GraLoRA: Granular Low-Rank Adaptation for Parameter-Efficient Fine-Tuning

Yeonjoon Jung[1,2*] **Daehyun Ahn**[1] **Hyungjun Kim**[1] **Taesu Kim**[1] **Eunhyeok Park**[2†]
[1]SqueezeBits [2]POSTECH
{yeonjoon.jung, daehyun.ahn, hyungjun.kim, taesu.kim}@squeezebits.com
{yeonjoon.jung, eh.park}@postech.ac.kr

Abstract

Low-Rank Adaptation (LoRA) is a popular method for parameter-efficient fine-tuning (PEFT) of generative models, valued for its simplicity and effectiveness. Despite recent enhancements, LoRA still suffers from a fundamental limitation: overfitting when the bottleneck is widened. It performs best at ranks 32–64, yet its accuracy stagnates or declines at higher ranks, still falling short of full fine-tuning (FFT) performance. We identify the root cause as LoRA's structural bottleneck, which introduces gradient entanglement to the unrelated input channels and distorts gradient propagation. To address this, we introduce a novel structure, Granular Low-Rank Adaptation (GraLoRA) that partitions weight matrices into sub-blocks, each with its own low-rank adapter. With negligible computational or storage cost, GraLoRA overcomes LoRA's limitations, effectively increases the representational capacity, and more closely approximates FFT behavior. Experiments on code generation, commonsense reasoning, mathematical reasoning, general language understanding, and image generation benchmarks show that GraLoRA consistently outperforms LoRA and other baselines, achieving up to +8.5% absolute gain in Pass@1 on HumanEval+. These improvements hold across model sizes and rank settings, making GraLoRA a scalable and robust solution for PEFT.

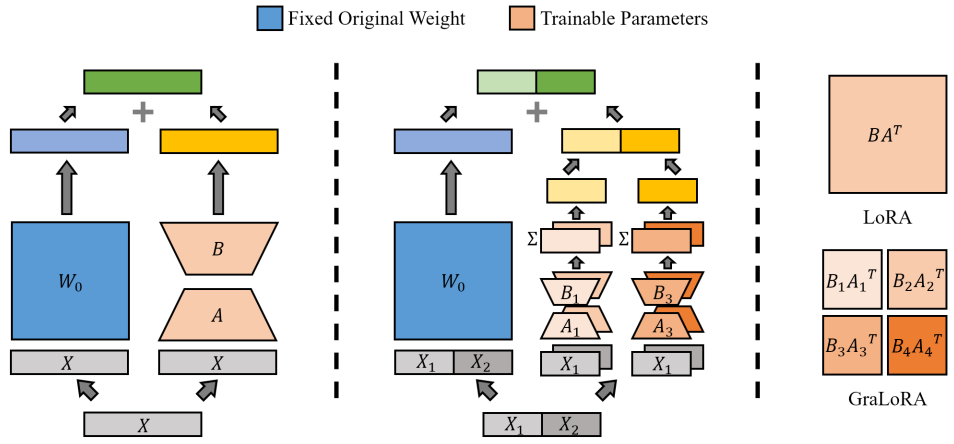
1 Introduction

Task-specific fine-tuning enables a wide range of applications and significantly improves the quality and effectiveness of generative models. However, the massive scale of these models poses substantial challenges for practical deployment. To address these limitations, Parameter-Efficient Fine-Tuning (PEFT) methods have emerged as a cost-effective alternative [12, 32]. Among them, Low-Rank Adaptation (LoRA) [13] has gained particular attention for its simplicity and effectiveness, introducing trainable low-rank matrices while keeping the pre-trained model weights frozen. Although the imposed rank-r bottleneck may lead to slight performance degradation compared to full fine-tuning (FFT), its efficiency has led to widespread adoption in practice.

To maximize the benefits of LoRA, various studies have proposed techniques such as improved initialization [4, 21, 23, 29] and structural refinements [10, 15, 16, 17] to enhance fine-tuning quality. While these efforts have advanced performance, a substantial quality gap remains compared to FFT, largely due to the inherent upper bound on the rank. Although using a higher rank, within hardware limits, appears to be a natural solution, unfortunately, current implementations of LoRA and its variants do not support such flexibility. Simply increasing the rank often leads to degraded accuracy in many scenarios.

In this paper, we present a theoretical analysis identifying the root cause of the rank limitation in LoRA. Our analysis reveals a fundamental issue in LoRA's structure, channel dominance in the gradient, where a small subset of outlier channels disproportionately influences the update direction.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).



(a) LoRA Workflow (b) GraLoRA Workflow (c) Weight Update

Figure 1: Illustration of LoRA architecture and GraLoRA architecture. GraLoRA consists of k^2 small adapter pairs, where each input and output dimension is k times smaller than the original LoRA.

This dominance suppresses contributions from other channels, leading to under-utilization of the available rank and degraded performance in tasks that require nuanced or distributed representations.

To overcome these expressivity bottlenecks, we propose Granular Low-Rank Adaptation (GraLoRA), a novel architectural extension of LoRA. As shown in Figure 1, GraLoRA divides the weight matrix into multiple sub-blocks and applies independent LoRA modules to each, enabling fine-grained updates. This design enhances the model's capacity to capture complex, localized, or multi-faceted patterns, effectively mitigating the channel dominance issue and improving performance—especially at higher ranks.

Extensive experiments show that GraLoRA consistently outperforms vanilla LoRA across a range of NLP benchmarks, particularly in scenarios with high input heterogeneity or task complexity. These results position GraLoRA as a principled and practical advancement in the PEFT landscape.

2 Details and Limitations of LoRA

2.1 Introduction to LoRA

LoRA is one of the most widely adopted strategies for PEFT. Given a pre-trained weight matrix $W_0 \in \mathbb{R}^{M\times N}$, where M and N represent the input and output channel dimension, respectively, LoRA keeps W_0 frozen and introduces a trainable low-rank update defined as:

$$R = sBA^\top, \quad A \in \mathbb{R}^{N\times r}, \quad B \in \mathbb{R}^{M\times r}, \quad s = \frac{\alpha}{r}. \tag{1}$$

Here, rank r and α are user-defined hyperparameters. Then, for a given input $X \in \mathbb{R}^{N\times T}$, the output of the LoRA-adapted layer is $Y = W_0X + RX \in \mathbb{R}^{M\times T}$, where T denotes the batch or token dimension. This low-rank decomposition allows the model to adapt using significantly fewer trainable parameters and reduced memory overhead.

While FFT updates the entire weight matrix, LoRA only updates the decomposed low-rank matrices A and B. Note that we assume s = 1 for simplicity, the gradient of the loss with respect to R is:

$$\nabla_R L = \frac{\partial L}{\partial R} = \frac{\partial L}{\partial Y}X^\top \in \mathbb{R}^{M\times N} \tag{2}$$

From this, the gradients with respect to the LoRA parameters B and A are given by:

$$\frac{\partial L}{\partial B} = \frac{\partial L}{\partial Y}X^\top A, \quad \frac{\partial L}{\partial A^\top} = B^\top \frac{\partial L}{\partial Y}X^\top. \tag{3}$$

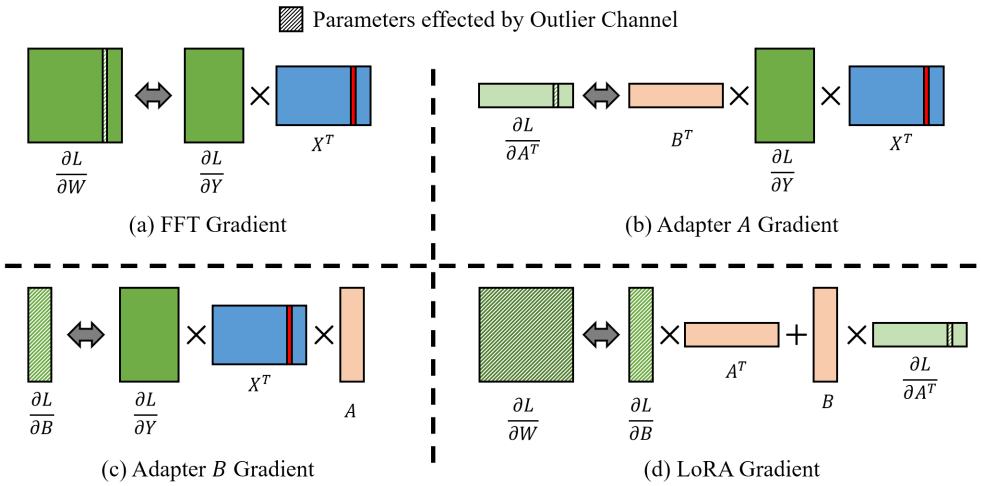


Figure 2: Gradient dynamics of FFT and LoRA in the presence of an outlier input channel. The red channel in input X denotes the outlier. While FFT localizes the gradient impact, LoRA's entire gradient update becomes disproportionately influenced by the single outlier.

These result in the following reconstructed update in the fused weight space:

$$\tilde{\nabla}_R L = \frac{\partial L}{\partial B} A^\top + B \frac{\partial L}{\partial A^\top} = \frac{\partial L}{\partial Y} X^\top A A^\top + B B^\top \frac{\partial L}{\partial Y} X^\top. \tag{4}$$

This expression reveals how the structure of LoRA introduces non-trivial interactions between the gradients and the input, particularly through the rank-r matrices.

2.2 Why Does LoRA Suffer from a Larger Rank?

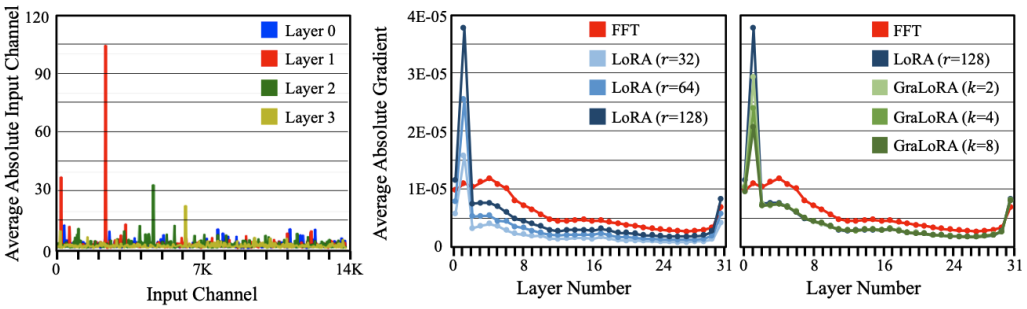
When fine-tuning with a large LoRA rank (e.g., $r > 64$), it is often observed that accuracy degrades compared to using a moderate rank. This counterintuitive behavior arises from the distinct gradient dynamics of LoRA, which differ significantly from those of FFT.

LoRA's structural design makes its gradients inherently sensitive to the entire input space, as illustrated in Figure 2. In particular, we observe that **outlier channels**, input channels with abnormally high activations, can disproportionately dominate the gradient signal.

In FFT, the effect of such outliers is typically localized, affecting only a single column of the weight matrix W that directly interacts with the outlier channel. In contrast, LoRA's low-rank constraint causes the entire gradient of the adapter matrix B, denoted $\partial L/\partial B$, to be influenced by these outliers. This results in distorted weight updates in the fused weight space, where the gradient signal from outlier channels overwhelms the contributions from other inputs. Consequently, LoRA fails to accurately replicate the gradient dynamics of FFT, limiting its ability to match FFT-level performance.

We observe that in certain layers, most notably the down-projection matrix of Layer 1 in LLaMA3.1–8B, input activations exhibit severe channel-wise imbalance (Figure 3 (a)). As shown in Figure 4, these outlier channels disproportionately impact the adapter's gradient updates. Figure 3 further illustrates that the gap between LoRA and FFT gradient updates widens as the LoRA rank increases.

These findings reveal a fundamental misalignment between LoRA updates and the gradient landscape shaped by FFT. The entangled influence of input channels caused by the low-rank projection limits LoRA's ability to selectively learn from salient features, particularly under skewed input statistics. While the negative impact of outliers has been well recognized in the context of quantization [31] [18], their influence on LoRA's behavior has not been systematically studied until now.



(a) Outlier Channel Analysis (b) LoRA Gradients (c) GraLoRA Gradients

Figure 3: (a) Mean input channel values for the down-projection matrices across layers in LLaMA3.1–8B. A pronounced outlier exists in Layer 1, channel 198 and 2427. (b) Gradient deviation between LoRA and FFT increases with rank, showing LoRA's susceptibility to input outliers. (c) GraLoRA gradient results at rank 128. GraLoRA noticeably reduces gradient deviation between FFT.

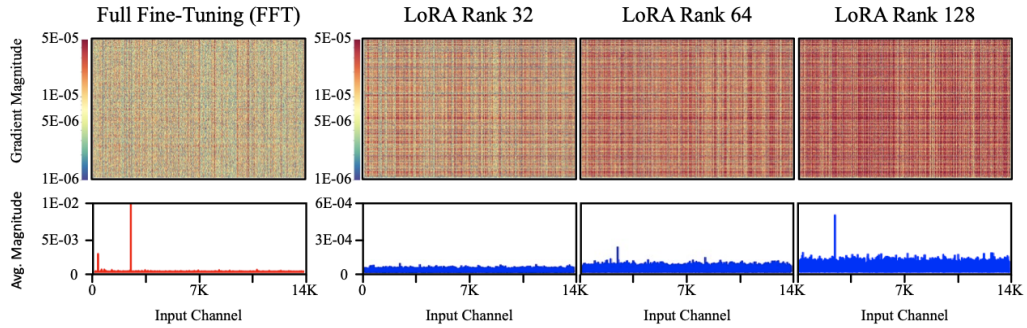


Figure 4: Gradient distribution in Layer 1 down-projection matrix. LoRA gradients show poor alignment with FFT, outlier channel increases the overall gradient scale, while less emphasizing the corresponding outlier channel.

3 Method

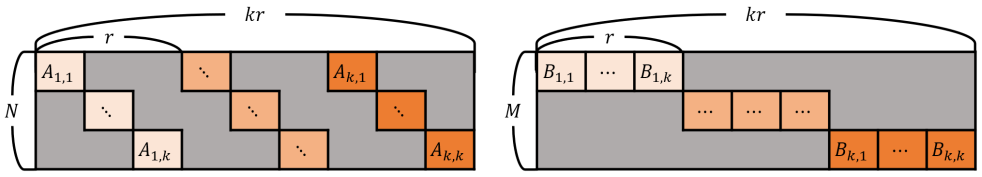
3.1 GraLoRA: Granular Low-Rank Adaptation

Motivated by observation in previous section, we propose **GraLoRA**, a fine-grained and modular extension of LoRA. As illustrated in Figure 1, GraLoRA addresses the limitations of standard LoRA by partitioning the weight matrix into a grid of $k \times k$ independent blocks, each equipped with its own local low-rank adapter. Here, k is a hyperparameter that determines the number of splits along the input and output dimensions. When $k = 1$, GraLoRA reduces to the vanilla LoRA formulation.

Specifically, the weight update $R \in \mathbb{R}^{M \times N}$ is expressed as the concatenation of block-wise updates:

$$R_{\text{GraLoRA}} = \begin{bmatrix} B_{1,1}A_{1,1}^\top & \cdots & B_{1,k}A_{1,k}^\top \\ \vdots & \ddots & \vdots \\ B_{k,1}A_{k,1}^\top & \cdots & B_{k,k}A_{k,k}^\top \end{bmatrix}, \quad A_{i,j} \in \mathbb{R}^{\frac{N}{k} \times \frac{r}{k}}, \quad B_{i,j} \in \mathbb{R}^{\frac{M}{k} \times \frac{r}{k}} \tag{5}$$

This block-wise reparameterization provides localized control over each spatial subregion of the parameter space. As detailed in Section 3.4, GraLoRA incurs the same parameter count and computational overhead as standard LoRA when using the same rank. However, it introduces two key advantages; (1) **Enhanced Expressivity** and (2) **Robustness to Input Outliers**. By enabling independent adaptation across k^2 subspaces, GraLoRA supports more fine-grained and specialized feature learning. In addition, Localized gradient updates ensure that only the adapters associated with



(a) Structure of $A_{GraLoRA}$ (b) Structure of $B_{GraLoRA}$

Figure 5: Regularized form of GraLoRA as multiplication of sparse two matrices, A_{GraLoRA} and B_{GraLoRA}.

the affected input regions receive large gradients, thereby reducing global gradient distortion and preserving inter-channel signal balance.

3.2 Expression Power Analysis

While the weight update of GraLoRA was expressed as concatenation of block-wise updates in (5), it can also be regularized as the form of multiplication of two matrices as in the vanilla LoRA. The sparse matrix $A_{\text{GraLoRA}} \in \mathbb{R}^{N \times kr}$ can be constructed as Figure 5 (a), where $A_{i,j}$ for $i, j \in \{n \in \mathbb{N} \mid n \leq k\}$ is located in position $(i + (j-1) \times k, j)$ of A_{GraLoRA}. Other elements are masked out, thus the total number of parameter becomes $N \times r$.

Then, $B_{\text{GraLoRA}} \in \mathbb{R}^{N \times kr}$ is constructed as Figure 5 (b), where matrix $B_{i,j}$ for $i, j \in \{n \in \mathbb{N} \mid n \leq k\}$ is located in position $(i, j + (i-1) \times k)$ of B_{GraLoRA}, Similarly, other composition of the matrix is masked, therefore the total number of parameter becomes $M \times r$. Then the weight update of GraLoRA can be expressed as $W = W_0 + R_{\text{GraLoRA}} = W_0 + B_{\text{GraLoRA}} A_{\text{GraLoRA}}^{\top}$.

Assuming that all columns of $[B_{i,1}, \cdots, B_{i,k}]$ are linearly independent, the rank of B_{GraLoRA} becomes $\mathbf{R}(B_{\text{GraLoRA}}) = kr$. Similarly, if all columns of $[A_{1,j}, \cdots, A_{k,j}]$ are linearly independent, the rank of A_{GraLoRA} is $\mathbf{R}(A_{\text{GraLoRA}}) = kr$. Applying Sylvester's rank inequality to derive the lower bound and the matrix product theorem for the upper bound, we obtain:

$$\mathbf{R}(B_{\text{GraLoRA}}) + \mathbf{R}(A_{\text{GraLoRA}}^{\top}) - kr \leq \mathbf{R}(B_{\text{GraLoRA}} A_{\text{GraLoRA}}^{\top}) \leq min(\mathbf{R}(B_{\text{GraLoRA}}), \mathbf{R}(A_{\text{GraLoRA}}^{\top})) \quad (6)$$

Thus, the effective rank of R_{GraLoRA} becomes kr, which is k times higher than that of the vanilla LoRA method—effectively enhancing the model's expressive capacity. The rank analysis of fine-tuned LoRA and GraLoRA, summarized in Table 5 in Appendix, demonstrates that GraLoRA linearly scales the representational power of the adaptation matrix in practical settings.

3.3 Gradient Dynamics Under Outlier Activation

GraLoRA effectively localizes the influence of outlier channels to a limited subset of adapter blocks. Because each block processes only a specific slice of the input, only the k adapter pairs intersecting with the outlier channel are exposed to amplified gradients. In contrast, the remaining $k^2 - k$ adapters maintain gradient magnitudes close to baseline levels. This selective gradient propagation resembles the behavior of FFT, where only weights directly connected to active inputs are significantly updated.

GraLoRA's impact on gradient dynamics can be observed by comparing gradient distributions of the down-projection matrix in Layer 1 with standard LoRA. As illustrated in the Figure 3 (c) and Figure 6, GraLoRA reduces the gradient deviation and limits the influence of outlier channels, overcoming the limitations of standard LoRA with larger ranks.

3.4 Tradeoff Analysis

As discussed, GraLoRA provides several advantages over standard LoRA. However, these benefits do not come without cost. In this section, we provide deeper analysis on the overhead introduced by GraLoRA.

Computation Overhead Analysis: First, we analyze the expected computational cost of LoRA in terms of FLOPs. To take advantage of the low-rank structure, LoRA computes the projection in two sequential steps. The first computes $A^{\top} X \in \mathbb{R}^{r \times T}$, followed by the reconstruction $B(A^{\top} X) \in$

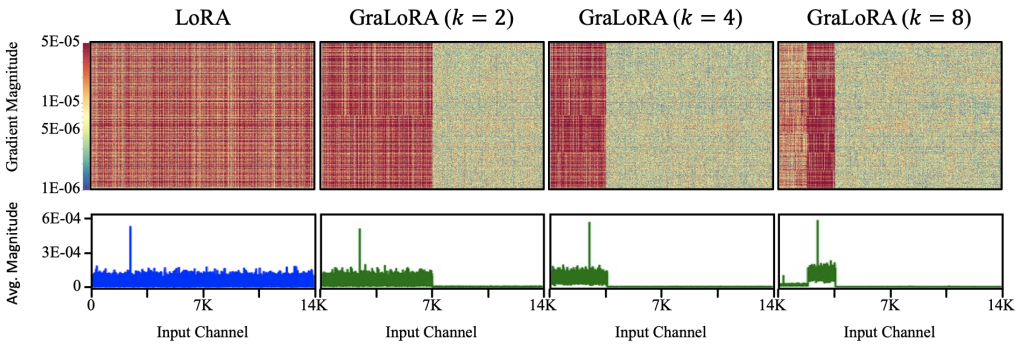


Figure 6: Comparison of gradient distributions under outlier activation. In GraLoRA, only the blocks interacting with the outlier exhibit elevated gradients, mitigating global distortion and aligning with FFT behavior.

$\mathbb{R}^{M\times T}$. These steps require $2NrT$ and $2rMT$ FLOPs, respectively, resulting in a total complexity of $O\left(r(M+N)T\right)$.

Similarly, GraLoRA divides the computation into two steps involving k^2 adapter blocks. In the first step, the projection computes $A_{i,j}^\top X_j \in \mathbb{R}^{\frac{r}{k}\times T}$ for each of the k^2 blocks, incurring a total cost of $2\cdot\frac{N}{k}\cdot\frac{r}{k}\cdot T\cdot k^2 = 2NrT$. In the second step, each intermediate output is processed by its corresponding $B_{i,j}$, producing $B_{i,j}(A_{i,j}^\top X_j) \in \mathbb{R}^{\frac{M}{k}\times T}$. This step adds another $2\cdot\frac{r}{k}\cdot\frac{M}{k}\cdot T\cdot k^2 = 2rMT$. FLOPs to the total cost. Hence, the overall computational cost of GraLoRA remains $O\left(r(M+N)T\right)$, maintaining efficiency comparable to vanilla LoRA while significantly enhancing expressive power. A detailed analysis of computational overhead is provided in Appendix C.

Table 1: Maximal allocated memory during training LLaMA3.1–8B model with batch size 1. Input length was set to 1024 and memory allocated for weight was removed for direct comparison.

	LoRA	GraLoRA (k=2)	GraLoRA (k=4)	GraLoRA (k=8)
Vanilla Backward (GB)	10.0	10.1	10.2	10.4
Gradient Checkpointing (GB)	2.6	2.6	2.6	2.6

Memory Overhead Analysis: As with classical LoRA, GraLoRA can be merged into the original weight matrix at inference time. Therefore, our analysis focuses on the memory overhead incurred during training. Although the number of parameters and FLOPs are identical to those of LoRA, the intermediate latent representation $A_{\text{GraLoRA}}^\top X$ becomes k times larger than the corresponding $A^\top X$ in standard LoRA. This expanded latent space allows for greater information preservation, which can be beneficial. However, it also leads to increased memory consumption during training time. Fortunately, the rank r is typically much smaller than the input and output dimensions, thus the additional memory required remains marginal—even for large k, as demonstrated in Table 1. Moreover, by applying recent techniques such as gradient checkpointing, the memory overhead from the expanded latent space can be effectively hidden, making the impact negligible in practice.

Selection of k While GraLoRA increases the total rank from r to kr, each individual block, represented as $B_{i,j}A_{i,j}^\top \in \mathbb{R}^{\frac{M}{k}\times\frac{N}{k}}$, is constrained to a reduced rank of $\frac{r}{k}$. As a result, increasing k beyond a certain threshold can degrade performance due to limited expressiveness within each block. This effect is especially pronounced when the overall rank r is small. Empirically, we observed that maintaining a minimum block expressiveness of approximately $r/k^2 \approx 8$ yields stable performance across various configurations. Based on this observation, we adopted $k = 2$ for ranks 16 and 32, and $k = 4$ for ranks 64 and 128 in our experiments. Detailed k-sweep results can be found in Section 4.7.

3.5 Hybrid GraLoRA

On the other hand, for smaller ranks—typically rank 16 or below—using $k = 2$ may still lead to performance degradation or yield only marginal gains. To address this limitation, we introduce a hybrid approach that combines the strengths of LoRA and GraLoRA. This method retains the

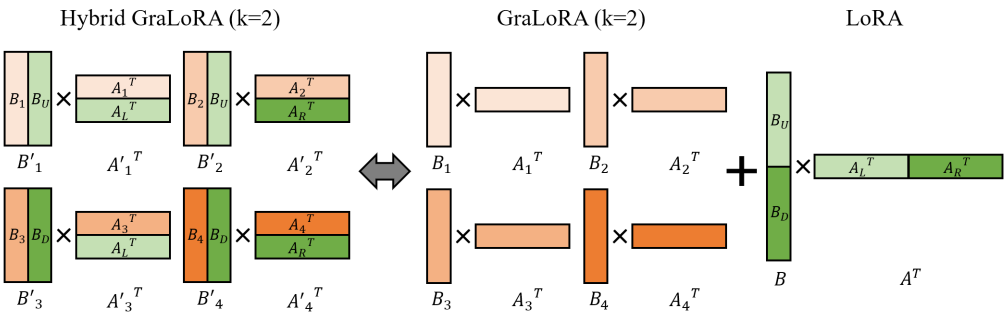


Figure 7: Hybrid GraLoRA architecture when GraLoRA $k = 2$. LoRA parameter becomes shared across small GraLoRA adapters in the same row or same column.

fine-grained input handling and increased total rank offered by GraLoRA, while preserving the expressive power of larger block units through LoRA. Since LoRA shares the same parameters across both rows and columns, it can be naturally integrated with GraLoRA in a concatenated form, which we refer to as *Hybrid GraLoRA* (see Figure 7). Empirically, we found that allocating up to $\frac{1}{2}$ of the total rank to the LoRA component mitigated the limitations of GraLoRA in low-rank scenarios ($\gamma <= 16$), while fully allocating the rank to GraLoRA better performed in high-rank circumstances.

4 Experiments

In order to validate the superiority of the proposed idea, we conduct an extensive analysis on large-scale dataset with the state-of-the art LLMs. We evaluate GraLoRA across five challenging domains: **code generation**, **commonsense reasoning**, **mathematical reasoning**, **general language understanding**, and **personalized image generation**. Our experiments are designed to assess whether the proposed granular adaptation mechanism improves performance across varying model sizes, LoRA ranks, and tasks that require nuanced reasoning and high representational fidelity.

4.1 Experimental Setup

Code Generation. We fine-tuned LLaMA3.1–8B ([9]) with 4 A100 80G GPU on the Magicoder-Evol-Instruct-110k [30] train dataset, a curated and decontaminated subset of WizardCoder [20], comprising high-quality instruction–response pairs for programming tasks. Evaluation was conducted on the Humaneval+ test dataset following He et al. [10], which samples 50 completions per problem using a temperature of 0.2. We report Pass@1, Pass@5, and Pass@10 accuracy following standard protocol via BigCode Evaluation Harness [1].

Commonsense Reasoning We fine-tuned LLaMA3.2–3B, LLaMA3.1–70B, Qwen-2.5-1.5B, and Qwen-2.5-7B ([33]) across 8 commonsense tasks: BoolQ [6], PIQA [3], SIQA [27], HellaSwag [35], WinoGrande [26], ARC-Challenge, ARC-Easy [7], and OpenBookQA [22]. We followed the training pipeline proposed by LLM-Adapters [14]. Training was performed on 2 H100 80G GPUs for 1.5-8B models, and on 8 A100 80G GPUs for the 70B model. LLaMA3.1–70B, Qwen-2.5-1.5B, and Qwen-2.5-7B were trained with rank 64, using the optimal configurations proposed by Biderman et al. [2]. LLaMA3.2–3B was trained with rank 32, following the settings of Ponkshe et al. [25] to ensure a fair comparison with results reported in the original paper.

Mathematical Reasoning We fine-tuned LLaMA3.2–3B on MetaMathQA [34] train dataset using 4 H100 80G GPUs. Evaluation was done on MATH [11] dataset, following the evaluation procedure and settings from He et al. [10].

General Language Understanding We trained and evaluated RoBERTa-base [19], an encoder-only architecture model, on the GLUE [28] benchmark composed of eight sub-tasks. Following the protocol from prior works ([17] [8]), we excluded MNLI and QQP—two time-intensive tasks—which also meant we did not apply the MNLI-based tricks for MRPC, RTE, and STS-B (as used in the original LoRA paper). Accordingly, we retrained LoRA on these tasks without this optimization and report updated results. All trainings were done on a single H100 80G GPU.

Table 2: Pass@1, Pass@5, and Pass@10 results on LLaMA3.1–8B using LoRA, MoRA, RaSA, and GraLoRA across different ranks. Best results per group are in bold. * indicates Hybrid GraLoRA.

Rank	Method	Training Time	Relative Time	Pass@1	Pass@5	Pass@10
16	LoRA	6.2h	1.00×	56.1%	65.3%	68.1%
	MoRA	8.8h	1.42×	53.6%	62.2%	64.5%
	RaSA	6.7h	1.08×	53.7%	64.4%	66.7%
	GraLoRA*	6.7h	1.08×	**58.0%**	**67.1%**	**70.1%**
32	LoRA	6.5h	1.00×	58.4%	**68.0%**	69.9%
	MoRA	9.1h	1.40×	58.3%	66.7%	69.0%
	RaSA	6.8h	1.05×	57.2%	67.9%	**70.5%**
	GraLoRA	6.9h	1.06×	**58.9%**	67.0%	69.0%
64	LoRA	6.7h	1.00×	58.1%	66.4%	68.5%
	MoRA	9.7h	1.45×	57.2%	66.4%	69.2%
	RaSA	6.9h	1.03×	56.6%	65.4%	67.9%
	GraLoRA	7.2h	1.07×	**60.5%**	**71.2%**	**72.6%**
128	LoRA	7.0h	1.00×	55.8%	64.8%	68.6%
	MoRA	9.9h	1.41×	52.8%	62.3%	65.3%
	RaSA	7.6h	1.09×	57.5%	65.5%	67.5%
	GraLoRA	7.7h	1.10×	**64.3%**	**71.7%**	**73.7%**

Personalized Image Generation We fine-tuned SDXL [24] following the official training setup from Huggingface diffusers repository, using the Naruto-Blip-Captions [5] dataset on a single H100 80G GPU. The dataset was split 90% for training and 10% for evaluation. The quality was measured through CLIP similarity and DINOv2 similarity scores.

Training Details We conducted experiments on five open-sourced LLMs—LLaMA3.1–8B, LLaMA3.1–70B, LLaMA3.2–3B, Qwen-2.5-1.5B, and Qwen-2.5-7B—covering diverse architecture and sclaes across code generation, commonsense reasoning, and mathematical reasoning tasks. Following common practice ([16, 17]), we used pre-trained models rather than instruction-tuned models. All PEFT methods were applied to the linear modules in both the attention (W_q, W_k, W_v, W_o)and the feed-forward networks ($W_{up}, W_{down}, W_{gate}$). We adopted alpaca-chat instruction template for training and evaluation. We compared GraLoRA to three representative PEFT methods: LoRA, MoRA [16] and RaSA [10]. We have also handled RoBERTa-base and SDXL, to show the robustness and scalability of our method across differnt models and tasks. Hyperparameters for GraLoRA followed those introduced in Kopiczko et al. [17], except for learning rate, which was reduced by a factor of 5–10, as VeRA uses a learning rate approximately 10 times larger than LoRA. Detailed training parameters can be found in Appendix E.

4.2 Results on Code Generation

As shown in Table 2, GraLoRA outperformed LoRA, MoRA, and RaSA across all tested ranks for Pass@1 accuracy. At rank 64, GraLoRA achieved an absolute improvement of +2.4% in Pass@1, +4.8% in Pass@5, and +4.1% in Pass@10 over LoRA. At rank 128, the gains were even more pronounced, with increases of +8.5% in Pass@1, +6.9% in Pass@5, and +5.1% in Pass@10. Notably, while other methods struggled to fully utilize the increasing rank capacity—often reaching performance plateaus at lower ranks—GraLoRA maintained a consistent upward trajectory, effectively overcoming the limitations of LoRA.

Even in low-rank settings (e.g., rank 16), where expressive capacity is typically constrained, the hybrid variant of GraLoRA demonstrated superior performance. These improvements highlight GraLoRA's enhanced capability to preserve diverse gradient signals and resist suppression from dominant outliers. The strong results on the HumanEval+ benchmark further underscore the benefits of fine-grained adaptation in tackling complex, high-precision code generation tasks.

4.3 Results on Commonsense Reasoning

As shown in Table 3, GraLoRA outperformed other methods across a wide range of models and tasks. Notably, GraLoRA demonstrated superior performance across models of varying scales, achieving a 1.1% improvement in average accuracy on both Qwen2.5-1.5B and LLaMA3.1-70B. It also yielded a

Table 3: Commonsense reasoning accuracy across models and tasks. Bold indicates the best performance per column. HS means HellaSwag, and WG WinoGrande. † indicates values reported by Ponkshe et al. [25]

Model	Method	BoolQ	PIQA	SIQA	HS	WG	ARC-c	ARC-e	OBQA	Avg.
Qwen2.5-1.5B	LoRA	66.5%	84.0%	74.9%	83.6%	73.7%	75.2%	88.1%	83.4%	78.7%
	MoRA	65.9%	82.2%	74.7%	82.6%	73.4%	72.6%	86.5%	82.8%	77.6%
	RaSA	**67.5%**	83.7%	75.7%	85.3%	72.9%	76.4%	89.8%	83.8%	79.4%
	GraLoRA	67.2%	**84.2%**	**75.9%**	**85.7%**	**73.8%**	**77.5%**	**89.9%**	**84.4%**	**79.8%**
Qwen2.5-7B	LoRA†	72.3%	88.2%	**79.2%**	92.9%	**84.7%**	84.0%	93.6%	89.6%	85.6%
	MoRA	69.9%	85.3%	78.5%	83.7%	81.4%	77.5%	88.6%	85.0%	81.2%
	RaSA	72.0%	88.5%	78.9%	**93.6%**	81.8%	86.1%	94.2%	90.2%	85.7%
	GraLoRA	**73.4%**	**89.7%**	79.0%	93.0%	84.0%	**86.9%**	**94.5%**	**90.6%**	**86.4%**
LLaMA3.2–3B	LoRA	70.0%	85.2%	79.1%	90.7%	82.2%	74.3%	86.9%	81.9%	81.3%
	MoRA	72.4%	86.1%	80.1%	92.3%	84.8%	76.8%	88.8%	**84.8%**	83.3%
	RaSA	73.1%	**87.5%**	**81.1%**	93.7%	85.3%	78.9%	88.9%	83.6%	84.0%
	GraLoRA	**74.1%**	86.5%	80.8%	**93.8%**	**87.5%**	**79.9%**	**89.5%**	**84.8%**	**84.6%**
LLaMA3.1–70B	LoRA	81.7%	93.4%	82.2%	97.5%	93.1%	90.2%	96.5%	95.6%	91.3%
	GraLoRA	**83.1%**	**94.7%**	**83.6%**	**97.9%**	**93.8%**	**92.3%**	**97.8%**	**96.2%**	**92.4%**

Table 4: MATH dataset accuracy results on Qwen2.5-1.5B using LoRA and GraLoRA across different ranks. Best results per group are in bold.

Rank	Method	Training Time	Relative Time	Accuracy
64	LoRA	5.3h	1.00×	23.6%
	GraLoRA	6.2h	1.17×	25.7%
128	LoRA	5.5h	1.00×	24.7%
	GraLoRA	6.6h	1.20×	28.9%

0.9% gain on the widely used mid-sized model, Qwen2.5-7B. Moreover, GraLoRA achieved a 3.3% improvement on LLaMA3.2-3B, surpassing a broad range of baselines as presented in Table 6.

Furthermore, GraLoRA achieved the best results on 26 out of 32 tasks, consistently outperforming alternatives across benchmarks. These results support our analysis in Section 3.3, showing that GraLoRA's localized updates enhance alignment with FFT and promote robust generalization in multi-aspect reasoning tasks.

4.4 Results on Mathematical Reasoning

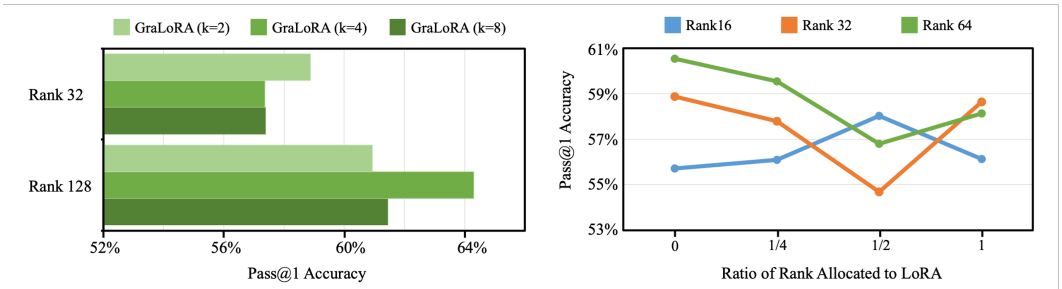
In mathematical reasoning task, regarded as one of the most challenging benchmarks, GraLoRA consistently outperformed LoRA across all configurations. Notably, in the high rank setting of $r = 128$, GraLoRA achieved a 4.2% improvement in accuracy (Table 4), mirroring the performance trends observed in the code generation experiments. These results further highlight the robustness of GraLoRA, demonstrating its capability to fully exploit the advantages enabled by increased rank capacity, thereby overcoming the inherent expressiveness constraints of previous PEFT methods.

4.5 Results on General Language Understanding

GraLoRA demonstrates strong performance even in the low-rank regime, outperforming all baselines in terms of average score. The Hybrid GraLoRA variant achieves the most robust results, attaining the best performance on four out of six tasks, while both the original and hybrid versions consistently surpass all other baselines, as shown in Table 7. Compared with LoRA, the best GraLoRA configuration yields a 1.8% improvement in average accuracy, with gains observed across all sub-tasks. These findings indicate that GraLoRA maintains high effectiveness even under constrained parameter budgets and generalizes well to non-LLM architectures.

4.6 Results on Personalized Image Generation

In the image generation task, GraLoRA consistently outperformed LoRA in both CLIP and DINOv2 similarity metrics, achieving 0.5% and 2.1% improvements, respectively (Table 8). These results



(a) GraLoRA k Sweep Result (b)Hybrid GraLoRA Ratio Sweep Result

Figure 8: (a) GraLoRA k sweep results and (b) Hybrid GraLoRA Ratio sweep results for LLaMA3.1–8B on code generation task. Ratio 0 implies default GraLoRA and ratio 1 vanilla LoRA in (b).

further demonstrate the generality and effectiveness of GraLoRA beyond language models, extending its applicability to vision–language and generative architectures such as diffusion models.

4.7 Ablation Study

GraLoRA k Sweep We evaluated the impact of varying k on code generation accuracy. As shown in Figure 8 (a), $k = 2$ yielded the best performance at rank 32, while $k = 4$ was optimal at rank 128. These results are consistent with the theoretical prediction that a smaller k is preferable for lower ranks, as reduced sub-block rank can be particularly detrimental when the overall rank is limited.

Hybrid GraLoRA Ratio Sweep We assessed performance across different LoRA-to-GraLoRA rank allocation ratios for the Hybrid GraLoRA configuration (Figure 8 (b)). At rank 16, partially allocating the rank to LoRA led optimal accuracy. However, for larger ranks, allocating rank to LoRA resulted in degraded performance. This suggests that Hybrid GraLoRA is advantageous in low-rank regimes, where the sub-block rank of GraLoRA alone may be insufficient. In contrast, under higher-rank settings where GraLoRA's sub-blocks are expressive enough, introducing LoRA components may lead to gradient entanglement, thereby hindering effective learning.

5 Conclusion

In this work, we introduced **GraLoRA**, a novel PEFT method that extends LoRA with granular, block-wise decomposition. Motivated by a rigorous analysis of LoRA's gradient behavior, we identified that input outliers can dominate the low-rank update, suppressing meaningful contributions from other input channels and misaligning with the localized gradient propagation observed in FFT.

GraLoRA addresses this limitation by dividing the adaptation space into k^2 independently trained low-rank adapters, enabling spatially localized and context-aware updates. Our theoretical analysis shows that this design increases expressivity by a factor of k, without additional parameters or computational cost. Moreover, under outlier activations, GraLoRA effectively mitigates the global gradient distortion seen in vanilla LoRA and better preserves inter-channel balance. Empirically, GraLoRA consistently outperforms standard LoRA and strong baselines such as RaSA across diverse tasks and model scales. On the code generation benchmark HumanEval+, it achieves up to +8.5% absolute gain in Pass1. GraLoRA also delivers significant improvements across other 4 additional tasks, highlighting its robustness and scalability across heterogeneous architectures and model sizes.

Future Work. While GraLoRA improves gradient locality and expressive power, its current design assumes uniform partitioning. Future extensions may explore adaptive or learned partitioning schemes, sparsity-aware block activation, or task-driven dynamic rank allocation. Additionally, applying GraLoRA to vision transformers, multimodal architectures, or continual learning setups may further highlight its potential for robust and efficient model adaptation.

Overall, GraLoRA represents a principled and practical step forward in the design of PEFT methods, bridging the gap between global low-rank reparameterization and local, fine-grained adaptation.

Acknowledgments and Disclosure of Funding

This work was partly supported by the Technology development Program of MSS [RS-2023-00258531], the Starting growth Technological R&D Program of MSS [S3358777], and the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2025-02304183, Development of Optimization Code Conversion Technology for Heterogeneous Al Semiconductor-Based Large-Scale Models).

References

[1] Loubna Ben Allal, Niklas Muennighoff, Logesh Kumar Umapathi, Ben Lipkin, and Leandro von Werra. A framework for the evaluation of code generation models. `https://github.com/bigcode-project/bigcode-evaluation-harness`, 2022.

[2] Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. LoRA learns less and forgets less. *Transactions on Machine Learning Research*, 2024.

[3] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[4] Kerim Büyükakyüz. Olora: Orthonormal low-rank adaptation of large language models. *arXiv preprint arXiv:2406.01775*, 2024.

[5] Eole Cervenka. Naruto blip captions. `https://huggingface.co/datasets/lambdalabs/naruto-blip-captions/`, 2022.

[6] 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 Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[7] 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. *arXiv preprint arXiv:1803.05457*, 2018.

[8] Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform, 2024.

[9] Aaron Grattafiori et al. The llama 3 herd of models, 2024.

[10] Zhiwei He, Zhaopeng Tu, Xing Wang, Xingyu Chen, Zhijie Wang, Jiahao Xu, Tian Liang, Wenxiang Jiao, Zhuosheng Zhang, and Rui Wang. RaSA: Rank-sharing low-rank adaptation. In *Proceedings of the 2025 International Conference on Learning Representations (ICLR)*, 2025.

[11] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021.

[12] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019.

[13] Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[14] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy 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*, pages 5254–5276, 2023.

[15] Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. HiRA: Parameter-efficient hadamard high-rank adaptation for large language models. In *Proceedings of the 2025 International Conference on Learning Representations (ICLR)*, 2025.

[16] Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. MoRA: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*, 2024.

[17] Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. VeRA: Vector-based random matrix adaptation. In *Proceedings of the 2024 International Conference on Learning Representations (ICLR)*, 2024.

[18] Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13355–13364, 2024.

[19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[20] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. In *The Twelfth International Conference on Learning Representations*, 2024.

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

[22] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018.

[23] Fabian Paischer, Lukas Hauzenberger, Thomas Schmied, Benedikt Alkin, Marc Peter Deisenroth, and Sepp Hochreiter. One initialization to rule them all: Fine-tuning via explained variance adaptation. *arXiv preprint arXiv:2410.07170*, 2024.

[24] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023.

[25] Kaustubh Ponkshe, Raghav Singhal, Eduard Gorbunov, Alexey Tumanov, Samuel Horvath, and Praneeth Vepakomma. Initialization using update approximation is a silver bullet for extremely efficient low-rank fine-tuning, 2025.

[26] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[27] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

[28] 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, 2019.

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

[30] Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with OSS-instruct. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 52632–52657. PMLR, 21–27 Jul 2024.

[31] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2024.

[32] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023.

[33] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[34] 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. *arXiv preprint arXiv:2309.12284*, 2023.

[35] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

A Rank Analysis in Real-World Scenarios

Table 5: Average rank size in each projection layer across LoRA and GraLoRA variants. Rank r was set to 128 in all methods.

	q_proj	k_proj	v_proj	o_proj	up_proj	down_proj	gate_proj
LoRA	128	128	128	128	128	128	128
GraLoRA (k=2)	256	256	256	256	256	256	256
GraLoRA (k=4)	512	512	512	512	512	512	512
GraLoRA (k=8)	1024	1016	1022	1024	1024	1024	1024

As shown in Table 5, GraLoRA denoted linearly increasing ranks as the k increased. The observation aligns with our theoretical analysis that increasing GraLoRA k leads to higher expression power by increasing the latent space from r to kr.

B Gradient Distribution of LoRA and GraLoRA

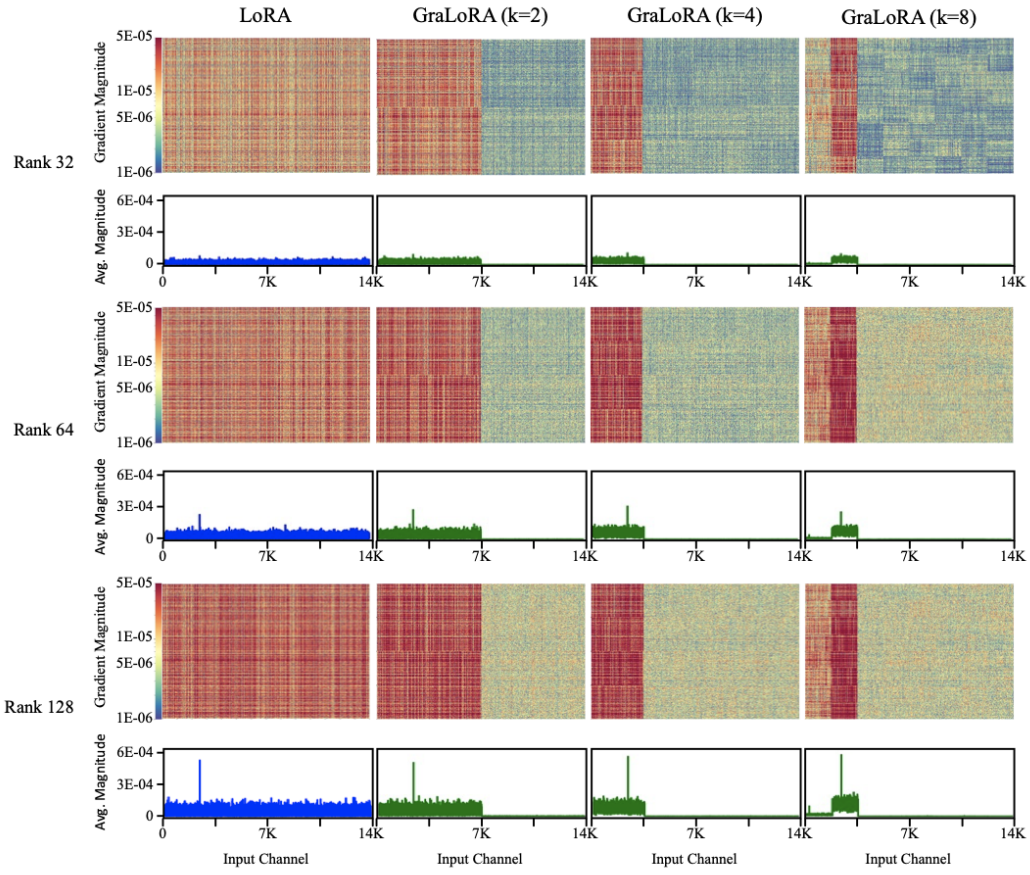


Figure 9: Comparison of gradient distributions under outlier activation for rank 32, 64, and 128 in LLaMA3.1-8B Layer 1 down-projection matrix.

Figure 9 displays gradient distributions of LoRA and GraLoRA for varying ranks. In GraLoRA, only the blocks interacting with the outlier exhibit elevated gradients, structurally solving the gradient entanglement discovered in vanilla LoRA. This enables to mitigate global distortion and align with FFT behavior in all ranks.

C Precise Analysis on Computation Overhead

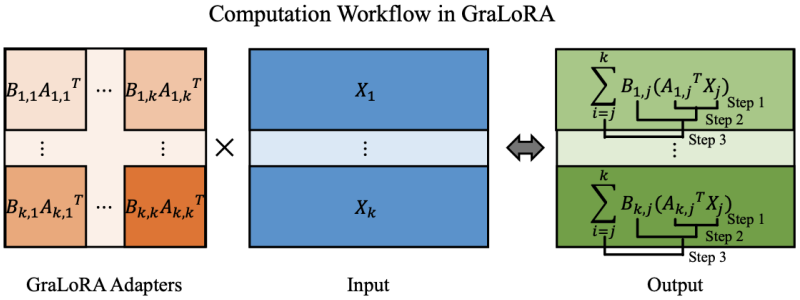


Figure 10: Computation workflow in GraLoRA is composed of 3 steps: two sub-block matrix multiplications and a following matrix addition.

In the previous *"Computation Overhead Analysis"* section 3.4 we compared the computation of LoRA and GraLoRA with the big O notation on the two major matrix multiplication steps. In this section we further examine the exact computation requirement and compare their efficiency.

LoRA FLOPs LoRA performs the projection in two sequential steps to leverage its low-rank structure. In the first step, the computation of $A^\top X \in \mathbb{R}^{r\times T}$ requires $(2N-1)rT$ FLOPs. In the second step, the reconstruction $B(A^\top X) \in \mathbb{R}^{M\times T}$ incurs $(2r-1)MT$ FLOPs. Therefore, the total FLOPs for LoRA is:

$$\begin{aligned}\text{LoRA}_{\text{FLOPs}} &= (2N-1)rT + (2r-1)MT \\ &= 2r(M+N)T - (r+M)T.\end{aligned}$$

GraLoRA FLOPs In practice, GraLoRA computations can be divided into three stages, involving k^2 adapter blocks: two matrix multiplications followed by a matrix addition as shown in Figure 10.

In the first stage (projection), each adapter block computes $A_{i,j}^\top X_j \in \mathbb{R}^{\frac{r}{k}\times T}$, which requires $\left(2\frac{n}{k}-1\right)\frac{r}{k}T$ FLOPs. Since there are k^2 such blocks, the total FLOPs for this step is $(2n-k)rT$.

In the second stage (reconstruction), each adapter block performs $B_{i,j}(A_{i,j}^\top X_j) \in \mathbb{R}^{\frac{m}{k}\times T}$, which costs $\left(2\frac{r}{k}-1\right)\frac{m}{k}T$ FLOPs. With k^2 blocks, the total becomes $(2r-k)mT$.

The final stage involves aggregating the outputs across k projections for each row:

$$\sum_{j=1}^{k} B_{i,j}(A_{i,j}^\top X_j) \in \mathbb{R}^{\frac{m}{k}\times T},$$

which requires $\left(\frac{m}{k}\times T\right)(k-1) = \frac{mT(k-1)}{k}$ FLOPs per row. Across k rows, the total cost becomes $(k-1)mT$.

Combining all three stages, the total FLOPs for GraLoRA is:

$$\begin{aligned}\text{GraLoRA}_{\text{FLOPs}} &= (2n-k)rT + (2r-k)mT + (k-1)mT \\ &= 2r(m+n)T - k(r+m)T + (k-1)mT \\ &= 2r(m+n)T - krT - mT.\end{aligned}$$

This can also be expressed as:

$$\text{GraLoRA}_{\text{FLOPs}} = \text{LoRA}_{\text{FLOPs}} - (k-1)rT,$$

demonstrating that GraLoRA introduces reduced computation compared to LoRA.

D Additional Experiment Results

D.1 Results on Commonsense Reasoning with extensive baseline comparison

Table 6: Commonsense reasoning accuracy across models and tasks. All values are percentages; bold indicates the best performance per row. HS means HellaSwag, and WG WinoGrande. † indicates values reported by Ponkshe et al. [25]

Method	Rank	#Parmas	BoolQ	PIQA	SIQA	HS	WG	ARC-c	ARC-e	OBQA	Avg.
Full-FT†	-	3.21B	70.4%	85.6%	80.5%	91.9%	85.0%	75.3%	88.5%	81.9%	82.4%
LoRA-XS†	96	1.81M	67.3%	83.4%	78.7%	89.0%	82.1%	72.6%	85.2%	78.9%	79.6%
LoRA-SB†	96	1.81M	70.3%	84.8%	80.2%	91.6%	84.6%	74.7%	87.9%	81.2%	81.9%
LoRA†	32	48.63M	70.0%	85.2%	79.1%	90.7%	82.2%	74.3%	86.9%	81.9%	81.3%
MELoRA	32	48.63M	71.3%	85.0%	78.6%	93.0%	79.7%	73.7%	85.5%	79.0%	80.7%
rsLoRA†	32	48.63M	69.8%	85.1%	78.9%	90.5%	82.0%	74.2%	86.7%	81.7%	81.1%
PiSSA†	32	48.63M	70.1%	85.4%	79.4%	90.9%	82.7%	74.6%	87.2%	81.8%	81.5%
DoRA†	32	49.40M	70.4%	85.6%	79.7%	90.8%	82.9%	74.9%	87.6%	82.0%	81.7%
BOFT	32	48.48M	72.3%	84.6%	79.1%	91.3%	84.5%	73.7%	87.8%	80.6%	81.7%
LoRA-Pro†	32	48.63M	71.3%	85.8%	79.4%	90.9%	83.4%	75.3%	87.2%	81.7%	81.9%
MoRA	32	48.63M	72.4%	86.1%	80.1%	92.3%	84.8%	76.8%	88.8%	**84.8%**	83.3%
RaSA	32	48.63M	73.1%	**87.5%**	**81.1%**	93.7%	85.3%	78.9%	88.9%	83.6%	84.0%
GraLoRA	32	48.63M	**74.1%**	86.5%	80.8%	**93.8%**	**87.5%**	**79.9%**	**89.5%**	**84.8%**	**84.6%**

D.2 Results on General Language Understanding (GLUE)

Table 7: GLUE dataset accuracy results on RoBERTa-base across different tasks. Best results per task are in bold.

Method	#Params	SST-2 (%)	MRPC (%)	CoLA (%)	QNLI (%)	RTE (%)	STS-B (%)	Avg (%)
Full-FT	125M	94.8	90.2	63.6	92.8	78.7	**91.2**	85.2
LoRA	0.3M	95.1	86.5	63.4	93.3	76.2	90.6	84.2
VeRA	0.043M	94.6	89.5	65.6	91.8	78.7	90.7	85.2
FourierFT	0.024M	94.2	90.0	63.8	92.2	79.1	90.8	85.0
GraLoRA	0.3M	**95.2**	89.7	**65.3**	93.0	**80.9**	91.1	85.8
Hybrid GraLoRA	0.3M	**95.2**	**90.2**	64.1	**93.4**	79.8	**91.2**	85.6
Best GraLoRA	0.3M	**95.2**	**90.2**	**65.3**	**93.4**	**80.9**	**91.2**	**86.0**

D.3 Results on Personalized Image Generation

Table 8: SDXL fine-tuning results personalized image generation.

Method	CLIP Similarity	DINOv2 Similarity
LoRA	91.4%	79.2%
GraLoRA	**91.9%**	**81.3%**

E Experiment Details

Baseline Methods. We compared GraLoRA with three main baseline methods. Key idea for each method is as follows:

- **LoRA** freezes pretrained model weights and injects trainable low-rank matrices into selected layers, allowing efficient fine-tuning with significantly fewer parameters, approximating weight updates as a product of two small matrices.
- **MoRA** employs a single square matrix instead of low-rank matrices to achieve high-rank updating while maintaining the same number of trainable parameters.
- **RaSA** enhances LoRA by sharing partial low-rank components across layers while keeping layer-specific updates.

Table 9: Hyperparameters for Code Generation, Commonsense Reasoning, Mathematical Reasoning, and Personalized Image Generation tasks.

Task	Model	Method	Rank	LR	Batch size	Epochs	Optimizer
Code Generation	LLaMA3.1–8B	LoRA MoRA RaSA GraLoRA	{16, 32, 64, 128}	2e-4	192	2	LionW
Commonsense Reasoning	Qwen-2.5-1.5B	LoRA MoRA RaSA GraLoRA	64	2e-4	192	2	LionW
	Qwen-2.5-7B	LoRA MoRA RaSA GraLoRA	64	4e-4	192	2	LionW
	LLaMA3.2–3B	BOFT MeLORA MoRA RaSA GraLoRA	32	4e-4 4e-4 2e-4 4e-4 4e-4	192	2	AdamW
	LLaMA3.1–70B	LoRA GraLoRA	64	3e-4	192	1	LionW
Mathematical Reasoning	Qwen-2.5-1.5B	LoRA GraLoRA	64	2e-4	192	4	AdamW
		LoRA GraLoRA	128	4e-4	192	4	AdamW
Personalized Image Generation	SDXL	LoRA GraLoRA	128	1e-4	1	2	AdamW

Table 10: Detailed hyperparameter settings for each sub-tasks in General Language Understanding.

Model	Task	Method	Rank	LR	Head-LR	Batch size	Epochs	Optimizer
RoBERTa-base	SST-2	GraLoRA Hybrid GraLoRA	8	4e-4	4e-3 4e-4	128	60	AdamW
	MRPC	GraLoRA Hybrid GraLoRA	8	4e-4	4e-4 2e-4	128	30	AdamW
	CoLA	GraLoRA Hybrid GraLoRA	8	5e-4 8e-4	5e-3 8e-4	128 256	80	AdamW
	QNLI	GraLoRA Hybrid GraLoRA	8	5e-4	2e-3	128	25	AdamW
	RTE	GraLoRA Hybrid GraLoRA	8	2e-4	2e-4	128	160	AdamW
	STS-B	GraLoRA Hybrid GraLoRA	8	1e-3	1e-2	128	80	AdamW

We fixed LoRA $\alpha = 2r$ which is known to be generally applicable in different models with different ranks [2]. Detailed hyperparameter settings for our experiments are denoted in Table 9.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, we strongly denote our main claims in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: Yes, we discuss the limitations of our method and evaluate it's impact in the "Tradeoff Analysis" section. We further examine how to overcome the limitation in "Hybrid GraLoRA" section and it's practical result in "Results on Code Generation" section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Yes, we do provide complete and correct proof for each theoretical result in the "Expression Power Analysis" section.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

Justification: Yes, we do disclose information needed to reproduce the results in "Experiment Setup" section and in Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Yes, we do provide open access to code with sufficient instructions as supplemental material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, we specify all the training and test details necessary to understand the results in "Experimental Setup" section and in Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: No, error bars are not reported because it would be too computationally expensive.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

Justification: Yes, we do provide sufficient information on the computer resources in "Tradeoff Analysis" section and in "Experiments" section.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, our research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We propose a generic method for optimizing efficient training of neural networks. It is not directly related to social impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our method solely works for fine-tuning a pretrained model, thus the paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Yes, the assets used in the paper are properly cited in "Experiments" section and in overall writings.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We do not introduce new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: Our core method development does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.