LoRA-MGPO: Mitigating Double Descent in LoRA Fine-Tuning via Momentum-Guided Perturbation Optimization

Anonymous ACL submission

Abstract

Parameter-efficient fine-tuning (PEFT) methods, such as Low-Rank Adaptation (LoRA), enable efficient adaptation of large language models (LLMs) via low-rank matrix optimization with frozen weights. However, LoRA typically exhibits "double descent" in training loss as rank increases, characterized by a three-phase dynamics: initial convergence, transient divergence, and eventual stabilization. This non-monotonic behavior delays convergence and impairs generalization through unstable gradients and attraction to sharp minima. To address these challenges, we propose **LoRA-MGPO**, a novel LoRA-based framework incorporating Momentum-Guided Perturbation Optimization (MGPO). First, MGPO eliminates Sharpness-Aware Minimization (SAM)'s dual gradient computations by reusing momentum vectors from optimizer states to guide perturbation directions. This retains SAM's training stability and flat minima preference with maintained efficiency. Second, MGPO incorporates adaptive perturbation normalization, scaling perturbation intensity via exponential moving average (EMA)-smoothed gradient magnitudes. Experiments on natural language understanding and generation benchmarks demonstrate that LoRA-MGPO outperforms LoRA and state-of-the-art PEFT methods. Further analysis confirms its ability to stabilize training and reduce sharp minima attraction, with smoother loss curves and improved convergence behavior.

1 Introduction

Large language models (LLMs) have revolutionized natural language processing, setting new performance benchmarks for tasks such as text generation and semantic understanding (Chang et al., 2024; Wei et al., 2022). However, full-parameter fine-tuning (Full FT) of these models requires updating billions of parameters, imposing prohibitive memory and computational costs. To address this challenge, parameter-efficient fine-tuning (PEFT) methods have emerged as a promising alternative by selectively optimizing essential model components (Lester et al., 2021; Fu et al., 2023).

Among these, Low-Rank Adaptation (LoRA) (Hu et al., 2021) stands out for its computational efficiency and architectural simplicity. Specifically, LoRA approximates weight changes via low-rank matrices by decomposing the update as $\Delta W = BA$, where $W_0 \in \mathbb{R}^{m\times n}$ represents the frozen pre-trained weights, and $B \in \mathbb{R}^{m\times r}$, $A \in \mathbb{R}^{r\times n}$ are trainable matrices with rank $r \ll \min(m, n)$. The updated weights $W' = W_0 + \Delta W$ retain the original model functionality while reducing the number of trainable parameters from $\mathcal{O}(mn)$ to $\mathcal{O}(r(m + n))$. For input $X \in \mathbb{R}^{p\times m}$, the output $Y = X(W_0 + BA) \in \mathbb{R}^{p\times n}$ preserves the original model functionality, enabling efficient downstream adaptation through low-rank updates.

Despite its efficiency, LoRA exhibits the "double descent" phenomenon during training (Belkin et al., 2019). As illustrated in Figure 1, LoRA fine-tunes LLaMA-2-7B (Touvron et al., 2023) on MetaMathQA (Yu et al., 2024) with rank (r) and alpha (α) parameters set to identical values ($r = \alpha \in \{32, 64, 128\}$) and a fixed learning rate of $5e-4$. This typically produces a double-descent loss curve characterized by initial convergence, abrupt divergence, and eventual stabilization. Notably, the double descent phenomenon becomes increasingly pronounced with larger rank values. Moreover, Full FT exhibits even more severe double descent behavior, aligning with the observation in (Nakkiran et al., 2019) that larger model sizes exacerbate this phenomenon. The non-monotonic behavior delays convergence and impairs generalization through unstable gradient dynamics and attraction to sharp minima during training (Li et al., 2024b).

To address the unstable gradient dynamics driving double descent, Sharpness-Aware Minimiza-

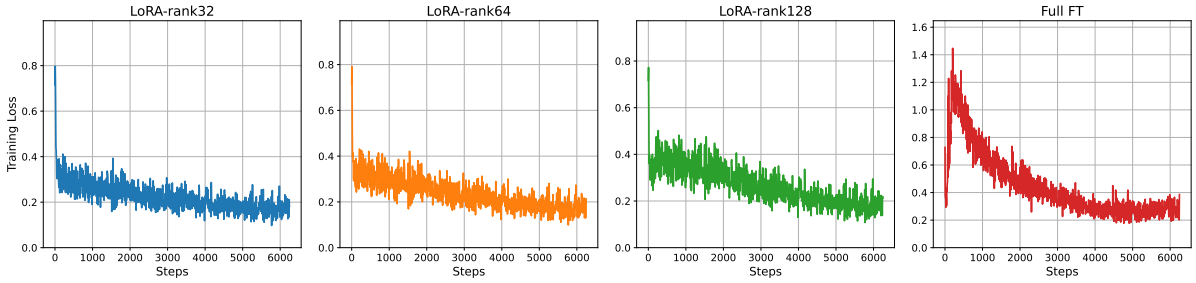

Figure 1: Training loss curves of Full FT and LoRA (Hu et al., 2021) methods with LLaMA-2-7B (Touvron et al., 2023) on the MetaMathQA dataset (Yu et al., 2024). For LoRA, rank (r) and alpha (α) parameters are set to identical values ($r = \alpha \in \{32, 64, 128\}$), while the learning rate is fixed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

tion (SAM) serves as a regularization strategy in LoRA optimization, demonstrated to enhance convergence and generalization (Foret et al., 2020; Li et al., 2024a). By seeking flat minima through gradient perturbation, SAM suppresses parameter oscillations during training (Li et al., 2024b). However, SAM's scalability is constrained by dual gradient computations per iteration, doubling computational costs (Becker et al., 2024; Li et al., 2024c). Unlike SAM's costly dual gradient computations, momentum-guided SAM reuses momentum vectors from optimizer states to guide perturbation directions, eliminating additional gradient evaluations while preserving directional sharpness-aware updates (Becker et al., 2024). Despite these efficiency, standalone momentum-guided SAM cannot ensure stable convergence or robust generalization. Thus, exponential moving average (EMA)-based smoothing (Wang et al., 2021) is introduced to suppress parameter oscillations and decouple perturbation intensity from optimization dynamics.

Building on these insights, we propose **LoRA-MGPO**, a novel LoRA-based framework incorporating Momentum-Guided Perturbation Optimization (MGPO) to mitigate the detrimental effects of double descent. First, MGPO eliminates SAM's dual gradient computations by reusing momentum vectors from optimizer states to guide perturbation directions. This preserves SAM's training stability and flat minima preference while maintaining computational efficiency. Second, MGPO introduces adaptive perturbation normalization, which scales perturbation intensity via EMA-smoothed gradient magnitudes. This decouples it from optimization dynamics and suppresses double descent. We evaluate LoRA-MGPO on natural language understanding (NLU) and generation (NLG) benchmarks, demonstrating consistent improvements over standard LoRA and other state-of-the-art PEFT baselines. Further analysis shows that LoRA-MGPO stabilizes training dynamics and mitigates sharp minima, with smoother loss curves and faster convergence rates as empirical evidence.

2 Related Work

Parameter-Efficient Fine-Tuning (PEFT). The scale of LLMs and their increasing deployment have motivated the development of PEFT techniques (Houlsby et al., 2019). While full-parameter fine-tuning methods (Howard and Ruder, 2018; Devlin, 2018) incur high computational and storage costs, PEFT methods reduce trainable parameters by selectively updating subsets of model weights, achieving competitive performance (Houlsby et al., 2019; Ding et al., 2023; Han et al., 2024). Among these, Low-Rank Adaptation (LoRA) (Hu et al., 2021) and its variants—such as Rank-Stabilized LoRA (rsLoRA) (Kalajdzievski, 2023), LoRA+ (Hayou et al., 2024), PiSSA (Meng et al., 2024), DoRA (Liu et al., 2024b), and AdaLoRA (Zhang et al., 2023)—have gained prominence by approximating weight updates through low-rank matrix decompositions while preserving model performance. These methods primarily focus on architectural modifications or adaptive rank allocation to enhance learning efficiency and performance. Other notable PEFT methods include Adapter Layers (Houlsby et al., 2019), BitFit (Zaken et al., 2021), and Prefix-Tuning (Li and Liang, 2021), which achieve efficient adaptation with minimal parameter updates. Recent advancements further explore gradient approximation strategies, such as LoRA-GA (Wang et al., 2024a) and LoRA-Pro (Wang et al., 2024b), which accelerate convergence by approximating full fine-tuning gradients.

Optimization Stability in PEFT

The optimization stability of PEFT methods, such as LoRA, is notably influenced by low-rank

adaptation strategies. Empirical studies show that performance improves with increasing rank up to a certain threshold, beyond which overfitting occurs, destabilizing training dynamics in LLMs. This behavior mirrors the double descent phenomenon, first characterized by (Belkin et al., 2019), where larger model capacities exacerbate gradient instability during training. (Nakkiran et al., 2019) further demonstrates that both increased model scale and data quantity may degrade performance in high-dimensional parameter spaces. To mitigate these challenges, Sharpness-Aware Minimization (SAM) (Foret et al., 2020) has been proposed to promote flat minima through gradient perturbation. However, it incurs significant computational overhead due to the need for dual gradient computations (Becker et al., 2024; Li et al., 2024c).

Recent advances in directional perturbation strategies have improved stability: gradient-guided approaches (Liu et al., 2024a; Ballas and Diou, 2025; Guo et al., 2024) leverage historical gradient information to control the magnitude of perturbations, offering greater robustness to sharp minima compared to stochastic noise injection methods. Momentum-guided methods, such as Momentum-SAM, reduce computational costs by reusing optimizer momentum vectors, thus eliminating the need for additional gradient evaluations (Becker et al., 2024). However, their integration with PEFT remains underexplored. Additionally, exponential moving average (EMA)-based smoothing (Wang et al., 2021) stabilizes training dynamics by controlling gradient magnitudes. Despite these advancements, existing methods generally address directional perturbation and dynamic regularization separately, highlighting the need for unified frameworks that optimize both efficiency and stability in PEFT.

3 Method

In this section, we first review the core framework of LoRA. Building on LoRA, we propose our LoRA-MGPO, which (i) integrates Momentum-Guided Perturbation Optimization (MGPO) to avoid additional gradient evaluations by reusing optimizer momentum vectors for directional perturbation updates, and (ii) introduces adaptive perturbation normalization to decouple perturbation intensity from regularization scheduling via EMA-smoothed intensity scaling.

3.1 Review of LoRA

Full Fine-tuning involves directly updating the entire weight matrix $W_0 \in \mathbb{R}^{m\times n}$, which incurs high computational and memory costs, making it difficult to deploy in practice. In contrast, Low-Rank Adaptation (LoRA) (Hu et al., 2021) is a parameter-efficient fine-tuning technique that introduces low-rank matrices $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$ to approximate weight updates while keeping pretrained weights W_0 frozen to retain model knowledge and simplify optimization. Vanilla LoRA initializes A with Kaiming normal distribution (He et al., 2015): $A \sim \mathcal{N}(0, \sigma^2)$ where $\sigma = \sqrt{\frac{2}{r}}$, and sets $B = 0$. Given an input matrix $X \in \mathbb{R}^{p\times m}$ (where p is the batch size, m is the input dimensionality), the output is computed as:

$$Y = XW' \in \mathbb{R}^{p\times n}, \tag{1}$$

where the adapted weight matrix is defined as:

$$W' = W_0 + \eta BA \in \mathbb{R}^{m\times n}. \tag{2}$$

Here, the rank r is chosen to satisfy $r \ll \min(m, n)$, which ensures the trainable parameters are negligible compared to the original model size. The scaling factor $\eta = \frac{\alpha}{r}$ balances the update magnitude and learning dynamics (Hu et al., 2021). While rsLoRA (Kalajdzievski, 2023) demonstrates that adaptive scaling of r can enhance performance, excessive rank expansion without corresponding optimization induces a pronounced double descent regime during fine-tuning, delaying convergence and impairing generalization through unstable gradient dynamics and attraction to sharp minima (Li et al., 2024b).

3.2 LoRA Fine-Tune with Momentum-Guided Perturbation Optimization

To address the challenge arising from double descent in LoRA training, we propose **LoRA-MGPO**, a novel LoRA-based framework integrating Momentum-Guided Perturbation Optimization (MGPO). MGPO eliminates SAM's dual gradient computations by reusing momentum vectors v_t to guide weight-space perturbations, while adaptive normalization dynamically scales perturbation intensity through EMA-smoothed gradient magnitudes.

3.2.1 Integrating SAM into LoRA

The objective function of Sharpness-Aware Minimization (SAM) (Foret et al., 2020) minimizes

loss landscape sharpness by perturbing in the full weight space:

$$\min_{A,B} \max_{\|\epsilon\| \leq \rho} \mathcal{L}\left(W_0 + \eta BA + \epsilon\right).$$

Here, A and B are trainable LoRA matrices, and ϵ is a weight-space perturbation vector derived from historical gradients. By maintaining LoRA's low-rank parameterization ηBA, our method preserves parameter efficiency while enabling curvature estimation aligned with SAM's principle (Becker et al., 2024). However, direct application of SAM to LoRA introduces two critical limitations: (1) dual gradient computations significantly increase computational overhead, and (2) additional weight storage for perturbation contradicts LoRA's principle of parameter-efficient fine-tuning. These challenges are resolved by MGPO through momentum reuse and adaptive normalization, as detailed below.

3.2.2 Momentum-Guided Perturbation Optimization (MGPO)

Building on Momentum-SAM (Becker et al., 2024), we reformulate SAM for LoRA by eliminating its dual gradient computations. Instead of computing gradients twice per iteration, MGPO reuses optimizer momentum vectors v_t to guide perturbation directions in the weight space:

$$\min_{A,B} \mathcal{L}(W_0 + \eta BA + \epsilon), \tag{3}$$

where the perturbation vector ϵ is defined as:

$$\epsilon = -\rho \cdot \frac{v_t}{\|v_t\|_2} \cdot \left(\bar{g}_l^{(t)}\right)^{-1}, \tag{4}$$

where ρ denotes the perturbation radius hyperparameter, $\boldsymbol{v}_t$ represents the momentum state vector guiding perturbation directions, and $\bar{g}_l^{(t)}$ stands for the exponential moving average of gradient magnitudes used for adaptive normalization. This approach leverages historical gradient information through the momentum update:

$$v_t = \mu v_{t-1} + \nabla \mathcal{L}(W_t), \tag{5}$$

where μ controls the contribution of historical gradients. By reusing v_t, MGPO avoids SAM's computationally expensive second gradient pass while maintaining stability against sharp minima attraction. Moreover, momentum reuse eliminates the need for additional weight storage during perturbation, inherently preserving LoRA's parameter efficiency.

Two-Stage Weight Update Mechanism To enable efficient curvature estimation and weight restoration, we adopt a two-stage strategy:

$$\widetilde{\boldsymbol{W}}_t = \boldsymbol{W}_t - \rho \cdot \frac{\boldsymbol{v}_t}{\|\boldsymbol{v}_t\|_2}, \tag{6}$$

where $\boldsymbol{W}_t = W_0 + \eta B_t A_t$ represents LoRA-parameterized weights, and $\widetilde{\boldsymbol{W}}_t$ denotes perturbed weights for curvature estimation. The training concludes with:

$$\boldsymbol{W}_T = \widetilde{\boldsymbol{W}}_T + \rho \cdot \frac{\boldsymbol{v}_T}{\|\boldsymbol{v}_T\|_2}, \tag{7}$$

where $\widetilde{\boldsymbol{W}}_T$ is the final perturbed state at iteration T, and $\boldsymbol{W}_T$ restores unperturbed LoRA weights while retaining curvature information.

3.2.3 Adaptive Perturbation Normalization

To complement MGPO's core design, we normalize ϵ via exponential moving average (EMA)-smoothed gradient magnitudes:

$$\bar{g}_l^{(t)} = \beta \bar{g}_l^{(t-1)} + (1-\beta)\|\nabla_{\Delta W_l} \mathcal{L}\|_2, \tag{8}$$

where β balances responsiveness to recent gradients with noise suppression. This adaptive normalization enhances MGPO's double descent suppression capability by ensuring scale-invariant perturbations across training stages and network layers.

4 Experiments

Baselines To evaluate LoRA-MGPO, we compare it with several baselines. Full Fine-Tuning updates all model parameters but is computationally expensive for LLMs. Vanilla LoRA (Hu et al., 2021) introduces a low-rank matrix product BA into linear layers, initializing A with Kaiming initialization and B as zero, offering parameter efficiency. We also examine *LoRA Variants with Modified Structure*. For instance, DoRA (Liu et al., 2024b) enhances expressiveness by incorporating learnable magnitudes. AdaLoRA (Zhang et al., 2023) dynamically prunes less significant weights via singular value decomposition (SVD), reallocating rank to optimize resource use. Additionally, we explore *LoRA Variants with Original Structure*. rsLoRA (Kalajdzievski, 2023) stabilizes training with a scaling factor. LoRA+ (Hayou et al., 2024) applies separate learning rates for A and B, improving adaptability. PiSSA (Meng et al., 2024) leverages SVD on the weight matrix W at initialization, using larger singular values for better initialization.

Table 1: Performance of T5-Base Fine-Tuning with Full Fine-Tuning and Various LoRA Variants on a Subset of GLUE Tasks. The best and second-best results are highlighted in **bold** and underline.

Method	MNLI	SST2	CoLA	QNLI	MRPC	Avg
Full FT	86.33$_{\pm 0.00}$	**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	**86.58**$_{\pm 0.11}$	94.72$_{\pm 0.46}$	**82.32**$_{\pm 0.18}$	**93.79**$_{\pm 0.46}$	**86.62**$_{\pm 0.68}$	**88.81**

LoRA-GA (Wang et al., 2024a) aligns low-rank gradients with full fine-tuning for faster convergence. Similarly, LoRA-Pro (Wang et al., 2024b) dynamically adjusts gradients to align the entire parameter update trajectory with full fine-tuning.

Implementation Details We replicate the experimental setup of LoRA-GA (Wang et al., 2024a). By default, we fine-tune the models using the AdamW optimizer (Loshchilov and Hutter, 2019), weight decay is set to 0, and a cosine learning rate schedule with a 0.03 warmup ratio is used. LoRA adapters are applied to all linear layers, excluding embedding, normalization, and classification layers. The rank r is set to 8, and the scaling factor α is set to 16 unless specified otherwise. For NLU tasks, 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. $\rho = 0.05$, $\mu = 0.9$, $\beta = 0.9$. For the NLG tasks, we fine-tune LLaMA-2-7B (Touvron et al., 2023) with a learning rate of 2×10^{-5}, a sequence length of 1024, and a macro batch size of 32. $\rho = 0.01$, $\mu = 0.8$, $\beta = 0.8$. Experiments were conducted on NVIDIA H20-NVLink (96GB) GPUs, repeated three times with different random seeds, and results are reported as averages with standard deviation.

4.1 Results on Natural Language Understanding (NLU) Tasks

We evaluate the performance of our proposed method, LoRA-MGPO, across several NLU tasks from the GLUE benchmark (Wang et al., 2018), including MNLI, SST-2, CoLA, QNLI, and MRPC. The experiments are conducted using the T5-base model (Raffel et al., 2020), with the LoRA rank fixed at 8. To ensure a comprehensive comparison, we include several baseline methods: full fine-tuning, standard LoRA (Hu et al., 2021), and its variants—DoRA (Liu et al., 2024b) and AdaLoRA (Zhang et al., 2023), which modify the LoRA structure. Additionally, we consider rsLoRA (Kalajdzievski, 2023), LoRA+ (Hayou et al., 2024), PiSSA (Meng et al., 2024), and LoRA-GA (Wang et al., 2024a), which retain the original LoRA structure. The results are summarized in Table 1.

LoRA-MGPO outperforms all other methods across all tasks. It achieves the highest accuracy on MNLI (86.58), CoLA (82.32), QNLI (93.79), and MRPC (86.62). For SST2, it ranks second with 94.72, just behind full fine-tuning (94.75). LoRA-MGPO's average performance is 88.81, surpassing LoRA-GA's 87.77 by 1.04 points. These results demonstrate LoRA-MGPO's robust performance, driven by its ability to optimize the loss landscape through MGPO and adaptive perturbation normalization, improving generalization, especially on smaller datasets.

4.2 Results on Natural Language Generation (NLG) Tasks

Experiments were conducted using the LLaMA-2-7B model (Touvron et al., 2023), with a LoRA rank set to 8. We also explored the effects of higher ranks (32 and 128) on performance. For dialogue generation, the model was fine-tuned on a 52k subset of the WizardLM dataset (Xu et al., 2024) and evaluated on the MT-Bench dataset (Zheng

Table 2: Fine-tuning results for LLaMA-2-7B model on NLG-related tasks. The best and second-best results are highlighted in **bold** and underline.

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

et al., 2024a). For mathematical reasoning, the model was fine-tuned on a 100k sample from the MetaMathQA dataset (Yu et al., 2024) and evaluated on the GSM8K test set (Cobbe et al., 2021). For code generation, the model was fine-tuned on a 100k subset of the CodeFeedback dataset (Zheng et al., 2024b) and evaluated on the HumanEval dataset (Chen et al., 2021). We compared LoRA-MGPO with several baselines, including full fine-tuning, standard LoRA (Hu et al., 2021), and its variants—DoRA (Liu et al., 2024b), AdaLoRA (Zhang et al., 2023), rsLoRA (Kalajdzievski, 2023), LoRA+ (Hayou et al., 2024), PiSSA (Meng et al., 2024), and LoRA-GA (Wang et al., 2024a).

The results, summarized in Table 2, show that LoRA-MGPO consistently outperforms all other methods across tasks. Specifically, LoRA-MGPO achieves the highest score on MT-Bench (6.27), indicating superior performance in dialogue generation. On GSM8K, LoRA-MGPO scores 54.56, placing second to full fine-tuning (59.36), demonstrating strong performance in mathematical reasoning. For HumanEval, LoRA-MGPO scores 21.02, outperforming LoRA (14.76), DoRA (19.75), and other variants, though it still falls short of full fine-tuning (35.31). Increasing the rank to 32 and 128 further improves LoRA-MGPO's performance. At rank 128, it achieves 6.48 on MT-Bench, 56.96 on GSM8K, and 24.87 on HumanEval, narrowing the gap to full fine-tuning and demonstrating scalability with higher ranks. These results validate LoRA-MGPO's effectiveness in enhancing training stability, achieving robust performance across various NLG tasks, and showcasing its scalability with increasing rank.

4.3 Effectiveness Analysis on Mitigating Double Descent

The primary objective of this experiment is to assess the effectiveness of LoRA-MGPO in mitigating the double descent phenomenon during fine-tuning. We fine-tuned LLaMA-2-7B (Touvron et al., 2023) on the MetaMathQA dataset (Yu et al., 2024) using LoRA, LoRA-MGPO, and Full FT. The first analysis (Figure 2) sets the rank (r) and alpha (α) to identical values, $r = \alpha \in \{16, 32, 64, 128\}$, with a fixed learning rate of $5e-4$. The second analysis (Figure 3) investigates learning rate sensitivity, evaluating $\{2e-4, 3e-4, 4e-4, 6e-4\}$ with $r = \alpha = 128$.

The results presented in Figures 2 and 3 lead to the following key observations: First, both LoRA and Full FT exhibit more pronounced double descent phenomena as the rank ($r \geq 64$) increases, marked by performance dips followed by recovery phases in their training curves. Second, higher learning rates ($lr \geq 4e-4$) exacerbate this issue, resulting in sharper fluctuations and delayed convergence in the baseline methods. Third, LoRA-MGPO consistently demonstrates smooth loss trajectories across all settings. These findings confirm

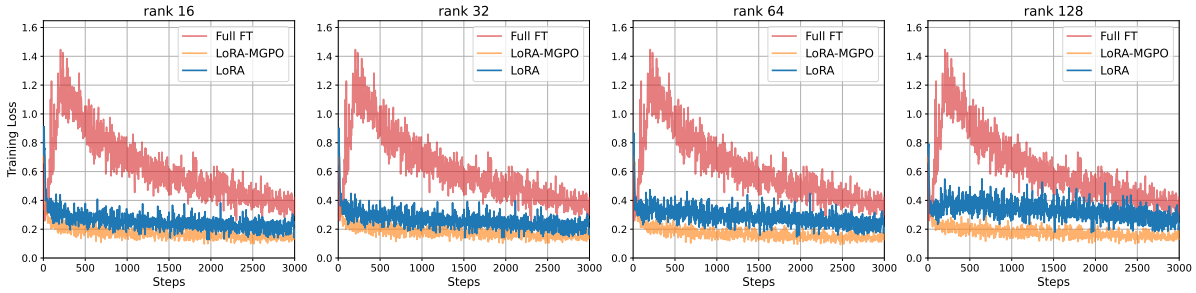

Figure 2: Training loss dynamics across rank configurations: Comparative analysis of LoRA, LoRA-MGPO, and full fine-tuning on LLaMA-2-7B and MetaMathQA. Rank (r) and alpha (α) follow $r = \alpha \in \{16, 32, 64, 128\}$ with fixed learning rate $5e-4$.

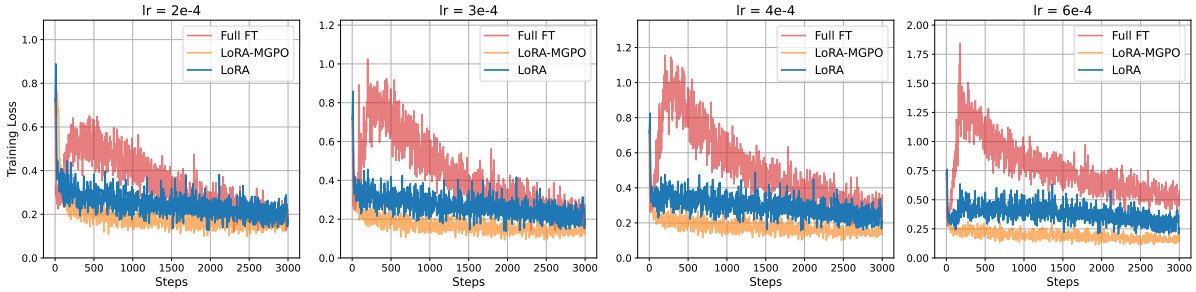

Figure 3: Learning rate sensitivity analysis: Training loss comparison of LoRA, LoRA-MGPO, and full fine-tuning on LLaMA-2-7B and MetaMathQA. Evaluation spans learning rates $\{2e-4, 3e-4, 4e-4, 6e-4\}$ with rank (r) and alpha (α) fixed at 128.

that LoRA-MGPO effectively mitigates the double descent phenomenon, ensuring stable convergence throughout the fine-tuning process.

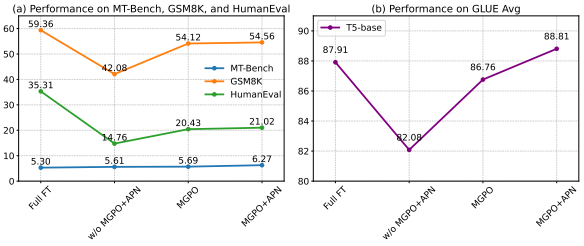

Figure 4: Ablation Study of LoRA Variants on NLG and NLU Tasks: (a) NLG Performance on MT-Bench, GSM8K, and HumanEval (LLaMA-2-7B); (b) Average NLU Performance on GLUE Benchmarks (MNLI, SST2, CoLA, QNLI, MRPC) with T5-Base. Methods compared: w/o MGPO+APN (standard LoRA), MGPO (LoRA+MGPO), and LoRA-MGPO (LoRA+MGPO+APN).

4.4 Ablation Study

This ablation study evaluates the individual contributions of the MGPO and Adaptive Perturbation Normalization (APN) components within the proposed LoRA-MGPO framework. For Natural Language Generation (NLG) tasks, we assessed LLaMA-2-7B (Touvron et al., 2023) on MT-Bench (Zheng et al., 2024a) (dialogue generation), GSM8K (Cobbe et al., 2021) (mathematical reasoning), and HumanEval (Chen et al., 2021) (code generation). For Natural Language Understanding (NLU) tasks, we tested T5-Base on five GLUE (Wang et al., 2018) subtasks: MNLI, SST2, CoLA, QNLI, and MRPC, with results averaged across these tasks. In all experiments, the LoRA rank was fixed at 8, and $\alpha = 16$ to ensure consistent comparisons.

As shown in Figure 4, LoRA-MGPO consistently achieves the highest performance. On NLU tasks, it outperforms full fine-tuning by +0.90 points (88.81 vs. 87.91) and surpasses MGPO (86.76) by +2.05 points, demonstrating the significant role of APN in enhancing generalization when combined with MGPO. For NLG tasks, LoRA-MGPO sets new state-of-the-art results on MT-Bench (6.27) and HumanEval (21.02), while maintaining competitive performance on GSM8K (54.56). In contrast, the baseline model without APN (standard LoRA) consistently underperforms across all tasks, such as 14.76 on HumanEval and 42.08 on GSM8K. These results confirm that the synergistic integration of MGPO and APN improves parameter-efficient fine-tuning.

Table 3: Comparative analysis of computational efficiency and performance in LoRA, LoRA-MGPO, and Full FT methods trained for a single epoch using LLaMA-2-7B on the WizardLM dataset.

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: Comparison of optimization methods with random noise analysis results for the LLaMA-3.1-8B-Base.

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	$\mathbf{7.51}_{\pm 0.07}$	$\mathbf{70.23}_{\pm 1.08}$	$\mathbf{45.13}_{\pm 0.63}$

4.5 Comparative Analysis with Random Noise

In this experiment, we evaluated several optimization methods using the LLaMA-3.1-8B-Base model (Dubey et al., 2024), with a fixed LoRA rank of 8 (alpha = 16). The methods compared include Full FT, LoRA, LoRA combined with Random Noise (Li et al., 2024b), and our proposed LoRA-MGPO. The model was fine-tuned on a 100K subset of the MetaMathQA (Yu et al., 2024), Code-Feedback (Zheng et al., 2024b), and WizardLM datasets, targeting mathematical reasoning, coding, and dialogue tasks, respectively. Training was performed using a peak learning rate of 5e-5 with BF16 mixed precision. Performance metrics for each method are summarized in Table 4, with task-specific evaluations: accuracy on GSM8k for mathematical reasoning, PASS@1 on HumanEval for coding, and average MTBench scores for dialogue generation, as detailed in (Zheng et al., 2023).

The results indicate that methods incorporating random noise (LoRA + Random Noise) yield minor improvements on certain tasks but show inconsistent performance across evaluation metrics. In contrast, LoRA-MGPO consistently outperforms all other methods on MTBench, GSM8k, and HumanEval. These findings suggest that LoRA-MGPO is more effective at enhancing model robustness, particularly in noisy environments, and provides a more reliable optimization strategy than traditional random noise-based methods.

4.6 Computational Cost Analysis

We compare the computational cost and overall performance of LoRA-MGPO, LoRA, and Full FT on NLG tasks. The experiments are conducted on a single NVIDIA H100-NVLink (96GB) GPU with 1 epoch of training, optimized using DeepSpeed (Rasley et al., 2020) and ZeRO-2 stages. The model is LLaMA-2-7B (Touvron et al., 2023), and the dataset is WizardLM (Xu et al., 2024).

The results, summarized in Table 3, show that Full FT requires the highest computational resources, with 6,738M parameters and over 96 GB of memory. In comparison, both LoRA and LoRA-MGPO use significantly fewer parameters (320M) and consume less memory, with LoRA-MGPO slightly exceeding LoRA in memory usage (90.56 GB vs. 81.73 GB). The training time for LoRA-MGPO is 6 hours and 52 minutes, slightly longer than the 5 hours and 48 minutes for LoRA. Importantly, LoRA-MGPO delivers substantial performance improvements over LoRA. This trade-off between computational cost and performance underscores the practical viability of LoRA-MGPO.

5 Conclusion

This work addresses the double descent phenomenon in LoRA, a parameter-efficient fine-tuning method for LLMs, characterized by non-monotonic training dynamics with transient divergence and delayed convergence. We propose LoRA-MGPO, a framework integrating Momentum-Guided Perturbation Optimization (MGPO), which leverages optimizer momentum vectors to guide perturbation directions and introduces adaptive normalization via gradient magnitude smoothing. This approach stabilizes training, reduces attraction to sharp minima, and maintains computational efficiency. Empirical results show LoRA-MGPO outperforms conventional LoRA and state-of-the-art PEFT methods on natural language tasks, achieving smoother loss curves and accelerated convergence. By resolving LoRA's instability while preserving its efficiency, LoRA-MGPO provides a practical solution for adapting LLMs with minimal parameter updates.

Limitations

First, LoRA-MGPO's reliance on momentum vectors for perturbation directions assumes stable optimizer dynamics, which may limit its effectiveness in early training stages or under highly non-stationary gradient conditions. Second, while adaptive perturbation normalization via EMA-smoothed gradients enhances robustness, its performance could be sensitive to abrupt shifts in gradient magnitude distributions, potentially requiring task-specific adjustments to smoothing hyperparameters.

References

Aristotelis Ballas and Christos Diou. 2025. Gradient-guided annealing for domain generalization. *arXiv preprint arXiv:2502.20162*.

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, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. 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.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proc. Int. Workshop Paraphrasing*.

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.

Yingqing Guo, Hui Yuan, Yukang Yang, Minshuo Chen, and Mengdi Wang. 2024. Gradient guidance for diffusion models: An optimization perspective. *arXiv preprint arXiv:2404.14743*.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Preprint*, arXiv:1502.01852.

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*.

Bingcong Li, Liang Zhang, and Niao He. 2024a. Implicit regularization of sharpness-aware minimization for scale-invariant problems. *arXiv preprint arXiv:2410.14802*.

Tao Li, Zhengbao He, Yujun Li, Yasheng Wang, Lifeng Shang, and Xiaolin Huang. 2024b. 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. 2024c. Revisiting random weight perturbation for efficiently improving generalization. *arXiv preprint arXiv:2404.00357*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Gang Liu, Hongyang Li, Zerui He, and Shenjun Zhong. 2024a. Enhancing generalization in medical visual question answering tasks via gradient-guided model perturbation. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2220–2224. IEEE.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. 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.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. pages 784–789.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *SIGKDD*, pages 3505–3506.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. pages 1631–1642.

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.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguist.*, 7:625–641.

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*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proc. Conf. North Am. Chapter Assoc. Comput. Linguist.*, pages 1112–1122.

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*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

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. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

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 of Datasets

A.1 Details of Models

In this work, we primarily utilize two pre-trained language models: LLaMA-2-7B and T5-base. Below, we provide a brief overview of these models along with their respective configurations.

- **LLaMA-2-7B**: This is a large language model developed by Meta, featuring 7 billion parameters. It is part of the LLaMA-2 series, which is known for its strong performance across various natural language understanding and generation tasks. The model architecture is based on the transformer decoder, making it particularly effective for autoregressive tasks such as text generation. More details about the model can be found on its Hugging Face page[*].

- **T5-base**: The Text-to-Text Transfer Transformer (T5) is a versatile encoder-decoder model introduced by Google. The base version of T5 consists of approximately 220 million parameters and is widely used for a variety of tasks, including translation, summarization, and question-answering. Its unified text-to-text framework allows for seamless adaptation to different downstream applications. Additional information about T5-base is available on its Hugging Face repository[†].

Both models were fine-tuned on our specific datasets to align with this study's objectives experiments were conducted using the implementations provided by Hugging Face's Transformers library.

[*] 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

A.2 Details of Datasets

Table 5 summarizes the GLUE benchmark datasets and their respective evaluation metrics (Wang et al., 2018). The GLUE benchmark encompasses a variety of natural language understanding tasks, such as grammatical correctness (CoLA (Warstadt et al., 2019)), sentiment classification (SST-2 (Socher et al., 2013)), paraphrase identification (MRPC (Dolan and Brockett, 2005)), natural language inference (MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2018)). The datasets differ significantly in size, with training examples ranging from as many as 393,000 in MNLI. These tasks involve either binary or multi-class classification. Each task uses specific evaluation metrics, including accuracy, F1 score, Matthews correlation coefficient, and Pearson/Spearman correlation coefficients, depending on the nature of the task. This extensive collection serves as a standardized benchmark for evaluating and comparing model performance across a broad spectrum of linguistic challenges.

Specific metrics are applied to natural language generation (NLG) tasks. For instance, Accuracy is used for GSM8K; Pass@1 is employed for HumanEval, representing the proportion of initially generated code snippets that successfully pass all unit tests; and GPT-4 Evaluation is utilized for MT-Bench, where GPT-4 evaluates the quality of the model's outputs.

B Baselines

To evaluate the effectiveness of LoRA-MGPO, we compare it with several baseline approaches. First, **Full Fine-Tuning** updates all model parameters but demands substantial computational resources, making it impractical for LLMs. In contrast, **Vanilla LoRA** (Hu et al., 2021) incorporates a low-rank matrix product BA into linear layers, where A is initialized using Kaiming initialization and B is set to zero, providing a more efficient alternative in terms of parameter usage.

Next, we examine **LoRA variants with structural modifications**, which introduce architectural changes to enhance performance. For instance, **DoRA** (Liu et al., 2024b) improves model expressiveness by incorporating learnable magnitudes, offering greater flexibility in representation. Meanwhile, **AdaLoRA** (Zhang et al., 2023) dynamically prunes less significant weights during fine-tuning using singular value decomposition (SVD), reallocating rank to critical areas within a fixed parameter budget to optimize resource utilization.

Finally, we explore **LoRA variants that retain the original structure**, which refine the implementation while preserving the fundamental framework. For example, **rsLoRA** (Kalajdzievski, 2023) introduces a scaling factor to stabilize the scale of LoRA during training. **LoRA+** (Hayou et al., 2024) applies separate learning rates for matrices A and B, enhancing adaptability and training efficiency. Additionally, **PiSSA** (Meng et al., 2024) leverages singular value decomposition (SVD) on the weight matrix W at the start of training, initializing A and B based on components with larger singular values to improve initialization quality. Furthermore, **LoRA-GA** (Wang et al., 2024a) accelerates convergence by approximating low-rank matrix gradients with those of full fine-tuning, ensuring better gradient flow throughout training. Similarly, **LoRA-Pro** (Wang et al., 2024b) dynamically adjusts the gradient in each optimization step to ensure that the parameter update trajectory of the entire training process is approximated with full fine-tuning, rather than just the initialization phase.