LoRA-MGPO: Mitigating Double Descent in Low-Rank Adaptation via Momentum-Guided Perturbation Optimization

Yupeng Chang[1] **Chenlu Guo**[1] **Yi Chang**[1,2,3] **Yuan Wu**[1*]
[1]School of Artificial Intelligence, Jilin University
[2]Engineering Research Center of Knowledge-Driven Human-Machine Intelligence, MOE, China
[3]International Center of Future Science, Jilin University
{changyp23, guocl23}@mails.jlu.edu.cn, {yichang, yuanwu}@jlu.edu.cn

Abstract

Parameter-efficient fine-tuning (PEFT), particularly Low-Rank Adaptation (LoRA), adapts large language models (LLMs) by training only a small fraction of parameters. However, as the rank of the low-rank matrices used for adaptation increases, LoRA often exhibits an unstable "double descent" phenomenon, characterized by transient divergence in the training loss, which delays convergence and impairs generalization by causing instability due to the attraction to sharp local minima. To address this, we introduce **LoRA-MGPO**, a framework that incorporates Momentum-Guided Perturbation Optimization (MGPO). MGPO stabilizes training dynamics by mitigating the double descent phenomenon and guiding weight perturbations using momentum vectors from the optimizer's state, thus avoiding dual gradient computations. Additionally, an adaptive normalization scheme scales the magnitude of perturbations based on an exponential moving average (EMA) of gradient norms, further enhancing stability. While EMA controls the magnitude of the perturbations, MGPO guides their direction, ensuring a more stable optimization trajectory. Experiments on a suite of natural language understanding and generation benchmarks show that LoRA-MGPO consistently achieves superior performance over LoRA and other PEFT methods. The analysis indicates that LoRA-MGPO leads to smoother loss curves, faster convergence, and improved generalization by stabilizing the training process and mitigating the attraction to sharp minima. The code is publicly available at https://github.com/llm172/LoRA-MGPO.

1 Introduction

Large language models (LLMs) have driven significant advancements in natural language processing, establishing new performance benchmarks on tasks ranging from text generation to semantic understanding (Chang et al., 2024b; Wei et al., 2022). However, the conventional method of full-parameter fine-tuning (Full FT) requires updating billions of parameters, incurring prohibitive memory and computational costs. To overcome this limitation, parameter-efficient fine-tuning (PEFT) methods have emerged as an effective alternative, enabling efficient adaptation by optimizing only a small subset of model parameters (Lester et al., 2021; Fu et al., 2023).

Among these methods, Low-Rank Adaptation (LoRA) (Hu et al., 2021) is distinguished by its computational efficiency and architectural simplicity. LoRA approximates the weight update matrix ΔW as a low-rank decomposition, where the original pre-trained weights W_0 remain frozen. The trainable matrices B and A, with rank $r \ll \min(m, n)$, drastically reduce the number of trainable parameters, improving efficiency without altering the model architecture.

Despite its efficiency, LoRA's training dynamics can be unstable. As shown in Figure 1, fine-tuning LLaMA-2-7B (Touvron et al., 2023) on MetaMathQA (Yu et al., 2024) often exhibits a "double descent" trajectory with initial convergence, transient divergence, and eventual stabilization. This phenomenon worsens with higher ranks and is not unique to LoRA; Full FT can exhibit even more severe double descent, highlighting the general challenge of stabilizing fine-tuning in high-capacity models (Nakkiran et al., 2019). Such non-monotonic behavior delays convergence and impairs generalization due to unstable gradients and the attraction to sharp local minima (Li et al., 2024a).

Addressing these stability issues is crucial. Sharpness-Aware Minimization (SAM) (Foret et al., 2020) improves generalization by seeking flatter minima. However, its application is hindered by the dual gradient computation require-

*Corresponding authors

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 648–659
November 4-9, 2025

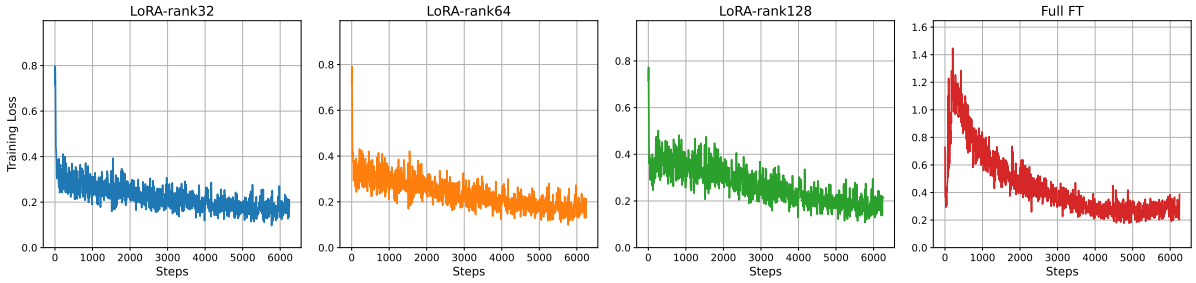

Figure 1: Training loss curves of Full FT and LoRA (Hu et al., 2021) methods with LLaMA-2-7B (Touvron et al., 2023) on the MetaMathQA dataset (Yu et al., 2024). For LoRA, rank (r) and alpha (α) are set to the same values ($r = \alpha \in \{32, 64, 128\}$), with a fixed learning rate of $5e-4$.

ment, which doubles the training cost (Becker et al., 2024; Li et al., 2024b). More efficient variants like momentum-guided SAM reuse optimizer states to avoid this overhead but may not guarantee stable convergence. To further enhance stability, complementary techniques such as applying an exponential moving average (EMA) to smooth optimization dynamics have been shown to suppress parameter oscillations and improve convergence in certain scenarios (Wang et al., 2021).

Building on these insights, we propose **LoRA-MGPO**, a novel framework that integrates Momentum-Guided Perturbation Optimization (MGPO) into LoRA to mitigate the detrimental effects of double descent. Our contributions are twofold:

1. **Mitigating Double Descent:** MGPO stabilizes training by addressing double descent, typically observed at higher ranks in LoRA. By reusing momentum vectors, it guides weight perturbations towards flatter minima, preventing transient divergences in loss.

2. **Adaptive Perturbation Normalization:** MGPO introduces an adaptive scheme that scales perturbation magnitude based on an exponential moving average (EMA) of gradient norms, decoupling perturbation intensity from optimization dynamics and further enhancing stability.

We evaluate LoRA-MGPO on a suite of natural language understanding (NLU) and generation (NLG) benchmarks. Our results show that it consistently achieves superior performance over standard LoRA and other state-of-the-art PEFT methods. Crucially, we demonstrate that LoRA-MGPO effectively mitigates the double descent phenomenon, leading to more stable training dynamics, smoother loss curves, and faster convergence, all of which contribute to better generalization and the avoidance of sharp minima.

2 Method

In this section, we first provide a concise overview of the Low-Rank Adaptation (LoRA) framework. We then introduce **LoRA-MGPO**, an extension of LoRA that integrates Momentum-Guided Perturbation Optimization (MGPO) to enhance its stability and efficiency. We describe how MGPO reuses optimizer momentum for guided perturbations of the trainable parameters and incorporates an adaptive normalization scheme to stabilize training.

2.1 Review of LoRA

While full fine-tuning directly updates the entire pre-trained weight matrix $W_0 \in \mathbb{R}^{m\times n}$, its prohibitive computational cost makes it impractical for large-scale models. Low-Rank Adaptation (LoRA) (Hu et al., 2021) offers a parameter-efficient alternative. LoRA freezes W_0 and injects a trainable low-rank decomposition, $\Delta W = BA$, where $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$ are trainable matrices with rank $r \ll \min(m, n)$. The weight update is incorporated into the forward pass as:

$$Y = X(W_0 + \frac{\alpha}{r}BA), \tag{1}$$

where X is the input, α is a scaling hyperparameter, and r is the rank of the decomposition. Typically, A is initialized with a Kaiming normal distribution, and B with zeros. While effective, LoRA can suffer from training instability, particularly the double descent phenomenon, when r increases without appropriate optimization strategies to maintain stability (Li et al., 2024a).

2.2 LoRA with Momentum-Guided Perturbation Optimization

To address the training instabilities in LoRA, we propose **LoRA-MGPO**, which integrates Momentum-Guided Perturbation Optimization (MGPO). Inspired by Sharpness-Aware Minimization (SAM), MGPO is redesigned for computational efficiency and parameter efficiency. It directly perturbs the trainable LoRA parameters by reusing the optimizer's first-moment estimate, guiding the perturbations toward stable directions. Additionally, MGPO incorporates adaptive normalization to dynamically scale the perturbation, enhancing training stability.

2.2.1 Motivation: SAM for LoRA and Its Limitations

The goal of SAM (Foret et al., 2020) is to find parameters in flat loss regions to improve generalization. A direct application to LoRA would involve perturbing the full weight matrix, solving $\min_{A,B} \max_{\|\epsilon\|_F \leq \rho} \mathcal{L}(W_0 + BA + \epsilon)$. This approach is ill-suited for PEFT due to two critical flaws: (1) its dual gradient computation requirement doubles the training cost, and (2) creating and storing the full-space perturbation ϵ counteracts the memory savings of LoRA. MGPO is explicitly designed to resolve these inefficiencies.

2.2.2 Momentum-Guided Perturbation of LoRA Parameters

MGPO achieves the stability benefits of SAM by perturbing the trainable parameters $\theta = (A, B)$ directly, using information readily available in the optimizer's state. At each training step t, instead of computing a new gradient for the perturbation direction, it reuses the optimizer's first-moment vector (momentum) from the previous step, $\boldsymbol{m}_{t-1}$. The optimization objective is:

$$\min_{\theta} \mathcal{L}(\theta_t + \epsilon_{\theta_t}), \tag{2}$$

where the perturbation ϵ_{θ_t} applied to the LoRA parameters $\theta_t = (A_t, B_t)$ is constructed using the state from step $t-1$:

$$\epsilon_{\theta_t} = \rho \cdot \frac{\boldsymbol{m}_{t-1}}{\|\boldsymbol{m}_{t-1}\|_2} \cdot \frac{1}{\bar{g}^{(t-1)}}. \tag{3}$$

Here, ρ is the perturbation radius. Using the historical momentum vector is a deliberate design choice, as it represents a smoothed average of past gradients, filtering out the noise from any single mini-batch and providing a more stable direction for assessing landscape sharpness. This vector is maintained by the optimizer itself. After computing the gradient on the perturbed parameters, the momentum for the current step is updated as:

$$\boldsymbol{m}_t = \mu \boldsymbol{m}_{t-1} + \nabla_{\tilde{\theta}_t} \mathcal{L}. \tag{4}$$

The decay factor μ (e.g., 'beta1' in AdamW) is reused from the optimizer's standard settings. The scalar $\bar{g}^{(t-1)}$ is a global normalization factor, detailed next. This formulation entirely avoids the second gradient computation and any operations in the full weight space.

Two-Stage Update Mechanism MGPO is implemented efficiently within each training step t. First, using the state from step $t-1$, we compute the perturbation ϵ_{θ_t} and apply it to the current parameters θ_t to get a perturbed version, $\tilde{\theta}_t$:

$$\tilde{\theta}_t = \theta_t + \epsilon_{\theta_t}. \tag{5}$$

Second, the loss and its gradient are computed with respect to these perturbed parameters: $\nabla_{\tilde{\theta}_t} \mathcal{L}$. This single gradient is then used by the optimizer to update both the original parameters from θ_t to θ_{t+1} and the momentum from $\boldsymbol{m}_{t-1}$ to $\boldsymbol{m}_t$. For inference, the final, unperturbed parameters θ_T are used.

2.2.3 Adaptive Perturbation Normalization

To ensure robustness across training stages, we introduce an Adaptive Perturbation Normalization (APN) scheme. The normalization factor $\bar{g}^{(t)}$ used in Equation 3 is a scalar computed via an exponential moving average (EMA) of the global L2-norm of the LoRA parameter gradients. Following the principle of using the actually computed gradient, the update rule is:

$$\bar{g}^{(t)} = \beta \bar{g}^{(t-1)} + (1-\beta) \|\nabla_{\tilde{\theta}_t} \mathcal{L}\|_2, \tag{6}$$

where β is the EMA decay rate. This mechanism makes the perturbation scale-invariant relative to the gradient dynamics. For instance, during early training with large gradients, the normalization factor increases, reducing the effective perturbation size to prevent destabilization. Conversely, in later stages, it ensures the perturbation remains sufficiently large to be effective. This adaptive scaling enhances training stability.

3 Experiments

3.1 Experimental Setup

Baselines To provide a comprehensive evaluation, we compare LoRA-MGPO against a carefully

Table 1: Performance of T5-Base on five GLUE tasks, comparing LoRA-MGPO with full fine-tuning and other LoRA variants (rank $r = 8$). Scores are reported for the primary metric of each task, averaged over 3 runs, with standard deviations shown in subscripts. **Bold** indicates the best score, while underlining denotes the second best.

Method	**MNLI**	**SST2**	**CoLA**	**QNLI**	**MRPC**	**Avg**
Train Size	*393k*	*67k*	*8.5k*	*105k*	*3.7k*	
Full FT	$86.33_{\pm 0.00}$	$\mathbf{94.75}_{\pm 0.21}$	$80.70_{\pm 0.24}$	$93.19_{\pm 0.22}$	$84.56_{\pm 0.73}$	87.91
LoRA	$85.30_{\pm 0.04}$	$94.04_{\pm 0.11}$	$69.35_{\pm 0.05}$	$92.96_{\pm 0.09}$	$68.38_{\pm 0.01}$	82.08
LoRA Variants with Modified Structure						
DoRA	$85.67_{\pm 0.09}$	$94.04_{\pm 0.53}$	$72.04_{\pm 0.94}$	$93.04_{\pm 0.06}$	$68.08_{\pm 0.51}$	82.57
AdaLoRA	$85.45_{\pm 0.11}$	$93.69_{\pm 0.20}$	$69.16_{\pm 0.24}$	$91.66_{\pm 0.05}$	$68.14_{\pm 0.28}$	81.62
LoRA Variants with Original Structure						
PiSSA	$85.75_{\pm 0.07}$	$94.07_{\pm 0.06}$	$74.27_{\pm 0.39}$	$93.15_{\pm 0.14}$	$76.31_{\pm 0.51}$	84.71
rsLoRA	$85.73_{\pm 0.10}$	$94.19_{\pm 0.23}$	$72.32_{\pm 1.12}$	$93.12_{\pm 0.09}$	$52.86_{\pm 2.27}$	79.64
LoRA+	$85.81_{\pm 0.09}$	$93.85_{\pm 0.24}$	$77.53_{\pm 0.20}$	$93.14_{\pm 0.03}$	$74.43_{\pm 1.39}$	84.95
LoRA-GA	$85.70_{\pm 0.09}$	$94.11_{\pm 0.18}$	$80.57_{\pm 0.20}$	$93.18_{\pm 0.06}$	$85.29_{\pm 0.24}$	87.77
LoRA-MGPO	$\mathbf{86.58}_{\pm 0.11}$	$94.72_{\pm 0.46}$	$\mathbf{82.32}_{\pm 0.18}$	$\mathbf{93.79}_{\pm 0.46}$	$\mathbf{86.62}_{\pm 0.68}$	**88.81**

selected set of baselines. These include Full Fine-Tuning (Full FT), serving as a strong performance benchmark, and vanilla LoRA (Hu et al., 2021), our primary point of comparison. We further include two categories of state-of-the-art LoRA variants. The first category, *variants with architectural modifications*, comprises methods that alter the LoRA structure itself, such as DoRA (Liu et al., 2024), which introduces learnable magnitude vectors, and AdaLoRA (Zhang et al., 2023), which dynamically allocates rank budgets. The second category, *variants improving the training process or initialization*, includes rsLoRA (Kalajdzievski, 2023), which stabilizes update magnitudes; LoRA+ (Hayou et al., 2024), which employs different learning rates for the LoRA matrices; and PiSSA (Meng et al., 2024), which refines initialization using SVD. Finally, we compare against methods focused on *gradient alignment*, such as LoRA-GA (Wang et al., 2024a) and LoRA-Pro (Wang et al., 2024b), which aim to align LoRA's gradient updates more closely with those of full fine-tuning.

Datasets Our experiments span a range of tasks in natural language understanding and generation. For NLU, we evaluate on five tasks from the widely-used General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018): MNLI, SST-2, CoLA, QNLI, and MRPC. These tasks cover natural language inference, sentiment analysis, grammatical acceptability, and paraphrase identification.

For NLG, we fine-tune the LLaMA-2-7B (Touvron et al., 2023) model on a 52k randomly sampled subset of the WizardLM dataset (Xu et al., 2024). We evaluate the model on the MT-Bench dataset (Zheng et al., 2024a), which consists of 80 multi-turn questions designed to assess conversational abilities across various aspects. The quality of the responses is evaluated by GPT-4, and we report the first-turn score as the primary evaluation metric.

For mathematical reasoning, we use a 100k random sample from MetaMathQA (Yu et al., 2024), with evaluation on the GSM8K test set (Cobbe et al., 2021). For code generation, fine-tuning is performed on a 100k randomly sampled subset of the CodeFeedback dataset (Zheng et al., 2024b), with evaluation on HumanEval (Chen et al., 2021).

Implementation Details For fair comparison, our experimental setup closely follows that of LoRA-GA (Wang et al., 2024a). Across all experiments, we use the AdamW optimizer (Loshchilov and Hutter, 2019) with weight decay set to 0 and a cosine learning rate schedule with a warm-up ratio of 0.03. LoRA adapters are applied to all linear layers within the transformer blocks, with the rank r set to 8 and scaling factor α to 16 by default. For our two task families, the settings are as follows. For Natural Language Understanding (NLU) on GLUE, we fine-tune T5-base (Raffel et al., 2020) with a learning rate of 1×10^{-4}, a sequence length of 128, and a batch size of 32. The MGPO hy-

Table 2: Fine-tuning results of LLaMA-2-7B on MT-Bench, GSM8K, and HumanEval. Performance is evaluated using primary task metrics: MT-Bench score, GSM8K accuracy, and HumanEval Pass@1. PEFT methods are tested with rank $r = 8$, and additional tests at ranks 32 and 128 are included to evaluate performance scaling. Results are averaged over three random seeds, with standard deviations provided. **Bold** and underlining denote the best and second-best scores, respectively.

Method	MT-Bench	GSM8K	HumanEval	Avg
Full FT	$5.30_{\pm 0.11}$	$\mathbf{59.36}_{\pm 0.85}$	$\mathbf{35.31}_{\pm 2.13}$	**33.32**
LoRA	$5.61_{\pm 0.10}$	$42.08_{\pm 0.04}$	$14.76_{\pm 0.17}$	20.82
DoRA	$\underline{5.97}_{\pm 0.02}$	$53.07_{\pm 0.75}$	$19.75_{\pm 0.41}$	26.26
AdaLoRA	$5.57_{\pm 0.05}$	$50.72_{\pm 1.39}$	$17.80_{\pm 0.44}$	24.70
PiSSA	$5.30_{\pm 0.02}$	$44.54_{\pm 0.27}$	$16.02_{\pm 0.78}$	21.95
rsLoRA	$5.25_{\pm 0.03}$	$45.62_{\pm 0.10}$	$16.01_{\pm 0.79}$	22.29
LoRA+	$5.71_{\pm 0.08}$	$52.11_{\pm 0.62}$	$18.17_{\pm 0.52}$	25.33
LoRA-GA	$5.95_{\pm 0.16}$	$53.60_{\pm 0.30}$	$19.81_{\pm 1.46}$	26.45
LoRA-GA (rank=32)	$5.79_{\pm 0.09}$	$55.12_{\pm 0.30}$	$20.18_{\pm 0.19}$	27.03
LoRA-GA (rank=128)	$6.13_{\pm 0.07}$	$55.07_{\pm 0.18}$	$23.05_{\pm 0.37}$	28.08
LoRA-MGPO	$\mathbf{6.27}_{\pm 0.12}$	$\underline{54.56}_{\pm 0.44}$	$\underline{21.02}_{\pm 0.39}$	<u>27.28</u>
LoRA-MGPO (rank=32)	$6.21_{\pm 0.15}$	$55.74_{\pm 0.21}$	$21.34_{\pm 0.47}$	27.76
LoRA-MGPO (rank=128)	$6.48_{\pm 0.23}$	$56.96_{\pm 0.35}$	$24.87_{\pm 0.54}$	29.44

perparameters are $\rho = 0.05$, $\mu = 0.9$ (AdamW's 'beta1'), and $\beta = 0.9$. For Natural Language Generation (NLG), we fine-tune LLaMA-2-7B (Touvron et al., 2023) with a learning rate of 2×10^{-5} and a sequence length of 1024. We use a per-device batch size of 4 with 8 gradient accumulation steps for an effective batch size of 32. The MGPO hyperparameters are $\rho = 0.01$, $\mu = 0.8$ (AdamW's 'beta1'), and $\beta = 0.8$. All experiments were conducted on NVIDIA H20 96GB GPUs, repeated three times with different random seeds, and we report the average and standard deviation of the results. Further details on optimizer settings, specific LoRA target modules, and the software environment are provided in the Appendix.

3.2 Main Results

Performance on Natural Language Understanding (NLU) We first evaluated LoRA-MGPO on a standard suite of NLU tasks from the GLUE benchmark (Wang et al., 2018), using the T5-base model. As detailed in Table 1, our method demonstrates strong and consistent performance. The improvements are particularly notable on challenging, low-resource benchmarks such as CoLA and MRPC, where LoRA-MGPO surpasses not only all other PEFT methods but also full fine-tuning. Success on these tasks often hinges on capturing subtle linguistic nuances. The stability afforded by LoRA-MGPO likely prevents the fine-tuning process from corrupting the rich knowledge encoded in the base model; by preventing erratic weight updates, our method may better preserve the pre-trained model's nuanced understanding of syntax and semantics. Quantitatively, LoRA-MGPO achieves the highest scores among all PEFT methods on five out of five tasks, obtains the best average score, and outperforms the next-best PEFT method, LoRA-GA, by a margin of 1.04 points.

Performance on Natural Language Generation (NLG) We further assessed our method on three challenging NLG tasks using the LLaMA-2-7B model, with results summarized in Table 2. LoRA-MGPO consistently secures top performance among all PEFT baselines. On the conversational MT-Bench, its top score suggests that stable training helps maintain the model's coherence and instruction-following capabilities. For structured reasoning tasks like mathematical problem-solving (GSM8K) and code generation (HumanEval), where logical consistency is paramount, LoRA-MGPO again emerges as the strongest PEFT method. A stable optimization trajectory may reduce the risk of the model deviating from a correct reasoning path during fine-tuning, as each update step is more measured, preventing

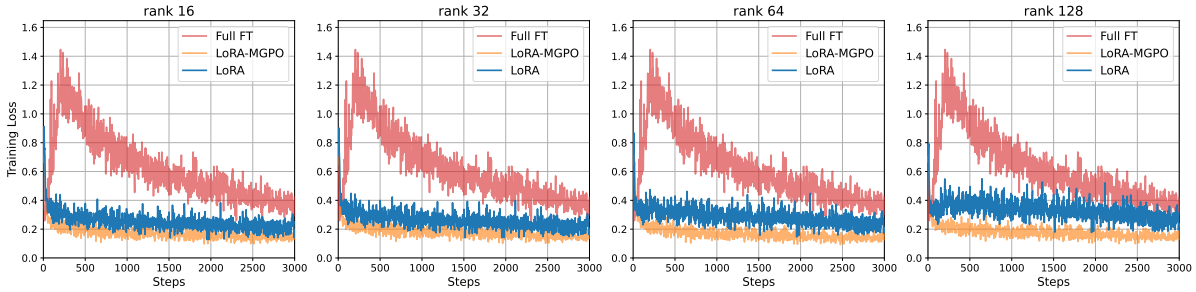

Figure 2: Training loss dynamics across different rank configurations: A comparative analysis of LoRA, LoRA-MGPO, and full fine-tuning on LLaMA-2-7B with MetaMathQA. Rank (r) and alpha (α) follow $r = \alpha \in \{16, 32, 64, 128\}$ with a fixed learning rate of $5e-4$.

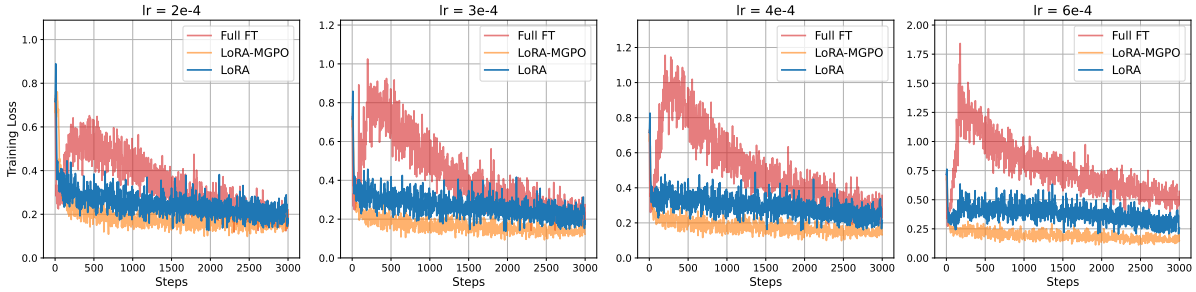

Figure 3: Learning rate sensitivity analysis: A comparison of training loss for LoRA, LoRA-MGPO, and full fine-tuning on LLaMA-2-7B with MetaMathQA. The analysis spans learning rates $\{2e-4, 3e-4, 4e-4, 6e-4\}$, with rank ($r$) and alpha ($\alpha$) fixed at 128.

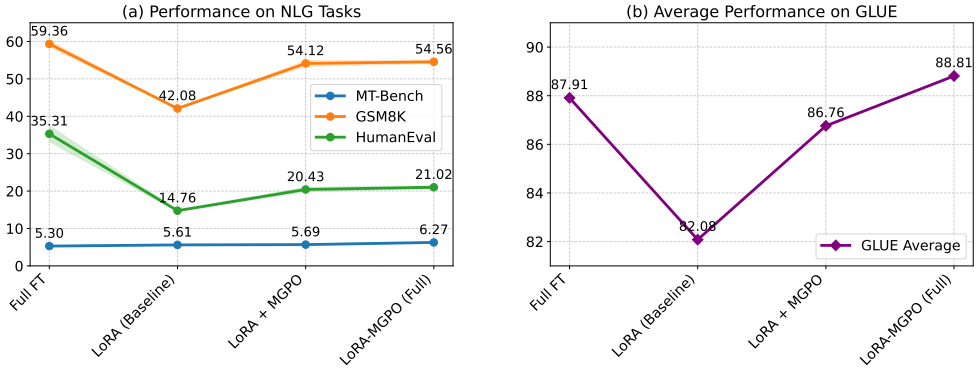

Figure 4: Ablation study of LoRA-MGPO on NLG and NLU tasks. (a) LLaMA-2-7B performance across three NLG tasks. (b) T5-Base performance on the GLUE benchmark. "LoRA (Baseline)" refers to standard LoRA, "LoRA + MGPO" refers to an ablation with only momentum-guided perturbation, and "LoRA-MGPO (Full)" includes both momentum-guided perturbation and adaptive normalization.

catastrophic error accumulation common in multi-step generation. While full fine-tuning still holds an edge on the reasoning tasks, our method narrows the gap and outperforms it on MT-Bench. Notably, as the LoRA rank increases from 8 to 128, the performance of LoRA-MGPO scales gracefully, validating its ability to effectively leverage a higher parameter budget while maintaining the training stability that standard LoRA often lacks at higher ranks.

3.3 Analysis and Ablation Studies

Effectiveness in Mitigating Double Descent To empirically validate LoRA-MGPO's core claim of mitigating double descent, we conducted a controlled analysis of its training dynamics, focusing on the impacts of rank and learning rate. The results, presented in Figure 2 and Figure 3, offer compelling visual evidence of our method's stability. Figure 2 illustrates that as the LoRA rank r increases, the double descent phenomenon in standard LoRA becomes progressively more se-

Table 3: Comparison of computational efficiency and performance across LoRA, LoRA-MGPO, and Full FT methods, trained for one epoch on the WizardLM dataset using LLaMA-2-7B.

Method	#Params	Memory Cost	Training Time	MT-Bench	GSM8K	HumanEval
Full FT	6738M	>96 GB	-	$5.30_{\pm 0.11}$	$59.36_{\pm 0.85}$	$35.31_{\pm 2.13}$
LoRA	320M	81.73 GB	5h 48min	$5.61_{\pm 0.10}$	$42.08_{\pm 0.04}$	$14.76_{\pm 0.17}$
LoRA-MGPO	320M	90.56 GB	6h 52min	$6.27_{\pm 0.12}$	$54.56_{\pm 0.44}$	$21.02_{\pm 0.39}$

Table 4: Ablation study of LoRA-MGPO vs. random noise perturbation on three NLG benchmarks. Experiments use LLaMA-3.1-8B-Base (Dubey et al., 2024) with rank $r = 8$. Scores are averaged over three random seeds, with standard deviations in subscripts. **Bold** indicates the best method.

Method	**MTBench**	**GSM8k**	**HumanEval**
Full FT	$5.88_{\pm 0.23}$	$73.69_{\pm 0.28}$	$51.63_{\pm 1.27}$
LoRA	$6.15_{\pm 0.02}$	$67.78_{\pm 1.25}$	$43.09_{\pm 0.35}$
LoRA + Random Noise	$6.43_{\pm 0.26}$	$68.05_{\pm 1.12}$	$42.92_{\pm 0.41}$
LoRA-MGPO	$\mathbf{7.51}_{\pm 0.07}$	$\mathbf{70.23}_{\pm 1.08}$	$\mathbf{45.13}_{\pm 0.63}$

vere, exhibiting a sharp rebound at $r = 128$. In stark contrast, LoRA-MGPO's loss curve remains smooth and monotonically decreasing across all ranks. Similarly, Figure 3 shows that while higher learning rates induce significant oscillations in standard LoRA, LoRA-MGPO maintains a stable convergence path. These findings provide strong empirical evidence that our method effectively stabilizes fine-tuning and potentially broadens the effective learning rate window.

Ablation Study To rigorously dissect the individual and combined contributions of our method's two key components—Momentum-Guided Perturbation (MGPO) and Adaptive Perturbation Normalization (APN)—we conducted a detailed ablation study, with results shown in Figure 4. The findings clearly validate our design choices. The first ablation step, labeled **'LoRA + MGPO'**, applies only the MGPO component and yields a substantial performance lift over the vanilla **'LoRA (Baseline)'**. On the NLU task suite, for instance, this single component boosts the average score from 82.08 to 86.76, demonstrating that the core strategy of using momentum to guide perturbations towards flatter loss regions is fundamentally effective.

However, the full potential is unlocked when introducing APN. Our complete model, labeled **'LoRA-MGPO (Full)'**, combines both components and achieves the final NLU score of 88.81. The significant improvement from 86.76 to 88.81 underscores the critical role of adaptive normalization. It suggests that while MGPO provides a stable perturbation *direction*, its effectiveness is maximized only when the perturbation *magnitude* is dynamically scaled in response to the gradient landscape. The consistent superiority of the full model across all NLU and NLG tasks confirms that these two components are not merely additive but work in synergy, fulfilling the design goals of our framework.

Comparison with Random Noise Perturbation To further validate that our performance gains stem from a principled optimization strategy rather than simple regularization, we compared LoRA-MGPO to LoRA augmented with undirected, isotropic random noise. The results in Table 4 are revealing: adding random noise provides only inconsistent and marginal benefits, and can even be detrimental in some cases (e.g., HumanEval). In contrast, LoRA-MGPO yields consistent and significant improvements across all tasks.

This disparity highlights a fundamental difference in mechanism. Random noise acts as a general regularizer by pushing parameters out of their immediate trajectory, which can occasionally help escape sharp minima by chance. However, the direction is arbitrary and uncorrelated with the loss landscape's structure. Our momentum-guided perturbation, conversely, is *informed*. It leverages the recent history of the optimization path—a strong indicator of relevant high-curvature directions—to perform a targeted exploration. This principled approach makes the search for flat minima non-stochastic and significantly more effective and reliable than undirected noise injection.

Computational Cost Analysis Finally, we analyzed the practical overhead of our method (Table 3). As expected, LoRA-MGPO operates with the same minimal number of trainable parameters as standard LoRA, making it vastly more memory-efficient than Full FT. In terms of training time, LoRA-MGPO introduces a modest and acceptable

overhead compared to vanilla LoRA (6h 52m vs. 5h 48m in our NLG setup). Given the significant performance improvements it delivers, this analysis confirms that LoRA-MGPO presents a highly favorable trade-off between computational cost and model performance, underscoring its practical viability.

4 Related Work

Parameter-Efficient Fine-Tuning (PEFT) The prohibitive computational and storage costs of full-parameter fine-tuning (Howard and Ruder, 2018; Devlin, 2018) have spurred the development of PEFT techniques for adapting large language models (Houlsby et al., 2019; Ding et al., 2023). By selectively updating a small subset of parameters, PEFT methods can achieve performance competitive with full fine-tuning while being significantly more efficient (Han et al., 2024). Among the diverse PEFT strategies, Low-Rank Adaptation (LoRA) (Hu et al., 2021) has gained prominence for its simplicity and effectiveness. Recent works have enhanced LoRA along several directions. One line of work introduces *architectural modifications*; for instance, DoRA (Liu et al., 2024) integrates learnable magnitude vectors, while AdaLoRA (Zhang et al., 2023) dynamically allocates rank budgets. Another direction focuses on *improving the training process and initialization*, such as adjusting scaling factors in rsLoRA (Kalajdzievski, 2023), using separate learning rates in LoRA+ (Hayou et al., 2024), or refining initialization with PiSSA (Meng et al., 2024) and NLoRA (Guo et al., 2025). A third direction aims to improve the quality of the parameter updates, for instance by alleviating training biases with BA-LoRA (Chang et al., 2024a) or by more closely aligning LoRA's gradients with those of full fine-tuning, as seen in LoRA-GA (Wang et al., 2024a) and LoRA-Pro (Wang et al., 2024b). Additional work has further explored LoRA's application in multi-task learning, such as (Liu et al., 2025b,a). Distinct from these approaches, our work focuses directly on the underlying optimization dynamics. Rather than altering LoRA's architecture or mimicking full fine-tuning gradients, we introduce a novel training framework to stabilize the optimization process itself.

Optimization Stability in PEFT The training stability of PEFT methods, particularly LoRA, is a critical concern. Empirical studies have revealed that as LoRA's rank increases, performance can degrade after an initial improvement, a behavior analogous to the double descent phenomenon (Belkin et al., 2019; Nakkiran et al., 2019). This instability highlights the challenge of navigating high-dimensional and non-convex loss landscapes during fine-tuning. To promote smoother optimization and find flatter minima, Sharpness-Aware Minimization (SAM) (Foret et al., 2020) has been influential. However, its requirement for dual gradient computations imposes a significant computational burden (Becker et al., 2024; Li et al., 2024b). More recent work has explored more efficient directional perturbation strategies. Momentum-guided methods, for example, reuse optimizer momentum to avoid the extra gradient step, reducing computational cost without sacrificing the directional guidance (Becker et al., 2024). Other techniques, such as applying an exponential moving average (EMA) to model weights, also contribute to stability by smoothing the trajectory of parameter updates (Wang et al., 2021). While these components—efficient perturbation and smoothing—are individually effective, they are typically studied in isolation. This leaves a clear gap for a unified framework that synergistically combines these strategies to enhance both the efficiency and stability of PEFT. Our work, LoRA-MGPO, is designed to fill this gap.

5 Conclusion

In this work, we addressed the double descent phenomenon in Low-Rank Adaptation (LoRA), an instability that can affect the fine-tuning of large language models. We proposed **LoRA-MGPO**, an optimization framework that integrates Momentum-Guided Perturbation Optimization (MGPO). This method aims to find flatter minima by reusing optimizer momentum to guide weight perturbations, combined with an adaptive normalization scheme to improve robustness. Our experimental results across a range of natural language understanding (NLU) and natural language generation (NLG) tasks show that LoRA-MGPO provides improved performance over standard LoRA and other common PEFT baselines. This improvement is reflected in more stable convergence trajectories and reduced training instability. LoRA-MGPO offers a practical approach to overcoming some of the optimization challenges in LoRA while maintaining its parameter efficiency. Future research may explore extending this framework to other parameter-

efficient methods or adapting it for different domains, such as vision and speech.

Limitations

First, LoRA-MGPO's use of momentum vectors for perturbation directions assumes relatively stable optimizer dynamics, which might limit its effectiveness during early training stages or in the presence of highly non-stationary gradient conditions. Second, while the adaptive perturbation normalization via EMA-smoothed gradients improves robustness, its performance may be sensitive to sudden changes in gradient magnitude distributions, potentially requiring adjustments to the smoothing hyperparameters depending on the specific task.

Ethics Statement

Our research focuses on LoRA-MGPO, a general-purpose optimization algorithm designed to improve the stability of parameter-efficient fine-tuning (PEFT). The experiments use publicly available, pre-trained models (LLaMA-2-7B, T5-base) and standard academic benchmarks. We acknowledge that these foundational models may inherit and potentially amplify societal biases present in their training data. The primary goal of this work is to provide a more reliable and resource-efficient tool for adapting and studying such models within the research community. By enhancing PEFT techniques, our work contributes to broader efforts aimed at reducing the computational costs involved in large-scale model adaptation.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No.2023YFF0905400), the National Natural Science Foundation of China (No.U2341229) and the Reform Commission Foundation of Jilin Province (No.2024C003).

References

Marlon Becker, Frederick Altrock, and Benjamin Risse. 2024. Momentum-sam: Sharpness aware minimization without computational overhead. *arXiv preprint arXiv:2401.12033*.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. 2019. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854.

Yupeng Chang, Yi Chang, and Yuan Wu. 2024a. Balora: Bias-alleviating low-rank adaptation to mitigate catastrophic inheritance in large language models. *arXiv preprint arXiv:2408.04556*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024b. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*.

Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2023. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 12799–12807.

Chenlu Guo, Yuan Wu, and Yi Chang. 2025. Nlora: Nystr\" om-initiated low-rank adaptation for large language models. *arXiv preprint arXiv:2502.14482*.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

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

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

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Tao Li, Zhengbao He, Yujun Li, Yasheng Wang, Lifeng Shang, and Xiaolin Huang. 2024a. Flat-lora: Low-rank adaption over a flat loss landscape. *arXiv preprint arXiv:2409.14396*.

Tao Li, Qinghua Tao, Weihao Yan, Zehao Lei, Yingwen Wu, Kun Fang, Mingzhen He, and Xiaolin Huang. 2024b. Revisiting random weight perturbation for efficiently improving generalization. *arXiv preprint arXiv:2404.00357*.

Jinda Liu, Yi Chang, and Yuan Wu. 2025a. R-lora: Random initialization of multi-head lora for multi-task learning. *arXiv preprint arXiv:2502.15455*.

Jinda Liu, Bo Cheng, Yi Chang, and Yuan Wu. 2025b. Align, don't divide: Revisiting the lora architecture in multi-task learning. *arXiv preprint arXiv:2508.05078*.

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

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *Preprint*, arXiv:2404.02948.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. 2019. Deep double descent: Where bigger models and more data hurt. *Preprint*, arXiv:1912.02292.

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

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Shaowen Wang, Linxi Yu, and Jian Li. 2024a. Lora-ga: Low-rank adaptation with gradient approximation. *Preprint*, arXiv:2407.05000.

Yizhou Wang, Yue Kang, Can Qin, Huan Wang, Yi Xu, Yulun Zhang, and Yun Fu. 2021. Rethinking adam: A twofold exponential moving average approach. *arXiv preprint arXiv:2106.11514*.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. 2024b. Lora-pro: Are low-rank adapters properly optimized? *Preprint*, arXiv:2407.18242.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *ICLR*.

Longhui Yu, Weisen Jiang, Han Shi, YU Jincheng, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *ICLR*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *Preprint*, arXiv:2303.10512.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024a. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS*.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue.

2024b. OpenCodeInterpreter: Integrating code generation with execution and refinement. In *Findings of ACL*.

Appendix

Contents

A Models and Datasets

A.1 Details of Models

In this work, we primarily utilize two pre-trained language models: LLaMA-2-7B and T5-base.

- **LLaMA-2-7B**: A 7-billion parameter, decoder-only transformer model from the LLaMA-2 series, primarily used for generation tasks. More details are available at its Hugging Face repository[*].
- **T5-base**: A 220-million parameter encoder-decoder transformer model, widely used for a variety of natural language understanding tasks. More details are available at its Hugging Face repository[†].

Our experiments were conducted using the implementations of these models provided by the Hugging Face Transformers library.

A.2 Details of Datasets

Table 5 summarizes the GLUE benchmark datasets (Wang et al., 2018). For our Natural Language Generation (NLG) experiments, we used the following evaluation metrics: Accuracy for GSM8K; Pass@1 for HumanEval; and a score based on GPT-4 evaluation for MT-Bench.

B Baselines and Implementation

B.1 Baseline Methods

Our study includes several baseline methods for a comprehensive comparison. **Full Fine-Tuning** serves as a strong performance benchmark. **Vanilla**

[*] https://huggingface.co/meta-llama/LLaMA-2-7B
[†] https://huggingface.co/t5-base

Table 5: GLUE Benchmark Datasets and Evaluation Metrics

Dataset	Task Type	Classes	Train Examples	Metric	Description
CoLA	Acceptability	2	8.5k	Matthews Corr.	Grammatical acceptability
SST-2	Sentiment	2	67k	Accuracy	Sentiment analysis
MRPC	Paraphrase	2	3.7k	Accuracy/F1	Paraphrase detection
MNLI	NLI	3	393k	Accuracy	Multi-genre NLI
QNLI	NLI/QA	2	108k	Accuracy	QA/NLI converted from SQuAD

LoRA (Hu et al., 2021) is our primary point of comparison from the PEFT literature. We also compare against LoRA variants that introduce **structural modifications** (DoRA (Liu et al., 2024), AdaLoRA (Zhang et al., 2023)) and those that **refine the training process or initialization** (rsLoRA (Kalajdzievski, 2023), LoRA+ (Hayou et al., 2024), PiSSA (Meng et al., 2024)). Finally, we include methods focused on **gradient alignment** (LoRA-GA (Wang et al., 2024a), LoRA-Pro (Wang et al., 2024b)).

B.2 Implementation Details

LoRA Configuration. As stated in the main text, LoRA adapters were applied to all linear layers within the transformer blocks for both LLaMA-2-7B and T5-base models.

Initialization of MGPO. The implementation of our method requires an initial state for the momentum vector and the adaptive normalization factor. Following standard optimizer practice, the momentum '$\boldsymbol{m}$' is initialized to zeros. The adaptive normalization factor '$\bar{g}$' is initialized using the L2-norm of the gradient computed in the first training step.

Hyperparameters. Our method introduces two primary hyperparameters: the perturbation radius 'ρ' and the EMA decay rate 'β'. 'ρ' controls the magnitude of the weight perturbation, influencing the search for flatter minima. 'β' controls the temporal smoothing window for the adaptive normalization. The values used in our main experiments were effective across the evaluated tasks, as evidenced by the strong performance reported in Section 3.

B.3 Hyperparameter Settings for Baselines

To ensure a fair and robust comparison, we adhered to the hyperparameter settings recommended in the original papers or official codebases of our baseline methods wherever possible. General settings, such as the learning rate schedule and batch size, were kept consistent across all methods as described in Section 3. Key method-specific hyperparameters are detailed below.

- **DoRA (Liu et al., 2024):** We utilized the official implementation provided by the authors, maintaining its default configuration for the magnitude and directional components.
- **AdaLoRA (Zhang et al., 2023):** We followed the setup from the original paper, with the rank budget dynamically allocated starting from a higher initial rank and pruned during training.
- **LoRA+ (Hayou et al., 2024):** Following the authors' recommendation, the learning rate for the LoRA matrix A was set to our default value (1×10^{-4} for NLU, 2×10^{-5} for NLG), while the learning rate for matrix B was set 16 times higher.
- **LoRA-GA and LoRA-Pro (Wang et al., 2024a,b):** For these methods focused on gradient alignment, we used the hyperparameter settings as specified in their respective papers and official implementations to ensure a faithful comparison.

For all other baselines, we used their standard, publicly available implementations without modification to their core components.