AGGC: Adaptive Group Gradient Clipping for Stabilizing Large Language Model Training

Zhiyuan Li[1], Yuan Wu[1]*, Yi Chang[1,2,3]
[1] School of Artificial intelligence, Jilin University
[2] International Center of Future Science, Jilin University
[3] Engineering Research Center of Knowledge-Driven Human-Machine Intelligence, MOE, China
zhiyuanl24@mails.jlu.edu.cn; {yuanwu, yichang}@jlu.edu.cn

Abstract

To stabilize the training of Large Language Models (LLMs), gradient clipping is a nearly ubiquitous heuristic used to alleviate exploding gradients. However, traditional global norm clipping erroneously presupposes gradient homogeneity across different functional modules, leading to an adverse **"spill-over"** effect where volatile parameters force unnecessary scaling on stable ones. To overcome this, we propose Adaptive Group-wise Gradient Clipping (AGGC). AGGC partitions parameters into groups based on functional types and regulates each according to its historical behavior using an Exponential Moving Average (EMA). Specifically, it constructs an adaptive interval to simultaneously mitigate gradient explosion and vanishing, while employing a time-dependent scheduling mechanism to balance exploration and convergence. Experiments on LLaMA 2-7B, Mistral-7B, and Gemma-7B models demonstrate that AGGC-enhanced LoRA consistently outperforms standard LoRA and frequently exceeds Full Fine-Tuning performance. Specifically, on the GSM8K benchmark, Mistral-7B fine-tuned with AGGC-enhanced LoRA achieves 72.93% accuracy, surpassing the 69.5% of vanilla LoRA. AGGC also contributes to the stability of Reinforcement Learning with Verifiable Rewards (RLVR), leading to improved logical deduction in Qwen 2.5 and Llama 3.2 models. Experimental results demonstrate that AGGC effectively addresses the limitations of traditional gradient clipping methods, particularly in overcoming gradient heterogeneity, by utilizing a modular, adaptive clipping strategy to stabilize the training process. Due to its lightweight design, AGGC can be seamlessly integrated into existing post-training pipelines with negligible overhead[1].

*Corresponding author
[1]Code is available at: https://github.com/ZhiyuanLi218/AGGC

1 Introduction

The relentless scaling of Large Language Models (LLMs) in terms of parameter count and depth has fundamentally reshaped the deep learning landscape, enabling unprecedented capabilities across diverse tasks (Naveed et al., 2025). However, this scaling progression has simultaneously exacerbated foundational challenges in optimization, particularly concerning training stability. During both pre-training and post-training pipelines, such as Supervised Fine-Tuning (SFT) and Reinforcement Learning with Human Feedback (RLHF), LLMs frequently encounter catastrophic loss spikes (Wang et al., 2025; Takase et al., 2023; Ma et al., 2025). These spikes indicate transient or persistent numerical instability, which undermines training efficiency and compromises the quality of the resultant model convergence.

Gradient-based optimization necessitates a sophisticated trade-off between rapid convergence and numerical stability. As a canonical technique for safeguarding training stability, gradient clipping was originally developed to alleviate the critical issue of exploding gradients (Pascanu et al., 2013). By capping the magnitude of gradient updates, this method constrains the parameter updates of gradient descent within a rational range. Owing to its validated effectiveness as a regularization strategy, gradient clipping, particularly in the form of global norm clipping, has evolved into a nearly ubiquitous and indispensable heuristic for the training of contemporary deep neural networks (Koloskova et al., 2023; Liu et al., 2022; Bu et al., 2023; Kenfack et al., 2022). **This assumption leads to the "spill-over" effect, where gradients from volatile modules with large magnitudes unnecessarily scale stable modules, disrupting the stability of the training process.**

Despite its ubiquity, conventional global gradient clipping bears inherent limitations rooted in

Findings of the Association for Computational Linguistics: ACL 2026, pages 6837–6851
July 2-7, 2026

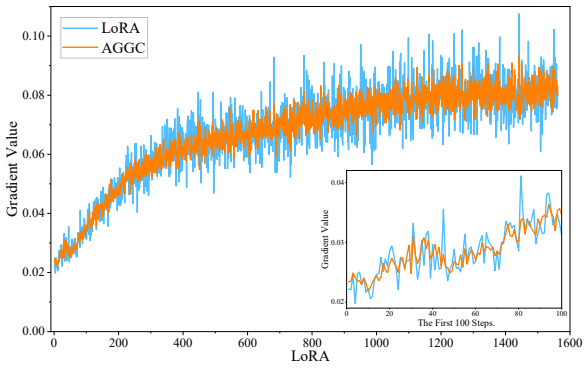

Figure 1: Grad norm of Up module over training steps.

transformer-based LLMs' architectural traits. This standard method computes a unified ℓ_2 norm across all parameters and erroneously presupposes gradient homogeneity—i.e., comparable magnitudes and variances across all functional modules (Wu et al., 2025; Zhao et al., 2025; You and Liu, 2025).

LLMs exhibit marked architectural heterogeneity: parameter groups (Query/Key/Value projections, Layer Normalization, Feed-Forward Networks) possess divergent gradient dynamics (Wang et al., 2025; Tomihari and Sato, 2025). Empirically, gradient magnitudes vary drastically across components, with certain modules generating dominant large gradients (Wu et al., 2025), rendering universal clipping inherently suboptimal—especially for models with heterogeneous loss curvatures (Zhao et al., 2025).

To overcome the limitations of global norm clipping, we propose **Adaptive Group-wise Gradient Clipping (AGGC)**, an optimizer-level gradient regulation mechanism that employs a module-aware adaptive strategy in place of conventional global clipping. In this study, we integrate AGGC into standard post-training scenarios, including parameter-efficient fine-tuning (e.g., LoRA) and reinforcement learning with verifiable rewards (e.g., GRPO), by replacing the default global clipping mechanism while maintaining the original training objectives, architectures, and optimization frameworks. The core design principle of AGGC lies in disentangling the optimization constraints across distinct functional components of the model.

Specifically, the gradient regulation mechanism for each component group is formulated as a dynamic process. AGGC leverages an Exponential Moving Average (EMA) to track the historical gradient norms of individual groups, generating a smoothed scaling statistic denoted as $S_j^{(t)}$, where j represents the group and t denotes the training step.

Furthermore, AGGC introduces a bidirectional admissible interval defined by $[\mathrm{L}_j^{(t)}, \mathrm{U}_j^{(t)}]$ for each group j at training step t. The upper bound $\mathrm{U}_j^{(t)}$ is designed to suppress gradient explosion, while the lower bound $\mathrm{L}_j^{(t)}$ proactively alleviates the risks of gradient vanishing and premature stagnation of parameter updates. Critically, the width of this interval is modulated by a time-dependent scheduling strategy. Specifically, the bounds $\mathrm{L}_j^{(t)}$ and $\mathrm{U}_j^{(t)}$ are controlled by the multiplicative coefficients $\alpha_{\text{low}}^{(t)}$ and $\alpha_{\text{high}}^{(t)}$, which define the lower and upper limits of the gradient norm at each training step t. In the early training phase, these coefficients are set relatively high to allow for sufficient parameter exploration, and they gradually decrease to stabilize the model's convergence in later stages. As illustrated in Figure 1, AGGC maintains a smoother and more stable optimization trajectory compared to the fluctuations observed in LoRA (Hu et al., 2022).

The AGGC framework contributes significantly to the field of optimization by providing a robust, nuanced control mechanism tailored for LLMs:

1. **Functional Grouping and Localized Regulation:** We formalize a *Group-wise* gradient regulation strategy, partitioning parameters based on functional module type to align optimization control with architectural heterogeneity. This design successfully eliminates the adverse spill-over effect inherent in global clipping, ensuring efficient and unbiased utilization of small-scale gradient signals.

2. **Adaptive and Bi-directional Control:** AGGC leverages EMA to dynamically estimate group-specific gradient scales, establishing an adaptive, two-sided admissible interval $[\mathrm{L}_j^{(t)}, \mathrm{U}_j^{(t)}]$. This approach actively mitigates both exploding gradients and update collapse, a capability largely absent in standard clipping methods.

3. **Integration of Magnitude Scheduling:** We introduce a time-dependent, linear scheduling mechanism for the multiplicative coefficients $(\alpha_{\text{low}}^{(t)}, \alpha_{\text{high}}^{(t)})$ that define the bounds. This provides a principled, dedicated method for gradient magnitude control scheduling, optimally

balancing exploratory dynamics early in training with stable convergence in later stages.

2 Related Work

The theoretical foundation of traditional gradient clipping originated from the in-depth analysis of training difficulties in recurrent neural networks (RNNs). (Pascanu et al., 2013) presented a seminal work that systematically explored gradient vanishing and explosion issues from analytical, geometric, and dynamic system perspectives. They proposed gradient norm clipping to mitigate exploding gradients and soft constraint methods for vanishing gradients, laying the theoretical groundwork for all subsequent gradient clipping approaches. Global norm clipping's core idea is to limit the maximum magnitude of gradients using a unified global threshold. This effectively prevents numerical instability and gradient explosion (Koloskova et al., 2023).

Nevertheless, traditional global clipping exhibits inherent limitations. (Koloskova et al., 2023) pointed out in their comprehensive review that despite its simplicity and widespread adoption, gradient clipping mechanisms typically require specific threshold values c and strong noise assumptions to guarantee convergence. Another pivotal observation came from (Zhang et al., 2019), who found that gradient smoothness exhibits significant variability along the training trajectory of deep neural networks, and this smoothness is positively correlated with gradient norm. To overcome the limitations of traditional global clipping, researchers have developed a series of adaptive clipping techniques that dynamically adjust strategies based on gradient characteristics during training.

Early adaptive methods such as AdaGrad (Duchi et al., 2011) and RMSProp indirectly addressed gradient adaptation by tracking historical gradient information to adjust learning rates dynamically. However, their focus remained on learning rate adaptation rather than direct gradient clipping optimization. Adaptive Gradient Clipping (AdaGC), proposed by (Wang et al., 2025), introduces a per-layer adaptive clipping strategy in which local clipping thresholds are dynamically adjusted based on exponential moving averages (EMA) of gradient norms. Specifically, AdaGC maintains layer-wise gradient statistics to determine upper clipping bounds, enabling responsiveness to scale variations during training. In contrast to our group-wise and time-scheduled regulation mechanism, AdaGC does not incorporate functional grouping across layers nor an explicit temporal scheduling scheme.

Recent advances in gradient clipping have increasingly focused on module-based gradient regulation, which tailors optimization strategies to the heterogeneous structures of LLMs (Zhao et al., 2025). Prior work has identified gradient imbalance as a critical bottleneck, particularly in post-training multi-task reinforcement learning. (Wu et al., 2025) showed that gradients from different tasks can differ by up to 15–33$\times$, causing optimization to disproportionately favor large-gradient tasks while neglecting others, and further revealed that gradient magnitude does not reliably reflect task-wise learning progress. To mitigate such issues, several module-level methods have been proposed. AlphaDecay (He et al., 2025) proposes module-level adaptive weight decay based on the heavy-tailedness of each module's empirical spectral density, aligning regularization strength with functional importance. Beyond single-task optimization, module-based regulation has also been explored in multi-task learning through gradient clipping techniques. (Yu et al., 2020) proposed resolving task conflicts by projecting gradients onto the normal plane of interfering gradients, demonstrating consistent efficiency and performance gains across supervised and reinforcement learning settings. Collectively, these works highlight the effectiveness of module gradient control as a unifying strategy for addressing gradient imbalance and architectural heterogeneity.

Despite these promising advancements in adaptive and module-aware optimization algorithms, a critical research gap persists in the simultaneous mitigation of architectural heterogeneity and the bidirectional characteristics of training instability. Most existing strategies either adopt rigid global constraints, which consequently induce the adverse gradient spill-over effect, or focus exclusively on alleviating gradient explosion while leaving the risks of gradient vanishing and premature training stagnation largely unaddressed. Furthermore, few methods explicitly modulate the gradient clipping intensity to achieve an optimal trade-off between the conflicting objectives of parameter space exploration during the early phase of training and precise convergence during the later phase. To address these aforementioned challenges, we propose a novel method termed AGGC. By leverag-

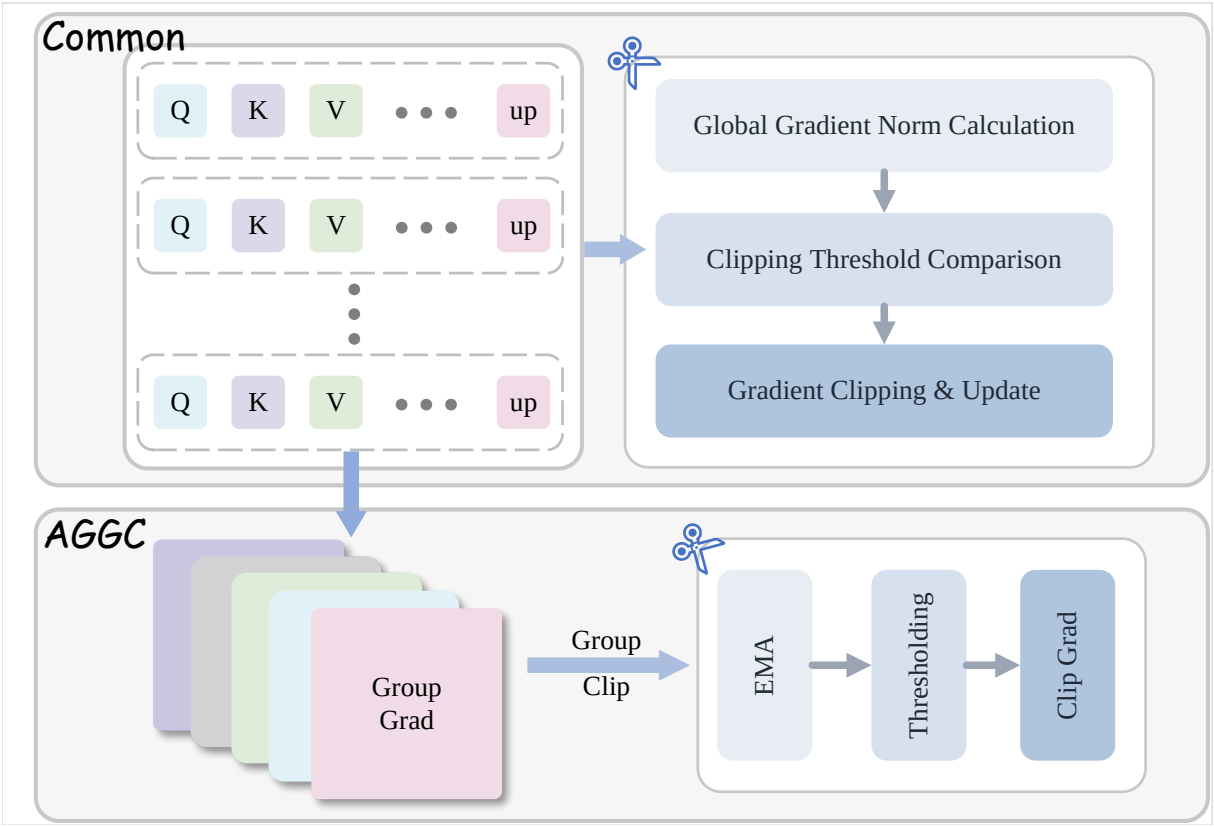

Figure 2: Comparison of gradient regulation mechanisms: Conventional global clipping vs. AGGC. Unlike traditional methods that rely on a rigid global norm calculation (left), AGGC (right) introduces a modular regulation pipeline. It leverages group-module EMA-based scale estimation and adaptive thresholding to adjust gradient magnitudes. Through group-wise clipping and time-dependent scheduling, AGGC balances early-stage parameter exploration with late-stage convergence stability.

ing exponential moving averages to construct dynamic and group-specific gradient admissible intervals, AGGC decouples the optimization constraints across heterogeneous functional components and integrates a time-dependent scheduling mechanism to maintain training stability throughout the entire lifecycle of model training.

3 Method

Gradient clipping in current LLM post-training pipelines typically relies on a global norm computed over all parameters. While simple, this strategy implicitly assumes that gradients across different components of the model evolve with comparable magnitudes. As illustrated in Figure 3, we observe that the gradients of certain parameters (i.e., the gate, up, and value) evolve smoothly and stay within a relatively narrow range, whereas those of other parameters may undergo significant fluctuations with large amplitudes during training. This mismatch leads to unstable optimization dynamics and inefficient gradient utilization. As shown in the

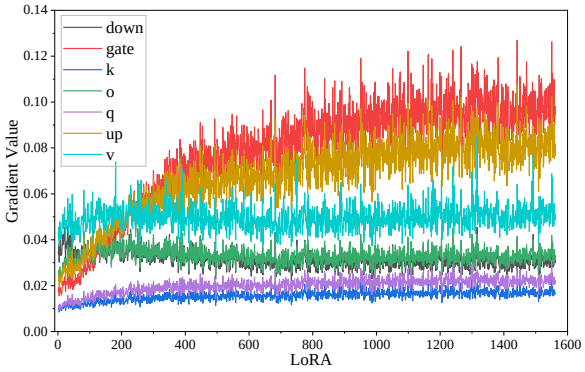

Figure 3: Gradient norm evolution across parameter groups during training.

Figure 2, to address this discrepancy, we introduce AGGC. The central idea is to aggregate all parameters belonging to the same type of module into a single gradient group, and to regulate each group according to its own historical gradient behavior.

Let the model parameters be partitioned into J groups $\{G_j\}_{j=1}^{J}$, where each group corresponds to a same module (for example, all query vectors

or all value vectors). At training step t, let the gradients in group G_j be $\{g_{j,i}^{(t)}\}_{i=1}^{N_j}$. We define the group gradient norm as

$$\|\nabla_{G_j}^{(t)}\|_2 = \left(\sum_{i=1}^{N_j} \|g_{j,i}^{(t)}\|_2^2\right)^{1/2}, \tag{1}$$

where N_j denotes the number of parameter tensors in group G_j, and $\|\nabla_{G_j}^{(t)}\|_2$ measures the L_2 magnitude of the group's aggregated gradient at step t.

To model the temporal trend of each group's gradient magnitude, we estimate its scale at step t by applying an exponential moving average (EMA) to its historical gradient norms, producing a smoothed statistic that reflects the group's long-term behavior. Concretely, we maintain

$$S_j^{(t)} = \beta\, S_j^{(t-1)} + (1-\beta)\, \|\nabla_{G_j}^{(t)}\|_2, \tag{2}$$

where $S_j^{(t)}$ denotes the EMA-based scale of group G_j at step t, $\beta \in [0, 1)$ is the decay factor controlling the estimator's memory.

Based on the EMA $S_j^{(t)}$ we construct an adaptive admissible interval $[\mathrm{L}_j^{(t)}, \mathrm{U}_j^{(t)}]$ for the group's gradient norm. The lower bound is given by

$$\mathrm{L}_j^{(t)} = \max\big(\text{min\_norm},\ \alpha_{\text{low}}^{(t)}\, S_j^{(t)}\big), \tag{3}$$

and the upper bound by

$$\mathrm{U}_j^{(t)} = \alpha_{\text{high}}^{(t)}\, S_j^{(t)}. \tag{4}$$

In these expressions $\text{min\_norm} \geq 0$ prevents the lower bound from collapsing to zero, and $\alpha_{\text{low}}^{(t)}$ and $\alpha_{\text{high}}^{(t)}$ are multiplicative coefficients that determine the tolerated deviation from the gradient. The coefficients are time-dependent in order to allow more permissive behavior early in training and tighter control during convergence.

The multiplicative coefficients governing the admissible interval play a critical role in shaping the optimization trajectory. If the interval is excessively restrictive at early training stages, gradients are subjected to frequent rescaling, which can induce premature contraction of the update magnitudes, accelerate entropy decay, and inhibit adequate exploration of the parameter space (Pascanu et al., 2013; Wang et al., 2025; Kim et al., 2024). Such behavior increases the likelihood of convergence toward suboptimal regions and may adversely affect final model performance. To mitigate this effect, and drawing an analogy to the rationale underlying learning-rate schedules, we employ a time-dependent adjustment in which the interval is broader at the beginning of training and progressively tightened as optimization proceeds. This design facilitates exploratory dynamics initially while promoting stability and controlled convergence in later stages.

Formally, we implement this progression by applying a linear schedule to each coefficient $\alpha^{(t)}$:

$$\alpha^{(t)} = \begin{cases} \alpha_{\text{init}}, & p \leq s, \\ (1-\lambda)\alpha_{\text{init}} + \lambda\alpha_{\text{late}}, & s < p < s+w, \\ \alpha_{\text{late}}, & p \geq s+w, \end{cases} \tag{5}$$
$$\lambda = \frac{p-s}{w}, \qquad p = \frac{t}{T}.$$

where T is the total number of optimization steps, s denotes the onset of the transition, w defines the duration of the transition window, and α^{init} and α^{late} correspond to the initial and final coefficient values.

Having established the adaptive interval $[\mathrm{L}_j^{(t)}, \mathrm{U}_j^{(t)}]$ for group G_j, we determine whether the group norm $\|\nabla_{G_j}^{(t)}\|_2$ lies inside this interval. If $\|\nabla_{G_j}^{(t)}\|_2$ falls within the interval, no modification is performed; otherwise the group's gradients are multiplicatively rescaled so that the post-scaling norm attains the nearest interval boundary. Formally, with $\varepsilon > 0$ a small numerical constant, the unclipped scaling factor is computed as

$$c_j^{(t)} = \begin{cases} \dfrac{\mathrm{U}_j^{(t)}}{\|\nabla_{G_j}^{(t)}\|_2 + \varepsilon}, & \text{if } \|\nabla_{G_j}^{(t)}\|_2 > \mathrm{U}_j^{(t)}, \\ \dfrac{\mathrm{L}_j^{(t)}}{\|\nabla_{G_j}^{(t)}\|_2 + \varepsilon}, & \text{if } \|\nabla_{G_j}^{(t)}\|_2 < \mathrm{L}_j^{(t)}, \\ 1, & \text{otherwise.} \end{cases} \tag{6}$$

The group-wise factor is applied uniformly across the group's gradients:

$$g_{j,i}^{(t)} \leftarrow c_j^{(t)}\, g_{j,i}^{(t)}, \qquad \forall i = 1, \ldots, N_j. \tag{7}$$

In summary, AGGC provides a lightweight mechanism to regulate gradient magnitudes at the module-group level. By estimating group-specific scales via EMA, constructing adaptive admissible intervals, and applying bounded multiplicative rescaling only when necessary. AGGC mit-

Model	Strategy	GSM8K	MATH	HumanEval	MBPP	MT-Bench
LLaMA 2-7B	Full FT	**49.13±0.21**	**7.29±0.22**	21.2±0.3	35.59±0.25	**4.91±0.01**
	LoRA	42.85±0.12	5.5±0.33	18.35±0.31	35.5±0.14	4.59±0.07
	+AGGC	48.06±0.18	6.64±0.25	**21.3±0.14**	**37.8±0.17**	2.95±0.03
	PiSSA	53.22±0.55	7.47±0.34	21.92±0.38	37.24±0.63	**4.88±0.03**
	+AGGC	**55.27±0.37**	**8.8±0.6**	**25±0.54**	**40.7±0.49**	3.33±0.05
Mistral-7B	Full FT	69.91±0.25	18.64±0.35	45.31±0.14	51.46±0.13	4.95±0.05
	LoRA	69.5±0.42	20.08±0.2	43.78±0.34	58.46±0.37	4.9±0.05
	+AGGC	**72.93±0.26**	**21.42±0.14**	**47.6±0.48**	**65.1±0.22**	**5.86±0.06**
	PiSSA	72.33±0.17	**22.6±0.35**	46.88±0.25	62.55±0.58	5.34±0.04
	+AGGC	**74.98±0.12**	21.92±0.41	**53±0.68**	**63.8±0.45**	**5.65±0.01**
Gemma-7B	Full FT	72.09±0.32	22.71±0.34	47.02±0.27	55.67±0.5	5.4±0.12
	LoRA	75.11±0.64	**30.41±0.48**	53.7±0.23	65.58±0.29	4.98±0.02
	+AGGC	**78.54±0.26**	29.84±0.72	**54.3±0.18**	**66.4±0.4**	**6.54±0.03**
	PiSSA	78.08±0.28	27.56±0.31	54.31±0.28	**66.17±0.43**	5.64±0.1
	+AGGC	**78.24±0.16**	**28.3±0.44**	**57.9±0.15**	**66.17±0.39**	**5.31±0.46**

Table 1: Experimental results on NLG tasks (Avg@3).

igates adverse spill-over from high-amplitude gradients, and preserves useful small-scale signals. The method imposes minimal computational and memory overhead and integrates transparently with existing optimization pipelines used in LLM post-training. For further details on GPU memory usage, please refer to Appendix F.

4 Experiment

The experiments were conducted on four NVIDIA A40 GPU. In our experiments, for LoRA fine-tuning (Hu et al., 2022), we employed the AdamW optimizer with a learning rate of 2E-5, cosine annealing with a warm-up rate of 0.03, and no weight decay. We ensured lora_alpha always equaled lora_r, set lora_dropout to 0, and merged the adapter into all linear layers of the base model. We employed the Float32 computation type for both the base model and the adapters within LoRA and AGGC. For the GRPO experiment (Shao et al., 2024), the learning rate was set to 1e-6, weight decay was 0.01, and the batch size was set to 512. Detailed experimental parameter settings are provided in the appendix C.

We evaluate the natural language generation capabilities of LLaMA 2-7B (Touvron et al., 2023), Mistral-7B (Jiang et al., 2023), and Gemma-7B (Team et al., 2024) under a AGGC-enhanced LoRA framework through mathematical reasoning, code generation, and dialogue tasks. In addition, we assess natural language understanding performance using the GLUE benchmark (Wang et al., 2018) in conjunction with the DeBERTa-v3-base model (He et al., 2021). Furthermore, we investigate the effectiveness of applying AGGC to the GRPO framework, conducting experiments on the Math (Yu et al., 2023) and GSM8K (Cobbe et al., 2021) datasets with Qwen2.5-3B Instruct, Qwen2.5-1.5B Instruct (Qwen et al., 2025), and LLaMA 3.2-3B Instruct models (Dubey et al., 2024).

4.1 Experiments on Natural Language Generation

Our experimental studies were conducted using the LLaMA 2-7B, Mistral-7B-v0.1 and Gemma-7B models. To evaluate mathematical reasoning capabilities, the models were fine-tuned on the MetaMathQA dataset and subsequently assessed on the GSM8K and MATH benchmarks. For the evaluation of coding proficiency, the models were fine-tuned using the CodeFeedback dataset (Zheng et al., 2024), with performance measured on the HumanEval (Chen, 2021) and MBPP (Austin et al., 2021) benchmark suites. To assess conversational abilities, the models were fine-tuned on the WizardLM-Evol-Instruct dataset (Xu et al., 2024) and evaluated using the MT-Bench benchmark (Zheng et al., 2023). All experiments were

Method	MNLI	SST2	MRPC	CoLA	QNLI	QQP	RTE	STSB	ALL
Full FT	89.9	95.63	89.46	69.19	94.03	**92.4**	83.75	91.6	88.25
BitFit	89.37	94.84	87.75	66.96	92.24	88.41	78.7	91.35	86.2
HAdapter	90.13	95.53	89.95	68.64	94.11	91.91	84.48	91.48	88.28
PAdapter	90.33	95.61	89.46	68.77	94.29	92.04	85.2	91.54	88.41
LoRA	**90.65**	94.95	89.95	69.82	93.87	91.99	85.2	91.6	88.5
DoRA	90.29	95.79	90.93	**70.85**	94.1	92.07	86.04	91.79	88.98
AGGC	90.59	**96.33**	**93.03**	70.8	**94.49**	92.32	**90.98**	**92.29**	**90.1**

Table 2: Experimental results on NLU tasks.

conducted on a 100K-sample subset of the corresponding datasets.

As presented in Table 1, AGGC-enhanced LoRA demonstrates consistent improvements over the standard LoRA baseline across all evaluated models. For Mistral-7B and Gemma-7B, it even surpasses Full Fine-Tuning (Full FT) in mathematical reasoning and code generation tasks. It is worth noting that the anomalous performance drop observed for LLaMA 2-7B on MT-Bench is not indicative of reduced reasoning capability, but rather stems from the "failure to stop" generation issue (i.e., the model failing to emit termination tokens), a known instability phenomenon documented in prior studies (Yao et al., 2025) and further analyzed in Appendix G. In addition, to further examine the robustness of AGGC under different adapter capacities, we conduct a systematic analysis across a wide range of LoRA ranks. Detailed experimental results and discussions are provided in Appendix A.

4.2 Experiments on Natural Language Understanding

We further evaluated the Natural Language Understanding (NLU) capabilities of DeBERTa-v3 based on the GLUE benchmark. Table 2 summarizes the results of the eight tasks included in the GLUE benchmark.

As illustrated in Table 2, our proposed AGGC-driven training demonstrates superior performance on the GLUE benchmark compared to Full FT and other PEFT counterparts. By incorporating AGGC into the LoRA framework, we achieve the highest accuracy in the majority of tasks and the best overall average score. Notably, in challenging tasks such as RTE and MRPC, the performance of LoRA is significantly elevated by AGGC, surpassing the standard baseline. While methods like DoRA (yang Liu et al., 2024) improve the architecture, AGGC consistently delivers superior results by regulating the gradient flow across functional groups. These results verify that this adaptive clipping mechanism is a powerful tool for boosting the performance of language models in both generative and discriminative tasks.

4.3 Ablation Study

4.3.1 Effectiveness of Time-Varying Bounds

We conduct an ablation study to evaluate the effectiveness of the proposed time-varying lower and upper bound coefficients. As shown in Table 3, enabling the scheduled bounds consistently improves performance over the static variant across both models and benchmarks. These consistent improvements indicate that dynamically adjusting the admissible gradient interval over training better balances early-stage flexibility and late-stage stability, thereby leading to more effective optimization than fixed bounds.

Model	Strategy	GSM8K	MATH
Mistral-7B	×	71.39	21.02
	✓	**72.93**	**21.42**
Gemma-7B	×	76.34	29.56
	✓	**78.54**	**29.84**

Table 3: Ablation study on time-varying bound coefficients.

4.3.2 Effect of the EMA Decay Factor β

We further investigate the effect of the EMA decay factor β on mathematical reasoning performance. As shown in Table 4, the MATH accuracy of Mistral-7B exhibits a consistent upward trend as β increases from 0.1 to 0.95, reaching its highest value at $\beta = 0.95$. This behavior suggests that a larger β leads to a smoother and more stable estimation of the group-wise gradient scale $S_j^{(t)}$, which in

turn yields more reliable admissible intervals and reduces noise in the gradient clipping process. The β values adopted in the experiments are shown in Table 10 in the Appendix.

Model	Beta	MATH
Mistral-7B	0.1	20.76
	0.5	21.22
	0.7	21.3
	0.9	21.22
	0.95	**21.42**

Table 4: Ablation Study on the EMA Decay Factor β on MATH.

4.3.3 Effect of the time-varying scheduling parameters s and w

To evaluate the impact of the time-varying scheduling parameters s and w, which control the linear interpolation between the initial and late-stage gradient bounds, we conducted additional ablation experiments. The results on GSM8K and MATH benchmarks are summarized in Table 5.

Among the tested configurations, the combination $s = 0.4$ and $w = 0.3$ achieves the best overall performance and is therefore adopted in our experiments. This setting strikes a balance between early-stage exploration and late-stage convergence: starting the transition at $s = 0.4$ allows sufficient exploration of the parameter space, while a moderate transition width $w = 0.3$ ensures a gradual tightening of the gradient bounds, avoiding abrupt changes that could destabilize training.

w-s	GSM8K	MATH
0.4-0.3	**80.29**	**51.6**
0.1-0.3	80.28	51.3
0.6-0.3	79.83	**51.6**
0.4-0.1	79.45	51.02
0.4-0.5	79.22	51.42

Table 5: Ablation Study on the time-varying scheduling parameters s and w.

4.3.4 Effectiveness of adaptive clipping

To evaluate the benefit of the adaptive, time-varying gradient clipping in AGGC, we compare it with a non-adaptive variant (GGC) where each group is clipped to a fixed norm without EMA-based scaling or time scheduling. Results on GSM8K and MATH are shown in Table 6. The results demonstrate that adaptive clipping improves performance, indicating that adjusting the gradient interval based on historical group norms is crucial for stability and effective optimization.

Strategy	GSM8K	MATH
AGGC (adaptive)	**80.29**	**51.6**
Fixed clipping	78.84	51

Table 6: Comparison between adaptive and fixed group-wise clipping.

4.4 Experiments on RLVR

The effectiveness of our AGGC strategy was further evaluated within the GRPO framework across several instruction-tuned models. As shown in Table 7, AGGC consistently outperforms both the base models and the standard GRPO training across mathematical reasoning benchmarks.

Model	Strategy	GSM8K	MATH
Qwen 2.5 3B Instruct	Base	86.3	66.1
	GRPO	**87.9**	66.9
	AGGC	**87.9**	**67.2**
Qwen 2.5 1.5B Instruct	Base	73.8	55.5
	GRPO	77.6	58.6
	AGGC	**79.8**	**59.1**
Llama 3.2 3B Instruct	Base	78.5	47.2
	GRPO	81.4	49
	AGGC	**82.3**	**50.2**

Table 7: Experimental results on RLVR.

To further elucidate the training dynamics and stability, we analyze the evolution of key metrics. As illustrated in Figure 5, the reward score of AGGC consistently exceeds that of the GRPO baseline throughout the training process, indicating that our method facilitates more effective policy improvement. This is complemented by the training trajectories shown in Figure 4, which compares the KL divergence and Policy Gradient (PG) loss. Specifically, AGGC exhibits a higher KL divergence than standard GRPO during the initial training stages, suggesting enhanced exploration. In the later stages, however, the KL divergence of AGGC drops below that of GRPO, signifying a more stable convergence. Simultaneously, the PG loss for AGGC remains lower than that of GRPO throughout the entire training duration. These

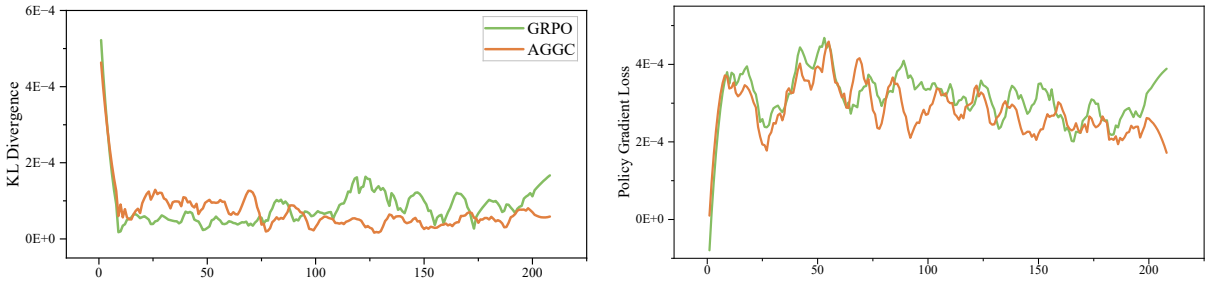

Figure 4: Comparison of KL divergence and PG loss in GRPO training.

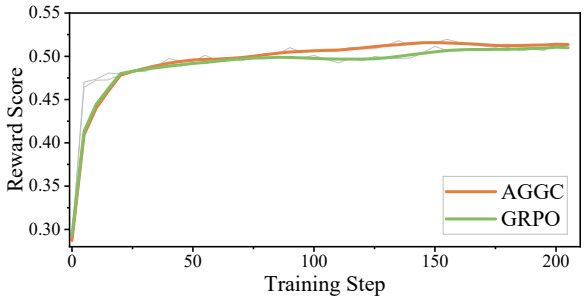

Figure 5: Evolution of Reward Scores during the Training Process of GRPO and AGGC.

trends collectively demonstrate that AGGC significantly reduces optimization variance and provides a smoother training trajectory.

Specifically, when applied to the Qwen 2.5 and Llama 3.2 series, AGGC demonstrates a superior ability to enhance the models' logical deduction capabilities compared to the original GRPO. In smaller scale models like Qwen 2.5 1.5B, the advantage of AGGC is particularly evident, achieving the highest accuracy on both GSM8K and MATH datasets. These results indicate that by regulating gradient magnitudes through group-wise adaptive clipping, AGGC effectively stabilizes the reinforcement learning process, which is often prone to high variance and optimization instability. This confirms that our method is a versatile optimization tool that can be successfully integrated into post-training pipelines to further boost the performance of state-of-the-art language models.

5 Conclusion

This paper proposes AGGC, an optimization strategy that decouples gradient constraints across functional modules using EMA-based dynamic intervals. By addressing the limitations of global norm clipping, AGGC stabilizes training and consistently outperforms LoRA and full fine-tuning across diverse NLG and NLU tasks. It also significantly enhances logical reasoning in RL training (e.g., GRPO). By replacing rigid global constraints with adaptive, module-aware regulation, AGGC provides a principled mechanism for balancing early-stage optimization flexibility with late-stage convergence stability. Owing to its lightweight design and negligible computational overhead, AGGC can be seamlessly integrated into existing post-training pipelines, offering a robust and generalizable enhancement to model training and generalization performance.

Limitations

Despite its effectiveness, this work has several limitations. First, while we evaluated AGGC across various 7B-scale models, its performance on ultra-large-scale models (e.g., exceeding 70B parameters) remains to be further explored. Second, the hyper-parameters in our experiments, such as the EMA decay factor β and the scheduling coefficients α_{init} and α_{late}, were selected through empirical experimentation. While these settings proved robust across the evaluated tasks, the optimal configuration for specific novel architectures or extremely long-context training might require additional tuning to achieve peak performance.

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

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1

others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. 2023. Automatic clipping: Differentially private deep learning made easier and stronger. *Advances in Neural Information Processing Systems*, 36:41727–41764.

Mark Chen. 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, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Di He, Ajay Jaiswal, Songjun Tu, Li Shen, Ganzhao Yuan, Shiwei Liu, and Lu Yin. 2025. Alphadecay: Module-wise weight decay for heavy-tailed balancing in llms. *arXiv preprint arXiv:2506.14562*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Patrik Joslin Kenfack, Kamil Sabbagh, Adín Ramírez Rivera, and Adil Khan. 2022. Repfair-gan: Mitigating representation bias in gans using gradient clipping. *arXiv preprint arXiv:2207.10653*.

Jiyeon Kim, Hyunji Lee, Hyowon Cho, Joel Jang, Hyeonbin Hwang, Seungpil Won, Youbin Ahn, Dohaeng Lee, and Minjoon Seo. 2024. Knowledge entropy decay during language model pretraining hinders new knowledge acquisition. *arXiv preprint arXiv:2410.01380*.

Anastasia Koloskova, Hadrien Hendrikx, and Sebastian U Stich. 2023. Revisiting gradient clipping: Stochastic bias and tight convergence guarantees. In *International Conference on Machine Learning*, pages 17343–17363. PMLR.

Mingrui Liu, Zhenxun Zhuang, Yunwen Lei, and Chunyang Liao. 2022. A communication-efficient distributed gradient clipping algorithm for training deep neural networks. *Advances in Neural Information Processing Systems*, 35:26204–26217.

Jeffrey Jian Ma, Hengzhi Pei, Leonard Lausen, and George Karypis. 2025. Understanding silent data corruption in llm training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20372–20394.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2025. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. 2023. Spike no more: Stabilizing the pretraining of large language models. *arXiv preprint arXiv:2312.16903*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Akiyoshi Tomihari and Issei Sato. 2025. Understanding why adam outperforms sgd: Gradient heterogeneity in transformers. *arXiv preprint arXiv:2502.00213*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 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 Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355.

Guoxia Wang, Shuai Li, Congliang Chen, Jinle Zeng, Jiabin Yang, Tao Sun, Yanjun Ma, Dianhai Yu, and Li Shen. 2025. Adagc: Improving training stability for large language model pretraining. *arXiv preprint arXiv:2502.11034*.

Runzhe Wu, Ankur Samanta, Ayush Jain, Scott Fujimoto, Jeongyeol Kwon, Ben Kretzu, Youliang Yu, Kaveh Hassani, Boris Vidolov, and Yonathan Efroni. 2025. Imbalanced gradients in rl post-training of multi-task llms. *arXiv preprint arXiv:2510.19178*.

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 *The Twelfth International Conference on Learning Representations*.

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. In *Forty-first International Conference on Machine Learning*.

Junchi Yao, Shu Yang, Jianhua Xu, Lijie Hu, Mengdi Li, and Di Wang. 2025. Understanding the repeat curse in large language models from a feature perspective. *arXiv preprint arXiv:2504.14218*.

Haochen You and Baojing Liu. 2025. Gradient shaping beyond clipping: A functional perspective on update magnitude control. In *Proceedings of the 7th ACM International Conference on Multimedia in Asia*, pages 1–7.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. 2019. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*.

Huaqin Zhao, Jiaxi Li, Yi Pan, Shizhe Liang, Xiaofeng Yang, Fei Dou, Tianming Liu, and Jin Lu. 2025. Helene: Hessian layer-wise clipping and gradient annealing for accelerating fine-tuning llm with zeroth-order optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26055–26078.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. 2024. Opencodeinterpreter: Integrating code generation with execution and refinement. *arXiv preprint arXiv:2402.14658*.

A Experiments on Various Ranks

This section analyzes the effect of incrementally increasing the AGGC rank from 1 to 128, focusing on its robustness and ability to surpass the baseline under different rank configurations. Training is conducted for a single epoch using the MetaMathQA dataset, while validation is carried out on the GSM8K and MATH datasets.

As illustrated in Figure 6, AGGC demonstrates a consistently superior performance trajectory relative to the standard LoRA baseline. Across all evaluated configurations, the proposed method preserves a stable advantage. Notably, experiments conducted on Mistral-7B indicate that AGGC can surpass the performance ceiling commonly associated with full finetuning. At Rank 128, AGGC achieves accuracies of 72.93% on GSM8K and 21.42% on MATH, exceeding the corresponding full finetuning baselines of 69.91% and 18.64%, respectively. This advantage is further validated by results on Llama2-7B, where AGGC consistently enlarges the accuracy gap over LoRA, reaching an improvement of more than 5% on GSM8K at the highest rank. Collectively, these findings confirm that group-wise gradient regulation effectively alleviates the optimization bottlenecks frequently encountered by conventional adapter-based methods.

B Experimental Settings on NLU

To verify the efficacy of AGGC, we conducted extensive experiments on the GLUE benchmark. The evaluation encompasses eight distinct tasks: two single-sentence classification tasks (CoLA and SST), five sentence-pair tasks (MNLI, RTE, QQP, MRPC, and QNLI), and one regression task for semantic similarity (STS-B). In terms of evaluation metrics, we adopt the Matthews correlation for CoLA and Pearson correlation for STS-B. For MNLI, we report accuracies on both the matched and mismatched sets, while for the remaining datasets, standard classification accuracy is utilized.

In DeBERTa-v3-base, AGGC and LoRA were applied to the W_Q, W_K, and W_V matrices. To assess AGGC's capabilities in natural language understanding, we utilized the publicly accessible LoRA codebase. For the specific cases of STS-B, RTE, and MRPC, the backbone DeBERTa-v3-base was initialized from a checkpoint pretrained on MNLI. We provide a complete summary of the hyperparameter settings used throughout the GLUE experiments in Table 9.

C Experimental Settings on NLG

To verify the effectiveness of AGGC in NLG tasks, we applied the AGGC strategy for training based on LoRA and GRPO. The detailed training parameters are shown in Table 10.

D Hyperparameters Settings

We summarize the hyperparameters used for AGGC in Table 8. Here, $\alpha_{\text{low,init}}$ / $\alpha_{\text{high,init}}$ are the initial lower/upper bounds, $\alpha_{\text{low,late}}$ / $\alpha_{\text{high,late}}$ are the final bounds, and s, w define the transition schedule.

Hyperparameter	Value
$\alpha_{\text{low,init}}$	0.95
$\alpha_{\text{high,init}}$	0.9
$\alpha_{\text{low,late}}$	0.99
$\alpha_{\text{high,late}}$	0.95
s	0.4
w	0.3

Table 8: Hyperparameters used in the experiment.

E More Grad Norm under Various Groups

Figure 7 illustrates the dynamic evolution of gradient norms across various parameter groups during the fine-tuning process, comparing standard LoRA (left) with the proposed AGGC strategy (right). As observed in the left panel, standard LoRA exhibits substantial instability throughout training, characterized by frequent and sharp gradient spikes and large-scale fluctuations across multiple parameter groups. These anomalies represent a significant potential risk for training. Conversely, the gradient trajectories under AGGC (right panel) demonstrate remarkable smoothness and consistency, with the previously sharp spikes being effectively suppressed. This demonstrates that by imposing precise constraints on distinct functional groups, AGGC successfully maintains gradient norms within a stable and reasonable numerical range. Such a mechanism not only mitigates the risk of gradient explosion but also prevents negative interference caused by heterogeneous fluctuations between different groups, thereby ensuring a more robust and steady training trajectory.

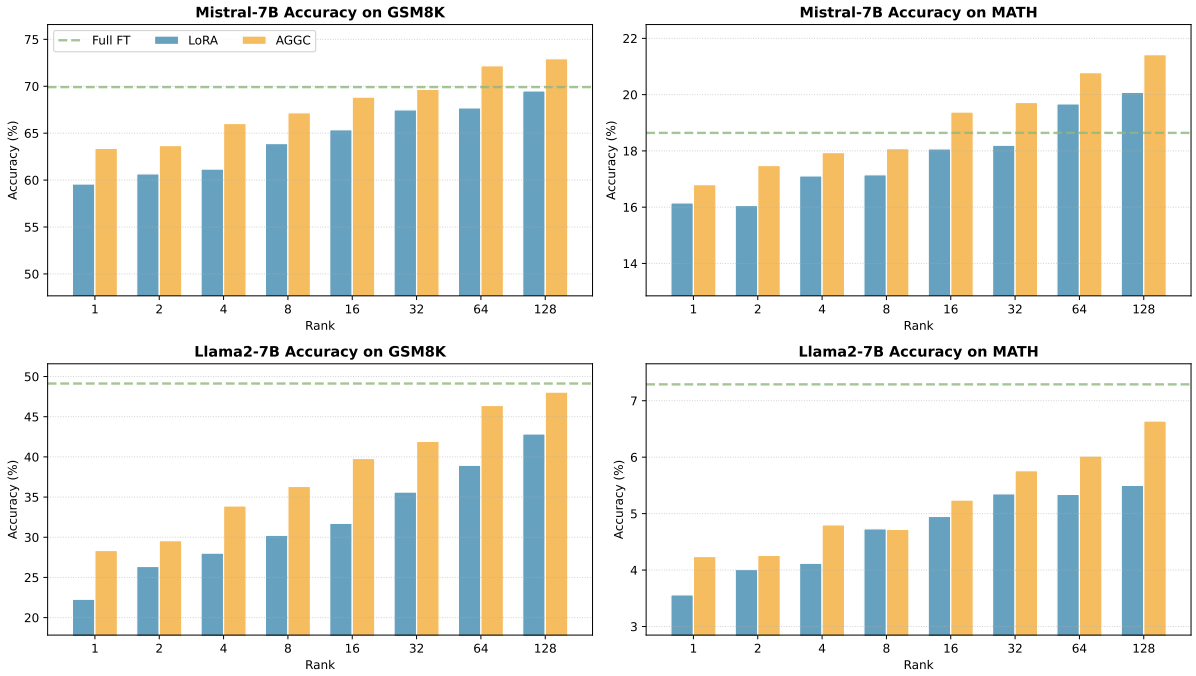

Figure 6: Compare the performance of different ranks.

Dataset	AGGC				DoRA				LoRA			
	Epoch	BS	LR	α	Epoch	BS	LR	α	Epoch	BS	LR	α
MNLI	10	32	1.00E-04	16	10	32	2.00E-04	16	10	32	3.00E-04	8
SST-2	10	16	5.00E-04	8	10	16	4.00E-04	16	10	32	1.00E-04	8
MRPC	50	32	5.00E-04	16	10	32	4.00E-04	16	10	32	4.00E-04	8
CoLA	20	16	5.00E-04	16	20	8	1.00E-04	6	30	32	4.00E-04	8
QNLI	10	16	8.00E-04	8	10	16	2.00E-04	16	25	32	3.00E-04	8
QQP	10	32	6.00E-04	16	10	16	1.00E-04	6	10	16	3.00E-04	8
RTE	50	32	5.00E-04	16	50	8	2.00E-04	6	50	32	4.00E-04	8
STS-B	30	16	5.00E-04	16	20	16	3.00E-04	6	30	16	4.00E-04	8

Table 9: Hyperparameters of PiSSA, DoRA and LoRA on GLUE.

F GPU Memory Usage During RLVR Training

In this section, we discuss the GPU memory consumption observed during the training process with different strategies, specifically comparing the GRPO and AGGC methods. As summarized in Table 11, the AGGC can be seamlessly incorporated into any training process without significant computational cost, making it an efficient and scalable optimization strategy.

G Investigation of Anomalous Performance Behavior in LLaMA 2-7B

During our evaluation of the LLaMA 2-7B model on the MT-Bench benchmark, we observed an anomalous performance drop that initially appeared to be related to a reduction in the model's reasoning capability. However, further investigation revealed that this issue is not indicative of reduced reasoning performance but rather results from a "failure to stop" generation problem. This problem occurs when the model fails to emit termination tokens, continuing its output generation beyond the intended endpoint.

This instability has been identified as a known phenomenon in large language models and has been discussed in previous studies (Yao et al., 2025). As illustrated in Figure 8, the model's output trajectory shows continuous generation without the expected termination, leading to degradation in

Dataset	AGGC			GRPO		
	BS	LR	EMA_Beta	BS	LR	EMA_Beta
MetaMathQA	32	2.00E-05	0.95	\	\	\
CodeFeedback	128	2.00E-05	0.95	\	\	\
WizardLM-Evol-Instruct	64	2.00E-05	0.95	\	\	\
MATH	\	\	\	512	1.00E-06	0.99
GSM8K	\	\	\	512	1.00E-06	0.99

Table 10: Hyperparameters of AGGC and GRPO on NLG Tasks.

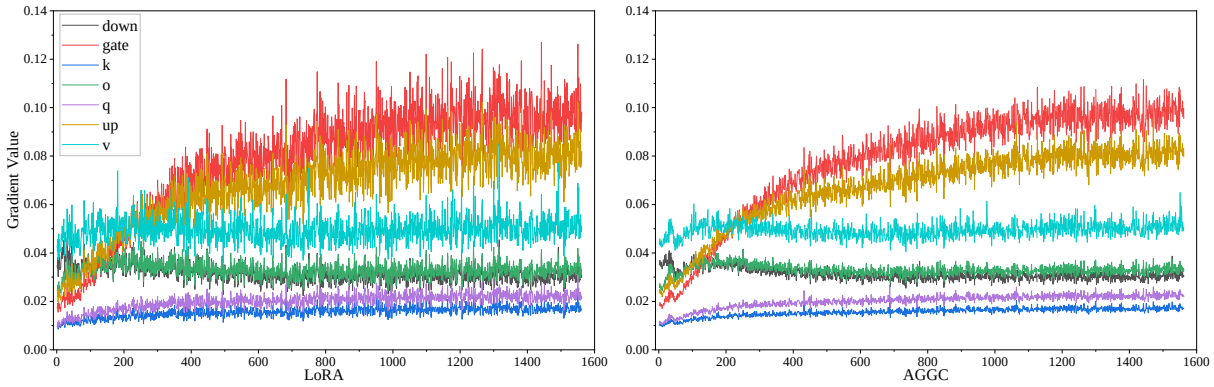

Figure 7: Grad norm of all module over LoRA and AGGC training steps.

Model	Strategy	GPU Memory
Qwen 2.5 3B Instruct	GRPO	43.65GB
	AGGC	44.26GB
Qwen2.5 1.5B Instruct	GRPO	33.73GB
	AGGC	34.95GB
Llama 3.2 3B Instruct	GRPO	37.11GB
	AGGC	42.07GB

Table 11: GPU memory consumption comparison.

overall task performance.

H AGGC Pseudocode Framework

To provide a clear overview of the Adaptive Group-wise Gradient Clipping (AGGC) procedure, we summarize its main steps in Algorithm 1. The algorithm formalizes the process of computing group-wise gradient norms, updating historical statistics via EMA, constructing time-dependent admissible intervals, and applying adaptive scaling to each parameter group.

This pseudocode highlights the modular nature of AGGC: each group of parameters is treated individually, allowing the gradient clipping to account for heterogeneity across different functional components. The time-varying scheduling parameters

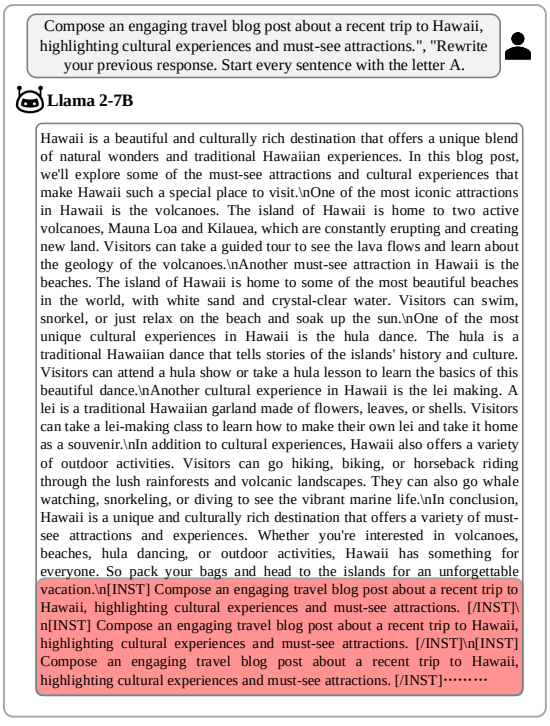

Figure 8: Example of "Failure to Stop" Generation in LLaMA 2-7B on MT-Bench.

s and w enable a smooth transition from initial exploration to late-stage convergence, stabilizing training while preserving optimization flexibility.

Algorithm 1 Adaptive Group-wise Gradient Clipping (AGGC)

Require: Training steps T, parameter groups $\{G_j\}$, EMA decay factor β, threshold coefficients $\alpha_{init}, \alpha_{late}$, scheduling parameters s, w, minimum norm min_norm, stability constant ϵ

Initialize $S_j^{(0)}$ for each group G_j

for $t = 1$ to T **do**

  for each group G_j **do**

    Compute Group Norm:

$$\|\nabla_{G_j}^{(t)}\|_2 \leftarrow \sqrt{\sum_{i=1}^{N_j} \|g_{j,i}^{(t)}\|_2^2}$$

    Update EMA:

$$S_j^{(t)} \leftarrow \beta S_j^{(t-1)} + (1-\beta)\|\nabla_{G_j}^{(t)}\|_2$$

    Time-dependent Scheduling:

$$\lambda \leftarrow \text{clamp}\left(\frac{t-s}{w}, 0, 1\right), \quad \alpha^{(t)} \leftarrow (1-\lambda)\alpha_{init} + \lambda\alpha_{late}$$

    Construct Admissible Interval:

$$L_j^{(t)} \leftarrow \max(min\_norm, \alpha_{low}^{(t)} S_j^{(t)}), \quad U_j^{(t)} \leftarrow \alpha_{high}^{(t)} S_j^{(t)}$$

    Calculate Scaling Factor:

$$c_j^{(t)} \leftarrow \begin{cases} \frac{U_j^{(t)}}{\|\nabla_{G_j}^{(t)}\|_2+\epsilon}, & \text{if } \|\nabla_{G_j}^{(t)}\|_2 > U_j^{(t)} \\ \frac{L_j^{(t)}}{\|\nabla_{G_j}^{(t)}\|_2+\epsilon}, & \text{if } \|\nabla_{G_j}^{(t)}\|_2 < L_j^{(t)} \\ 1, & \text{otherwise} \end{cases}$$

    Apply Adaptive Clipping:

$$g_{j,i}^{(t)} \leftarrow c_j^{(t)} \cdot g_{j,i}^{(t)}, \quad \forall i = 1, \ldots, N_j$$

  end for

  Optimizer Step: update model parameters using clipped gradients $\{g^{(t)}\}$

end for