonsense Reasoning. We fine-tune Llama-3-8B with all compared methods on eight commonsense reasoning benchmark datasets and report the mean and standard deviation over five runs. The results are summarized in Table 3. As shown in Table 3, BDA-LoRA outperforms all compared methods on every task. Its consistent gains over BDA-LoRA* with the same initialization support the benefit of preserving balance during training. BDA-LoRA* performs comparably to LoRA-One on average, supporting the effectiveness of our gradient-approximate initialization.

Overall Comparison. Across the reported settings, BDA-LoRA* achieves higher average scores than LoRA-One (Zhang et al., 2025b) without computing the full parameter gradient matrix, supporting the practical effectiveness of our initialization. BDA-LoRA further improves on BDA-LoRA* on every evaluated task; since the two variants share the same initialization, this comparison directly supports the contribution of balance-preserving updates. BDA-LoRA also outperforms RefLoRA (Zhang et al., 2025a) and BaLoRA (Castin et al., 2026), consistent with the optimizer-related limitations discussed above.

Table 3: Test accuracy (%) of Llama-3-8B with all compared methods on commonsense reasoning benchmarks. The best PEFT result is in **bold** and the second best is underlined.

Method	BoolQ	PIQA	SIQA	Hella.	WinoG.	ARC-c	ARC-e	OBQA
LoRA	75.42 ± 0.01	89.39 ± 0.19	82.13 ± 0.16	95.35 ± 0.06	88.50 ± 0.24	74.89 ± 0.45	91.06 ± 0.21	85.73 ± 0.25
LoRA-One	75.64 ± 0.33	89.79 ± 0.27	82.22 ± 0.09	95.54 ± 0.06	88.45 ± 0.58	78.26 ± 0.45	91.64 ± 0.34	88.27 ± 0.82
LoRA-GA	75.58 ± 0.28	89.28 ± 0.31	82.36 ± 0.48	95.13 ± 0.10	88.50 ± 0.04	71.96 ± 2.43	90.66 ± 0.52	87.00 ± 0.16
PiSSA	75.66 ± 0.45	89.70 ± 0.07	82.18 ± 0.48	95.37 ± 0.14	88.32 ± 0.19	77.53 ± 0.20	91.55 ± 0.14	86.80 ± 0.75
DoRA	75.29 ± 0.07	89.54 ± 0.40	81.97 ± 0.58	95.36 ± 0.08	88.40 ± 0.45	75.09 ± 0.79	91.18 ± 0.31	85.60 ± 0.59
RefLoRA	75.61 ± 0.34	89.63 ± 0.40	82.01 ± 0.51	95.19 ± 0.09	88.56 ± 0.13	76.99 ± 0.45	91.51 ± 0.11	87.40 ± 0.91
BaLoRA	74.31 ± 0.19	89.72 ± 0.12	81.53 ± 0.07	94.83 ± 0.16	87.82 ± 0.33	70.76 ± 0.45	90.26 ± 0.15	84.87 ± 0.50
AltLoRA	73.73 ± 0.33	88.96 ± 0.35	81.93 ± 0.36	94.14 ± 0.09	87.19 ± 0.23	76.02 ± 0.85	87.05 ± 0.11	82.13 ± 0.52
BDA-LoRA*	75.67 ± 0.47	89.77 ± 0.13	82.45 ± 0.05	95.62 ± 0.06	88.63 ± 0.37	78.27 ± 0.70	91.72 ± 0.26	88.20 ± 0.50
BDA-LoRA	$\mathbf{76.71 \pm 0.76}$	$\mathbf{90.35 \pm 0.14}$	$\mathbf{83.28 \pm 0.11}$	$\mathbf{96.14 \pm 0.08}$	$\mathbf{89.58 \pm 0.28}$	$\mathbf{79.44 \pm 0.18}$	$\mathbf{92.80 \pm 0.09}$	$\mathbf{89.33 \pm 0.62}$

Training-Loss Analysis Figure 1 compares the EMA-smoothed training losses of Llama-3-8B on CodeFeedback-100K and MetaMathQA-100K. BDA-LoRA exhibits an overall decreasing loss trend and remains among the lowest-loss methods throughout the later stages of both runs. It achieves the lowest final smoothed losses: 0.448 on CodeFeedback-100K and 0.141 on MetaMathQA-100K. These curves are consistent with Theorem 1. Even without balance preservation, BDA-LoRA* shows loss trajectories comparable to or better than LoRA-One, supporting the effectiveness of our gradient-approximate initialization without computing and storing the full parameter gradient matrix. RefLoRA exhibits slower loss reduction on both tasks, consistent with our earlier discussion that, under adaptive optimizers, gradient preconditioning may produce updates different from explicit balance preservation due to the lack of rotation equivariance.

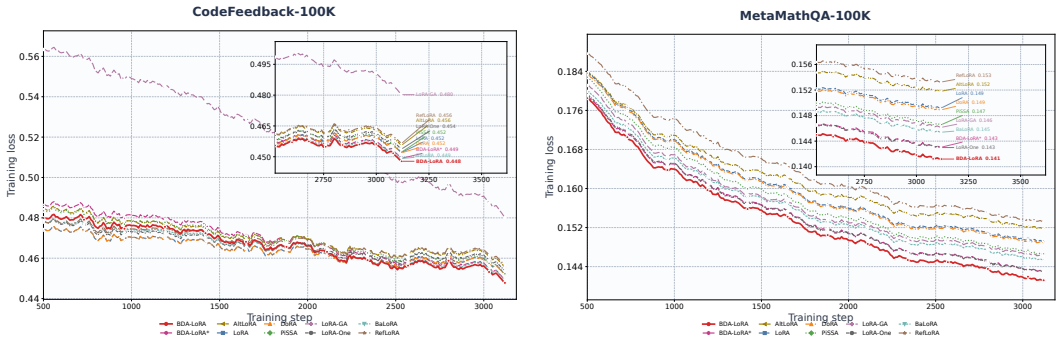


Figure 1: EMA-smoothed training loss on CodeFeedback-100K and MetaMathQA-100K.

5 Conclusion

We studied the conditions on $\mathbf{B}^t$ and $\mathbf{A}^t$ that yield the most loss descent: the largest and smallest singular values of both LoRA matrices equal the square roots of the corresponding singular values of their product, and their associated internal singular subspaces align. We reveal that the balance $(\mathbf{B}^t)^\top \mathbf{B}^t = \mathbf{A}^t (\mathbf{A}^t)^\top$ is a simple and easy-to-implement case of the conditions on $\mathbf{B}^t$ and $\mathbf{A}^t$. Based on this insight, we proposed BDA-LoRA, which uses LoRA-based randomized SVD to obtain a gradient-approximate balanced initialization without computing or storing the full parameter gradient matrix, and uses a post-optimizer balance-preserving matrix to maintain exact balance while approximately minimizing changes to the raw optimizer update directions. Experiments show that BDA-LoRA achieves superior overall performance over all compared PEFT methods, including existing methods that explicitly incorporate balance into LoRA optimization.

AI USE STATEMENT

We used generative AI primarily to assist with writing, translation, literature search, and checking proofs. We did not use AI to propose the core research ideas, formulate mathematical claims, or design experiments. We reviewed all AI-assisted content and take responsibility for the final work.

REFERENCES

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*, pp. 7432–7439, 2020.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Valérie Castin, Kimia Nadjahi, Pierre Ablin, and Gabriel Peyré. Balanced LoRA: Removing parameter invariance to accelerate convergence. *CoRR*, abs/2605.31484, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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) (NAACL'19)*, pp. 2924–2936, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

Shu Ding, Yang Peng, Hangan Zhou, Xinyu Lu, Shangwei Chen, Junhua Huang, Mingxuan Yuan, and Wei Wang. Towards understanding the dynamics of low-rank adaptation. In *Proceedings of the 43rd International Conference on Machine Learning (ICML'26)*, 2026.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the 3rd International Workshop on Paraphrasing, (IWP@IJCNLP'05)*, 2005.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53 (2):217–288, 2011.

Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17554–17571. PMLR / OpenReview.net, 2024.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. The impact of initialization on LoRA finetuning dynamics. In *Advances in Neural Information Processing Systems 37 (NeurIPS'24)*, 2024a.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, pp. 17783–17806, 2024b.

Haonan He, Peng Ye, Yuchen Ren, Yuan Yuan, Luyang Zhou, Shucun Ju, and Lei Chen. GoRA: Gradient-driven adaptive low rank adaptation. In *Advances in Neural Information Processing Systems 38 (NeurIPS'25)*, 2025.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The 11th International Conference on Learning Representations (ICLR'23)*, 2023.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *The 10th International Conference on Learning Representations (ICLR'22)*, 2022.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. LLM-Adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP'23)*, pp. 5254–5276, 2023.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with LoRA. *CoRR*, abs/2312.03732, 2023.

Mingjie Li, Wai Man Si, Michael Backes, Yang Zhang, and Yisen Wang. SaLoRA: Safety-alignment preserved low-rank adaptation. In *The 13th International Conference on Learning Representations (ICLR'25)*, 2025a.

Shiwei Li, Xiandi Luo, Xing Tang, Haozhao Wang, Hao Chen, Weihong Luo, Yuhua Li, Xiuqiang He, and Ruixuan Li. Beyond zero initialization: Investigating the impact of non-zero initialization on LoRA fine-tuning dynamics. In *Proceedings of the 42nd International Conference on Machine Learning (ICML'25)*, pp. 35519–35535, 2025b.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. ReLoRA: High-rank training through low-rank updates. In *The 12th International Conference on Learning Representations (ICLR'24)*, 2024.

Weicheng Lin, Yi Zhang, Jiawei Dang, and Liang-Jie Zhang. TLoRA: Task-aware low rank adaptation of large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL'26)*, pp. 29252–29269, 2026. doi: 10.18653/v1/2026.acl-long.1348.

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. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, pp. 32100–32121, 2024.

Llama Team. The Llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

Grigory Malinovsky, Umberto Michieli, Hasan Abed Al Kader Hammoud, Taha Ceritli, Hayder Elesedy, Mete Ozay, and Peter Richtárik. Randomized asymmetric chain of lora: The first meaningful theoretical framework for low-rank adaptation. *CoRR*, abs/2410.08305, 2024.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS'24)*, pp. 121038–121072, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP'18)*, pp. 2381–2391, 2018.

Théodore Papadopoulo and Manolis I. A. Lourakis. Estimating the Jacobian of the singular value decomposition: Theory and applications. In *Proceedings of the 6th European Conference on Computer Vision, Part I (ECCV'00)*, pp. 554–570, 2000.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*, pp. 2383–2392, 2016.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*, pp. 8732–8740, 2020.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*, pp. 4463–4473, 2019.

Valeria Simoncini. Computational methods for linear matrix equations. *SIAM Rev.*, 58(3):377–441, 2016.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, pp. 1631–1642, 2013.

Pengwei Tang, Xiaolin Hu, Yong Liu, Lizhong Ding, Dongjie Zhang, Xing Wu, and Debing Zhang. Put the space of LoRA initialization to the extreme to preserve pre-trained knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'26)*, volume 40, pp. 33232–33240, 2026. doi: 10.1609/aaai.v40i39.40608.

Gemma Team. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295, 2024.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 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 *The 7th International Conference on Learning Representations (ICLR'19)*, 2019.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. MiLoRA: Harnessing minor singular components for parameter-efficient LLM finetuning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers) (NAACL'25)*, pp. 4823–4836, 2025a. doi: 10.18653/v1/2025.naacl-long.248.

Shaowen Wang, Linxi Yu, and Jian Li. LoRA-GA: Low-rank adaptation with gradient approximation. In *Advances in Neural Information Processing Systems (NeurIPS'24)*, pp. 54905–54931, 2024.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. LoRA-Pro: Are low-rank adapters properly optimized? In *The 13th International Conference on Learning Representations (ICLR'25)*, 2025b.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'18)*, pp. 1112–1122, 2018.

Shuo Xie, Mohamad Amin Mohamadi, and Zhiyuan Li. Adam exploits ℓ_∞-geometry of loss landscape via coordinate-wise adaptivity. In *The 13th International Conference on Learning Representations (ICLR'25)*, 2025.

Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Leon Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. CorDA: Context-oriented decomposition adaptation of large language models for task-aware parameter-efficient fine-tuning. In *Advances in Neural Information Processing Systems 37 (NeurIPS'24)*, 2024.

Can Yaras, Peng Wang, Laura Balzano, and Qing Qu. Compressible dynamics in deep overparameterized low-rank learning & adaptation. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, pp. 56946–56965, 2024.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *The 12th International Conference on Learning Representations (ICLR'24)*, 2024.

Xin Yu, Yujia Wang, Jinghui Chen, and Lingzhou Xue. AltLoRA: Towards better gradient approximation in low-rank adaptation with alternating projections. In *Advances in Neural Information Processing Systems (NeurIPS'25)*, pp. 37484–37522, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL'19)*, pp. 4791–4800, 2019.

Fangzhao Zhang and Mert Pilanci. Riemannian preconditioned LoRA for fine-tuning foundation models. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, pp. 59641–59669, 2024.

Yilang Zhang, Bingcong Li, and Georgios B. Giannakis. RefLoRA: Refactored low-rank adaptation for efficient fine-tuning of large models. In *Advances in Neural Information Processing Systems (NeurIPS'25)*, pp. 92847–92878, 2025a.

Yilang Zhang, Xiaodong Yang, Yiwei Cai, and Georgios B. Giannakis. ScaloRA: Optimally scaled low-rank adaptation for efficient high-rank fine-tuning. In *Forty-third International Conference on Machine Learning*, 2026.

Yuanhe Zhang, Fanghui Liu, and Yudong Chen. LoRA-One: One-step full gradient could suffice for fine-tuning large language models, provably and efficiently. In *Proceedings of the 42nd International Conference on Machine Learning (ICML'25)*, pp. 75513–75574, 2025b.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, pp. 61121–61143, 2024.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. OpenCodeInterpreter: Integrating code generation with execution and refinement. In *Findings of the Association for Computational Linguistics: ACL 2024 (ACL'24)*, pp. 12834–12859, 2024.

Jiacheng Zhu, Kristjan H. Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in low-rank adapters of foundation models. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, pp. 62369–62385, 2024.

A Theoretical Derivations

The symbols $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ denote the Frobenius inner product and norm, respectively; $\|\cdot\|_2$ denotes the spectral norm.

A.1 Proof of the Optimal Descent Conditions in Theorem 1

Fix iteration t and the common starting point $\mathbf{W}^t$. Define

$$\mathbf{M} := \mathbf{W}^t - \mathbf{W}^0 = \mathbf{B}^t\mathbf{A}^t, \qquad \mathbf{W}^t = \mathbf{W}^0 + \mathbf{M}, \qquad \operatorname{rank}(\mathbf{M}) = r.$$

The comparison set consists of all full-rank pairs $(\mathbf{B}^t, \mathbf{A}^t)$ satisfying $\mathbf{B}^t\mathbf{A}^t = \mathbf{M}$. Hence every pair represents the same $\mathbf{W}^t$ and shares the gradient $\mathbf{G}^t := \nabla\mathcal{L}(\mathbf{W}^t)$. The compact SVD of the fixed product is $\mathbf{M} = \mathbf{U}^t\mathbf{\Sigma}^t(\mathbf{V}^t)^\top$, where $\mathbf{U}^t \in \mathbb{R}^{m\times r}$ and $\mathbf{V}^t \in \mathbb{R}^{n\times r}$ have orthonormal columns and $\mathbf{\Sigma}^t \in \mathbb{R}^{r\times r}$ is positive diagonal. Let $\eta > 0$ be the learning rate, and write $\mathbf{P}_{\mathbf{U}^t} = \mathbf{U}^t(\mathbf{U}^t)^\top$ and $\mathbf{P}_{\mathbf{V}^t} = \mathbf{V}^t(\mathbf{V}^t)^\top$ and define

$$\mathcal{T}_{\mathbf{M}} := \{\mathbf{X} : (\mathbf{I} - \mathbf{P}_{\mathbf{U}^t})\mathbf{X}(\mathbf{I} - \mathbf{P}_{\mathbf{V}^t}) = 0\} = \mathcal{T}_{\mathbf{B}^t\mathbf{A}^t}.$$

Every gradient has the orthogonal decomposition

$$\mathbf{G}^t = \overline{\mathbf{G}}^t + \mathbf{G}^t_\perp,$$

where

$$\overline{\mathbf{G}}^t = \mathbf{P}_{\mathbf{U}^t}\mathbf{G}^t + \mathbf{G}^t\mathbf{P}_{\mathbf{V}^t} - \mathbf{P}_{\mathbf{U}^t}\mathbf{G}^t\mathbf{P}_{\mathbf{V}^t} \in \mathcal{T}_{\mathbf{M}}, \qquad \mathbf{G}^t_\perp = (\mathbf{I} - \mathbf{P}_{\mathbf{U}^t})\mathbf{G}^t(\mathbf{I} - \mathbf{P}_{\mathbf{V}^t}) \in \mathcal{T}_{\mathbf{M}}^\perp.$$

Because $\operatorname{col}(\mathbf{B}^t) = \operatorname{col}(\mathbf{U}^t)$ and $\operatorname{col}((\mathbf{A}^t)^\top) = \operatorname{col}(\mathbf{V}^t)$, both factor gradients depend only on $\overline{\mathbf{G}}^t$.

One simultaneous Euclidean factor step satisfies

$$\mathbf{B}^{t+1} = \mathbf{B}^t - \eta\mathbf{G}^t(\mathbf{A}^t)^\top, \qquad \mathbf{A}^{t+1} = \mathbf{A}^t - \eta(\mathbf{B}^t)^\top\mathbf{G}^t,$$

and therefore

$$\Delta\mathbf{M}^t := \mathbf{B}^{t+1}\mathbf{A}^{t+1} - \mathbf{B}^t\mathbf{A}^t = -\eta\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t) + \eta^2\mathcal{R}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t), \tag{10}$$

where

$$\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{X}) := \mathbf{B}^t(\mathbf{B}^t)^\top\mathbf{X} + \mathbf{X}(\mathbf{A}^t)^\top\mathbf{A}^t, \qquad \mathcal{R}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{X}) := \mathbf{X}(\mathbf{A}^t)^\top(\mathbf{B}^t)^\top\mathbf{X}.$$

Thus Eq. (10) is the operator form of the product update substituted into Eq. (3). Let x_1 and x_r be the largest and smallest eigenvalues of $(\mathbf{B}^t)^\top\mathbf{B}^t$, and let y_1 and y_r be those of $\mathbf{A}^t(\mathbf{A}^t)^\top$. Define

$$d_-(\mathbf{B}^t, \mathbf{A}^t) := \min\{x_r, y_r\}, \qquad d_+(\mathbf{B}^t, \mathbf{A}^t) := x_1 + y_1, \qquad p := \|\mathbf{B}^t\|_2\|\mathbf{A}^t\|_2 = \sqrt{x_1 y_1}. \tag{11}$$

For a fixed candidate pair, abbreviate $d_-(\mathbf{B}^t, \mathbf{A}^t)$ and $d_+(\mathbf{B}^t, \mathbf{A}^t)$ as d_- and d_+, respectively.

For any product perturbation $\mathbf{\Delta} \in \mathbb{R}^{m\times n}$, the descent lemma gives

$$\mathcal{L}(\mathbf{W}^t + \mathbf{\Delta}) \leqslant \mathcal{L}(\mathbf{W}^t) + \mathcal{U}_L(\mathbf{G}^t, \mathbf{\Delta}), \qquad \mathcal{U}_L(\mathbf{G}^t, \mathbf{\Delta}) := \langle\mathbf{G}^t, \mathbf{\Delta}\rangle + \frac{L}{2}\|\mathbf{\Delta}\|^2. \tag{12}$$

For later use, set

$$q_\eta(\lambda) := -\eta\lambda + \frac{L\eta^2}{2}\lambda^2, \qquad H_\eta(\mathbf{B}^t, \mathbf{A}^t) := \max\{q_\eta(d_-), q_\eta(d_+)\}. \tag{13}$$

The $\mathcal{O}(\eta^2)$ remainder from the exact product update. Expanding Eq. (10) inside the smooth upper bound gives

$$\begin{aligned}&\mathcal{U}_L(\mathbf{G}^t, \Delta\mathbf{M}^t) - \mathcal{U}_L\big(\mathbf{G}^t, -\eta\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t)\big)\\ &= \eta^2\langle\mathbf{G}^t, \mathcal{R}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t)\rangle - L\eta^3\langle\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t), \mathcal{R}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t)\rangle + \frac{L\eta^4}{2}\|\mathcal{R}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t)\|^2.\end{aligned} \tag{14}$$

Here, ε_η^t is the finite-step remainder contributed by the second-order term $\eta^2\mathcal{R}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t)$ in the exact product update.

$$\|\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t)\| \le d_+\|\overline{\mathbf{G}}^t\|, \qquad \|\mathcal{R}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t)\| \le p\|\overline{\mathbf{G}}^t\|^2.$$

Assume $\|\overline{\mathbf{G}}^t\| \le C_e$, $\|\mathbf{G}_\perp^t\| \le C_\perp$, and $d_+(\mathbf{B}^t,\mathbf{A}^t) \le d$. Set $C_G := \sqrt{C_e^2 + C_\perp^2}$. Equation (14) and the Cauchy–Schwarz inequality imply

$$\left|\mathcal{U}_L(\mathbf{G}^t,\Delta\mathbf{M}^t) - \mathcal{U}_L\big(\mathbf{G}^t,-\eta\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t)\big)\right| \le \eta^2\|\overline{\mathbf{G}}^t\|^2\left(C_G p + L\eta d_+ p\|\overline{\mathbf{G}}^t\| + \frac{L\eta^2}{2}p^2\|\overline{\mathbf{G}}^t\|^2\right). \tag{15}$$

Since $p = \sqrt{x_1 y_1} \le \frac{(x_1+y_1)}{2} = \frac{d_+(\mathbf{B}^t,\mathbf{A}^t)}{2} \le \frac{d}{2}$, the right-hand side is bounded by

$$E_\eta := \eta^2 C_e^2\left(C_G\frac{d}{2} + L\eta d\frac{d}{2}C_e + \frac{L\eta^2}{2}\left(\frac{d}{2}\right)^2 C_e^2\right) = \mathcal{O}(\eta^2).$$

Therefore, under these boundedness assumptions,

$$\mathcal{U}_L(\mathbf{G}^t,\Delta\mathbf{M}^t) = \mathcal{U}_L\big(\mathbf{G}^t,-\eta\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t)\big) + \varepsilon_\eta^t, \qquad |\varepsilon_\eta^t| \le E_\eta = \mathcal{O}(\eta^2). \tag{16}$$

Substituting Eq. (16) into Eq. (12) yields

$$\begin{aligned}\mathcal{L}(\mathbf{W}^0 + \mathbf{B}^{t+1}\mathbf{A}^{t+1}) \leqslant \mathcal{L}(\mathbf{W}^0 + \mathbf{B}^t\mathbf{A}^t) \\ - \eta\Big\langle \mathbf{G}^t, \mathbf{B}^t(\mathbf{B}^t)^\top\mathbf{G}^t + \mathbf{G}^t(\mathbf{A}^t)^\top\mathbf{A}^t\Big\rangle \\ + \frac{L\eta^2}{2}\Big\|\mathbf{B}^t(\mathbf{B}^t)^\top\mathbf{G}^t + \mathbf{G}^t(\mathbf{A}^t)^\top\mathbf{A}^t\Big\|^2 + \varepsilon_\eta^t.\end{aligned} \tag{17}$$

If the same bounds C_e, $C_\perp$, and d hold for all candidates, then $|\varepsilon_\eta^t| = \mathcal{O}(\eta^2)$ uniformly over these candidates. Hence, for small η, their leading-order comparison is determined by $H_\eta(\mathbf{B}^t,\mathbf{A}^t)$.

Let $\{\mathbf{E}_i\}$ be an orthonormal eigenbasis of $\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}|_{\mathcal{T}_\mathbf{M}}$, with

$$\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{E}_i) = \lambda_i\mathbf{E}_i, \qquad \langle\mathbf{E}_i,\mathbf{E}_j\rangle = \delta_{ij}, \qquad \lambda_i \in [d_-, d_+].$$

Here δ_{ij} is the Kronecker delta: $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise. Write

$$\overline{\mathbf{G}}^t = \sum_i c_i\mathbf{E}_i.$$

By linearity,

$$\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\overline{\mathbf{G}}^t) = \sum_i c_i\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{E}_i) = \sum_i c_i\lambda_i\mathbf{E}_i. \tag{18}$$

Using $\langle\mathbf{E}_i,\mathbf{E}_j\rangle = \delta_{ij}$,

$$\langle\overline{\mathbf{G}}^t, \mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\overline{\mathbf{G}}^t)\rangle = \left\langle\sum_i c_i\mathbf{E}_i, \sum_j c_j\lambda_j\mathbf{E}_j\right\rangle = \sum_{i,j} c_ic_j\lambda_j\langle\mathbf{E}_i,\mathbf{E}_j\rangle = \sum_i c_i^2\lambda_i, \tag{19}$$

and

$$\|\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\overline{\mathbf{G}}^t)\|^2 = \left\langle\sum_i c_i\lambda_i\mathbf{E}_i, \sum_j c_j\lambda_j\mathbf{E}_j\right\rangle = \sum_{i,j} c_ic_j\lambda_i\lambda_j\langle\mathbf{E}_i,\mathbf{E}_j\rangle = \sum_i c_i^2\lambda_i^2. \tag{20}$$

Since $\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t) = \mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\overline{\mathbf{G}}^t)$ and $\mathbf{G}_\perp^t \perp \mathcal{T}_\mathbf{M}$,

$$\begin{aligned}\mathcal{U}_L\big(\mathbf{G}^t,-\eta\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t)\big) &= -\eta\langle\overline{\mathbf{G}}^t,\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\overline{\mathbf{G}}^t)\rangle + \frac{L\eta^2}{2}\|\mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\overline{\mathbf{G}}^t)\|^2 \\ &= -\eta\sum_i c_i^2\lambda_i + \frac{L\eta^2}{2}\sum_i c_i^2\lambda_i^2 \\ &= \sum_i c_i^2\left(-\eta\lambda_i + \frac{L\eta^2}{2}\lambda_i^2\right) \\ &= \sum_i c_i^2 q_\eta(\lambda_i).\end{aligned} \tag{21}$$

The convexity of q_η implies $q_\eta(\lambda_i) \leqslant \max\{q_\eta(d_-), q_\eta(d_+)\} = H_\eta(\mathbf{B}^t, \mathbf{A}^t)$. Moreover, $\sum_i c_i^2 = \|\overline{\mathbf{G}}^t\|^2$. Therefore,

$$\mathcal{U}_L\big(\mathbf{G}^t, -\eta \mathcal{K}_{\mathbf{B}^t,\mathbf{A}^t}(\mathbf{G}^t)\big) \leqslant \|\overline{\mathbf{G}}^t\|^2 H_\eta(\mathbf{B}^t, \mathbf{A}^t). \tag{22}$$

Consequently, Eqs. (17) and (22) give

$$\mathcal{L}(\mathbf{W}^0 + \mathbf{B}^{t+1}\mathbf{A}^{t+1}) \leqslant \mathcal{L}(\mathbf{W}^0 + \mathbf{B}^t\mathbf{A}^t) + \|\overline{\mathbf{G}}^t\|^2 H_\eta(\mathbf{B}^t, \mathbf{A}^t) + \varepsilon_\eta^t, \tag{23}$$

The product singular-value inequalities give

$$\sigma_r(\mathbf{M}) = \sigma_r(\mathbf{B}^t\mathbf{A}^t) \geq \sigma_r(\mathbf{B}^t)\sigma_r(\mathbf{A}^t) = \sqrt{x_r y_r} \geq \min\{x_r, y_r\} = d_-,$$

while

$$\sigma_1(\mathbf{M}) \leq \sigma_1(\mathbf{B}^t)\sigma_1(\mathbf{A}^t) = \sqrt{x_1 y_1}.$$

Consequently,

$$d_-(\mathbf{B}^t, \mathbf{A}^t) \leq \sigma_r(\mathbf{M}), \qquad d_+(\mathbf{B}^t, \mathbf{A}^t) = x_1 + y_1 \geq 2\sqrt{x_1 y_1} \geq 2\sigma_1(\mathbf{M}).$$

For a factorization $(\mathbf{B}^t, \mathbf{A}^t)$, define the internal extreme singular subspaces

$$\mathcal{V}^+_{\mathbf{B}^t} := \{\mathbf{z} : (\mathbf{B}^t)^\top \mathbf{B}^t \mathbf{z} = \sigma_1^2(\mathbf{B}^t)\mathbf{z}\}, \qquad \mathcal{U}^+_{\mathbf{A}^t} := \{\mathbf{z} : \mathbf{A}^t(\mathbf{A}^t)^\top \mathbf{z} = \sigma_1^2(\mathbf{A}^t)\mathbf{z}\},$$
$$\mathcal{V}^-_{\mathbf{B}^t} := \{\mathbf{z} : (\mathbf{B}^t)^\top \mathbf{B}^t \mathbf{z} = \sigma_r^2(\mathbf{B}^t)\mathbf{z}\}, \qquad \mathcal{U}^-_{\mathbf{A}^t} := \{\mathbf{z} : \mathbf{A}^t(\mathbf{A}^t)^\top \mathbf{z} = \sigma_r^2(\mathbf{A}^t)\mathbf{z}\},$$

where each $\mathbf{z}$ lies in $\mathbb{R}^r$ and σ_r denotes the smallest nonzero singular value. Let $\mathcal{V}^+_{\mathbf{M}}$ and $\mathcal{V}^-_{\mathbf{M}}$ be the right singular subspaces of $\mathbf{M}$ associated with $\sigma_1(\mathbf{M})$ and $\sigma_r(\mathbf{M})$, and set $k_\pm := \dim(\mathcal{V}^\pm_{\mathbf{M}})$. Let $(\mathbf{B}^t_*, \mathbf{A}^t_*)$ satisfy

$$\sigma_1(\mathbf{B}^t_*) = \sigma_1(\mathbf{A}^t_*) = \sqrt{\sigma_1(\mathbf{M})}, \qquad \sigma_r(\mathbf{B}^t_*) = \sigma_r(\mathbf{A}^t_*) = \sqrt{\sigma_r(\mathbf{M})},$$
$$\dim(\mathcal{V}^+_{\mathbf{B}^t_*} \cap \mathcal{U}^+_{\mathbf{A}^t_*}) \geq k_+, \qquad \dim(\mathcal{V}^-_{\mathbf{B}^t_*} \cap \mathcal{U}^-_{\mathbf{A}^t_*}) \geq k_-.$$

The first two equalities give

$$d_-(\mathbf{B}^t_*, \mathbf{A}^t_*) = \sigma_r(\mathbf{M}), \qquad d_+(\mathbf{B}^t_*, \mathbf{A}^t_*) = 2\sigma_1(\mathbf{M}).$$

Set

$$\alpha := \sigma_r(\mathbf{M}), \qquad \beta := 2\sigma_1(\mathbf{M}), \qquad \eta_c := \frac{2}{L(\alpha+\beta)}.$$

Since $0 < \sigma_r(\mathbf{M}) \leqslant \sigma_1(\mathbf{M})$, we have $0 < \alpha < \beta$. Since $d_-(\mathbf{B}^t_*, \mathbf{A}^t_*) = \alpha$ and $d_+(\mathbf{B}^t_*, \mathbf{A}^t_*) = \beta$,

$$H_\eta(\mathbf{B}^t_*, \mathbf{A}^t_*) = \max\{q_\eta(\alpha), q_\eta(\beta)\}.$$

If $0 < \eta \leq \eta_c$, then

$$q_\eta(\alpha) - q_\eta(\beta) = \eta(\beta - \alpha)\left[1 - \frac{L\eta}{2}(\alpha+\beta)\right] \geq 0.$$

Moreover,

$$q'_\eta(\lambda) = \eta(L\eta\lambda - 1).$$

For every $\lambda \in (0, \alpha]$,

$$L\eta\lambda \leqslant L\eta_c\alpha = \frac{2\alpha}{\alpha+\beta} < 1, \qquad \text{so} \qquad q'_\eta(\lambda) < 0.$$

Thus q_η is strictly decreasing on $(0, \alpha]$. Every competing candidate pair $(\mathbf{B}^t, \mathbf{A}^t)$ satisfies $d_-(\mathbf{B}^t, \mathbf{A}^t) \leq \alpha$, so

$$H_\eta(\mathbf{B}^t, \mathbf{A}^t) \geq q_\eta\big(d_-(\mathbf{B}^t, \mathbf{A}^t)\big) \geq q_\eta(\alpha) = H_\eta(\mathbf{B}^t_*, \mathbf{A}^t_*).$$

If $\eta \geq \eta_c$, then $q_\eta(\beta) \geq q_\eta(\alpha)$ and for every $\lambda \in [\beta, \infty)$,

$$L\eta\lambda \geq L\eta_c\beta = \frac{2\beta}{\alpha+\beta} > 1, \qquad \text{so} \qquad q'_\eta(\lambda) > 0.$$

Thus q_η is strictly increasing on $[\beta, \infty)$. Since $d_+(\mathbf{B}^t, \mathbf{A}^t) \geq \beta$,

$$H_\eta(\mathbf{B}^t, \mathbf{A}^t) \geq q_\eta\big(d_+(\mathbf{B}^t, \mathbf{A}^t)\big) \geq q_\eta(\beta) = H_\eta(\mathbf{B}^t_*, \mathbf{A}^t_*).$$

The two cases cover every $\eta > 0$ and prove

$$H_\eta(\mathbf{B}^t_*, \mathbf{A}^t_*) = \min_{\mathbf{B}^t, \mathbf{A}^t : \mathbf{B}^t\mathbf{A}^t = \mathbf{M}} H_\eta(\mathbf{B}^t, \mathbf{A}^t). \tag{24}$$

Combining Eqs. (24) and (22) shows that $(\mathbf{B}^t_*, \mathbf{A}^t_*)$ minimizes the H_η-dependent term in the loss upper bound, and consequently yields the most loss descent characterized by this bound.

A.2 Exact Balance-Preserving Adjustment

At iteration t, let $(\mathbf{B}^t)^\top \mathbf{B}^t = \mathbf{A}^t(\mathbf{A}^t)^\top$, and let $\mathbf{U}_B^t$ and $\mathbf{U}_A^t$ be the raw optimizer update directions. The exact finite-step minimum-change problem in Eq. (7) is

$$\begin{aligned} \min_{\mathbf{Z}_B^t, \mathbf{Z}_A^t} \quad & \frac{1}{2}\|\mathbf{Z}_B^t - \mathbf{U}_B^t\|^2 + \frac{1}{2}\|\mathbf{Z}_A^t - \mathbf{U}_A^t\|^2, \\ \text{s.t.} \quad & (\mathbf{B}^t - \eta\mathbf{Z}_B^t)^\top(\mathbf{B}^t - \eta\mathbf{Z}_B^t) = (\mathbf{A}^t - \eta\mathbf{Z}_A^t)(\mathbf{A}^t - \eta\mathbf{Z}_A^t)^\top. \end{aligned} \tag{25}$$

Using the pre-update balance condition, the constraint becomes

$$0 = (\mathbf{B}^t)^\top \mathbf{Z}_B^t + (\mathbf{Z}_B^t)^\top \mathbf{B}^t - \mathbf{Z}_A^t(\mathbf{A}^t)^\top - \mathbf{A}^t(\mathbf{Z}_A^t)^\top - \eta\big((\mathbf{Z}_B^t)^\top \mathbf{Z}_B^t - \mathbf{Z}_A^t(\mathbf{Z}_A^t)^\top\big). \tag{26}$$

Only the symmetric part of the Lagrange multiplier contributes to the constraint, and we denote the effective multiplier by $\mathbf{P}_\eta^t \in \mathbb{R}^{r\times r}$. The Lagrangian is

$$\begin{aligned} \mathcal{L} = & \frac{1}{2}\|\mathbf{Z}_B^t - \mathbf{U}_B^t\|^2 + \frac{1}{2}\|\mathbf{Z}_A^t - \mathbf{U}_A^t\|^2 \\ & + \frac{1}{2}\Big\langle \mathbf{P}_\eta^t, (\mathbf{B}^t)^\top \mathbf{Z}_B^t + (\mathbf{Z}_B^t)^\top \mathbf{B}^t - \mathbf{Z}_A^t(\mathbf{A}^t)^\top - \mathbf{A}^t(\mathbf{Z}_A^t)^\top - \eta\big((\mathbf{Z}_B^t)^\top \mathbf{Z}_B^t - \mathbf{Z}_A^t(\mathbf{Z}_A^t)^\top\big)\Big\rangle. \end{aligned}$$

Stationarity with respect to $\mathbf{Z}_B^t$ and $\mathbf{Z}_A^t$ gives

$$\mathbf{Z}_B^t - \mathbf{U}_B^t + \mathbf{B}^t\mathbf{P}_\eta^t - \eta\mathbf{Z}_B^t\mathbf{P}_\eta^t = 0, \tag{27}$$

$$\mathbf{Z}_A^t - \mathbf{U}_A^t - \mathbf{P}_\eta^t\mathbf{A}^t + \eta\mathbf{P}_\eta^t\mathbf{Z}_A^t = 0. \tag{28}$$

At any regular solution of the constrained problem, these stationarity equations imply

$$\begin{aligned} \mathbf{Z}_B^t &= (\mathbf{U}_B^t - \mathbf{B}^t\mathbf{P}_\eta^t)(\mathbf{I}_r - \eta\mathbf{P}_\eta^t)^{-1}, \\ \mathbf{Z}_A^t &= (\mathbf{I}_r + \eta\mathbf{P}_\eta^t)^{-1}(\mathbf{U}_A^t + \mathbf{P}_\eta^t\mathbf{A}^t). \end{aligned} \tag{29}$$

The multiplier $\mathbf{P}_\eta^t$ must also make Eq. (26) hold. Solving this nonlinear matrix equation together with the two inverse factors is less convenient in practice. For bounded $\mathbf{P}_\eta^t$, $(\mathbf{I}_r \mp \eta\mathbf{P}_\eta^t)^{-1} = \mathbf{I}_r + \mathcal{O}(\eta)$, motivating the inverse-free update directions in Eq. (8). Dropping the inverse factors approximates these exact stationary directions; it is not asserted to solve Eq. (25) exactly. Instead, the multiplier is chosen through a Lyapunov equation so that the resulting inverse-free update satisfies the finite-step balance constraint exactly.

Fix iteration t and assume that the current factors are full rank and balanced, with $\mathbf{S}^t := (\mathbf{B}^t)^\top\mathbf{B}^t = \mathbf{A}^t(\mathbf{A}^t)^\top \succ 0$. Fix $\eta > 0$ and introduce the balance-preserving matrix $\mathbf{P}_\eta^t \in \mathbb{R}^{r\times r}$. Define

$$\mathbf{Z}_B^t = \mathbf{U}_B^t - \mathbf{B}^t\mathbf{P}_\eta^t, \qquad \mathbf{Z}_A^t = \mathbf{U}_A^t + \mathbf{P}_\eta^t\mathbf{A}^t.$$

These are the inverse-free directions in Eq. (8). The corresponding discrete updates are

$$\mathbf{B}^{t+1} = \mathbf{B}^t - \eta(\mathbf{U}_B^t - \mathbf{B}^t\mathbf{P}_\eta^t), \tag{30}$$

$$\mathbf{A}^{t+1} = \mathbf{A}^t - \eta(\mathbf{U}_A^t + \mathbf{P}_\eta^t\mathbf{A}^t). \tag{31}$$

Direct expansion gives

$$\begin{aligned} (\mathbf{B}^{t+1})^\top\mathbf{B}^{t+1} &= \mathbf{S}^t - \eta\big((\mathbf{U}_B^t)^\top\mathbf{B}^t + (\mathbf{B}^t)^\top\mathbf{U}_B^t - \mathbf{P}_\eta^t\mathbf{S}^t - \mathbf{S}^t\mathbf{P}_\eta^t\big) \\ &\quad + \eta^2\big((\mathbf{U}_B^t)^\top\mathbf{U}_B^t - (\mathbf{U}_B^t)^\top\mathbf{B}^t\mathbf{P}_\eta^t - \mathbf{P}_\eta^t(\mathbf{B}^t)^\top\mathbf{U}_B^t + \mathbf{P}_\eta^t\mathbf{S}^t\mathbf{P}_\eta^t\big), \end{aligned} \tag{32}$$

$$\begin{aligned} \mathbf{A}^{t+1}(\mathbf{A}^{t+1})^\top &= \mathbf{S}^t - \eta\big(\mathbf{U}_A^t(\mathbf{A}^t)^\top + \mathbf{A}^t(\mathbf{U}_A^t)^\top + \mathbf{P}_\eta^t\mathbf{S}^t + \mathbf{S}^t\mathbf{P}_\eta^t\big) \\ &\quad + \eta^2\big(\mathbf{U}_A^t(\mathbf{U}_A^t)^\top + \mathbf{U}_A^t(\mathbf{A}^t)^\top\mathbf{P}_\eta^t + \mathbf{P}_\eta^t\mathbf{A}^t(\mathbf{U}_A^t)^\top + \mathbf{P}_\eta^t\mathbf{S}^t\mathbf{P}_\eta^t\big). \end{aligned} \tag{33}$$

Setting Eqs. (32) and (33) equal, canceling the common $\mathbf{P}_\eta^t\mathbf{S}^t\mathbf{P}_\eta^t$ term, and dividing by η yields

$$\begin{aligned} &\mathbf{P}_\eta^t\big(2\mathbf{S}^t - \eta((\mathbf{B}^t)^\top\mathbf{U}_B^t + \mathbf{A}^t(\mathbf{U}_A^t)^\top)\big) + \big(2\mathbf{S}^t - \eta((\mathbf{U}_B^t)^\top\mathbf{B}^t + \mathbf{U}_A^t(\mathbf{A}^t)^\top)\big)\mathbf{P}_\eta^t \\ &= \big((\mathbf{U}_B^t)^\top\mathbf{B}^t + (\mathbf{B}^t)^\top\mathbf{U}_B^t\big) - \big(\mathbf{U}_A^t(\mathbf{A}^t)^\top + \mathbf{A}^t(\mathbf{U}_A^t)^\top\big) - \eta\big((\mathbf{U}_B^t)^\top\mathbf{U}_B^t - \mathbf{U}_A^t(\mathbf{U}_A^t)^\top\big). \end{aligned} \tag{34}$$

Define

$$\mathbf{K}^t = 2\mathbf{S}^t - \eta\big((\mathbf{U}_B^t)^\top \mathbf{B}^t + \mathbf{U}_A^t(\mathbf{A}^t)^\top\big), \tag{35}$$

$$\mathbf{Q}^t = \big((\mathbf{U}_B^t)^\top \mathbf{B}^t + (\mathbf{B}^t)^\top \mathbf{U}_B^t\big) - \big(\mathbf{U}_A^t(\mathbf{A}^t)^\top + \mathbf{A}^t(\mathbf{U}_A^t)^\top\big) - \eta\big((\mathbf{U}_B^t)^\top \mathbf{U}_B^t - \mathbf{U}_A^t(\mathbf{U}_A^t)^\top\big). \tag{36}$$

Equation (34) is therefore the Lyapunov equation

$$\mathbf{P}_\eta^t(\mathbf{K}^t)^\top + \mathbf{K}^t\mathbf{P}_\eta^t = \mathbf{Q}^t.$$

Since $\mathbf{K}^t$ is a small perturbation of the positive-definite matrix $2\mathbf{S}^t$, all eigenvalues of $\mathbf{K}^t$ have positive real parts for a small learning rate. The Lyapunov equation then has a unique solution $\mathbf{P}_\eta^t$.

Finally, compare the modified update with the raw (uncorrected) factors generated by the same optimizer directions:

$$\widehat{\mathbf{B}}^{t+1} = \mathbf{B}^t - \eta\mathbf{U}_B^t, \qquad \widehat{\mathbf{A}}^{t+1} = \mathbf{A}^t - \eta\mathbf{U}_A^t.$$

The balance correction cancels in the first-order product update:

$$(\mathbf{U}_B^t - \mathbf{B}^t\mathbf{P}_\eta^t)\mathbf{A}^t + \mathbf{B}^t(\mathbf{U}_A^t + \mathbf{P}_\eta^t\mathbf{A}^t) = \mathbf{U}_B^t\mathbf{A}^t - \mathbf{B}^t\mathbf{P}_\eta^t\mathbf{A}^t + \mathbf{B}^t\mathbf{U}_A^t + \mathbf{B}^t\mathbf{P}_\eta^t\mathbf{A}^t = \mathbf{U}_B^t\mathbf{A}^t + \mathbf{B}^t\mathbf{U}_A^t. \tag{37}$$

For the exact discrete updates,

$$\mathbf{B}^{t+1}\mathbf{A}^{t+1} = \mathbf{B}^t\mathbf{A}^t - \eta(\mathbf{U}_B^t\mathbf{A}^t + \mathbf{B}^t\mathbf{U}_A^t) + \eta^2(\mathbf{U}_B^t - \mathbf{B}^t\mathbf{P}_\eta^t)(\mathbf{U}_A^t + \mathbf{P}_\eta^t\mathbf{A}^t),$$

$$\widehat{\mathbf{B}}^{t+1}\widehat{\mathbf{A}}^{t+1} = \mathbf{B}^t\mathbf{A}^t - \eta(\mathbf{U}_B^t\mathbf{A}^t + \mathbf{B}^t\mathbf{U}_A^t) + \eta^2\mathbf{U}_B^t\mathbf{U}_A^t.$$

Subtracting the two identities gives

$$\mathbf{B}^{t+1}\mathbf{A}^{t+1} - \widehat{\mathbf{B}}^{t+1}\widehat{\mathbf{A}}^{t+1} = \eta^2\big(\mathbf{U}_B^t\mathbf{P}_\eta^t\mathbf{A}^t - \mathbf{B}^t\mathbf{P}_\eta^t\mathbf{U}_A^t - \mathbf{B}^t(\mathbf{P}_\eta^t)^2\mathbf{A}^t\big), \tag{38}$$

Consequently,

$$\begin{aligned}\|\mathbf{B}^{t+1}\mathbf{A}^{t+1} - \widehat{\mathbf{B}}^{t+1}\widehat{\mathbf{A}}^{t+1}\| \leqslant \eta^2\big(&\|\mathbf{U}_B^t\|\|\mathbf{P}_\eta^t\|_2\|\mathbf{A}^t\|_2 + \|\mathbf{B}^t\|_2\|\mathbf{P}_\eta^t\|_2\|\mathbf{U}_A^t\| \\ &+ \|\mathbf{B}^t\|_2\|\mathbf{P}_\eta^t\|_2^2\|\mathbf{A}^t\|\big).\end{aligned} \tag{39}$$

Thus the matrix-based adjustment exactly enforces balance. The display above also shows that, provided $\mathbf{P}_\eta^t$ remains bounded, it changes the raw product step only at second order. Since $\mathbf{P}_\eta^t$ is applied to the update directions after the optimizer has produced $\mathbf{U}_B^t, \mathbf{U}_A^t$, no reparameterization or direct transformation of its moment estimates is required.

To make the second-order characterization explicit, we further bound the magnitude of the balance-preserving matrix. Define

$$\mu_\eta^t := \lambda_{\min}\left(\frac{\mathbf{K}^t + (\mathbf{K}^t)^\top}{2}\right). \tag{40}$$

Assume that $\mu_\eta^t > 0$. Since

$$\mathbf{P}_\eta^t(\mathbf{K}^t)^\top + \mathbf{K}^t\mathbf{P}_\eta^t = \mathbf{Q}^t, \tag{41}$$

the unique solution admits the Lyapunov representation

$$\mathbf{P}_\eta^t = \int_0^\infty e^{-\mathbf{K}^t s}\mathbf{Q}^t e^{-(\mathbf{K}^t)^\top s}\, ds. \tag{42}$$

Moreover,

$$\|e^{-\mathbf{K}^t s}\|_2 \leq e^{-\mu_\eta^t s}, \tag{43}$$

and hence

$$\|\mathbf{P}_\eta^t\|_2 \leq \frac{\|\mathbf{Q}^t\|_2}{2\mu_\eta^t}. \tag{44}$$

Let

$$\alpha := \lambda_{\min}(\mathbf{S}^t), \qquad \beta := \lambda_{\max}(\mathbf{S}^t). \tag{45}$$

Since $(\mathbf{B}^t)^\top \mathbf{B}^t = \mathbf{A}^t(\mathbf{A}^t)^\top = \mathbf{S}^t$, we have

$$\|\mathbf{B}^t\|_2 = \|\mathbf{A}^t\|_2 = \sqrt{\beta}. \tag{46}$$

Therefore,

$$\begin{aligned}\mu_\eta^t &= \lambda_{\min}\left(2\mathbf{S}^t - \frac{\eta}{2}\left((\mathbf{U}_B^t)^\top \mathbf{B}^t + (\mathbf{B}^t)^\top \mathbf{U}_B^t + \mathbf{U}_A^t(\mathbf{A}^t)^\top + \mathbf{A}^t(\mathbf{U}_A^t)^\top\right)\right) \\ &\geq 2\alpha - \frac{\eta}{2}\left\|(\mathbf{U}_B^t)^\top \mathbf{B}^t + (\mathbf{B}^t)^\top \mathbf{U}_B^t + \mathbf{U}_A^t(\mathbf{A}^t)^\top + \mathbf{A}^t(\mathbf{U}_A^t)^\top\right\|_2 \\ &\geq 2\alpha - \eta\sqrt{\beta}\left(\|\mathbf{U}_B^t\|_2 + \|\mathbf{U}_A^t\|_2\right). \end{aligned} \tag{47}$$

In particular, if

$$\eta\sqrt{\beta}\left(\|\mathbf{U}_B^t\|_2 + \|\mathbf{U}_A^t\|_2\right) \leq \alpha, \tag{48}$$

then $\mu_\eta^t \geq \alpha$. Furthermore,

$$\begin{aligned}\|\mathbf{Q}^t\|_2 \leq\ & 2\sqrt{\beta}\left(\|\mathbf{U}_B^t\|_2 + \|\mathbf{U}_A^t\|_2\right) \\ &+ \eta\left(\|\mathbf{U}_B^t\|_2^2 + \|\mathbf{U}_A^t\|_2^2\right). \end{aligned} \tag{49}$$

Combining this bound with Eq. (44) gives

$$\|\mathbf{P}_\eta^t\|_2 \leq \frac{2\sqrt{\beta}\left(\|\mathbf{U}_B^t\|_2 + \|\mathbf{U}_A^t\|_2\right) + \eta\left(\|\mathbf{U}_B^t\|_2^2 + \|\mathbf{U}_A^t\|_2^2\right)}{2\alpha}. \tag{50}$$

Thus, provided that the balanced Gram matrix remains nondegenerate, i.e., α is bounded away from zero, and the raw optimizer update directions remain bounded, we have

$$\|\mathbf{P}_\eta^t\|_2 = \mathcal{O}(1). \tag{51}$$

In particular, $\mathbf{P}_\eta^t$ cannot grow as $\mathcal{O}(\eta^{-1})$ under these conditions. Consequently,

$$\eta\|\mathbf{P}_\eta^t\|_2 = \mathcal{O}(\eta), \qquad \eta^2\|\mathbf{P}_\eta^t\|_2^2 = \mathcal{O}(\eta^2). \tag{52}$$

Therefore, the quadratic term involving $\mathbf{P}_\eta^t$ in Eq. (38) satisfies

$$\|\eta^2 \mathbf{B}^t(\mathbf{P}_\eta^t)^2\mathbf{A}^t\| \leq \eta^2\|\mathbf{B}^t\|_2\|\mathbf{P}_\eta^t\|_2^2\|\mathbf{A}^t\| = \mathcal{O}(\eta^2). \tag{53}$$

The other two terms in Eq. (38) have the same $\mathcal{O}(\eta^2)$ order under the stated boundedness assumptions. Hence, $\mathbf{P}_\eta^t$ remains bounded and the matrix-based adjustment changes the raw product update only at second order in η.

B Low-Rank Gradient Approximation Algorithms

B.1 Standard Randomized SVD

Standard randomized SVD applied directly to the minibatch gradient $\mathbf{g}_j$ is given in Algorithm 3.

B.2 Streaming Low-Rank Accumulation

At minibatch $i \geq 2$, write the current rank-r accumulator and the rank-r approximation of $\mathbf{g}_i$ as

$$\mathbf{H}^{i-1} = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^\top, \qquad \widetilde{\mathbf{g}}_i = \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^\top,$$

where $\mathbf{U}_1, \mathbf{U}_2 \in \mathbb{R}^{m\times r}$, $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2 \in \mathbb{R}^{r\times r}$, and $\mathbf{V}_1, \mathbf{V}_2 \in \mathbb{R}^{n\times r}$.

Algorithm 4 constructs orthonormal bases $\mathbf{U}_c$ and $\mathbf{V}_c$ for the joint column and row spaces. Therefore, $\mathbf{U}_c\mathbf{U}_c^\top\mathbf{U}_s = \mathbf{U}_s$ and $\mathbf{V}_c\mathbf{V}_c^\top\mathbf{V}_s = \mathbf{V}_s$ for $s \in \{1, 2\}$, which gives

$$\mathbf{H}^{i-1} + \widetilde{\mathbf{g}}_i = \mathbf{U}_c\mathbf{K}\mathbf{V}_c^\top, \tag{54}$$

where

$$\mathbf{K} = (\mathbf{U}_c^\top\mathbf{U}_1)\mathbf{\Sigma}_1(\mathbf{V}_1^\top\mathbf{V}_c) + (\mathbf{U}_c^\top\mathbf{U}_2)\mathbf{\Sigma}_2(\mathbf{V}_2^\top\mathbf{V}_c).$$

Thus, each merge is computed through a matrix of size at most $2r \times 2r$, without forming the full $m \times n$ sum. Computing the SVD of $\mathbf{K}$ and retaining its top r components produces the factors returned by Algorithm 4.

Algorithm 3 Standard Randomized SVD

Require: Minibatch gradient $\mathbf{g}_j \in \mathbb{R}^{m\times n}$, rank r, oversampling p, power iterations q
Ensure: Rank-r approximation $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ of $\mathbf{g}_j$

$\ell \leftarrow r + p$
$\mathbf{A} \in \mathbb{R}^{\ell\times n} \sim \mathcal{N}(0,1)$
$\mathbf{Y}_B \leftarrow \mathbf{g}_j\mathbf{A}^\top$
for $i = 1$ **to** q **do**
  $\mathbf{Q} \leftarrow \mathrm{QR}(\mathbf{Y}_B)$
  $\mathbf{Y}_A \leftarrow \mathbf{Q}^\top\mathbf{g}_j$
  $\mathbf{Q} \leftarrow \mathrm{QR}(\mathbf{Y}_A^\top)$
  $\mathbf{Y}_B \leftarrow \mathbf{g}_j\mathbf{Q}$
end for
$\mathbf{Q} \leftarrow \mathrm{QR}(\mathbf{Y}_B)$
$\mathbf{Y}_A \leftarrow \mathbf{Q}^\top\mathbf{g}_j$
$\widetilde{\mathbf{U}}, \mathbf{\Sigma}, \mathbf{V} \leftarrow \mathrm{SVD}(\mathbf{Y}_A)$; $\mathbf{U} \leftarrow \mathbf{Q}\widetilde{\mathbf{U}}$
return $\mathbf{U}_{:,1:r}$, $\mathbf{\Sigma}_{1:r,1:r}$, $\mathbf{V}_{:,1:r}$

Algorithm 4 Low Rank Merge for Streaming Accumulation

Require: Accumulated rank-r factors $(\mathbf{U}_1, \mathbf{\Sigma}_1, \mathbf{V}_1)$ of $\mathbf{H}^{i-1}$, rank-r factors $(\mathbf{U}_2, \mathbf{\Sigma}_2, \mathbf{V}_2)$ of $\widetilde{\mathbf{g}}_i$
Ensure: Merged rank-r factors $(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V})$

Compute the joint column-space basis: $\mathbf{U}_c \leftarrow \mathrm{QR}([\mathbf{U}_1, \mathbf{U}_2])$
Compute the joint row-space basis: $\mathbf{V}_c \leftarrow \mathrm{QR}([\mathbf{V}_1, \mathbf{V}_2])$
Compute the coordinates of the merged matrix in the joint bases: $\mathbf{K} \leftarrow (\mathbf{U}_c^\top\mathbf{U}_1)\mathbf{\Sigma}_1(\mathbf{V}_1^\top\mathbf{V}_c) + (\mathbf{U}_c^\top\mathbf{U}_2)\mathbf{\Sigma}_2(\mathbf{V}_2^\top\mathbf{V}_c)$
$\widetilde{\mathbf{U}}, \widetilde{\mathbf{\Sigma}}, \widetilde{\mathbf{V}} \leftarrow \mathrm{SVD}(\mathbf{K})$
$\mathbf{U} \leftarrow (\mathbf{U}_c\widetilde{\mathbf{U}})_{:,1:r}$
$\mathbf{\Sigma} \leftarrow \widetilde{\mathbf{\Sigma}}_{1:r,1:r}$
$\mathbf{V} \leftarrow (\mathbf{V}_c\widetilde{\mathbf{V}})_{:,1:r}$
return $(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V})$

C Additional Computational and Memory Costs

C.1 Complexity Summary

For one linear layer, let $\mathbf{W}^0 \in \mathbb{R}^{m\times n}$ be the frozen weight and $\mathbf{X} \in \mathbb{R}^{b\times n}$ its input, where b is the number of input vectors processed by the layer. The LoRA matrices are $\mathbf{B} \in \mathbb{R}^{m\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times n}$, where r is the LoRA rank, and q is the number of randomized-SVD power iterations. For randomized SVD, $\ell = r + p$ with oversampling size $p = \mathcal{O}(r)$, so $\ell = \mathcal{O}(r)$. The time columns count additional matrix operations, whereas the memory columns count method-specific parameters, saved activations, and temporary memory. The frozen weight and the backbone computation shared by all methods are excluded.

Tables 4–6 report the additional per-layer computation and memory introduced by each LoRA method during initialization, the forward pass, and backward/update, respectively. All entries are given in $\mathcal{O}(\cdot)$ notation.

Table 4: Additional per-layer initialization cost.

Method	Additional Time	Additional Memory
LoRA	$r(m+n)$	$r(m+n)$
RefLoRA	$r(m+n)$	$r(m+n)$
BaLoRA	$r(m+n)$	$r(m+n)$
AltLoRA	$r(m+n)$	$r(m+n)$
LoRA-One	$bmn + qmnr$	$mn + r(m+n) + b(m+n)$
LoRA-GA	$bmn + qmnr$	$mn + r(m+n) + b(m+n)$
DoRA	$mn + r(m+n)$	$r(m+n) + m$
PiSSA	$mn\min(m,n)$	$mn + r(m+n)$
BDA-LoRA	$qbr(m+n) + q(m+n)r^2 + r^3 + mnr$	$r(m+n)$

Table 5: Additional per-layer forward-pass cost.

Method	Additional Time	Additional Memory
LoRA	$br(m+n)$	$r(m+n) + br$
RefLoRA	$br(m+n)$	$r(m+n) + br$
BaLoRA	$br(m+n)$	$r(m+n) + br$
AltLoRA	$br(m+n)$	$r(m+n) + br$
LoRA-One	$br(m+n)$	$r(m+n) + br$
LoRA-GA	$br(m+n)$	$r(m+n) + br$
DoRA	$br(m+n) + mnr$	$r(m+n) + br + mn$
PiSSA	$br(m+n)$	$r(m+n) + br$
BDA-LoRA	$br(m+n)$	$r(m+n) + br$

Table 6: Additional per-layer backward/update cost.

Method	Additional Time	Additional Memory
LoRA	$br(m+n)$	$r(m+n)+b(n+r)$
RefLoRA	$br(m+n)+(m+n+r)r^2$	$r(m+n)+b(n+r)+r^2$
BaLoRA	$br(m+n)+(m+n+r)r^2$	$r(m+n)+b(n+r)+r^2$
AltLoRA	$br(m+n)+(m+n+r)r^2$	$r(m+n)+b(n+r)+r^2$
LoRA-One	$br(m+n)$	$r(m+n)+b(n+r)$
LoRA-GA	$br(m+n)$	$r(m+n)+b(n+r)$
DoRA	$br(m+n)$	$r(m+n)+b(n+r)+m$
PiSSA	$br(m+n)$	$r(m+n)+b(n+r)$
BDA-LoRA	$br(m+n)+(m+n+r)r^2$	$r(m+n)+b(n+r)+r^2$

C.2 Derivation of the Complexity Terms

LoRA. (Hu et al., 2022) Initializing and storing $\mathbf{B}$ and $\mathbf{A}$ requires $\mathcal{O}(r(m+n))$ time and memory. The forward pass evaluates $\mathbf{X}\mathbf{A}^\top$ and then multiplies the result by $\mathbf{B}^\top$, which costs $\mathcal{O}(br(n+m))$ and stores the $b \times r$ intermediate activation. Computing the two factor gradients has the same time order. Backpropagation additionally retains the layer input and the rank-r activation, giving $\mathcal{O}(r(m+n)+b(n+r))$ memory.

RefLoRA. (Zhang et al., 2025a) Its initialization and forward-pass costs match LoRA. RefLoRA constructs a pair of $r \times r$ preconditioners from the factor Gram matrices and applies them to the two factor gradients before the optimizer step. Forming the Gram matrices and applying the preconditioners costs $\mathcal{O}((m+n)r^2)$. Computing the matrix square roots and inverse square roots required by the preconditioners costs $\mathcal{O}(r^3)$. The total backward/update cost is therefore $\mathcal{O}(br(m+n)+(m+n+r)r^2)$, with an additional $\mathcal{O}(r^2)$ temporary memory.

BaLoRA. (Castin et al., 2026) Its initialization and forward-pass costs also match LoRA. BaLoRA balances the two matrices through an SVD-based refactorization that preserves their product. Thin QR factorizations of $\mathbf{B}$ and $\mathbf{A}^\top$ cost $\mathcal{O}((m+n)r^2)$, the SVD of the resulting $r \times r$ core costs $\mathcal{O}(r^3)$, and reconstructing the two matrices costs $\mathcal{O}((m+n)r^2)$. Hence its backward/update cost is $\mathcal{O}(br(m+n)+(m+n+r)r^2)$, with an additional $\mathcal{O}(r^2)$ temporary memory.

AltLoRA. (Yu et al., 2025) AltLoRA initializes and stores the same two rank-r factors as LoRA, and its forward pass uses the same two low-rank matrix products. At each training step it differentiates only one factor. Its alternating projection and momentum transport form an $r \times r$ Gram matrix, multiply the active factor or its gradient by an $r \times r$ matrix, and solve an $r \times r$ linear system. These operations cost $\mathcal{O}((m+n)r^2+r^3)$ in the worst case, on top of the $\mathcal{O}(br(m+n))$ factor-gradient computation. The factors and momentum states have $\mathcal{O}(r(m+n))$ memory order, and the small-matrix operations require $\mathcal{O}(r^2)$ temporary memory. Thus its backward/update time and memory are $\mathcal{O}(br(m+n)+(m+n+r)r^2)$ and $\mathcal{O}(r(m+n)+b(n+r)+r^2)$, respectively.

LoRA-One and LoRA-GA. (Zhang et al., 2025b; Wang et al., 2024) Constructing the dense $m \times n$ gradient from b input vectors costs $\mathcal{O}(bmn)$. Each randomized-SVD power iteration multiplies this gradient by a rank-r block and costs $\mathcal{O}(mnr)$, yielding $\mathcal{O}(bmn+qmnr)$ initialization time. The dense gradient, the LoRA matrices, and the layer activations require $\mathcal{O}(mn+r(m+n)+b(m+n))$ memory. After initialization, both methods have the same forward and backward/update orders as LoRA.

DoRA. (Liu et al., 2024) Computing the weight magnitudes during initialization costs $\mathcal{O}(mn)$, in addition to initializing the two rank-r matrices. DoRA stores the LoRA matrices and an m-dimensional magnitude vector. During the forward pass, constructing the normalized weight direction adds $\mathcal{O}(mnr)$ time and an $m \times n$ temporary matrix to the standard LoRA cost. Its backward/update time has the same leading order as LoRA, with the magnitude vector contributing an additional $\mathcal{O}(m)$ memory.

PiSSA. (Meng et al., 2024) The dense SVD of $\mathbf{W}^0$ costs $\mathcal{O}(mn\min\{m,n\})$ time and $\mathcal{O}(mn)$ temporary memory. Retaining the rank-r matrices adds $\mathcal{O}(r(m+n))$ memory. Its forward and backward/update costs then match those of LoRA.

BDA-LoRA. Each alternating factor-gradient evaluation costs $\mathcal{O}(br(m+n))$. The initial projection, q power iterations, and the final projection therefore contribute $\mathcal{O}(qbr(m+n))$ time after suppressing constant factors. The QR factorizations and low-rank basis transformations contribute $\mathcal{O}(q(m+n)r^2)$, and the small SVD costs $\mathcal{O}(r^3)$. Computing the initialization offset $\mathbf{B}^1\mathbf{A}^1$ adds $\mathcal{O}(mnr)$. The streaming accumulator stores the current and incoming rank-r matrices, requiring $\mathcal{O}(2r(m+n))$ memory. The forward-pass cost matches LoRA. During each update, constructing the $r \times r$ Lyapunov equation and applying its solution costs $\mathcal{O}((m+n)r^2)$, and solving it costs $\mathcal{O}(r^3)$. Hence the total backward/update cost is $\mathcal{O}(br(m+n)+(m+n+r)r^2)$ with an additional $\mathcal{O}(r^2)$ temporary memory.

D Experimental Details

D.1 NLU Setup

We fine-tune DeBERTa-v3-base on the official GLUE splits (Wang et al., 2019) for 10 epochs with LoRA rank $r = 8$, scale $\alpha = 16$, and a learning rate of 1×10^{-4}. The maximum sequence length is 128 and the effective training batch size is 128. We adapt `query`, `key`, `value`, and `output.dense`, using $N = 10$ initialization batches and initialization scale $\gamma = 1/128$. We report Matthews correlation coefficient (MCC) for CoLA (Warstadt et al., 2019) and accuracy for the other GLUE tasks. We use AdamW as the base optimizer for all methods except AltLoRA; for AltLoRA, we strictly follow the optimization procedure in its original paper (Yu et al., 2025). During training, we solve the corresponding matrix equation with SciPy's continuous Lyapunov solver (Virtanen et al., 2020).

D.2 NLG Setup

For both Meta-Llama-3-8B and Gemma-7B, we fine-tune on 100K samples from MetaMathQA (Yu et al., 2024) for GSM8K (Cobbe et al., 2021), and on 100K samples from CodeFeedback-Filtered-Instruction (Zheng et al., 2024) for HumanEval (Chen et al., 2021). The LoRA rank and scale are $(r, \alpha) = (8, 16)$ for GSM8K and $(32, 64)$ for HumanEval. For both tasks, we adapt `q_proj`, `k_proj`, `v_proj`, and `o_proj`, using $N = 10$ initialization batches and $\gamma = 1/128$. We train for one epoch with a maximum sequence length of 1024 and an effective training batch size of 32. We conducted a grid search over the learning rates $\{1 \times 10^{-1}, 1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}, 5 \times 10^{-5}, 2 \times 10^{-5}\}$. We report accuracy for GSM8K and pass@1(Chen et al., 2021) for HumanEval. We use AdamW as the base optimizer for all methods except AltLoRA; for AltLoRA, we strictly follow the optimization procedure in its original paper (Yu et al., 2025). During training, we solve the corresponding matrix equation with SciPy's continuous Lyapunov solver (Virtanen et al., 2020).

D.3 Commonsense Reasoning Setup

For commonsense reasoning, we separately fine-tune Meta-Llama-3-8B on BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), Social IQA (SIQA) (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-Challenge and ARC-Easy (Clark et al., 2018), and OpenBookQA (OBQA) (Mihaylov et al., 2018). For each dataset, we train for three epochs with a maximum sequence length of 256 and an effective training batch size of 32. We adapt `q_proj`, `k_proj`, `v_proj`, and `o_proj` with $r = 8$ and $\alpha = 16$. Initialization uses $N = 10$ batches and $\gamma = 1/128$. We conduct a grid search over the learning rates $\{1 \times 10^{-4}, 5 \times 10^{-5}, 2 \times 10^{-5}\}$ and report accuracy on each benchmark. We use AdamW as the base optimizer for all methods except AltLoRA; for AltLoRA, we strictly follow the optimization procedure in its original paper (Yu et al., 2025). During training, we solve the corresponding matrix equation with SciPy's continuous Lyapunov solver (Virtanen et al., 2020).

Table 7: Training configuration for the NLU experiments.

Configuration	GLUE
Backbone	DeBERTa-v3-base
LoRA rank r	8
LoRA alpha α	16
Initialization batches N	10
Initialization scale γ	$1/128$
Target modules	`query,key,value,output.dense`
Learning rate	1×10^{-4}
Epochs	10
Maximum sequence length	128
Effective train batch size	128

Table 8: Training configurations for the NLG experiments.

Configuration	GSM8K	HumanEval
Backbones	Meta-Llama-3-8B and Gemma-7B	
LoRA rank r	8	32
LoRA alpha α	16	64
Initialization batches N	10	10
Initialization scale γ	$1/128$	$1/128$
Target modules	`q_proj,k_proj,v_proj,o_proj`	
Learning-rate grid	$\{1\times10^{-1}, 1\times10^{-2}, 1\times10^{-3}, 1\times10^{-4}, 5\times10^{-5}, 2\times10^{-5}\}$	
Epochs	1	1
Maximum sequence length	1024	1024
Effective train batch size	32	32

Table 9: Training configuration for the commonsense reasoning experiment.

Configuration	Commonsense reasoning
Backbone	Meta-Llama-3-8B
LoRA rank r	8
LoRA alpha α	16
Initialization batches N	10
Initialization scale γ	$1/128$
Target modules	`q_proj,k_proj,v_proj,o_proj`
Learning-rate grid	$\{1\times10^{-4}, 5\times10^{-5}, 2\times10^{-5}\}$
Epochs	3
Maximum sequence length	256
Effective train batch size	32