LoSiA: Efficient High-Rank Fine-Tuning via Subnet Localization and Optimization

Anonymous ACL submission

Abstract

Parameter-Efficient Fine-Tuning (PEFT) methods, such as LoRA, significantly reduce the number of trainable parameters by introducing low-rank decomposition matrices. However, existing methods perform extensive matrix multiplications in domain specialization tasks, resulting in computational inefficiency and sub-optimal fine-tuning performance. Hence, we propose LoSiA[1] (**Lo**w-Resources **S**ubnet **I**ntegration **A**daptation), an innovative method that dynamically localizes and optimizes critical parameters during the training process. Specifically, it identifies a sub-network using gradient sparsity analysis and optimizes it as the trainable target. This design enables effective high-rank adaptation by updating only the sub-network parameters, reducing the additional matrix multiplication. We also present LoSiA-Pro, a faster implementation of LoSiA, which reduces the training latency by about 27% compared to LoRA. Extensive evaluations show that our method achieves minimal performance drop compared to full fine-tuning, while requiring the least training time across domain specialization and common-sense reasoning tasks. Further analysis shows that LoSiA also reduces forgetting during continued training.

1 Introduction

Large language models, when fine-tuned via supervised learning, can be effectively adapted to downstream tasks such as mathematics (Shao et al., 2024), programming (Hui et al., 2024), and domain knowledge reasoning (Wei et al., 2021). Although full parameter fine-tuning often yields the best performance, updating billions of parameters is computationally expensive and resource-intensive. To address this, parameter-efficient fine-tuning (PEFT) updates only a small subset of parameters to reduce GPU memory usage and communication overhead while maintaining performance comparable to full fine-tuning (Houlsby et al., 2019; Ding et al., 2023).

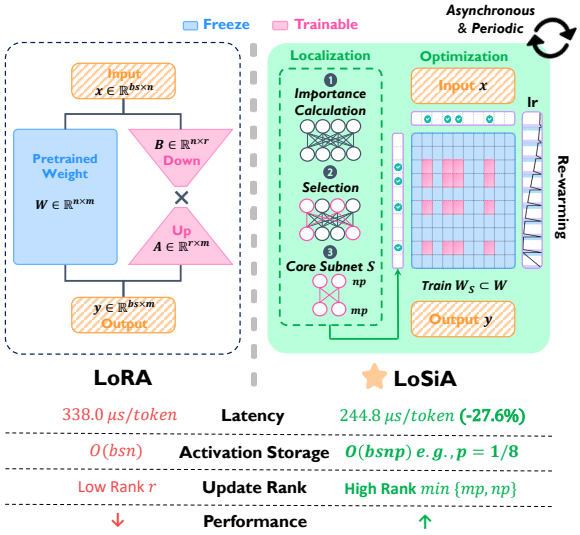


Figure 1: Overview of LoSiA. It locates and optimizes core sub-network in asynchronous periods.

Among PEFT approaches, LoRA (Hu et al., 2022) has gained widespread adoption by introducing low-rank matrices to approximate full weight updates, enabling competitive performance with significantly reduced computational and economic costs (Taori et al., 2023). Variants in the LoRA family further refine the method by biased fine-tuning modules (Zhu et al., 2024; Hayou et al., 2024a) or dimensions (Meng et al., 2024a) to accelerate convergence and achieve superior performance. However, constrained by the low-rank assumption, these paradigms often struggle to balance model performance and efficiency, particularly in domain-specific tasks (Yang et al., 2024; Ghosh et al., 2024) and continual learning scenarios (Shuttleworth et al., 2024a). In such settings, low rank configurations (e.g., 8 or 16) can lead to performance degradation and underfitting (Biderman et al., 2024). Although increasing the rank may mitigate these issues, it introduces additional

[1]The source code will be publicly available.

memory consumption, extensive floating point operations, and risks of overfitting or convergence difficulties (Kalajdzievski, 2023; Borse et al., 2024). Recent studies have attempted to approximate high-rank updates by accumulating multiple low-rank components. However, these approaches still suffer from issues such as locally low-rank updates (Meng et al., 2024b; Lialin et al., 2023) or increased computational complexity (Zhao et al., 2024a). Therefore, while the low-rank assumption offers notable improvements in efficiency, it also introduces inherent limitations.

The Lottery Ticket Hypothesis (Frankle and Carbin, 2019) suggests that dense neural networks contain trainable sub-networks capable of achieving comparable test accuracy. This prompts us to reconsider the route of PEFT and explore an alternative: **Can we identify and fine-tune such sub-networks within the backbone model to achieve high-quality adaptation more efficiently?**

To answer this question, we propose LoSiA (**Lo**w-Resources **S**ubnet **I**ntegration **A**daptation), a novel PEFT framework that dynamically localizes and optimizes critical sub-networks periodically, as illustrated in Figure 1. LoSiA asynchronously selects a core sub-network for each layer by calculating sensitivity-based importance scores and performing greedy selecting algorithms. Following localization, it fine-tunes the identified sub-network and applies a rewarming learning rate strategy to promote stable and robust training. The design enables real-time high-rank updates without introducing additional matrix multiplication overhead, which reduces training latency and ensures no steep increase in training time for high-rank updating. Additionally, LoSiA does not introduce extra architectural components and only requires an optimizer replacement for seamless deployment. Extensive experiments demonstrate its superior performance among PEFT baselines on domain-specific, commonsense reasoning tasks, while mitigating forgetting in continue learning. We also propose LoSiA-Pro, a more refined equivalent implementation of LoSiA, which significantly reduces the activation storage and computational complexity in backward propagation. LoSiA-Pro speeds up training $1.38\times$ compared to LoRA and $2.68\times$ compared to DoRA.

In summary, our contributions are as follows.

(1) Innovatively introduces **the structure of the sub-network** to the field of parameter-efficient fine-tuning. We develop periodic subnet localization, optimization and integration techniques to dynamically capture and adapt task-essential parameters.

(2) We propose **LoSiA**, a novel PEFT approach that dynamically localizes and optimizes sub-networks to achieve high performance, efficiency, and usability. To further improve practicality, we also introduce a faster variant, **LoSiA-Pro**.

(3) We conduct extensive **evaluations** across multiple models and benchmarks. LoSiA outperforms all advanced PEFT baselines on domain-specific and commonsense reasoning tasks, while also accelerating training by $1.15\times$ compared to LoRA. Moreover, its efficient variant, LoSiA-Pro, achieves a further speedup of $1.38\times$.

2 Related Work

Parameter-Efficient Fine-Tuning Full parameter fine-tuning (FFT) adapts pre-trained models to downstream tasks by updating all model parameters (Wei et al., 2022), but it is often computationally expensive. In contrast, parameter-efficient fine-tuning (PEFT) methods update only a small subset of parameters, significantly reducing training costs while still maintaining strong performance. LoRA (Hu et al., 2022) approximates parameter updates as the product of low-rank matrices, achieving promising performance for tasks such as instruction tuning (Ghosh et al., 2024). Enhanced variants such as PiSSA (Meng et al., 2024a) accelerate convergence by prioritizing major singular vectors, while DoRA (Liu et al., 2024) improves performance in low-rank by decomposing updates into directional and magnitude components. Other derivatives such as LoRA+ (Hayou et al., 2024b), LoRA-GA (Wang et al., 2024a) and LoRA-Dash (Si et al., 2025) refine the framework by directional or module biased fine-tuning.

However, recent studies (Jiang et al., 2024b; Biderman et al., 2024; Ghosh et al., 2024) reveal that the low-rank structure limits effectiveness in knowledge-intensive domains (e.g., mathematics, coding). Advanced solutions adopt novel strategies: 1) Architectural modifications through MoE-based LoRA combinations (Zadouri et al., 2023; Li et al., 2024; Wang et al., 2024b) for multitask scenarios; 2) High-rank fine-tuning via accumulated low-rank projections, such as ReLoRA (Lialin et al., 2023), MoRA (Jiang et al., 2024a) and GaLore (Zhao et al., 2024a) to enhance training effectiveness. However, these approaches incur an increase in architectural complexity or a drop in throughput. Rare methods achieve an optimal balance between performance,

training latency, and implementation simplicity.

Skill Localization and Pruning LLM pruning reduces neural network size by eliminating redundant or less critical parameters. Prior work demonstrates that sparse networks can play crucial roles (Frankle and Carbin, 2019; Yao et al., 2025). Panigrahi et al. (2023) identifies critical parameters in fine-tuned LMs by optimizing masks of grafted models, though such methods require additional training time and data. Alternatively, gradient- and sensitivity-based metrics enable real-time identification of task-aware parameters (Molchanov et al., 2019; Sanh et al., 2020; Zhang et al., 2022). Recent advances adapt the techniques to PEFT: Zhang et al. (2023) prunes LoRA trainable parameters, while Feng et al. (2024) applies the approach to continual learning scenarios.

3 Method

Definition Consider a model $f_0 : \mathcal{X} \to \mathcal{Y}$ trained on dataset $\mathcal{D} = \{B_i\}_{i=1}^{N}$, where each batch $B_i = \{(x_j, y_j)\}_{j=1}^{M}$ contains M samples. Let W denote the parameters and $\mathcal{L}$ the loss function. The neural network S_0 in f_0 can be represented as a tuple comprising input neurons X_{S_0}, output neurons Y_{S_0} and neural connections $W_{X_{S_0},Y_{S_0}}$, that is, $S_0 = (X_{S_0}, Y_{S_0}, W_{X_{S_0},Y_{S_0}})$. The notation $f_0 \xrightarrow[\mathcal{D}]{P_0} f$ denotes training model f_0 on $\mathcal{D}$ over full parameters $P_0 = \{W_{X_{S_0},Y_{S_0}}\}$ and produces model f. We investigate the following question: *Can we efficiently identify a parameter subset $P \subset P_0$ of fixed size, such that training $f \xrightarrow[\mathcal{D}]{P} f'$ minimizes the loss difference $\Delta\mathcal{L} = |\mathcal{L}(f', \mathcal{D}) - \mathcal{L}(f, \mathcal{D})|$?*

3.1 Structure of Gradients

Inspired by pruning techniques, we minimize the mean squared error (MSE) $\mathcal{L}_{\text{MSE}}$ between the outputs of models f_k, f'_k, where both are trained on f_{k-1} with trainable parameter set P_0 and P, respectively. For SGD optimizers, we derive the bound:

$$\mathcal{L}_{MSE} \leq \eta^2 \frac{\|1_{(i,j)\notin P} \cdot \nabla_{W_0} \mathcal{L}(\mathcal{B}_k)\|_F^2 \|x\|_F^2}{M} \quad (1)$$

For AdamW optimizers, $\mathcal{L}_{MSE}$ admits a similar bound through ∇W in most cases (Appendix A.1.1). Thus, gradient magnitudes in P provide an upper bound for the approximation error, while prioritizing larger gradients yields tighter bounds. We therefore seek optimal subsets P to capture the parameters with large gradient magnitudes.

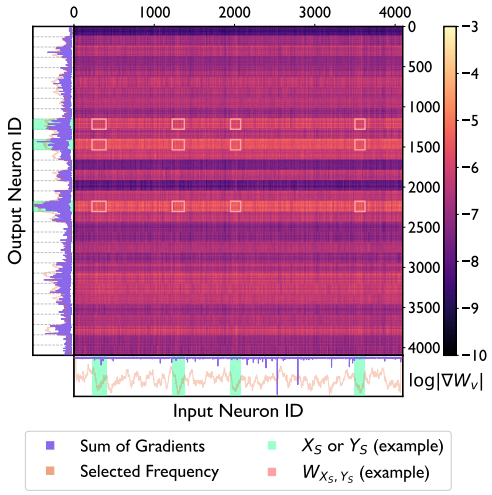


Figure 2: Gradient Magnitude Distribution of proj_v. Large gradients follow a sparse subnet distribution.

Ideally, selecting the *top-K* entries of ∇W would suffice, but storing and fine-tuning sparse matrices compromises efficiency. Instead, we claim that a suitable pattern for P corresponds to the parameters in subnet $S = (X_S, Y_S, W_{X_S,Y_S})$, i.e., all connections between the input neuron set X_S and the output neuron set Y_S.

To validate this selection paradigm, Figure 2 visualizes the gradient magnitude distributions in LLaMA-2 7B's `proj_v` layer: The 32 attention heads exhibit a significant disparity in attention scores, while gradients strongly correlate with output neurons Y_S. In particular, a consistent subset of input neurons X_S (green markers, x-axis) contributes dominantly to all attention heads. The sparse pattern remains consistent in MLP layers (Appendix A.2.1). We therefore focus on fine-tuning subnet $S = (X_S, Y_S, W_{X_S,Y_S})$ - termed the *core subnet* - rather than the entire network.

3.2 Subnet Localization

To localize core subnets efficiently, an ideal algorithm should satisfy three key requirements: 1) Efficiency: no additional data requirements or significant latency. 2) Lightweight: minimal GPU memory overhead. 3) Dynamic Awareness: enable real-time localizations during training. To meet these objectives, the subnet localization process is divided into two stages:

Parameter Importance Calculation To quantify parameter importance $I(\cdot)$, existing approaches (LeCun et al., 1989; Ma et al., 2023) observe the change in loss assuming $W_k = 0$ for the k-th parameter. Adopting second-order Taylor expansion,

element-wise $I(\cdot)$ is estimated as:

$$I = |\frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_k} W_k - \frac{1}{2} W_k H_{kk} W_k + o(W_k^2)| \quad (2)$$

Here, H stands for the Hessian matrix. However, Eq.2 is difficult to calculate in real-time. We derive a micro-batch approximation:

$$I_i = |\frac{\partial \mathcal{L}(\mathcal{B}_i)}{\partial W_k} W_k - \frac{1}{2} (\frac{\sum_j \frac{\partial \mathcal{L}(\mathcal{B}_{ij})}{\partial W_k}}{M} W_k)^2 + o(W_k^2)| \quad (3)$$

Furthermore, estimates by single micro-batch may introduce bias by ignoring training dynamics. Sensitivity smoothing and uncertainty quantification (Zhang et al., 2022) are used to handle the problem. For the training step i, compute an exponential moving average (EMA) $\overline{I}_i$ for I_i ,and uncertainty $\overline{U}_i$ for variation $\Delta I_i = |I_i - \overline{I}_i|$:

$$\overline{I}_i(W_k) = \beta_1 \overline{I}_{i-1}(W_k) + (1-\beta_1) I_i(W_k) \quad (4)$$

$$\overline{U}_i(W_k) = \beta_2 \overline{U}_{i-1}(W_k) + (1-\beta_2) |\Delta I_i(W_k)| \quad (5)$$

$$s(W_k) = \overline{I}(W_k) \cdot \overline{U}(W_k) \quad (6)$$

where $\beta_1, \beta_2 \in (0,1)$ is the EMA factors. We regard $s(\cdot)$ as a appropriate importance assessment. Notably, to obtain gradients W asynchronously, LoSiA use per-layer weight updates (Lv et al., 2024), executing the optimization during backpropagation without storing gradients.

Core Subnet Localization via Importance Scores For a subnet S selected from origin network $S_0 = (\{i\}_{i=1}^n, \{j\}_{j=1}^m, W)$, define its importance as:

$$s(S) = \sum_{i \in X_S} \sum_{j \in Y_S} s(W_{ij}) \quad (7)$$

The objective is to identify optimal subset S and maximize $s(S)$, while satisfying the memory constraint $\max\{\frac{|X_S|}{n}, \frac{|Y_S|}{m}\} \leq p$, where $p \in (0,1)$ represents the *rank factor*. However, the task is NP-Hard. Leveraging observations in Section 3.1 regarding gradient magnitude sparsity patterns, we develop greedy selection algorithms to identify critical input and output neuron sets (X_S, Y_S):

Algorithm 1 Greedy Strategy for Localization

```
function ROW2COLUMN(q)
    sums ← Sum(q, dim = 1)
    rows ← Top-K(sums, ⌊np⌋).indices
    sums ← Sum(q[rows, :], dim = 0)
    cols ← Top-K(sums, ⌊mp⌋).indices
    return (rows, cols)
end function
```

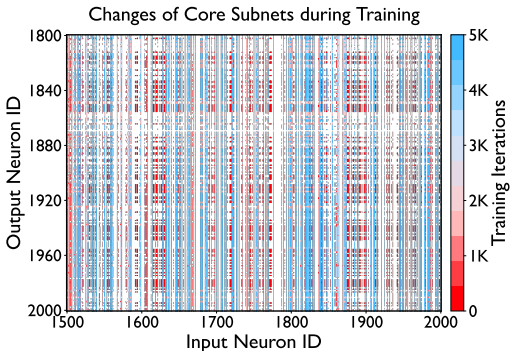


Figure 3: Core Subnet Distribution during Training. The subnet various across different training iterations.

Algorithm 1 implements a row-major greedy approach that selects the *top-K* rows by importance score summation, and then finds $\{Y_S\}$ to maximum $s(S)$ given fixed $\{X_S\}$. A column-major greedy algorithm *Column2Row* is also considered. The better selection result of two strategies is selected.

Dimensionality Reduction in Output Layer Fine-Tuning While prior work (Chen et al., 2024) has established the benefits of fine-tuning the output layer in conjunction with PEFT methods, the approach remains computationally prohibitive for large-vocabulary models (e.g., Gemma-2B). However, empirically, backward propagation through the output layer exhibits gradient sparsity, with only a limited subset of tokens receiving significant updates. Building on this insight, LoSiA easily implements an efficient optimization strategy by constructing a tunable subnet $S = (X_{S_0}, Y_S, W_{X_{S_0}, Y_S})$ of the output layer, where $|Y_S| = p_o |Y_{S_0}|$, and $p_o \in (0,1)$ denoting the *dimension reduction factor*.

3.3 Subnet Optimization and Intergration

During fine-tuning, the locations of core subnets may undergo dynamic shifts, as illustrated in Figure 3. Although a small subset of neurons is consistently selected, peripheral components exhibit significant temporal variability. Fine-tuning with a static subnet risks model underfitting and neuron over-specialization. To address the issue, we introduce an asynchronous and periodic subnet re-localization mechanism that adapts to the evolving network topology.

Naive periodic re-localization strategies can induce training instability and loss spikes (Lialin et al., 2023). Furthermore, the storage requirements for $\overline{I}(\cdot), \overline{U}(\cdot)$ in a synchronous period would lead to a scaling of GPU memory overhead. Therefore, we propose *asynchronous periodic*

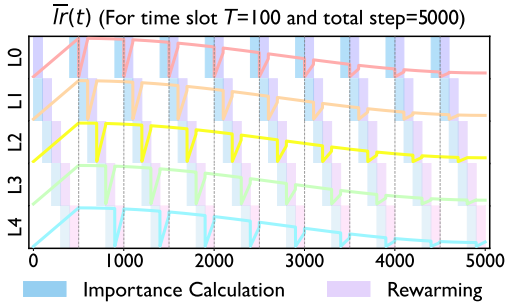


Figure 4: Asynchronous Periodic Subnet Reselection and Learning Rate Rewarming Mechanism (in a 5-layer model for example).

localization coupled with rewarmings of learning rate. Consider a model f with L decoder layers $\{D_l\}_{l=0}^{L-1}$, where each layer D_l contains K linear layers $\{W_{l,k}\}_{k=1}^{K}$ with corresponding core subnets $\{S_{l,k}\}_{k=1}^{K}$. The training timeline is divided into time slots of T steps, such that for time slots $[iT, (i+1)T), i = 1, 2, \ldots$, we:

1. Compute $\overline{I}(\cdot), \overline{U}(\cdot)$ for layer D_l in the time slot $[(kL + l - 1)T, (kL + l)T), k \in N$.

2. Sequentially reselect S_l by $s(\cdot)$ before step $t = (kL + l)T$, the end of time slots in 1.

This yields a reselection period of $\overline{T} = LT$ for each layer. The asynchronous design ensures that at any step, there is exactly one decoder layer calculating $\overline{I}(\cdot), \overline{U}(\cdot)$ and one rewarming, which greatly reduces additional GPU memory overheads for importance score calculation. We also implement a learning rate rewarm-up to further enhance training stability. Formally, the learning rate at step t is:

$$\overline{lr}(t) = \begin{cases} \frac{t-(kL+l)T}{T} \cdot \mathrm{lr}(t) & \text{if C} \\ \mathrm{lr}(t) & \text{otherwise} \end{cases} \tag{8}$$

The condition C is $t \in [(kL + l)T, (kL + l + 1)T)$ and $t > T_w$, where T_w is the warmup duration. Figure 4 illustrates the timelines of importance calculation and the learning rate rewarming across layers, with re-localization sandwiched between them.

3.3.1 Faster Implementation (LoSiA-Pro)

Through subnet fine-tuning, LoSiA can further mitigate activation storage and backward latency. The gradient for the subnet S can be factorized as:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial W_S} &= \frac{\partial \mathcal{L}}{\partial W}[X_S, :][:, Y_S] \\ &= (x^T[X_S, :])(\frac{\partial \mathcal{L}}{\partial y}[:, Y_S]) = \tilde{L}_S \tilde{R}_S \end{aligned} \tag{9}$$

Noticing $\tilde{L}_S \in R^{np \times bs}, \tilde{R}_S \in R^{bs \times mp}$, the input activation storage is reduced by a factor p, while the computational complexity of gradient calculation is reduced from $O(nmbs)$ to $O(nmbsp^2)$. We named the method LoSiA-Pro, a refined equivalent implementation of LoSiA. It offers a **27.6%** latency reduction compared to LoRA with GRADIENT CHECK-POINTING, while reducing **13.4GB** GPU memory consumption compared to LoSiA training without GRADIENT CHECK-POINTING.

4 Experiments

We evaluate LoSiA across a broad range of model scales and datasets, conducting rigorous comparisons with common baselines. On both domain-specific and common-sense reasoning tasks, the method demonstrates robust performance with significantly reduced training overheads. The experiments highlight that LoSiA effectively promotes both training efficiency and task proficiency.

4.1 Experimental Setup

Datasets Models are trained on downstream tasks in the domains of mathematics, coding, and general capabilities. Specifically, training sets are sampled by 50,000 random entries from MetaMathQA, Magicoder, and Alpaca-GPT4, respectively. The GSM8K, MBPP, and MMLU benchmark are for testing. Additionally, we also compared LoSiA with baseline methods on eight common sense reasoning tasks. More details regarding the datasets can be found in the Appendix.

Implementation Details We employ Gemma 2B, LLaMA-2 7B, and LLaMA-2 13B as the backbone models. The effectiveness of LoSiA is evaluated against parameter-efficient fine-tuning (PEFT) baselines, namely LoRA, DoRA, PiSSA, and GaLore. For control of consistency in memory consumption, the rank r of LoRA, DoRA, and PiSSA is set to 64. For GaLore, the gradient projection rank R is set to 512 with the full projection strategy. In the case of LoSiA, the rank factor p is set to $\frac{1}{8}$. The learning rate is 6×10^{-5} for MetaMathQA and 5×10^{-5} for the rest, with time slots T of 100 for MetaMathQA and 150 for the rest.

Additionally, both GaLore and LoSiA incorporate the output layer into the fine-tuning process. Dimension reduction factor p_0 is set to $\frac{1}{64}$ for Gemma 2B, $\frac{1}{8}$ for LLaMA-2 7B, and 1 for LLaMA-2 13B in LoSiA. The PEFT modules are applied to all linear layers within the transformer. The train-

Table 1: Comparison of PEFT Methods Across Models on Domain-Specific Tasks. Accuracy is reported, alongside with memory consumption (GB), maximum per-task training time (h). The numbers in parentheses indicate the training time of LoSiA-Pro, which is a refined and computationally equivalent implementation of LoSiA.

Model	Method	Mem(GB)	Time(h)	GSM8K		MBPP		MMLU		Avg.
				5-shot	0-shot,CoT	Pass@1	Pass@10	0-shot,PPL	5-shot,GEN	
Gemma 2B	FFT	50.1	11.0	46.4	50.4	33.0	43.4	36.1	37.0	41.05
	LoRA	36.1	15.0	35.7	41.1	26.0	36.6	34.9	31.2	34.25
	PiSSA	36.1	14.9	38.5	46.5	26.4	39.0	33.8	32.6	36.13
	DoRA	37.3	29.8	39.7	43.0	31.4	**43.2**	36.2	37.1	38.43
	GaLore	37.5	14.2	39.3	44.7	**31.6**	42.6	36.6	35.5	38.38
	LoSiA (-Pro)	36.9	**10.3** (**9.4**)	**42.8**	**49.7**	30.7	43.0	**37.5**	**37.4**	**40.18**
LLaMA 2-7B	FFT	64.1	27.5	46.6	46.9	29.9	40.2	45.2	42.5	41.88
	LoRA	23.7	33.3	42.9	46.7	26.0	37.8	42.3	37.3	38.83
	PiSSA	23.7	33.1	43.5	46.2	26.8	36.6	42.7	38.5	39.05
	DoRA	24.2	68.3	**45.0**	**47.2**	26.0	34.4	44.1	36.7	38.90
	GaLore	23.7	39.6	42.2	45.3	28.0	39.0	43.1	41.2	39.80
	LoSiA (-Pro)	21.9	**26.1** (**21.8**)	44.7	46.7	**28.4**	**39.4**	**45.0**	**41.5**	**40.95**
LLaMA 2-13B	FFT-8Bit	77.1	53.0	61.2	55.7	35.7	43.2	53.6	56.2	50.93
	LoRA	36.9	56.0	58.6	**56.4**	34.1	44.8	52.6	53.7	50.03
	PiSSA	36.9	55.5	53.4	55.2	34.5	44.8	52.0	48.8	48.11
	LoSiA (-Pro)	36.9	**46.5** (**38.6**)	**59.0**	54.0	**34.9**	**48.2**	**53.1**	**55.7**	**50.82**

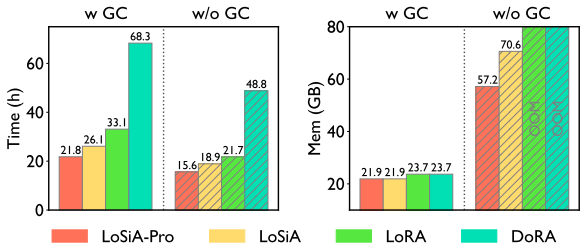


Figure 5: Overheads Comparison of PEFT methods training with and without Gradient Check-Pointing (GC). Taking training arguments in Table 1 as example.

ing batch size is set to 4, the warm-up ratio is set to 0.1 and the model is trained by 3 epochs. For training stability, the backbone model are in precision of BF16 and low-rank modules are upcasted to FP32. [2] All of the experiments are conducted on single NVIDIA A800 80GB GPU. Further details (including implementation details for common-sense reasoning tasks) can be found in Appendix A.3.

4.2 Main Results

Table 1 presents the performance of LoSiA compared to baseline methods across Gemma-2B, LLaMA2-7B, and LLaMA2-13B models. For GSM8K, we report 0-shot Chain-of-Thought (CoT) and 5-shot accuracy to reveal the model's reasoning capability and few-shot prompting performance. For MBPP, we report the Pass@1 and Pass@10 metrics. For MMLU, we report both 5-shot generation and perplexity-based results. The metrics are intended to measure the quality of generation and knowledge proficiency, respectively. Table 2 shows results on common-sense reasoning tasks, extracting the option with minimum perplexity and reporting ACC metric following lm-evaluation-harness. The test setup provides a robust measure of intrinsic knowledge acquisition.

LoSiA effectively reserves knowledge LoSiA demonstrates superior knowledge retention, as evidenced by perplexity-based evaluations. It outperforms LoRA by 2.48% on commonsense reasoning tasks and maintains an average 1.93% improvement on MMLU (0-shot, PPL). Unlike low-rank methods, LoSiA's sparse, high-rank fine-tuning approach enables localized knowledge retention while shifting likelihood toward correct answers.

LoSiA demonstrates superior performance in generalization In domain-specific tasks, LoSiA achieves average improvements of 1.75%, 1.15%, and 0.79% compared to the best baseline, respectively. High-rank update methods such as GaLore also exhibit relatively stable performance. The method shows its strength in problem-solving metrics (GSM8K, MBPP Pass@1, and MMLU 5-shot), suggesting that LoSiA provides strong generaliza-

[2] Trained with LLaMA-Factory (Zheng et al., 2024). Upcasting to FP32 only costs an additional 0.6GB of memory and trains faster than BF16 in practice.

Table 2: Comparison of PEFT Methods on Commen-Sense Reasoning Tasks, using LLaMA-2 7B as the backbone model. Evaluations are PPL-based in lm-evaluation-harness and we report the ACC metric.

Method	Mem(GB)	Time(h)	ARC-C	ARC-E	HellaSwag	Winogrande	PIQA	OBQA	SIQA	BoolQ	Avg.
LoRA	19.46	10.0	50.28	79.71	59.86	73.88	79.33	55.00	56.86	**88.07**	67.87
PiSSA	19.18	10.4	51.19	79.80	62.36	77.74	80.41	56.60	59.88	87.71	69.46
DoRA	20.42	25.6	51.71	79.34	59.86	**79.24**	79.98	59.60	59.57	88.04	69.67
GaLore	**18.24**	16.7	48.63	79.97	60.07	76.24	80.09	56.80	56.65	82.60	67.63
LoSiA	18.68	**9.2**	**52.22**	**80.26**	**65.05**	77.19	**81.50**	**61.40**	**61.05**	84.13	**70.35**

tion capabilities by applying learned knowledge to address various problems. Notably, while performing comparable to Full-Parameter Fine-Tuning (FFT) with only 0.64% of degradation in average, LoSiA significantly reduces computational resources, highlighting its practical efficiency.

LoSiA and LoSiA-Pro greatly improve training efficiency Figure 5 compares the training overheads of various PEFT methods. Contrast to baselines such as DoRA which incur significant additional FLOPs, LoSiA shows superior efficiency in both training time and memory usage. By eliminating extra matrix multiplication operations, LoSiA achieves faster training speeds. Its refined implement, LoSiA-Pro, further compresses activation storage by at least **22.8GB** (w GC) and reduce the training time up to **34%** (w/o GC) compared to LoRA by saving and computing on partial activations. A detailed training latency and GPU memory measurement is in Appendix A.4.

4.3 Ablation Study

This section assesses the functionality of sensitivity importance-aware localization, asynchronous mechanism, and re-warmups, alongside with robustness analysis of LoSiA. We present comprehensive ablation studies in Table 3 and training dynamics in Figure 6. Additional robustness tests for rank factor selection are provided in the Appendix.

Table 3: Ablation Study of LoSiA on GSM8K and MMLU, using LLaMA-2 7B as the backbone.

Model	GSM8K	MMLU	Avg.
Vanilla LoSiA	**44.66**	**44.95**	**44.81**
Synchronous Localization (*SL*)	42.76	44.13	43.45
Gradient-based Localization (*GL*)	43.00	44.88	43.94
w/o Warm-up during Selection (*WDS*)	38.06	44.21	41.14
w FFT lm_head (*FFTO*)	43.96	44.32	44.14

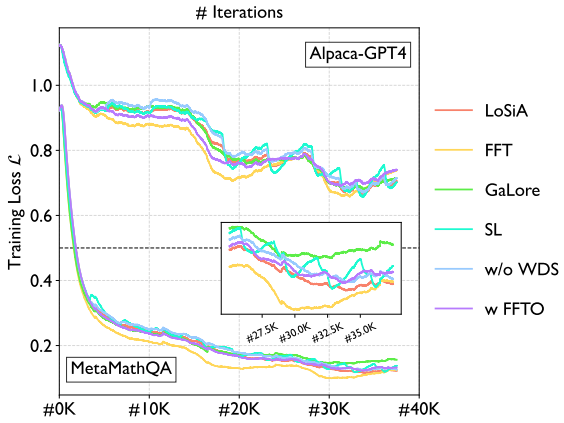


Figure 6: Loss Curves of Baselines and LoSiA Variants, training on MetaMathQA and Alpaca-GPT4.

Asynchronous mechanism yields more stable training Variant *SL* refers to using a synchronous layer-wise localization mechanism. However, it causes loss fluctuation, destabilizes later training, and degenerate the model performance by 1.36% in average, while asynchronous updates produce more stable loss curves.

Sensitivity-based Importance versus Gradient-based Importance Variant *GL* uses absolute gradients as the importance score. On MMLU, its performance remains comparable but is biased toward Humanities tasks (see Table 11), while its accuracy on GSM8K drops by 1.66%. Sensitivity-based scores, which aggregate multi-sample information, are more effective to capture general patterns in linear layers compared to biased gradients. However, gradient-based LoSiA exhibits promising results. In practice, the storage of $\overline{I}(\cdot), \overline{U}(\cdot)$ (about 1GB memory occupation on LLaMA-2 7B) can be eliminated using gradient-based importance if needed. Further discussion is provided in Appendix A.2.2.

Effect of rewarming and full fine-tuning the output layer The variant *w/o WDS*, which omits rewarm-ups, introduces instability of the loss, leads to under-fitting and ultimately impairs final performance. *w FFTO* fully fine-tunes the output layer,

shows a performance comparable to LoSiA with additional trainable parameters. It highlights the effectiveness of extracting tunable subnets on the output layer in LoSiA. In permissible GPU memory constraints, fully training the output layer shows promising performance and is also recommended.

Table 4: Robustness of Time Slot T Across a Series of Data Scales. Trained by MetaMathQA and evaluated by GSM8K on LLaMA-2 7B.

Method	@30K	@50K	@70K
LoRA	41.39	42.86	44.58
T		*LoSiA*	
25	**42.99**	**43.37**	42.07
50	**42.91**	42.46	42.15
75	41.09	**44.05**	**47.46**
100	40.49	**44.66**	**46.17**
125	39.88	42.23	**45.19**
150	39.12	40.41	42.84

Robustness across varying data scales and time slot lengths Table 4 assesses the performance of LoSiA across different training data scales. LoSiA consistently outperforms LoRA, demonstrating stability and robustness. Furthermore, the optimal time slot T is positively correlated with the size of training set, while LoSiA shows transcendent performance within a reasonable range of T.

4.4 Analysis

Selection Distribution We analyze the core subnet selection frequency of neurons in Figure 7. The frequently selected neurons remain similar under different rank factor p, while smaller p produces more concentrated distribution patterns. This indicates that LoSiA effectively identifies and optimizes critical neurons with limited training budgets, while simultaneously adjusting marginal parameters to further enhance generalization capability.

Reduce Intruder Dimensions Low-rank fine-tuning methods often introduce intruder dimensions (Shuttleworth et al., 2024b), resulting in spectral discrepancies between the fine-tuned and the pre-trained weights. This diminishes the adaptability of LoRA in sequential learning. Figure 8 illustrates the cosine similarity between the Top-500 singular vectors of the trained matrices and those of the original weights. Both LoRA and DoRA exhibit dimensional shifts due to their low-rank structures, whereas LoSiA demonstrates higher similarity and dimensional stability comparable to FFT.

To evaluate LoSiA's efficacy in continual learning, we perform sequential fine-tuning on Hellaswag, PiQA, BoolQ, SiQA, and Winogrande datasets on LLaMA-2 7B. We employs Average Performance (AP) (Chaudhry et al., 2018), Forward Transfer (FWT) (Lopez-Paz and Ranzato, 2017), and Backward Transfer (BWT) (Ke and Liu, 2023) to assess overall performance, knowledge transfer ability from previous tasks to current task, and level of forgetting, respectively. Details of the metrics and experiments are provided in A.3.4.

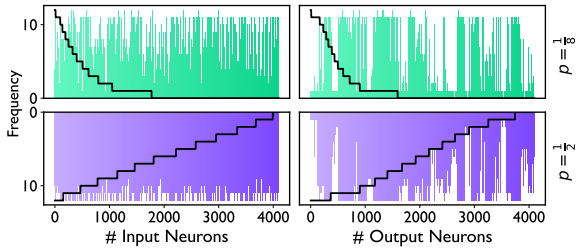


Figure 7: Selected Frequency Distributions of Neurons in Core Subnets. Sorted frequencies are ploted in black.

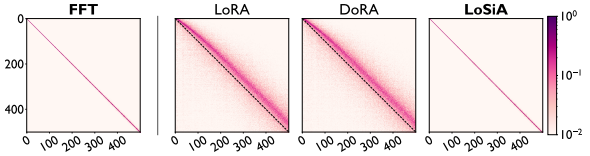


Figure 8: Similarities of Top-500 Largest Singular Vectors between Pre- and Post-Fine-Tuning Weights.

Table 5: Results of Continue Learning with Sequential PEFTs on Five Commen-Sense Reasoning Tasks.

Method	AP(↑)	FWT(↑)	BWT(↑)
Seq-LoRA	66.62	**1.46**	−8.04
Seq-LoSiA	**70.48**	−0.20	**−3.54**

The results in Table 5 demonstrate that LoSiA outperforms LoRA in mitigating forgetting with 4.5% in BWT and achieves a 3.86% improvement in average performance of sequential fine-tuning. This aligns with our hypothesis that LoSiA exhibits stronger robustness in continue learning, indicating that our method can adapt to more diverse application scenarios than existing baselines.

5 Conclusion

We present LoSiA, a novel PEFT framework that dynamically identifies and optimizes core subnetworks. Through sensitivity-based localization, asynchronous re-selection, and efficient high-rank adaptation, LoSiA achieves high throughput and low activation overhead. Extensive experiments show that LoSiA outperforms baselines on domain-specific and common-sense reasoning tasks while reducing forgetting. We hope that our work will inspire future research to further explore the intrinsic substructures in supervised fine-tuning.

6 Limitation

The innovative design of locating and optimizing sub-networks enables LoSiA to demonstrate outstanding advantages in terms of efficiency and performance. This work preliminarily validates the effectiveness of fine-tuning focused on substructures, yet there remains considerable room for further exploration and improvement. The effectiveness in scenarios such as multi-tasking, vision, and format alignment remains unclear. As for the method, the subnet localization in LoSiA is relatively rigid, and may still fail to precisely capture all critical neuron connections. More flexible and accurate approaches for the location of substructures, such as dynamically adjusting the rank factor for various layers, could further enhance performance.

Furthermore, while LoSiA can be conveniently integrated with other training platforms, additional efforts are required to improve its usability in real-world production scenarios. Currently, our work aims to provide individuals and small enterprises with a highly efficient single-GPU fine-tuning method, but the workflow could be further extended to multi-GPU environments. Moreover, to accommodate diverse datasets and practical deployment conditions, automated time slot selection mechanisms warrant further investigation.

References

2019. Winogrande: An adversarial winograd schema challenge at scale.

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*.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. 2024. Lora learns less and forgets less. *Preprint*, arXiv:2405.09673.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Shubhankar Borse, Shreya Kadambi, Nilesh Prasad Pandey, Kartikeya Bhardwaj, Viswanath Ganapathy, Sweta Priyadarshi, Risheek Garrepalli, Rafael Esteves, Munawar Hayat, and Fatih Porikli. 2024. Foura: Fourier low rank adaptation. *Preprint*, arXiv:2406.08798.

Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. *CoRR*, abs/1801.10112.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. Longlora: Efficient fine-tuning of long-context large language models. *Preprint*, arXiv:2309.12307.

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*.

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:1803.05457v1*.

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

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *Preprint*, arXiv:2307.08691.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, and 1 others. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

Yujie Feng, Xu Chu, Yongxin Xu, Guangyuan Shi, Bo Liu, and Xiao-Ming Wu. 2024. Tasl: Continual dialog state tracking via task skill localization and consolidation. *Preprint*, arXiv:2408.09857.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *Preprint*, arXiv:1803.03635.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Sreyan Ghosh, Chandra Kiran Reddy Evuru, Sonal Kumar, Ramaneswaran S, Deepali Aneja, Zeyu Jin, Ramani Duraiswami, and Dinesh Manocha. 2024. A closer look at the limitations of instruction tuning. *Preprint*, arXiv:2402.05119.

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

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

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

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.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

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

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and 1 others. 2024b. Mora: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*.

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

Zixuan Ke and Bing Liu. 2023. Continual learning of natural language processing tasks: A survey. *Preprint*, arXiv:2211.12701.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. 2024. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *Preprint*, arXiv:2404.15159.

Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. 2023. Relora: High-rank training through low-rank updates. *Preprint*, arXiv:2307.05695.

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*.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continuum learning. *CoRR*, abs/1706.08840.

Kai Lv, Hang Yan, Qipeng Guo, Haijun Lv, and Xipeng Qiu. 2024. Adalomo: Low-memory optimization with adaptive learning rate. *Preprint*, arXiv:2310.10195.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Preprint*, arXiv:2305.11627.

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

Xiangdi Meng, Damai Dai, Weiyao Luo, Zhe Yang, Shaoxiang Wu, Xiaochen Wang, Peiyi Wang, Qingxiu Dong, Liang Chen, and Zhifang Sui. 2024b. Periodiclora: Breaking the low-rank bottleneck in lora optimization. *Preprint*, arXiv:2402.16141.

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*.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. *Preprint*, arXiv:1906.10771.

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. 2023. Task-specific skill localization in fine-tuned language models. *Preprint*, arXiv:2302.06600.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Preprint*, arXiv:2005.07683.

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

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

Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. 2024a. Lora vs full fine-tuning: An illusion of equivalence. *Preprint*, arXiv:2410.21228.

Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. 2024b. Lora vs full fine-tuning: An illusion of equivalence. *arXiv preprint arXiv:2410.21228*.

Chongjie Si, Zhiyi Shi, Shifan Zhang, Xiaokang Yang, Hanspeter Pfister, and Wei Shen. 2025. Task-specific directions: Definition, exploration, and utilization in parameter efficient fine-tuning. *Preprint*, arXiv:2409.01035.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

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

Xujia Wang, Haiyan Zhao, Shuo Wang, Hanqing Wang, and Zhiyuan Liu. 2024b. Malora: Mixture of asymmetric low-rank adaptation for enhanced multi-task learning. *Preprint*, arXiv:2410.22782.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. *Preprint*, arXiv:2109.01652.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120*.

Menglin Yang, Jialin Chen, Yifei Zhang, Jiahong Liu, Jiasheng Zhang, Qiyao Ma, Harshit Verma, Qianru Zhang, Min Zhou, Irwin King, and Rex Ying. 2024. Low-rank adaptation for foundation models: A comprehensive review. *Preprint*, arXiv:2501.00365.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. 2025. Knowledge circuits in pretrained transformers. *Preprint*, arXiv:2405.17969.

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*.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. 2023. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

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

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2022. Platon: Pruning large transformer models with upper confidence bound of weight importance. *Preprint*, arXiv:2206.12562.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024a. Galore: Memory-efficient llm training by gradient low-rank projection. *Preprint*, arXiv:2403.03507.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024b. Sapt: A shared attention framework for parameter-efficient continual learning of large language models. *Preprint*, arXiv:2401.08295.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. 2024. Asymmetry in low-rank adapters of foundation models. *Preprint*, arXiv:2402.16842.

A Appendix

A.1 Derivations and Proofs

A.1.1 Proof for Formula 1

On a batch $\mathcal{B}$ composed of M samples, the MSE Loss between full fine-tuning (produces model f) and training with parameter set P (produces model f') is given by:

$$\mathcal{L}_{MSE} = \frac{\|y - y'\|_F^2}{M} = \frac{\|Wx - W'x\|_F^2}{M} \tag{10}$$

$$\leq \frac{\|W - W'\|_F^2 \|x\|_F^2}{M} \tag{11}$$

SGD In SGD optimizer, suppose learning rate is η, the difference in fine-tuned parameter is:

$$W - W' = -\eta 1_{(i,j)\notin P} \cdot \nabla_{W_0} \mathcal{L}(\mathcal{B}) \tag{12}$$

It derives a upper bound for the MSE Loss:

$$\mathcal{L}_{MSE} \leq \eta^2 \frac{\|1_{(i,j)\notin P} \cdot \nabla_{W_0} \mathcal{L}(\mathcal{B})\|_F^2 \|x\|_F^2}{M} \tag{13}$$

The result suggests that maximizing the sum of $\nabla_{W_0} \mathcal{L}(\mathcal{B})_{ij}$ where $(i,j) \in P$ ideally tightens the approximate error of training on subset.

AdamW In AdamW optimizer, at training step t, the first-order momentum M_t and second-order momentum V_t are calculated by:

$$G_t = \nabla_W \mathcal{L}(\mathcal{B}_t) \tag{14}$$

$$M_t = \beta_1 M_{t-1} + (1 - \beta_1) G_t \tag{15}$$

$$V_t = \beta_2 V_{t-1} + (1 - \beta_2) G_t^2 \tag{16}$$

$$\tilde{G}_t = \frac{M_t}{\sqrt{V_t + \epsilon}} \tag{17}$$

Similarly, since $W - W' = -\eta 1_{(i,j)\notin P} \cdot \tilde{G}_t$, we analysis relationship between $\tilde{G}_t$ and G_t:

$$\frac{\partial (\tilde{G}_t)^2}{\partial G_t} = 2M_t [\frac{(1-\beta_1)V_t}{V_t^2} - \frac{(1-\beta_2)G_t M_t}{V_t^2}] \tag{18}$$

Suppose $M_t > 0$, when $G_t < \frac{(1-\beta_1)V_t}{(1-\beta_2)M_t}$, $\frac{\partial (\tilde{G}_t)^2}{\partial G_t} > 0$. In practice, the typical settings are $\beta_1 = 0.9, \beta_2 = 0.999$. Therefore, when $G_t < 10^2 \frac{V_t}{M_t}$, $\tilde{G}_t$ increases with G_t, effectively covering a broad range of non-stationary optimization scenarios.

A.1.2 Proof for Formula 3

The foundational work was established by LeCun et al. (1989) and Kirkpatrick et al. (2016). However, for real-time importance calculation during training, approximations is necessary and is derived bellow. Element-wise importance score $I(\cdot)$ is formulated as:

$$\begin{aligned} I(W_k) =& |\Delta \mathcal{L}(\mathcal{D})| = |\mathcal{L}(\mathcal{D}) - \mathcal{L}_{W_k=0}(\mathcal{D})| \\ =& |\frac{\partial \mathcal{L}^T(\mathcal{D})}{\partial W_k} W_k - \frac{1}{2} W_k^T H_{kk} W_k \\ & + o(W_k^2)| \end{aligned} \tag{19}$$

where H donates the Hessian matrix. It is computational intensive for Hessian calculation, we therefore use fisher information matrix F for diagonal elements instead:

$$\begin{aligned} F_{kk} =& - H_{kk} = -E_{p(\theta|\mathcal{D})}[\frac{\partial^2 \mathcal{L}(\theta, D)}{\partial^2 \theta_k}|_{\theta=\theta^*}] \\ \approx& - E_{(x,y)\sim\mathcal{D}}[(\frac{\partial \mathcal{L}(\theta, x, y)}{\partial \theta_k}|_{\theta=\theta^*})^2] \end{aligned} \tag{20}$$

Approximating the expectation by dataset $\mathcal{D}$ based on Monte Carlo Method, it derives:

$$\begin{aligned} I(W_k) =& |\frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_k} W_k + o(W_k^2) \\ & - \sum_{(x,y)\in\mathcal{D}} \frac{1}{2|\mathcal{D}|} (\frac{\partial \mathcal{L}(x,y)}{\partial W_k})^2 W_k^2| \end{aligned} \tag{21}$$

During training, the dataset $\mathcal{D}$ is processed in batches $\mathcal{B}_i$, and the batch gradient is calculated as $\nabla_W \mathcal{L}(\mathcal{B}_i) = \frac{1}{M} \sum_{j=1}^{M} \nabla_W \mathcal{L}(\mathcal{B}_{ij})$. To avoid calculating the gradients separately for each sample in the batch, we approximate $\sum_{(x,y)\in\mathcal{B}_i} \frac{1}{M} (\frac{\partial \mathcal{L}(x,y)}{\partial W_k})^2$ to the term $(\frac{\sum_{(x,y)\in\mathcal{B}_i} \frac{\partial \mathcal{L}(x,y)}{\partial W_k}}{M})^2$. To analyze errors, assume $g = \frac{\partial \mathcal{L}(x,y)}{\partial W_k} \sim G$, we have:

$$\begin{aligned} \Delta =& |\frac{1}{M} \sum_{j=1}^{M} g_j^2 - (\frac{\sum_{i=1}^{M} g_j}{M})^2| \\ =& \frac{1}{M} \sum_{i=1}^{M} g_j^2 - (\frac{\sum_{j=1}^{M} g_j}{M})^2 \\ \leq& \frac{(\max g_j - \min g_j)^2}{4} = O(g^2) \end{aligned} \tag{22}$$

The approximation errors are bounded. We take the following for importance estimation:

$$I_i = |\frac{\partial \mathcal{L}(\mathcal{B}_i)}{\partial W_k} W_k - \frac{1}{2} (\frac{\sum_j \frac{\partial \mathcal{L}(\mathcal{B}_{ij})}{\partial W_k}}{M} W_k)^2 + o(W_k^2)| \tag{23}$$

A.1.3 Maximizing Formula 7 is NP-Hard

Task *Given an arbitrary matrix $A^{n\times m}$, select $\tilde{n}$ ($\tilde{n} \leq n$) rows X_S and $\tilde{m}$ ($\tilde{m} \leq m$) columns Y_S, to maximize the sum $\sum_{i\in X_S}\sum_{j\in Y_S} A_{ij}$.*

Lemma (The Maximum Clique Problem is NP-Complete) Given an undirected graph $G = (V, E)$, where:

- V is a set of vertices,
- $E \subseteq V \times V$ is a set of edges,

a clique $C \subseteq V$ is a subset of vertices such that every two distinct vertices in C are adjacent, i.e.,

$$\forall u, v \in C\ , u \neq v \Rightarrow (u, v) \in E.$$

The Maximum Clique Problem (MCP) seeks a clique of maximum cardinality in G. The problem is **NP-complete**, meaning:

- It is **NP**: a candidate solution can be verified in polynomial time, and
- It is **NP-Hard**: any problem in NP can be reduced to it in polynomial time.

Proof Construct a special form of $A^{n\times n}$ as the adjacent matrix of graph G with larger values on diagonal maximum, that is:

$$A_{uv} = \begin{cases} 1 & \text{if } (u, v) \in E, \\ n^2 + 1 & \text{if } u = v \\ 0 & \text{otherwise} \end{cases}$$

Then, MCP can be reduced to the task in polynomial time following the algorithm:

- Enumerate k in descending order $n, n-1 \ldots 1$
- Solve the task with $\tilde{n} = \tilde{m} = k$
- If the optimal solution equals to $(n^2 + k)k$, then there exist a clique $C = X_S$ of size k, terminate.

Therefore, a NP-Complete problem can be reduced to the task in polynomial time, which yields the conclusion that the task is NP-Hard.

A.2 Further Observations

A.2.1 Gradient Magnitude Distribution

To investigate the universality of the sparse subnetwork structure for large gradients, we analyze gradient magnitude distributions across different layers, as shown in Figure 9. Both Gradients of the Self-Attention and MLP modules exhibits the consistent structure of core subnets.

A.2.2 Gradient- or Sensitivity-Based

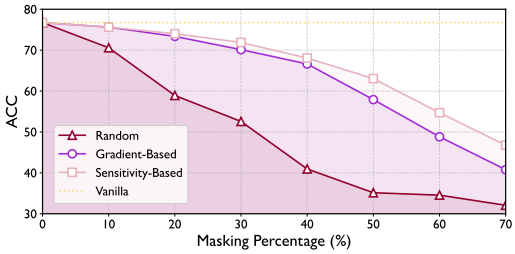


Figure 10: ARC-E Accuracy under Different Masking Percentage. Linear layers in the 10-th to the 25-th decoder layer of LLaMA-2 7B are masked with gradient-based and sensitivity-based subnet selection strategies.

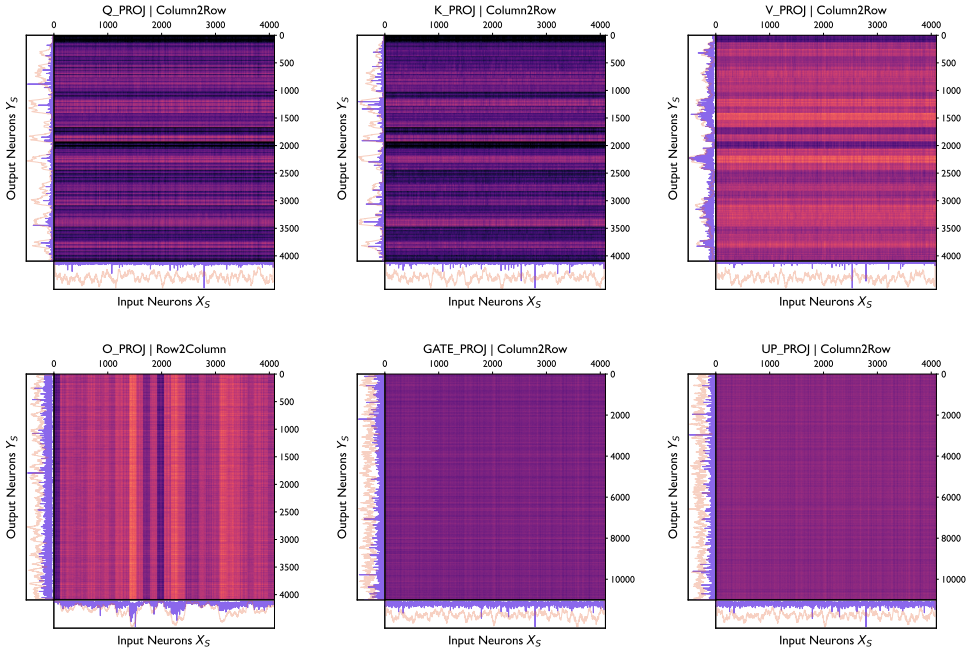
Figure 10 presents the performance on ARC-E across varying masking percentages. The gradient-based approach identifies the subnet based on magnitude of gradients while masking the remaining parameters. Among importance scoring strategies, the sensitivity-based approach, which is adopted by LoSiA, exhibits stronger robustness in higher masking ratios. However, tuning hyperparameters β_1, β_2 in the EMA of sensitivity-based importance calculation may result in marginal return for LoSiA, as evidenced by the minimal performance gap between the refined and unrefined selection methods.

A.3 Experiments Details

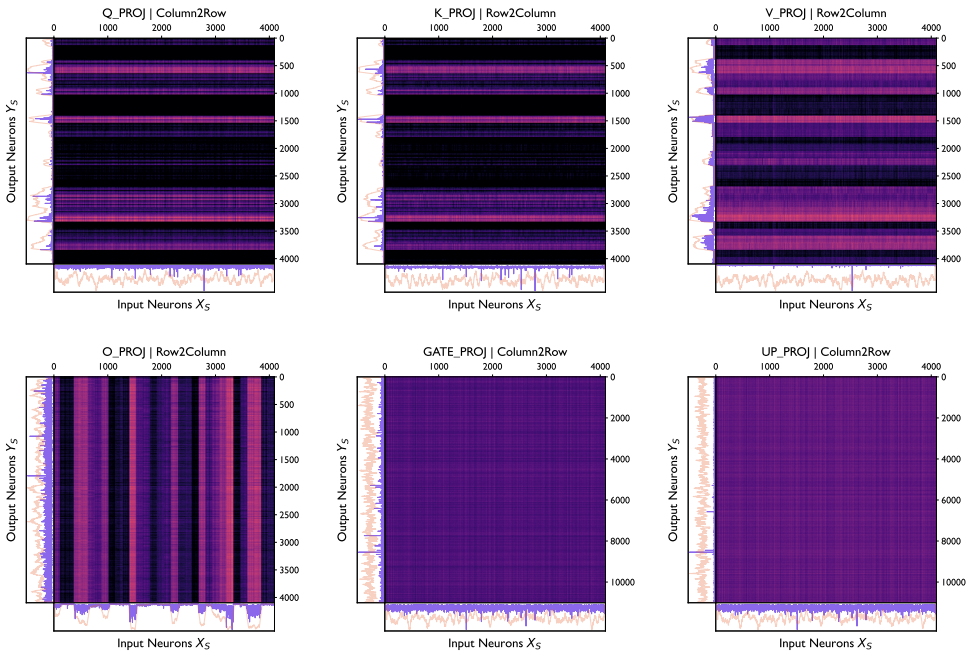
A.3.1 Domain Specific Tasks

We randomly sample 50K data from open-source training datasets: MetaMathQA (Yu et al., 2023), Magicoder (Wei et al., 2023) and Alpaca-GPT4 (Peng et al., 2023), and evaluate fine-tuned models on GSM8K (Cobbe et al., 2021), MBPP (Austin et al., 2021) and MMLU (Hendrycks et al., 2021b,a), respectively. Evaluations are conducted using lm-evaluation-harness (Gao et al., 2024), with baseline implementations from LLaMA-Factory (Zheng et al., 2024).

Table 6 shows the hyperparameters for fine-tuning LLaMA-2 7B on MetaMathQA. We follow the commonly used configurations for baselines, while aligning GPU memory consumptions. For LoSiA, the hyperparameters for each task and model are listed in Table 7. Rank factor p is set to $\frac{1}{8}$, and the gradient dimension of lm_head is compressed to a fraction by p_o. Time slot T and learning rate may various across tasks. All experiments are conducted with single run on a NVIDIA A800-80GB GPU and CentOS 7 on x86-64 CPUs. Pytorch version is 2.4.1.



(a) Decoder Layer 15



(b) Decoder Layer 25

Figure 9: Gradient Magnitude Distribution on LLaMA-2 7B for Different Decoder Layers and Modules. Purple curve: row/column gradient sums. Orange curve: smoothed neuron selecting frequency. Best selection strategy for each layers (Row2Column/Column2Row) are record in the title of subplots.

Table 6: Hyperparameter Configurations of Fine-Tuning LLaMA-2 7B on MetaMathQA. Note that β_1, β_2 are EMA smoothing factors in sensitivity-based importance calculation, and are fixed across all experiments. p and p_o are dimension factors determining the shape of core subnets. T of LoSiA refers to the time slot between re-selections.

	LoRA/DoRA	**PiSSA**	**GaLore**	**LoSiA**
Optimizer	AdamW			
Epochs	3			
Batch Size	4			
LR	$2e-4$	$1e-4$	$1e-4$	$6e-5$
Cutoff Length	2048			
Warm-up Ratio	0.1			
Rank Related	$r = 64$		$r = 512$	$p = \frac{1}{8},\ p_o = \frac{1}{8}$
Scale Related	$\alpha = 128$		$\alpha = 2.0$	-
Period Related	-		$T = 200$	$T = 100$
Others	-		Full Proj	$\beta_1 = \beta_2 = 0.85$
Implement Layer	proj_q,proj_k,proj_v,proj_o, up_proj,down_proj,gate_proj		proj_q,proj_k,proj_v,proj_o, up_proj,down_proj,gate_proj, lm_head	

Table 7: Hyperparameter Configurations of LoSiA across different tasks and models.

Datasets	**MetaMathQA**	**Magicoder**	**Alpaca-GPT4**
LR	$6e-5$	$5e-5$	$5e-5$
Time Slot T	100	150	150
Rank Factor p	$\frac{1}{8}$		
Models	**Gemma-2B**	**LLaMA-2 7B**	**LLaMA-2 13B**
Vocabulary Size	$256,000$	$32,000$	$32,000$
Dimension Factor p_o	$\frac{1}{64}$	$\frac{1}{8}$	1

A.3.2 Common-Sense Reasoning Tasks

Table 8: Datasets of Common-Sense Reasoning.

Datasets	**#Train**	**#Test**	**Task Type**
ARC-C(Clark et al., 2018)	1,120	1,170	Q & A
ARC-E (Clark et al., 2018)	2,250	2,380	Q & A
HellaSwag (Zellers et al., 2019)	39,905	10,042	Sentence Completion
Winogrande (ai2, 2019)	9,248	1,267	Fill the Blank
PIQA (Bisk et al., 2020)	16,100	1,840	Q & A
OBQA (Mihaylov et al., 2018)	4,957	500	Q & A
SIQA (Sap et al., 2019)	33,410	1,954	Q & A
BoolQ (Clark et al., 2019)	9,427	3,270	Text Classification

The datasets of common-sense reasoning tasks are presented in Table 8, while corresponding hyper-parameters detailed in Table 9. The GPU memory usage remain aligned. For each PEFT baselines, searches in learning rate are performed.

We report the accuracy metric evaluated by lm-evaluation-harness, which selects answers based on minimal perplexity. This approach mitigates the sensitivity of models to input phrasing variants, thereby enabling a more reliable measurement of the implicit knowledge encoded within the models.

A.3.3 Rank Factor Robustness

To evaluate the impact of the rank factor p, which determines the scale of the core subnets, we conduct an ablation study on MetaMathQA. The results demonstrate LoSiA's robustness across various subnet scales. Note that $p = 1/16$ may

Table 9: Hyperparameter Configurations of Fine-Tuning LLaMA-2 7B on Common-Sense Reasoning Datasets.

	LoRA/DoRA	PiSSA	GaLore	LoSiA
Optimizer	AdamW			
Epochs	3			
Batch Size	16			
LR	$\{1e-4, 2e-4\}$	$\{5e-5, 1e-4\}$	$\{1e-4, 2e-4\}$	$\{5e-5, 1e-4\}$
Cutoff Length	256			
Warm-up Ratio	0.1			
Rank Related	$r = 64$		$r = 512$	$p = \frac{1}{8},\ p_o = 1$
Scale Related	$\alpha = 128$		$\alpha = 2.0$	-
Period Related	-		$T = 200$	$T = 50$
Others	-		Full Proj	$\beta_1 = \beta_2 = 0.85$
Implement Layer	proj_q,proj_k,proj_v,proj_o, up_proj,down_proj,gate_proj		proj_q,proj_k,proj_v,proj_o, up_proj,down_proj,gate_proj, lm_head	

be relatively small for effective subnet fine-tuning, while increasing computational budget boosts the performance.

Table 10: Rank Factor Robustness on GSM8K

Model	1/16	1/8	1/4	1/2
Gemma-2B	37.53	42.84	45.03	45.64
LLaMA-2 7B	40.64	44.66	46.02	48.45

Table 11: The Detail of Ablation Study on MMLU. Note that the variant *GL* surpasses LoSiA on Humanities but shows performance drop on the rest of domains.

Model	MMLU				
	Humanities	Other	Social S	STEM	Avg.
Sensitivity-based Localization (LoSiA)	41.70	**52.23**	**50.89**	**36.82**	**44.95**
Gradient-based Localization (*GL*)	**42.64**	51.41	50.62	36.22	44.88

A.3.4 Continue Learning

To examine whether reduction of intruder dimensions in LoSiA mitigates forgetting in continue learning, we sequentially adapt LLaMA-2 7B through five common-sense reasoning tasks by the order HellaSwag, PIQA, BoolQ, SIQA and Winogrande. Learning rate for LoRA is $1e-4$ and for LoSiA is $5e-5$. The remaining hyperparameters are consistent with Table 9. LoRA modules are merged into the backbone before subsequent task adaptation.

Suppose the model learn sequentially on N tasks. Let $P_{i,j}$ denote the ACC on task j after training on task i. Following Zhao et al. (2024b), we formulate the metrics (AP, FWT and BWT) as bellow:

Average Performance: $AP = \frac{1}{N}\sum_{i=1}^{N} P_{N,i}$

Forward Transfer: The metric measures the transferability of learned knowledge from previous tasks to a new task. FWT $= \frac{1}{N}\sum_{i=1}^{N}(P_{i,i} - P_{0,i})$, where $P_{0,i}$ is the performance of individually training task i.

Backward Transfer: The metric evaluates the impact of learning later tasks on the model's performance on an earlier task, that is BWT $= \frac{1}{N-1}\sum_{i=1}^{N-1}(P_{N,i} - P_{i,i})$.

Table 12 shows the detailed result during sequential adaptation. After continuing learning through all tasks, Seq-LoSiA outperforms Seq-LoRA across all benchmarks, highlighting its efficiency in forgetting mitigating.

A.4 Resources Measurement

Figure 11 and 12 shows the memory and training time overheads for different PEFT methods on LLaMA-2 7B. With GRADIENT CHECKPOINTING, LoSiA and LoSiA-Pro display lower latency than low-rank methods across all ranks.

When disables GRADIENT CHECKPOINTING, LoSiA-Pro significantly reduces activation storage by at least 26% and supports 70% additonal training context length under consistent GPU memory constraints compared to LoRA.

A.4.1 Memory Estimate

Consider a model with L decoder layers, each containing K tunable matrices. The model use b-bit

Table 12: Details of Performances on Continue Learning Five Common-Sense Reasoning Tasks. The column stands for training order, while the label "ST" indicates the result in single-tasking training.

Method	Task	(#1) HellaS	(#2) PIQA	(#3) BoolQ	(#4) SIQA	(#5) WinoG	ST
Seq-LoRA	HellaSwag	59.86	55.64	59.10	57.86	54.36	59.86
	PIQA	76.01	80.52	77.86	78.73	77.64	79.33
	BoolQ	77.80	73.27	86.30	80.12	75.93	88.07
	SIQA	45.80	47.80	45.85	59.52	46.11	56.86
	Winogrande	64.25	68.35	68.82	69.93	79.08	73.88
Seq-LoSiA	HellaSwag	63.72	61.89	61.11	60.37	**56.43**	63.72
	PIQA	78.29	79.49	79.82	79.38	**77.75**	81.50
	BoolQ	77.52	70.76	83.24	82.54	**81.99**	84.13
	SIQA	47.80	48.26	48.26	59.93	**56.04**	61.05
	Winogrande	68.51	67.88	68.51	71.82	**80.19**	77.19

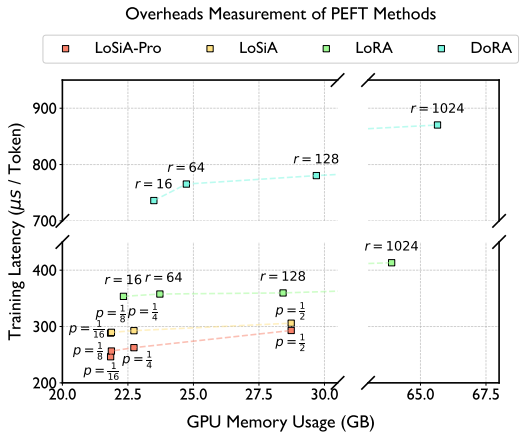


Figure 11: GPU Memory Usage and Training Latency Comparison w Gradient Checkpointing

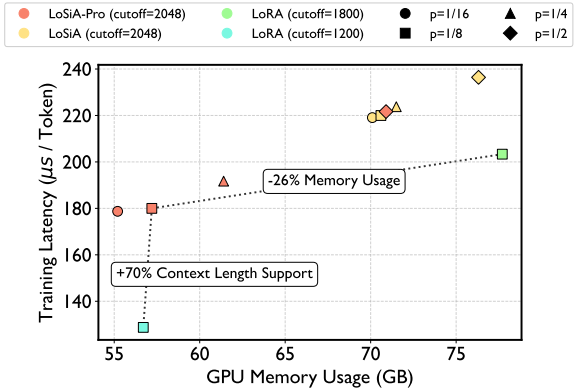


Figure 12: GPU Memory Usage and Training Latency Comparison w/o Gradient Checkpointing

precision storage, with hidden dimension d and vocabulary size V. Table 13 shows GPU memory consumption details of LoRA, GaLore and LoSiA.

Table 13: Comparison of Memory Consumptions. Cells in green highlight the components that may notably lower than other methods, while in red highlight the components that may cause relatively large memory consumption.

	LoRA	GaLore	**LoSiA**
Update Rank	r	R	pd
#Trainable	$2LKrdb$	LKR^2b+Vdb	$LKd^2p^2b+Vdp_ob$
#Optimizer	$4LKrdb$	$2(LKR^2b+Vdb)$	$2(LKd^2p^2b+Vdp_ob)$
#Gradient	$2LKrdb$	$\max\{d^2b, Vdb\}$	$\max\{d^2b, Vdb\}$
#Auxiliary	$2LKrdb$	$2LKRdb$	$2Kd^2b$
#Total	$8LKrdb$	$2(LKR^2b+Vdb)$ +$\max\{d^2b, Vdb\}$ +2LKRdb	$2(LKd^2p^2b+Vdp_ob)$ +$\max\{d^2b, Vdb\}$ +2Kd^2b

For optimizers like AdamW, LoSiA reduces the gradient dimension of the output layer to a fraction p_o, while GaLore performs full fine-tuning on the output layer of shape $d \times V$. Both GaLore and LoSiA utilize per-layer weight update techniques for gradient computation. In contrast, basic LoRA implementations require collecting all gradients after backward propagation. The reduced memory overhead of gradient storage allows GaLore and LoSiA to fully train the `lm_head` in LLaMA-2 models.

In terms of auxiliary parameters, GaLore requires storing down- and up-projection matrices. Since R is typically high-rank, GaLore's auxiliary parameters can be significantly larger than those of other methods.

For LoSiA, auxiliary parameters are used to compute the importance scores ($\overline{U}(\cdot)$ and $\overline{I}(\cdot)$). If gradient-based importance scoring is adopted, this component can be completely eliminated.

Regarding total memory consumption, increasing the rank in LoRA and GaLore incurs substantial overhead. However, in LoSiA, only the term

$2(LKd^2p^2b + Vdp_ob)$ scales with rank factor p, resulting in a more efficient memory footprint.

Table 14: Details of Trainable Parameters for LoRA and LoSiA under Different Hyperparameter Configurations on LLaMA-2 7B.

LoRA				
Rank r	16	64	256	1024
#Trainable	40.0M	160.0M	640.0M	2560.0M
Mem(GB)	22.33	23.72	28.42	63.90
LoSiA				
Factor p	1/16	1/8	1/4	1/2
Update Rank r	256	512	1024	2048
$\mathbf{p_o = 1/8}$				
#Trainable	42.8M	122.1M	439.3M	1700.8M
Mem(GB)	21.84	21.87	22.73	28.73
$\mathbf{p_o = 1}$				
#Trainable	158.0M	238.9M	562.2M	1855.7M
Mem(GB)	22.24	22.84	23.37	28.98

A.4.2 Latency Measurement

We measure the training latency (μs / token) fine-tuning with different PEFT methods on LLaMA-2 7B, and the results are shown in Table 15. The experiments are conducted with cutoff_len $= 2048$ and batch _size $= 4$.

Table 15: Comparison of Training Latency on LLaMA-2 7B. Latencies are reported in measuerments of μs per token, training with FLASH-ATTENTION 2 (Dao, 2023).

	Forward	**Backward**	**Other**	**Total**
w Gradient Check-Pointing				
$\text{LoRA}_{r=64}$	74.0	264.0	0	338.0
$\text{DoRA}_{r=64}$	104.2	552.2	0	656.4
$\text{GaLore}_{R=512}$	**70.1**	227.5	140.1 (574s / 500 step)	437.7
LoSiA$_{p=\frac{1}{8}}$	**70.0** (-5.6%)	220.4 (-16.5%)	0	290.4 (-14.1%)
LoSiA-Pro$_{p=\frac{1}{8}}$	71.4 (-3.5%)	**173.4** (-34.3%)	0	**244.8** (-27.6%)
w/o Gradient Check-Pointing				
$\text{LoRA}_{r=64}$	Out of Memory			
LoSiA$_{p=\frac{1}{8}}$	**70.0** (-5.6%)	146.5 (-44.5%)	0	216.5 (-35.1%)
LoSiA-Pro$_{p=\frac{1}{8}}$	71.4 (-3.5%)	**102.4** (-61.3%)	0	**173.8** (-49.6%)

While demonstrating superior performance among existing baselines, LoSiA reduces training latency by 14.1% compared to LoRA, 55.8% compared to DoRA. The acceleration is mainly due to the elimination of low-rank matrix multiplication. Specifically, during backward propagation with GRADIENT CHECKPOINTING, the production of low-rank matrices introduces significant overhead for activation recomputation and gradient calculation. Note that LoRA can avoid gradient calculations on backbone weights, but this requires specialized implementations and introduces a large coefficient of computational complexity.

For LoSiA-Pro, the computational complexity remains the same as LoSiA during the forward pass, but it only requires storing a proportion p of the input activations of the linear layers. During the backward pass, LoSiA-Pro reduces the computational cost to p^2 relative to full gradient computation, which significantly lowers the latency of backward propagation. This results in highly efficient training and lower GPU memory consumption.