GRASS: Gradient-based Adaptive Layer-wise Importance Sampling for Memory-efficient Large Language Model Fine-tuning

Kaiyuan Tian[1], Linbo Qiao[1]*, Yu Tang[2], Gongqingjian Jiang[1], Baihui Liu[1], Yifu Gao[1], Xialin Su[1], Dongsheng Li[1]*
[1]National University of Defense Technology
[2]Information Support Force Engineering University
{kyt,qiao.linbo,tangyu14,jianghydro,lbh,gaoyifu,suxialin25,dsli}@nudt.edu.cn

Abstract

Full-parameter fine-tuning of large language models is constrained by substantial GPU memory demands. Low-rank adaptation methods mitigate this challenge by updating only a subset of parameters. However, these approaches often limit model expressiveness and yield lower performance than full-parameter fine-tuning. Layer-wise fine-tuning methods have emerged as an alternative, enabling memory-efficient training through static layer importance sampling strategies. However, these methods overlook variations in layer importance across tasks and training stages, resulting in suboptimal performance on downstream tasks. To address these limitations, we propose GRASS, a gradient-based adaptive layer-wise importance sampling framework. GRASS utilizes mean gradient norms as a task-aware and training-stage-aware metric for estimating layer importance. Furthermore, GRASS adaptively adjusts layer sampling probabilities through an adaptive training strategy. We also introduce a layer-wise optimizer state offloading mechanism to further reduce memory usage while maintaining comparable training throughput. Extensive experiments across multiple models and benchmarks demonstrate that GRASS consistently outperforms state-of-the-art methods, achieving an average accuracy improvement of up to 4.38 points and reducing memory usage by up to 19.97%.

1 Introduction

Large language models (LLMs) serve as the foundation of modern natural language processing, delivering strong performance across diverse tasks (Brown et al., 2020; Li et al., 2021; Chang et al., 2024; Zhu et al., 2024). To adapt LLMs for downstream tasks, full-parameter fine-tuning (FFT) (Howard and Ruder, 2018) is widely adopted. Nevertheless, as model sizes scale, FFT becomes increasingly costly and impractical due to the prohibitive bottleneck of GPU memory (Wang et al., 2025; Tian et al., 2026). Parameter-efficient fine-tuning (PEFT) methods overcome this issue by updating only a small subset of parameters. Among these, LoRA (Hu et al., 2021) has gained widespread popularity because of its effective trade-off between efficiency and model performance. Despite their efficiency, PEFT methods inevitably yield inferior results to FFT. Recent work shows that LoRA (Hu et al., 2021), in particular, suffers from limited representational capacity due to its low-rank parameterization (Ding et al., 2022; Lialin et al., 2023), resulting in degraded performance compared to FFT (Xia et al., 2024).

Unlike LoRA, layer-wise fine-tuning methods (Zhu et al., 2023; Pan et al., 2024; Yao et al., 2024; Li et al., 2025) provide an alternative approach to reducing the memory cost of FFT. These methods avoid low-rank constraints and instead activate and update only a subset of layers during training, thereby reducing memory overhead while preserving the model's full capacity. However, existing layer-wise strategies rely on static, task-agnostic criteria for layer selection, implicitly assuming that layer importance remains constant across tasks and throughout training. Empirical evidence indicates that the relative importance of layers varies across downstream tasks and training stages. As a result, static strategies often misalign with training dynamics and yield suboptimal performance on certain benchmarks, as shown in Table 1.

To overcome these limitations, we propose GRASS, a gradient-based adaptive layer-wise importance sampling framework for memory-efficient LLM fine-tuning. First, we pinpoint the limitations of static layer-wise sampling methods on downstream tasks and compare the contribution of different layers to training. Our findings reveal the varying layer-wise importance across tasks and training stages. Driven by these insights, GRASS

*Corresponding authors.

Findings of the Association for Computational Linguistics: ACL 2026, pages 9777–9790

directly uses dynamic optimization signals rather than static heuristics or task-specific assumptions to guide layer selection. Specifically, GRASS leverages mean gradient norms (MGN) as a task-aware and training-stage-aware indicator to quantify the contribution of each layer to loss reduction. By periodically measuring layer-wise MGN and dynamically updating sampling probabilities, GRASS adaptively samples and activates a subset of layers for fine-tuning, thereby focusing on the most influential layers at different points during training. This mechanism ensures the full-parameter expressiveness of the model. In addition, we introduce a layer-wise optimizer state offloading mechanism to further improve memory efficiency. By overlapping optimizer state transfers with computation, GRASS maintains comparable training throughput.

The contributions of this work are as follows:

- We identify the limitations of the static layer-wise sampling strategy through experiments on different datasets, demonstrating that the importance of each layer varies across tasks and training stages.
- We propose GRASS, a novel gradient-based adaptive layer-wise fine-tuning framework. It assesses the importance of layers based on the mean gradient norms and adopts adaptive layer sampling probabilities to achieve an average performance improvement.
- We further address the memory bottleneck of layer-wise fine-tuning by introducing a layer-wise optimizer state offloading mechanism with overlapping, enabling GRASS to achieve high memory efficiency without sacrificing training throughput.

2 Related Work

2.1 Parameter-efficient Fine-tuning

As LLMs scale, FFT becomes increasingly impractical due to memory constraints. PEFT methods address this challenge by updating only a subset of the pre-trained parameters. Representative PEFT approaches can be categorized into: prompt-based methods (Lester et al., 2021; Li and Liang, 2021; Liu et al., 2022), adapter-based methods (Houlsby et al., 2019; Wang et al., 2022; He et al., 2022), and reparameterization-based methods (Hu et al., 2021; Valipour et al., 2023; Liu et al., 2024). Among them, LoRA (Hu et al., 2021) is widely adopted due to its simplicity and effectiveness. LoRA introduces two low-rank matrices to approximate the incremental parameters, and merges them into the pre-trained weights during inference, incurring no additional computational overhead. Despite its benefits, the expressiveness of LoRA is limited by the number of its trainable parameters and low-rank parameterization, thus falling short of FFT (Lialin et al., 2023; Xia et al., 2024; Zhao et al., 2024). Then, DoRA (Liu et al., 2024) revisits LoRA from a weight decomposition perspective. By explicitly decoupling weight magnitude and direction to enable learnable magnitude, DoRA enhances the expressiveness of LoRA without compromising its parameter efficiency. However, DoRA still uses a limited number of trainable parameters and requires more memory compared to LoRA.

Method	MultiA.	AddSub	GSM8K	AQuA	SingleEq	SVAMP	Avg. ↑
FFT	**94.43**	**66.08**	**47.31**	18.90	**80.51**	**55.50**	**60.46**
LISA	91.17	62.53	42.91	**20.08**	71.65	51.10	56.57

Table 1: Comparison of full-parameter fine-tuning (FFT) and LISA on arithmetic reasoning benchmarks. Although LISA approaches FFT performance on certain tasks, it exhibits notable degradation on others(e.g., GSM8K and SingleEq).

2.2 Static Layer-wise Selective Fine-tuning

Other works aim to reduce GPU memory cost during fine-tuning by updating only a subset of layers. LIFT (Zhu et al., 2023) proposes an extreme approach that updates only one Transformer block per iteration. It shortens the backward depth and reduces optimizer state memory requirements. However, LIFT utilizes a fixed update order (e.g., front-to-end), resulting in the allocation of computational budget to layers with low task relevance. Moreover, LIFT lacks a mechanism for prioritizing more informative layers. Pan et al. (2024) observe a skewed weight-norm distribution in LoRA. Inspired by importance sampling (Zhao and Zhang, 2015), they propose LISA, which uniformly samples a subset of layers to optimize during training. LISA achieves good performance while significantly reducing memory overhead. Nevertheless, the static sampling scheme in LISA inherently struggles to adapt to the varying characteristics across tasks and training stages. Recent advances further explore improved static or heuristic-driven layer selection strategies. IST (Yao et al., 2024) estimates layer importance via response suppression and reinforcement learning. OWS (Li et al., 2025)

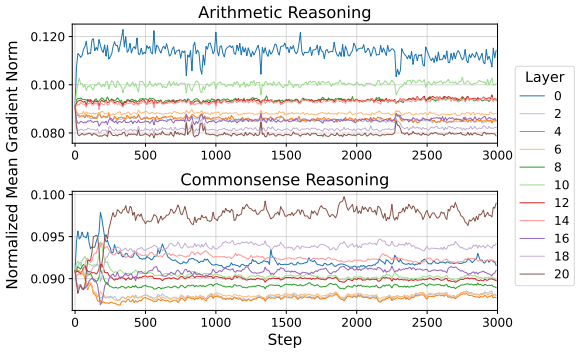

Figure 1: A comparison of normalized layer-wise mean gradient norm across different datasets on TinyLlama.

adopts outlier-weighted sampling with low-rank gradient projection. In contrast, GRASS employs gradient-based importance estimation that reflects real-time training dynamics and adaptively updates sampling probabilities. Our approach enables more task-aligned layer sampling and consistently yields improved performance across various models and benchmarks.

3 Method

3.1 Gradient-based Layer-wise Importance Measure

Selecting which layers to update is crucial to layer-wise fine-tuning. Previous methods, such as LISA, estimate layer importance using a static and weight-based strategy, which implicitly assumes that the relative importance of each layer remains constant across tasks and throughout training. Consequently, it overlooks the fact that layer importance can vary substantially across different downstream tasks and different optimization stages, leading to suboptimal layer selection. For example, as shown in Table 1, although LISA significantly reduces memory consumption, it underperforms FFT on most arithmetic reasoning benchmarks. This performance gap suggests that static strategies can misidentify which layers are truly influential for the current task, thereby limiting optimization effectiveness.

To develop an importance measure that more directly reflects the current training dynamics, we turn to the gradient statistics of each layer. Intuitively, gradients encode how sensitively the loss responds to parameter updates. Under the first-order Taylor approximation, we have:

$$\Delta\mathcal{L} \approx \langle \nabla_{\theta^{(l)}}\mathcal{L}, \Delta\theta^{(l)} \rangle \,, \tag{1}$$

where $\theta^{(l)}$ denotes the parameters of layer l, $\Delta\theta^{(l)}$ the parameter update applied to that layer at a training step, and $\Delta\mathcal{L}$ the corresponding change in training loss. The magnitude of the gradient with respect to layer l quantifies the potential impact of updating that layer on the training objective, indicating higher importance for optimization.

Based on this insight, we define the mean gradient norm (MGN) to measure layer-wise importance by aggregating gradient magnitudes over continuous training steps. Let $\mathbf{g}_t^{(l)}$ denote the gradients of layer l at training step t, $N_p^{(l)}$ the number of parameters in that layer, and T the number of continuous training steps. The MGN for layer l within a period composed of T training steps is defined as:

$$\mathbf{m}_l(T) = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{N_p^{(l)}}\|\mathbf{g}_t^{(l)}\|_2^2}\,. \tag{2}$$

MGN serves as a task-aware and training-stage-aware indicator of layer importance. In our formulation, layers with higher MGN values tend to be more important, as these layers have a greater impact on loss reduction when updated. Detailed discussion of the validity of MGN as a layer-wise importance measure is provided in Appendix B. Figure 1 illustrates the variation of normalized MGN values across layers when fine-tuning TinyLlama on different datasets. We observe significant differences in the relative importance of layers across tasks. Specifically, layer 20 consistently exhibits higher MGN under commonsense reasoning, whereas it is less prominent for arithmetic reasoning. This divergence suggests that layer importance is highly task-dependent and changes continuously as training proceeds. These observations motivate the use of MGN as the core signal for adaptive layer sampling in GRASS. It allows layer importance to be estimated in a task-aware and training-stage-aware manner, rather than relying on static criteria that ignore the evolving optimization dynamics. Additional analyses conducted on different models further confirm that MGN effectively captures layer-wise differentiation and task-specific patterns (see Appendix C.1).

3.2 Layer Sampling with Adaptive Update of Sampling Probabilities

Building upon MGN as a dynamic indicator of layer-wise importance introduces a new challenge: static layer sampling strategies are unable to exploit such time-varying signals. In particular, sampling

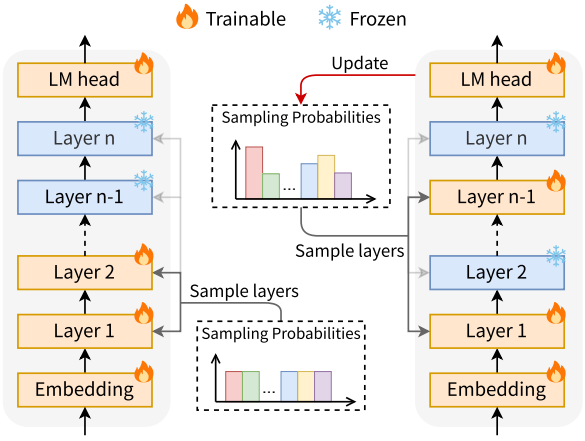

Figure 2: An overview of static layer-wise sampling (Left) and GRASS (Right). GRASS alternates between sampling a subset of model layers for tuning and updating sampling probabilities during training.

decisions based on initial statistics can quickly become misaligned with evolving optimization dynamics, resulting in suboptimal performance (see further discussion and analysis in § 4.4). To address this challenge, GRASS dynamically updates both MGN statistics and layer sampling probabilities throughout training.

GRASS proceeds in two stages: a probing phase for initializing layer-wise MGN values, followed by an adaptive fine-tuning phase guided by these MGNs. At the start of training, GRASS conducts a probing phase for the first T_p steps ($T_p \ll T$) to initialize each layer's MGN. During probing, standard forward and backward passes are performed as in FFT, but parameter updates are omitted. This stage allows us to collect gradient statistics without incurring the memory overhead of optimizer states. The probing phase yields an initial MGN estimate $\mathbf{m}_l(T_p)$ for each layer, which serves as the starting point for subsequent adaptive sampling.

After probing, GRASS transitions to the adaptive fine-tuning phase, where it periodically samples layers for update. At every probability update interval of T_u training steps, the current MGN values are converted into a probability distribution over layers:

$$\{p^{(l)}\}_{l=1}^{N_L} = \frac{\exp(\mathbf{m}_l(T_u)/\tau)}{\sum_{i=1}^{N_L} \exp(\mathbf{m}_i(T_u)/\tau)}, \quad (3)$$

where τ is a temperature hyperparameter controlling the sharpness of the distribution. This softmax-based mapping assigns higher sampling probabilities to layers with larger MGNs, while preserving exploration over less active layers. Based on $\{p^{(l)}\}_{l=1}^{N_L}$, GRASS samples γ layers out of N_L to update during the subsequent sampling period of length T_s, while all other layers remain frozen. The sampled layers in each sampling period are referred to as *trainable* layers, and the remaining layers are considered *frozen*, as shown in Figure 2. During each update interval, only the trainable layers produce gradients while the frozen layers do not. We update MGN using an exponential moving average:

$$\mathbf{m}_l(T) = \alpha \mathbf{m}_l(T_u) + (1-\alpha)\mathbf{m}_l(T - T_u), \quad (4)$$

where α controls the trade-off between responsiveness to newly observed MGNs and stability of the accumulated importance estimates. Frozen layers retain their previous MGN values during the current update period. This adaptive mechanism enables GRASS to progressively emphasize layers that consistently exhibit high importance, while also allowing layer importance to evolve with the training stage and downstream task characteristics.

In summary, GRASS first initializes layer-wise MGN values through a probing stage, then alternates between (i) sampling a small subset of layers for parameter updates and (ii) refreshing their importance estimates based on newly observed gradient statistics. Unlike existing layer-wise sampling methods that rely on static importance scores (Figure 2 (Left)), GRASS dynamically aligns layer selection with the ongoing optimization process by continuously updating sampling probabilities using MGN, as depicted in Figure 2 (Right).

3.3 Layer-wise Optimizer State Offloading

Layer-wise fine-tuning methods introduce a new memory challenge that differs from LoRA-style approaches, which maintain optimizer states for a fixed and small set of parameters. These methods dynamically activate different subsets of model layers during training. Consequently, the optimizer states associated with all potentially trainable layers need to be preserved throughout training. A straightforward solution is to keep all the optimizer states in GPU memory. However, this approach incurs prohibitive memory overhead, undermining the memory efficiency benefits of layer-wise fine-tuning. Alternatively, storing optimizer states on CPU reduces GPU memory consumption but introduces significant communication latency due to frequent data transfers between CPU and GPU. This dilemma between memory and throughput poses a challenge in layer-wise fine-tuning methods.

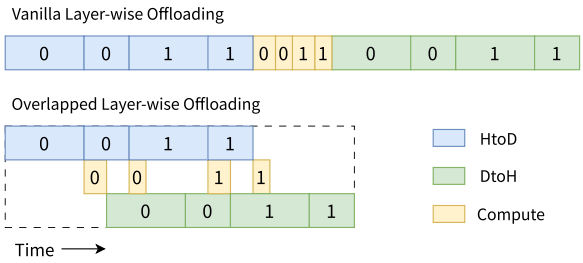

Figure 3: A comparison of vanilla and overlapped layer-wise optimizer state offloading. HtoD refers to host to device, while DtoH represents device to host. Each block with index i stands for a parameter in layer i.

GRASS addresses this dilemma by exploiting a key observation: at any training step, only a small subset of layers (i.e., $\gamma \ll N_L$) are active and require optimizer updates. Therefore, we adopt a layer-wise optimizer state offloading strategy, where optimizer states are managed at the granularity of individual layers. Specifically, at any given time, GPU memory only holds the optimizer states of one active layer that is currently undergoing updates, while the states of remaining layers are stored in CPU memory. To mitigate communication overhead, GRASS asynchronously prefetches optimizer states to perform updates, then puts them back to the CPU immediately after the updates layer by layer. By overlapping data transfers with computation, this approach reduces idle time during optimizer updates, thereby increasing training throughput (as shown in Figure 3), while minimizing the optimizer state memory footprint on the GPU. We provide quantitative analysis of the benefits of this technique in § 4.3.

4 Experiments

4.1 Arithmetic Reasoning

Settings To validate the effectiveness of GRASS, we compare it with various PEFT methods across three language models of different scales on arithmetic reasoning tasks. Specifically, we conduct experiments on TinyLlama (Zhang et al., 2024), Gemma-2B (Team et al., 2024), and LLaMA2-7B (Touvron et al., 2023), covering parameters counts from 1B to 7B. Each model is first fine-tuned with the math dataset collected by Hu et al. (2023), and subsequently evaluated on six arithmetic reasoning benchmarks, including MultiArith (Roy and Roth, 2015), AddSub (Hosseini et al., 2014), GSM8K (Cobbe et al., 2021), AQuA (Ling et al., 2017), SingleEq (Koncel-Kedziorski et al., 2015), and SVAMP (Patel et al., 2021). Additional experimental details are provided in Appendix A.

Results Table 2 reports the performance of different PEFT methods on arithmetic reasoning benchmarks across three model scales. Overall, GRASS yields competitive results across all models, surpassing state-of-the-art methods on most benchmarks, indicating its robustness and effectiveness. Specifically, GRASS consistently achieves higher average accuracy than both LoRA and DoRA. This result suggests that adaptive layer-wise updates offer greater representational flexibility compared to low-rank update methods. Compared to LISA, GRASS provides more consistent improvements across tasks and model scales. Although LISA performs well on certain benchmarks (e.g., AddSub and SingleEq on TinyLlama), its static sampling strategy can lead to performance declines on other tasks. In contrast, GRASS leverages MGN to dynamically adjust layer sampling probabilities during training, capturing both task-dependent and stage-dependent layer importance. This adaptive assessment allows GRASS to accommodate diverse tasks, leading to more balanced performance gains. Notably, GRASS outperforms FFT on both TinyLlama and Gemma-2B, and achieves an average accuracy improvement of up to 4.38 points over $\text{LoRA}_{\text{r=128}}$. These results suggest that adaptive layer-wise fine-tuning may serve as an implicit regularizer, consistent with findings from prior works (Pan et al., 2024).

4.2 Commonsense Reasoning

Settings To further demonstrate the generalization of GRASS across diverse tasks, we conduct fine-tuning experiments on commonsense reasoning. Following the methodology of Hu et al. (2023), we employ a training dataset comprising eight tasks and conduct evaluations on the individual test dataset for each task, respectively. The commonsense reasoning tasks include BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-c/e (Clark et al., 2018), and OBQA (Mihaylov et al., 2018).

Results As shown in Table 3, GRASS delivers consistent improvements across all three model scales on commonsense reasoning tasks. On TinyLlama, GRASS improves the average accuracy to 38.59%, outperforming LoRA, DoRA, and LISA.

Model	Method	MultiArith	AddSub	GSM8K	AQuA	SingleEq	SVAMP	Avg. ↑
TinyLlama	FFT	64.17	40.76	15.16	13.78	42.92	24.10	33.48
	$\text{LoRA}_{r=128}$	61.17	25.32	15.16	12.20	38.19	27.00	29.84
	$\text{LoRA}_{r=256}$	65.50	30.89	17.44	12.60	37.59	28.20	32.04
	DoRA	67.83	34.68	**18.35**	12.99	36.81	**29.50**	33.36
	LISA	65.00	**38.04**	17.74	12.99	**43.11**	24.90	33.63
	GRASS	**68.00**	35.19	17.13	**15.16**	42.52	27.30	**34.22**
Gemma-2B	FFT	86.67	66.84	42.53	32.68	80.12	52.10	60.16
	$\text{LoRA}_{r=128}$	91.50	66.84	42.61	25.20	77.36	49.00	58.75
	$\text{LoRA}_{r=256}$	92.50	65.06	42.53	23.23	**79.33**	52.30	59.16
	DoRA	92.17	**67.09**	42.68	23.62	78.94	50.90	59.23
	LISA	90.17	62.53	40.18	21.26	75.00	49.60	56.46
	GRASS	**93.50**	66.33	**43.06**	**29.13**	78.35	**53.50**	**60.65**
LLaMA2-7B	FFT	94.43	66.08	47.31	18.90	80.51	55.50	60.46
	$\text{LoRA}_{r=128}$	90.50	62.28	46.85	19.68	**78.74**	54.90	58.83
	$\text{LoRA}_{r=256}$	90.50	63.04	46.87	19.69	78.54	**56.00**	59.11
	DoRA	**93.16**	63.03	**48.52**	21.65	75.59	53.40	59.23
	LISA	91.17	62.53	42.91	20.08	71.65	51.10	56.57
	GRASS	92.00	**67.09**	42.15	**23.62**	77.56	55.10	**59.59**

Table 2: Accuracy comparison of multiple LLMs and PEFT methods on six math reasoning benchmarks. The best and second best results are marked in **bold** and underlined, respectively.

This improvement demonstrates that adaptive layer selection remains advantageous even for smaller models with limited parameter capacity, where static or task-agnostic update strategies may waste optimization effort on less relevant layers. For LLaMA-2-7B, GRASS achieves the closest performance to FFT (76.30% vs. 77.80%), outperforming all other PEFT methods. These results highlight that GRASS can adapt its sampling policy based on gradients and focus on layers that exhibit higher task-specific importance, yielding superior performance. Overall, GRASS demonstrates strong generalization beyond arithmetic reasoning and provides robust improvements on various commonsense reasoning benchmarks.

4.3 Memory Efficiency

We conduct experiments on the peak GPU memory consumption of different fine-tuning methods across several scenarios to showcase the memory efficiency of GRASS.

Table 4 presents the peak GPU memory usage across different model sizes. Compared to reparameterization-based methods such as LoRA and DoRA, GRASS consistently achieves a lower or comparable memory footprint. This advantage stems from a reduction in activation memory, as depicted in Figure 4 (Left). Regarding LISA, GRASS effectively reduces optimizer state memory consumption. Moreover, the memory usage of GRASS exhibits only a marginal increase as the number of active layers γ grows, demonstrating a significant

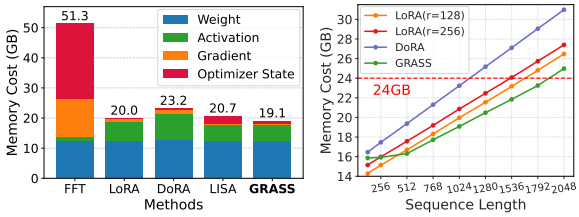

Figure 4: (Left) Peak memory consumption of fine-tuning LLaMA2-7B with different methods. (Right) Memory consumption of fine-tuning LLaMA2-7B across different sequence lengths.

reduction in memory growth, from 1.63 GB to just 0.14 GB. This reduction can be attributed to the layer-wise optimizer states offloading technique discussed in § 3.3.

To investigate how memory usage scales with input sequence length, we conducted experiments on LLaMA2-7B with a fixed batch size of 1. As shown in Figure 4 (Right), DoRA consistently consumes more memory than GRASS as the sequence length grows, and GRASS maintains the lowest memory usage under most scenarios (sequence length ≥ 512). Notably, when the sequence length reaches 1792, the memory consumption of LoRA (r=128/256) and DoRA exceeds the 24 GB limit, whereas GRASS remains below this threshold at 23.25 GB. These results highlight that GRASS achieves substantial reductions in peak memory usage while significantly enhancing long-context scalability, thus enabling memory-efficient training under hardware constraints where FFT and even some PEFT methods become infeasible.

Model	Method	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg. ↑
TinyLlama	FFT	61.90	51.25	36.64	29.72	50.43	27.95	25.85	31.00	39.34
	$\text{LoRA}_{r=128}$	58.35	**52.50**	34.34	26.17	50.75	25.04	25.68	24.80	37.20
	$\text{LoRA}_{r=256}$	58.69	50.76	33.52	25.59	**51.70**	26.64	**27.82**	26.20	37.62
	DoRA	57.71	51.74	34.39	25.98	50.83	26.01	27.05	26.40	37.51
	LISA	60.18	49.05	33.78	25.77	49.49	26.22	25.43	28.80	37.34
	GRASS	**61.29**	51.74	**34.75**	**27.01**	50.12	**27.74**	26.45	**29.60**	**38.59**
Gemma-2B	FFT	64.46	76.82	66.02	73.61	60.46	80.01	64.68	70.00	69.51
	$\text{LoRA}_{r=128}$	60.21	73.29	61.59	68.65	59.43	76.89	60.67	67.60	66.04
	$\text{LoRA}_{r=256}$	60.40	73.45	62.54	68.07	61.17	78.28	60.32	66.40	66.33
	DoRA	61.68	73.50	62.74	68.56	59.75	77.65	60.67	66.00	66.32
	LISA	62.63	**78.18**	65.61	72.87	60.77	78.91	**62.03**	68.00	68.63
	GRASS	**63.79**	76.01	**66.73**	**76.10**	**62.27**	**79.29**	61.60	**68.60**	**69.30**
LLaMA2-7B	FFT	71.16	80.96	75.13	85.88	71.35	86.28	72.27	79.40	77.80
	$\text{LoRA}_{r=128}$	66.70	78.35	73.85	82.90	70.24	84.85	68.03	77.40	75.29
	$\text{LoRA}_{r=256}$	67.40	78.35	74.05	83.10	71.59	84.97	**69.80**	78.00	75.91
	DoRA	68.53	79.65	73.13	83.37	**72.45**	**85.27**	69.20	77.60	76.15
	LISA	68.93	80.06	72.42	83.51	68.59	83.59	67.66	75.60	75.05
	IST	68.20	78.73	73.08	**84.19**	71.64	82.87	68.98	76.60	75.54
	OWS	**69.73**	79.92	**74.96**	82.93	70.43	82.70	66.21	77.80	75.59
	GRASS	68.87	**80.09**	74.31	83.71	71.35	84.72	69.11	**78.20**	**76.30**

Table 3: Accuracy comparison of multiple LLMs and PEFT methods on eight commonsense reasoning benchmarks. The best and second best results are marked in **bold** and underlined, respectively.

	FFT	LoRA		DoRA	LISA		GRASS	
Model	-	r=128	r=256	-	γ=2	γ=4	γ=2	γ=4
TinyLlama	8.76	4.71	5.03	5.71	4.93	5.31	4.49	4.55
Gemma-2B	21.19	11.26	11.81	11.64	13.33	14.27	12.45	12.46
LLaMA2-7B	51.32	19.97	20.86	23.23	20.68	22.31	19.08	19.22

Table 4: Comparison of peak GPU memory consumption (GB) for different models and fine-tuning methods. Batch size and sequence length are set to 1 and 1024, respectively.

4.4 Ablation Study

Static strategy We demonstrate the accuracy gains of adaptive layer sampling by comparing GRASS to a static variant (GRASS*), where layer sampling probabilities are initialized using the MGN obtained in the probing phase and remain fixed throughout training. As shown in Table 5, the results of arithmetic reasoning tasks show that adaptive sampling probability updates have brought comparable or significant accuracy improvements across different tasks and models. These results indicate that layer importance is not only task-dependent but also stage-dependent. While the initial probing phase provides a reasonable starting point, fixing the sampling distribution fails to adapt to changing gradient patterns during the optimization process. By continuously updating MGN and sampling probabilities, GRASS dynamically aligns layer updates with the current training stage, leading to improved generalization and accuracy.

Hyperparameters We investigate various combinations of the two key hyperparameters of GRASS: the sampling period T_s, which controls how frequently layers are resampled, and the number of active layers γ, which determines how many layers are updated in each sampling period. Experiments are conducted with LLaMA2-7B, and all settings adopt a learning rate of $\eta = 3 \times 10^{-5}$. Table 6 reports the average accuracy on arithmetic reasoning tasks under different configurations. We find that increasing the number of active layers consistently improves performance. Such results suggest that allocating a larger update budget allows GRASS to approximate or even surpass FFT, while remaining memory-efficient. As for the sampling period, T_s regulates the frequency of layer switching and the update of sampling probabilities. A small T_s (e.g., $T_s = 5$) leads to inferior performance, likely due to unstable sampling caused by rapidly changing gradient estimates. Conversely, overly large T_s delays the timely update of layer-wise importance. Across all settings, intermediate sampling periods ($T_s \in \{25, 50\}$) consistently yield the best results. Overall, these results suggest that *a moderate sampling period combined with a sufficient number of active layers enhances the performance of GRASS.* In addition to T_s and γ, GRASS involves a probing phase used for initializing MGN statistics. We find that GRASS is relatively insensitive to the choice of T_p within a reasonable range, and provide a detailed analysis in Appendix C.3.

Model	Method	MultiArith	AddSub	GSM8K	AQuA	SingleEq	SVAMP	Avg. ↑
TinyLlama	GRASS*	54.50	**39.24**	13.12	**15.75**	40.75	20.80	30.69
	GRASS	**68.00**	35.19	**17.13**	15.16	**42.52**	**27.30**	**34.22**
Gemma-2B	GRASS*	91.00	63.80	41.40	**31.10**	76.77	49.90	59.00
	GRASS	**93.50**	**66.33**	**43.06**	29.13	**78.35**	**53.50**	**60.65**
LLaMA2-7B	GRASS*	91.83	61.52	41.93	13.78	77.56	52.90	56.59
	GRASS	**92.00**	**67.09**	**42.15**	**23.62**	77.56	**55.10**	**59.59**

Table 5: Comparison of GRASS and its variant without adaptive update of sampling probabilities (GRASS*) on arithmetic reasoning tasks.

γ	2					4					8				
T_s	5	10	25	50	100	5	10	25	50	100	5	10	25	50	100
Avg.	57.68	57.12	**59.59**	58.40	58.85	58.13	59.64	60.34	**60.37**	59.32	62.22	61.87	62.32	**62.46**	60.30

Table 6: Comparison average accuracy of applying different GRASS configurations on arithmetic reasoning tasks.

Model	Seed 1	Seed 2	Seed 3
TinyLlama	34.36	34.22	34.12
Gemma-2B	60.64	60.65	60.85
LLaMA2-7B	59.45	59.59	59.37

Table 7: Average accuracy on arithmetic reasoning tasks with varying random seeds.

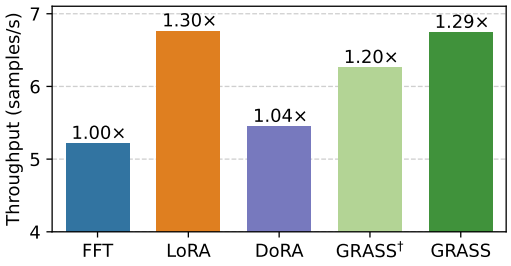

Figure 5: Training throughput of different methods on LLaMA2-7B. GRASS† represents a variant of GRASS that does not use overlap.

Random seed Since GRASS involves stochastic layer sampling, we further examine its robustness with respect to different random seeds. Table 7 reports the average accuracy on arithmetic reasoning tasks obtained by fine-tuning models with different random seeds. As shown, GRASS exhibits consistent performance across all model scales. The variation across seeds is minimal, with the maximum absolute difference being 0.24 for TinyLlama, 0.21 for Gemma-2B, and 0.22 for LLaMA2-7B. These results indicate that the stochasticity introduced by layer sampling does not lead to unstable training outcomes. Overall, these findings confirm the stability of GRASS across random seeds, further supporting its practical reliability.

Throughput To demonstrate the training throughput trade-off caused by initializing optimizer states in CPU memory, as well as the enhancement provided by the overlap technique, we conduct experiments on LLaMA2-7B. The batch size and the input sequence length are set to 4 and 1024, respectively. These results are shown in Figure 5. The results indicate that the overlap technique yields a throughput increase of 1.08× for GRASS, achieving a throughput comparable to LoRA, and all methods exhibit higher training throughput than FFT.

Computational overhead To quantify the additional computational costs introduced by GRASS, we provide a detailed runtime breakdown across three model scales. As shown in Table 8, the total overhead of GRASS decreases significantly as model size increases. For example, on LLaMA2-7B, the combined overhead of the probing phase and periodic MGN computation constitutes only 2.01% of the total training time. This reduction occurs because the dominant forward and backward pass computations scale linearly with model size, whereas the computational overhead of GRASS remains nearly constant. These results indicate that the additional cost introduced by probing and periodic MGN computation remains small, approximately 2-6% of total training time. Considering the memory savings, this overhead does not diminish the practical advantages of GRASS.

5 Conclusion

In this paper, we propose GRASS, a gradient-based adaptive layer-wise importance sampling framework. It dynamically samples and updates influential layers during training based on mean gradient

Model	Probing phase	MGN update	Total training time
Tinyllama	34 (3.24%)	32 (3.09%)	1050 (100%)
Gemma-2B	30 (0.99%)	42 (1.37%)	3036 (100%)
Llama2-7B	54 (0.95%)	60 (1.06%)	5708 (100%)

Table 8: Time breakdown of the additional runtime introduced by GRASS on the arithmetic reasoning dataset. Batch size and sequence length are set to 4 and 1024, respectively.

norms. By leveraging dynamically estimated layer importance rather than static heuristics, GRASS improves the performance of layer-wise fine-tuning methods. To reduce the memory of accumulated optimizer states, we introduce layer-wise optimizer state offloading, enabling GRASS to obtain comparable throughput with state-of-the-art methods. Extensive experiments over multiple models and tasks demonstrate that GRASS consistently outperforms PEFT baselines while maintaining matching or lower memory consumption.

Limitations

Despite its effectiveness, GRASS has several limitations that need further investigation. Firstly, it relies on gradient-based statistics to estimate layer importance, which introduces additional computation during training. Nevertheless, the relative overhead decreases as model scale increases, indicating favorable amortization behavior in larger-scale settings. Additionally, our evaluation focuses on decoder-only language models and widely used NLP benchmarks. The applicability of GRASS to other architectures and modalities remains an interesting direction for future work.

Acknowledgments

This work is sponsored in part by the National Natural Science Foundation of China (No. 62421002) and the Fundamental and Interdisciplinary Disciplines Breakthrough and Plan of the Ministry of Education of China (JYB2025XDXM202).

References

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL-HLT 2019*, pages 2924–2936. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, and Matthias Plappert. 2021. Training verifiers to solve math word problems.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, and Shengding Hu. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint*, arXiv:2203.06904.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP 2014*, pages 523–533. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, and Mona Attariyan. 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. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Lu Wang. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, and Xing Xu. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *EMNLP 2023*, pages 5254–5276. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP 2021*, pages 3045–3059. Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Pretrained language model for text generation: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4492–4499. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Pengxiang Li, Lu Yin, Xiaowei Gao, and Shiwei Liu. 2025. Outlier-weighed layerwise sampling for llm fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19460–19473.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL-IJCNLP 2021*, pages 4582–4597. Association for Computational Linguistics.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. 2023. Relora: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *ACL 2017*, pages 158–167. Association for Computational Linguistics.

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*, pages 32100–32121. PMLR.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *ACL 2022*, pages 61–68. Association for Computational Linguistics.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP 2018*, pages 2381–2391. Association for Computational Linguistics.

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. 2024. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. *Advances in Neural Information Processing Systems*, 37:57018–57049.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *NAACL-HLT 2021*, pages 2080–2094. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *EMNLP 2015*, pages 1743–1752. Association for Computational Linguistics.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *EMNLP-IJCNLP 2019*, pages 4463–4473. Association for Computational Linguistics.

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*.

Kaiyuan Tian, Linbo Qiao, Baihui Liu, Gongqingjian Jiang, Shanshan Li, and Dongsheng Li. 2026. A survey on memory-efficient transformer-based model training in ai for science. *Frontiers of Computer Science*, 20(11):2011355.

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*.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *EACL 2023*, pages 3274–3287. Association for Computational Linguistics.

Luping Wang, Sheng Chen, Linnan Jiang, Shu Pan, Runze Cai, Sen Yang, and Fei Yang. 2025. Parameter-efficient fine-tuning in large language models: A survey of methodologies. *Artificial Intelligence Review*, 58(8):227.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. In *EMNLP 2022*, pages 5744–5760. Association for Computational Linguistics.

Wenhan Xia, Chengwei Qin, and Elad Hazan. 2024. Chain of lora: Efficient fine-tuning of language models via residual learning. *arXiv preprint arXiv:2401.04151*.

Kai Yao, Penglei Gao, Lichun Li, Yuan Zhao, Xiaofeng Wang, Wei Wang, and Jianke Zhu. 2024. Layer-wise importance matters: Less memory for better performance in parameter-efficient fine-tuning of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1977–1992.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *ACL 2019*, pages 4791–4800. Association for Computational Linguistics.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. In *Forty-first International Conference on Machine Learning*.

Peilin Zhao and Tong Zhang. 2015. Stochastic optimization with importance sampling for regularized loss minimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1–9. PMLR.

Ligeng Zhu, Lanxiang Hu, Ji Lin, and Song Han. 2023. Lift: Efficient layer-wise fine-tuning for large model models.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2024. Multilingual machine translation with large language models: Empirical results and analysis. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2765–2781, Mexico City, Mexico. Association for Computational Linguistics.

A Experimental Details

In this section, we give more details about the experiment settings. All experiments are conducted using 2 NVIDIA H100 80G GPUs. For all accuracy-related experiments, we report the average results over three independent runs.

A.1 Hyperparameters

We provide detailed hyperparameter settings of arithmetic reasoning and commonsense reasoning tasks in Table 9. Notably, probing phase last for 150 steps ($T_p = 150$) in all experiments.

B Validity of MGN as a Layer-wise Importance Measure

In this section we discuss the rationale of using mean gradient norm (MGN) as a layer-wise importance indicator in GRASS.

B.1 Optimization-aligned Proxy

Gradients provide a direct signal of how sensitively the loss responds to parameter updates. For layer l with parameters $\theta^{(l)}$, a small update $\Delta\theta^{(l)}$ changes the loss approximately via a first-order Taylor expansion:

$$\Delta\mathcal{L} \approx \langle \nabla_{\theta^{(l)}}\mathcal{L}, \Delta\theta^{(l)} \rangle \,, \tag{1}$$

which implies that the magnitude of $\nabla_{\theta^{(l)}}\mathcal{L}$ reflects the potential impact of updating that layer on the training objective.

Motivated by this observation, we use the gradient norm as a local indicator of a layer's sensitivity to the loss, reflecting how small parameter perturbations at the current training stage affect the objective. To ensure comparability across layers of different sizes, we normalize gradient magnitudes by the number of parameters in each layer. Aggregating these normalized gradient norms over multiple training steps yields the mean gradient norm (MGN), i.e.

$$\mathbf{m}_l(T) = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{N_p^{(l)}}\|\mathbf{g}_t^{(l)}\|_2^2} \,, \tag{2}$$

which reduces stochastic noise and captures persistent optimization signals.

Importantly, MGN provides a relative ranking of layers in terms of their short-term optimization impact, rather than an absolute measure of importance.

B.2 Empirical Sufficiency for Adaptive Layer Sampling

While MGN is not a complete or causal measure of layer importance, our experiments demonstrate that it is sufficiently informative for guiding adaptive layer sampling. Across multiple models and tasks, we observe that: (i) sampling layers according to MGN often outperforms uniform sampling baseline, (ii) static importance estimates are inferior to dynamically updated MGN statistics, and (iii) MGN exhibits task-dependent and training-stage-dependent patterns across different architectures.

These observations indicate that MGN captures meaningful optimization signals that are directly relevant to the layer selection problem addressed by GRASS. Therefore, MGN serves as an effective and efficient proxy for allocating update resources under memory-constrained fine-tuning settings.

C Additional Experiments

C.1 Layer-wise MGN on Different Models

We conduct further experiments on Gemma-2B and LLaMA2-7B to demonstrate that MGN is a task-aware indicator that captures varying layer-wise importance across downstream tasks and training stages. The results are shown in Figure 6.

Similar to the observations on TinyLlama, we find that MGN values exhibit layer-wise differentiation and task-specific patterns across both models. In particular, the relative ordering of layer importance varies noticeably between arithmetic reasoning and commonsense reasoning tasks, indicating that different layers dominate the optimization dynamics depending on task characteristics. These results further support the validity of MGN as a indicator of layer importance, motivating its use for adaptive layer sampling in GRASS.

C.2 Convergence

Figure 7 illustrates the validation loss curves of different fine-tuning methods on the Alpaca-GPT4 dataset for TinyLlama, Gemma-2B, and LLaMA2-7B. Across all model scales, GRASS exhibits stable convergence behavior throughout training and consistently achieves lower validation loss in later training stages. The results indicate that GRASS effectively concentrates updates on the most influential components, leading to reliable convergence across different model sizes.

Hyperparameter		Arithmetic Reasoning			Commonsense Reasoning		
		TinyLlama	Gemma-2B	LLaMA2-7B	TinyLlama	Gemma-2B	LLaMA2-7B
FFT	Learning rate	1e-5			1e-5		
	Batch size	8	4	4	8	4	4
LoRA	Learning rate	1e-4			5e-5	2e-5	2e-5
	Batch size	8	4	4	8	4	4
	Rank	128/256			128/256		
	α	256/512			256/512		
DoRA	Learning rate	1e-4			5e-5	2e-5	2e-5
	Batch size	8	4	4	8	4	4
	Rank	128			128		
	α	256			256		
LISA	Learning rate	5e-5			5e-5	3e-5	2e-5
	Batch size	8	4	4	8	4	4
	γ	2			2		
	T_s	10			10		
GRASS	Learning rate	3e-5			3e-5	3e-5	2e-5
	Batch size	8	4	4	8	4	4
	γ	2			2		
	T_s	25			25		

Table 9: Hyperparameter configurations of different fine-tuning methods for arithmetic reasoning and common reasoning tasks.

T_p	25	50	100	150	200
Avg.	58.95	59.31	59.33	**59.59**	58.20

Table 10: Average accuracy on arithmetic reasoning tasks with varying lengths of probing phase.

C.3 Sensitivity to Probing Phase Length

At the start of training, GRASS employs a short probing phase of length T_p to initialize layer-wise MGN statistics. To examine the sensitivity of GRASS to the choice of T_p, we vary T_p from 25 to 200 steps and report the average accuracy on arithmetic reasoning benchmarks. As shown in Table 10, increasing T_p improves performance up to a moderate range, peaking at $T_p = 150$. When the probing phase is too short (e.g., $T_p = 25$), the initial MGN estimates can be noisy due to insufficient gradient observations, leading to slightly degraded performance. On the other hand, overly long probing phases (e.g., $T_p = 200$) may delay the onset of adaptive layer sampling and reduce effective training time for optimization, resulting in mild performance degradation. However, the performance variance between $T_p = 50$ and $T_p = 150$ remains small, indicating that GRASS is not sensitive to the choice of T_p as long as it falls within a reasonable range. This robustness allows T_p to be set flexibly without careful tuning, and we use $T_p = 150$ as the default setting in all experiments.

C.4 Scalability Analysis

To further validate the scalability of GRASS across larger models and varied architectures, we conduct experiments on LLaMA3-8B and Qwen3-14B. All methods were evaluated using the same eight commonsense reasoning benchmarks as in the main experiments, with identical hyperparameter settings. Table 11 summarizes the results. On LLaMA3-8B, GRASS achieves the highest average accuracy (79.52%), surpassing LoRA(rank=128), DoRA, and LISA. GRASS also shows superior performance on 6 of 8 tasks, indicating robust adaptation to larger models. For the larger Qwen3-14B model, GRASS continues to achieve the highest average accuracy. These results suggest that the adaptive layer sampling and optimizer offloading mechanisms in GRASS scale effectively to larger architectures, while preserving both performance and memory efficiency. Together, experiments on LLaMA3-8B and Qwen3-14B validate the scalability of GRASS across different model sizes and families, supporting its practical utility for memory-efficient fine-tuning of large language models.

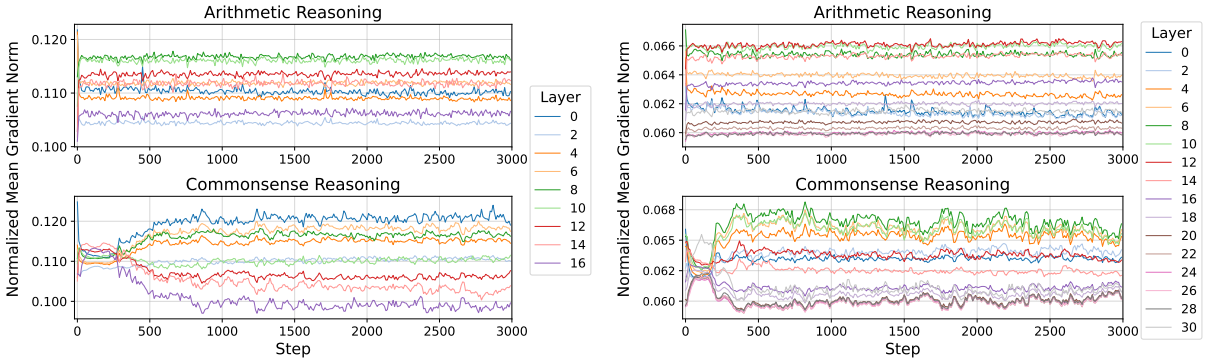

Figure 6: Comparison of layer-wise mean gradient norm across different datasets on Gemma-2B (Left) and LLaMA2-7B (Right).

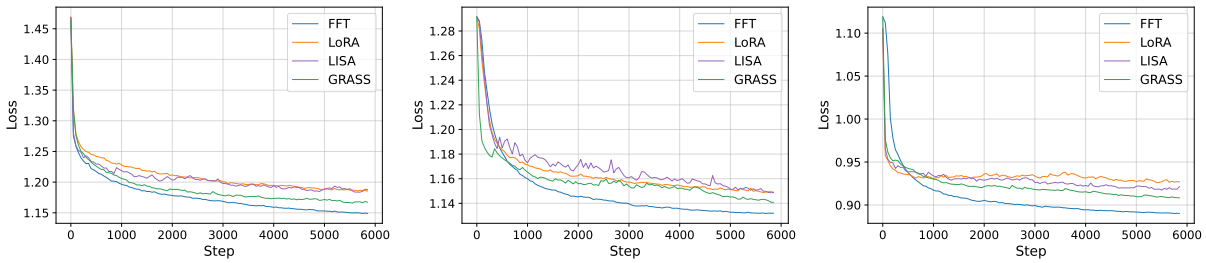

Figure 7: Loss curves of fine-tuning TinyLlama (Left), Gemma-2B (Middle), and LLaMA2-7B (Right) with different methods on Alpaca-GPT4 dataset.

Model	Method	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg. ↑
LLaMA3-8B	LoRA	66.57	80.90	73.75	85.80	74.03	87.67	75.77	**80.80**	78.16
	DoRA	67.39	81.23	74.10	85.62	74.19	88.77	76.13	79.60	78.38
	LISA	69.54	82.59	73.95	86.60	**77.27**	88.89	75.85	77.80	79.06
	GRASS	**70.80**	**83.46**	**74.26**	**87.26**	75.61	**89.27**	**76.51**	79.00	**79.52**
Qwen3-14B	LoRA	**74.50**	90.42	78.71	93.54	**81.61**	96.93	**93.26**	92.40	87.67
	DoRA	74.37	91.02	78.35	**93.61**	81.45	96.46	91.64	92.60	87.44
	LISA	73.52	89.66	**79.63**	92.17	75.37	97.35	92.24	92.00	86.49
	GRASS	73.49	**91.08**	79.07	93.56	81.06	**97.64**	93.00	**92.80**	**87.71**

Table 11: Accuracy comparison of larger-scale LLMs and PEFT methods on commonsense reasoning benchmarks. The best and second best results are marked in **bold** and underlined, respectively.