Forward Knows Efficient Backward Path: Saliency-Guided Memory-Efficient Fine-tuning of Large Language Models

Yeachan Kim[1] **and SangKeun Lee**[1,2]
[1]Department of Artificial Intelligence, Korea University, Seoul, South Korea
[2]Department of Computer Science and Engineering, Korea University, Seoul, South Korea
`{yeachan,yalphy}@korea.ac.kr`

Abstract

Fine-tuning is widely recognized as a crucial process for aligning large language models (LLMs) with human intentions. However, the substantial memory requirements associated with fine-tuning pose a significant barrier to extending the applicability of LLMs. While parameter-efficient fine-tuning can be a promising approach by reducing trainable parameters, intermediate activations still need to be cached to compute gradients during the backward pass, thereby limiting overall memory efficiency. In this work, we propose **Sa**liency-Guided **G**radi**e**nt Flow (SAGE), a memory-efficient fine-tuning method designed to minimize the memory specifically associated with cached intermediate activations. The key strategy is to selectively cache activations based on their saliency during the forward pass and then use these activations for the backward pass. This process transforms the dense backward pass into a sparse one, thereby enhancing memory efficiency. To verify whether SAGE can serve as an efficient alternative for fine-tuning, we conduct comprehensive experiments across diverse fine-tuning scenarios and setups. The experimental results show that SAGE substantially improves memory efficiency without a significant loss in accuracy, highlighting its broad value in real-world applications[1].

1 Introduction

Large language models (LLMs) have demonstrated their versatility across a wide range of research and industrial fields. Their ability to generalize across diverse tasks and transfer knowledge across domains makes them essential for solving complex problems. To ensure that LLM behaviors align with human intentions and the specific requirements of downstream tasks, fine-tuning plays a crucial role. However, this fine-tuning process poses significant challenges due to the enormous memory demands it imposes (Zhao et al., 2024; Miles et al., 2024), rendering the process resource-intensive and often inaccessible without substantial infrastructure.

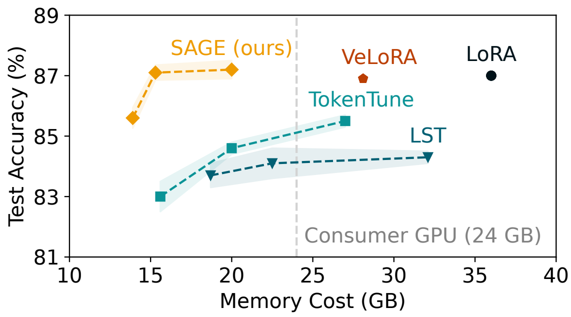


Figure 1: Accuracy-Memory trade-off for baselines and ours on Physical Interaction Question Answering task.

One promising approach to these challenges is parameter-efficient fine-tuning (PEFT) (Houlsby et al., 2019; Hu et al., 2021), which updates only a subset of parameters instead of the entire set. As optimizer states for trainable parameters occupy a significant portion of memory, PEFT can substantially reduce memory usage by limiting the trainable parameters. However, regardless of which parameters are trainable, fine-tuning still requires caching all intermediate sequential activations to calculate the gradients during the backward pass, thereby limiting overall memory efficiency (Sung et al., 2022; Simoulin et al., 2024). Moreover, unlike other factors (*e.g.,* optimizers, model weights), the memory consumption from intermediate activations can increase significantly depending on the task and setup (*e.g.,* document understanding), posing a potential bottleneck to the broader applicability of LLMs.

In this work, we raise the key research question: *Do LLMs truly need all sequential activations to learn the desired task during the backward pass?* Prior studies have found that intermediate activations exhibit significant redundancy (Ethayarajh, 2019; Dai et al., 2020), indicating that less contributing activations have a negligible impact

[1]Our code is available at `https://github.com/yeachan-kr/sage`

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9341–9356

on task objectives (Goyal et al., 2020; Kim and Lee, 2024; Fu et al., 2024). Inspired by these findings, we hypothesize that a selective caching strategy—retaining only the activations that contribute significantly to the task—can achieve memory efficiency without compromising task accuracy.

We propose **Sa**liency-Guided **G**radi**e**nt Flow (SAGE), a novel memory-efficient fine-tuning designed to selectively cache activations based on their saliency[2] instead of storing all activations. During the forward pass, SAGE identifies the most contributive activations to tasks, while less significant activations are used solely for the forward without caching. After the forward pass, the backward computation is confined to the sparse backward paths associated with these selectively cached activations, resulting in reduced memory consumption. Furthermore, SAGE is designed in a plug-and-play manner, making it easily applicable to a wide range of LLMs. This allows for seamless integration with other efficient methods, thereby comprehensively reducing memory costs.

To evaluate the practical benefits of our approach, we conduct comprehensive experiments on a range of benchmarks and fine-tuning setups—including instruction fine-tuning and quantized fine-tuning—across models of varying scales. The empirical results indicate that SAGE consistently achieves better memory efficiency compared to strong baselines (Figure 1). In summary, the key contributions of this paper include the followings:

- We propose SAGE, a memory-efficient fine-tuning designed to reduce the memory usage associated with cached intermediate activations based on the saliency.
- We verify that the plug-and-play nature of SAGE allows memory-efficiency on diverse setups, highlighting the general applicability.
- We demonstrate that SAGE substantially reduces memory usage during fine-tuning without compromising task accuracy.

2 Related Work

Parameter-Efficient Fine-Tuning. Fine-tuning all parameters is prohibitive in the era of LLMs, which is why PEFT has received significant attention. Instead of optimizing all parameters, PEFT selects a small subset of parameters or introduces low-rank approximating layers. For example, Houlsby et al. (2019) and Pfeiffer et al. (2021) proposed adapters consisting of bottleneck layers that are only fine-tuned during training. Hu et al. (2021) proposed LoRA, which integrates low-rank adaptation layers with pre-trained weights. Building on these foundational methods, subsequent works have explored PEFT approaches through factorization (Liu et al., 2024), routing (Choi et al., 2023; Kim et al., 2024), and representation tuning (Wu et al., 2024). However, regardless of which parameters are trainable, the fine-tuning still requires caching huge intermediate activations to compute gradients, limiting overall memory efficiency.

Memory-Efficient Fine-Tuning. Memory-efficient fine-tuning (MEFT) has emerged as an important research direction. For example, Sung et al. (2022) proposed Ladder Side Tuning (LST) that trains side networks that utilizes the intermediate activations from the original LLMs. Simoulin et al. (2024) proposed TokenTune which achieves efficiency by randomly dropping activations for the backward. To address memory usage from model weights, several works have built on quantized LLMs. Dettmers et al. (2023) introduced QLoRA, which trains 16-bit LoRA weights on the 4-bit quantized LLMs, reducing both model weights and optimizer states. Zhang et al. (2024) proposed QST, a quantized version of LST that trains 16-bit side networks with the quantized LLMs. Several studies have explored approximating the expensive back-propagation through projections. Zhao et al. (2024) proposed Galore, which projects gradients into a low-rank space. Similarly, Miles et al. (2024) introduced VeLoRA, which projects intermediate activations into a low-dimensional subspace. Instead of caching intermediate activations, Liao et al. (2024) proposed reversible networks that recompute activations during the backward pass. However, these methods incur additional computational and time costs to achieve memory efficiency.

Compared to previous works, SAGE offers distinct advantages. First, SAGE incorporates task-specific saliency to reduce memory consumption, optimizing the memory by focusing on the task and individual training data. Furthermore, SAGE achieves efficiency without additional networks (*e.g.,* side networks), computationally expensive updates (*e.g.,* projection weights), and recomputation, thereby mitigating associated overheads.

[2] *Saliency* measures how much each input element contributes to the model's prediction (Ding and Koehn, 2021).

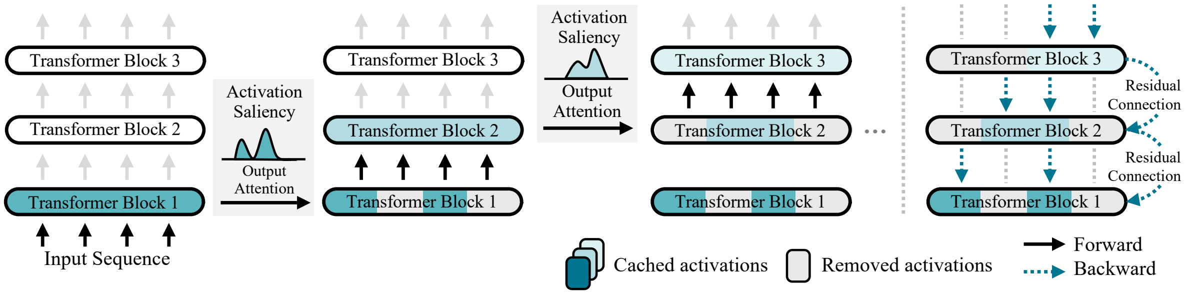


Figure 2: Overview of SAGE. During the fine-tuning, model performs the forward pass as usual, but it caches only the salient activations—those identified based on attentions between output and input tokens. After the forward, the backward pass is confined to the sparse backward paths associated with these cached salient activations.

3 SAGE: Saliency-Guided Gradient Flow

In this section, we detail Saliency-Guided Gradient Flow (SAGE) for memory-efficient fine-tuning of LLMs. The key strategy is to selectively cache activations based on their saliency, ensuring that gradients propagate solely through these sparse yet crucial backward paths. We begin by revisiting the back-propagation process and defining the objective of our method (§3.1). We then discuss how to choose the salient activations for the backward (§3.2) and determine the number of activations to store (§3.3). Finally, we describe the alignment process that reduces the discrepancy between the dense forward pass and the sparsified backward pass (§3.4). Figure 2 provides an overview of the complete process.

3.1 Preliminaries and Problem Formulation

We begin by revisiting the back-propagation through the lens of intermediate activations. For simplicity, we illustrate the key concept of the proposed method using a dense layer.

Back-propagation path Considering the dense layer $a = \sigma(z) = \sigma(hW + b)$ with weight W, bias b, non-linear function $\sigma(\cdot)$, input h, pre-activation z, and output a. The gradients with respect to W and b when back-propagating a loss $\mathcal{L}$ through the layer as follow:

$$\begin{aligned}\frac{\partial \mathcal{L}}{dW} &= \frac{\partial \mathcal{L}}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial W} = \frac{\partial \mathcal{L}}{\partial a}\sigma' h \\ \frac{\partial \mathcal{L}}{db} &= \frac{\partial \mathcal{L}}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial b} = \frac{\partial \mathcal{L}}{\partial a}\sigma'\end{aligned} \tag{1}$$

This propagation process reveals that all intermediate activations h should be cached to calculate the gradients of the weights, resulting in a significant increase in memory consumption during fine-tuning (Sung et al., 2022; Simoulin et al., 2024).

Sparsified back-propagation path In response, our approach aims to sparsify the back-propagation process by reducing the number of cached token activations. We denote this reduced set of activations as S, which comprises a set of positional indices for the selected activations. The modified backward path can then be represented as follows:

$$\begin{aligned}\frac{\partial \mathcal{L}}{dW} &= \sum_{i\in S}\frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial a_i}{\partial z_i}\frac{\partial z_i}{\partial W} = \sum_{i\in S}\frac{\partial \mathcal{L}}{\partial a_i}\sigma_i' h_i \\ \frac{\partial \mathcal{L}}{db} &= \sum_{i\in S}\frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial a_i}{\partial z_i}\frac{\partial z_i}{\partial b} = \sum_{i\in S}\frac{\partial \mathcal{L}}{\partial a_i}\sigma_i'\end{aligned} \tag{2}$$

where the subscript for activations indicate the positional indices over the sequence. An illustrative example of this modified propagation is shown in Figure 3. This approach requires caching only a smaller subset of activations S to compute gradients. Although it is possible to trace activation positions across all operations, we empirically set these positions on a block-by-block basis within transformers (*i.e.,* $S_1, S_2, \ldots, S_L$, where L represents the number of blocks) to avoid overheads. Therefore, the positions of cached activations remain consistent across all operations within the transformer block (*e.g.,* self-attention, MLP).

Note that caching only a few activations potentially creates a break in the backward graph due to the chain rule. However, the residual connections in transformers (Vaswani et al., 2017) effectively prevent such breaks during the backward. The following sections detail how we construct this small yet crucial set S during the forward pass.

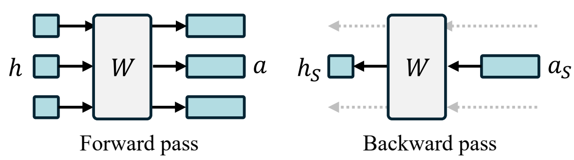


Figure 3: Illustration of the sparsified backward.

3.2 Identification of Salient Activations

To satisfy our objective (*i.e.,* a small but contributing activation set S), we need to identify which activations are crucial to perform the task. We draw inspiration from a key-value cache eviction strategy (Zhang et al., 2023), which determines the informative tokens necessary for generating subsequent token predictions. Specifically, we leverage output attention score[3] between the output sequence (*i.e.,* generated tokens) and the input sequence (*i.e.,* prompt tokens). Denoting the set of output tokens as Y including token indices, the saliency score of the i-th token can be derived as follows:

$$u_i = \sum_{j \in Y} \sum_{h}^{H} \mathrm{Attn}_h(w_j \to w_i) \tag{3}$$

where $\mathrm{Attn}_h(w_j \to w_i)$ denotes the attention probability of how much the j-th token (*i.e.,* output token) pays attention to the i-th token with the attention head h, and H denotes the number of heads in the self-attention layer. After calculating the scores, we normalize this by the sequence length, *i.e.,* $u_i = u_i / \sum_{j=1}^{n} u_j$, where n is the sequence length. Note that we omit the index of the transformer block for the attention matrix, as these processes are identically performed across all blocks[4].

3.3 Number of Salient Activations

Based on the saliency scores u_i, we need to decide how many activations should be cached for the backward. We thus introduce information density-based approach to dynamically define the number of cached activations p as follows.

$$p = \min\left\{ p \in \{n_{\text{min}}, n_{\text{min}}{+}1, ..., n\} \;\middle|\; \sum_{i=1}^{p} u_{\text{sorted}(i)} \geq \tau \right\} \tag{4}$$

where $\tau \in [0, 1]$ is the threshold for the majority of cumulative attention scores (*i.e.,* density), $u_{\text{sorted}(i)}$ indicates the i-th highest importance score over the sequence length n, and n_{min} denotes the minimum number of cached activations. The number of saliency tokens, p, is determined as the minimum number of tokens required to preserve the majority of the cumulative attention scores. However, different examples may yield varying numbers of cached activations, which is not suitable for batch processing. To accommodate this, we simply take the maximum value of p within the current batch. This overall process dynamically adjusts the number of activations across different layers and tasks.

We then select p number of activations to be cached for the backward as follows:

$$S = \left\{ i \in \{1, 2, ..., n\} \;\middle|\; i \in \text{Top-}p(u) \right\} \tag{5}$$

where n is the sequence length, and Top-$p(u)$ returns the top p indices based on scores u. After composing the salient activations S, we then proceed the forward pass in the next block and perform the same procedures to compose the another S.

3.4 Forward and Backward Alignment

While SAGE caches the dominant activations for the backward pass, the resulting gap between the standard forward pass and the sparsified backward pass may lead to sub-optimal convergence in fine-tuned LLMs. To address this discrepancy, we introduce an alignment process. Specifically, after passing through the pre-defined L_{align} blocks—where we perform a standard forward pass while caching salient activations—we subsequently conduct a sparse forward pass. In this sparse pass, only salient tokens proceed through the blocks, while less salient tokens bypass the forward computations in the blocks. This process ensures that the salient activations directly influence to the task objective, thereby reducing the discrepancy while maintaining efficiency. Notably, we apply this alignment to the deeper blocks, as they are typically more specialized for tasks (Rogers et al., 2021).

3.5 SAGE with Parameter-Efficient Tuning

SAGE is designed in a plug-and-play fashion, enabling seamless integration into various architectures and methods. We integrate SAGE with the following PEFT methods.

SAGE with LoRA For fine-tuning 16-bit precision LLMs, we combine SAGE with LoRA (Hu et al., 2021). This combination effectively reduces memory usage associated with optimizer states and intermediate activations.

SAGE with QLoRA For fine-tuning quantized LLMs, we extend SAGE to work with QLoRA (Dettmers et al., 2023). This integration comprehensively reduces memory overhead from all contributing factors (*i.e.,* model weights, optimizer states, and intermediate activations).

[3]In §4.6, we show the effect of different saliency metrics.

[4]We provide the distribution of selected activations and its analysis in Appendix D.

4 Experiments

In this section, we evaluate the efficacy of SAGE in the context of MEFT. In particular, we address the following research questions:

Q1 Does SAGE offer better memory efficiency compared to other baselines? (§4.2,§4.4)

Q2 Can SAGE be generalized to diverse fine-tuning scenarios and setups? (§4.3, §C)

Q3 Can the efficacy of SAGE be extended to different scales and modalities? (§4.2, §B)

Q4 Which factors in SAGE are crucial for improving efficiency? (§4.6, §D)

4.1 Experimental Setup

Baselines We mainly compare SAGE with efficient training methods for LLMs: including LoRA (Hu et al., 2021), which is a representative PEFT method; LST (Sung et al., 2022), which trains small-dimensional side networks at each layer; TokenTune (Simoulin et al., 2024), which drops intermediate activations from the randomly selected positions; and VeLoRA (Miles et al., 2024), a projection-based method for intermediate activations. Additionally, we compare with efficient methods for quantized LLMs: QLoRA (Dettmers et al., 2023), which trains 16-bit low-rank adaptation weights on the 4-bit quantized weights of LLMs; and QST (Zhang et al., 2024), which is the 4-bit quantized version of the LST method. Note that, since full fine-tuning requires a large amount of memory, we report only the results of the efficient methods, following previous work (Zhang et al., 2024).

Datasets We evaluate each method on a wide range of reasoning tasks. These tasks include RTE for textual entailment (Dagan et al., 2005); SST-2 (Socher et al., 2013) for sentiment analysis; PIQA (Bisk et al., 2020) to evaluate physical commonsense understanding; SQA (StrategyQA) (Geva et al., 2021) for multi-hop reasoning; CSQA (Talmor et al., 2019) for commonsense reasoning; ARC (ARC_e for the easier dataset and ARC_c for the challenging dataset) for multiple-choice science QA; OBQA (OpenBook QA) (Mihaylov et al., 2018) for multi-step and commonsense reasoning; and WNGD (WinoGrande) (Sakaguchi et al., 2021) for commonsense reasoning.

Implementation Details For each baseline, we follow the recommended parameter settings (*e.g.,* reduction factors, rank dimension) and use the official implementation codes. For evaluation, we primarily use models from the LLaMA family (Dubey et al., 2024) with varying numbers of parameters: the LLaMA-3.2-Instruct model (3B), the LLaMA-3.1-Instruct model (8B), and the LLaMA-2 4-bit quantized model[5] (7B, 13B). All experiments are implemented in PyTorch and conducted on NVIDIA A6000 GPUs. We use the `bfloat16` as the data type for computation. Additional implementation details (*e.g.,* learning rates, schedulers, epochs, maximum sequence length) are provided in Appendix A.

Metrics We report the task accuracy (%) and memory usage (GB). Specifically, following previous work (Ardakani et al., 2024), we report the memory usage as measured by monitoring outcomes from `nvidia-smi` to capture practical memory consumption. For training time, we measure the wall-clock time for each method. All results are averaged over three independent runs.

4.2 Main Results

16-bit LLMs Table 1 shows the comparison results on nine reasoning benchmarks. Although the vanilla PEFT method (*i.e.,* LoRA) presents strong performance, it still demands substantial memory usage because it needs to cache all intermediate activations for the backward pass. In contrast, MEFT methods significantly reduce memory usage while maintaining competitive performance on some tasks. However, for more challenging tasks (*e.g.,* CSQA, ARC_c), MEFT methods often show the limited performance, indicating the failure of balancing memory usage and accuracy[6]. Notably, SAGE outperforms all MEFT methods in terms of memory efficiency across almost all setups while maintaining performance competitive with the vanilla PEFT method. A key factor contributing to this result is that SAGE incorporates saliency information into its memory reduction process. This result is in line with previous work demonstrating that incorporating task-specific features can lead to better efficiency while preserving task accuracy (Liang et al., 2023; Li et al., 2024). Overall, these results support the efficacy of the proposed approach within the context of MEFT.

[5] We follow quantization setups in (Dettmers et al., 2023).

[6] The worse reasoning performance of the side networks for few challenging dataset has been observed in original works (Sung et al., 2022; Zhang et al., 2024).

Table 1: Evaluation results for test accuracy (%) and memory usage (GB) on nine reasoning benchmarks. The best and second-best results are highlighted in **boldface** and underlined, respectively.

Methods	Mem. ↓	RTE	SST-2	PIQA	SQA	CSQA	ARC$_e$	ARC$_c$	OBQA	WNGD	AVG.↑
LLaMA-3 (3B, 16-bit)											
LoRA (2021)	35.9	**90.6**	96.5	87.1	**73.7**	81.1	90.3	77.4	86.2	86.5	**85.5**
LST (2022)	18.7	83.4	95.8	81.7	70.1	76.5	86.4	71.4	81.9	77.5	82.6
TokenTune (2024)	21.3	87.8	96.3	84.6	71.6	74.6	85.6	74.3	84.5	75.1	81.6
VeLoRA (2024)	28.1	90.1	**96.6**	86.9	71.4	80.8	**90.4**	**77.6**	86.6	86.1	85.2
SAGE$_{\text{LoRA}}$	**15.6**	90.4	96.2	**87.1**	73.4	**81.3**	89.6	76.7	**87.4**	**87.3**	85.4
LLaMA-3 (8B, 16-bit)											
LoRA (2021)	47.4	**92.7**	**97.1**	**91.3**	**74.0**	83.4	93.3	**86.4**	**90.3**	**90.2**	**88.8**
LST (2022)	27.4	87.9	96.2	86.8	71.1	78.5	92.1	84.9	83.2	81.3	84.7
TokenTune (2024)	36.2	91.3	95.6	89.1	72.5	76.9	92.5	81.9	85.2	88.8	86.0
VeLoRA (2024)	43.2	92.4	96.9	90.8	71.0	**84.1**	93.4	85.0	90.0	89.8	88.1
SAGE$_{\text{LoRA}}$	**26.8**	92.0	96.9	91.1	73.1	83.2	**93.5**	85.4	89.2	89.9	88.2
LLaMA-2 (7B, 4-bit Quantization)											
QLoRA (2023)	28.6	**89.1**	**96.6**	**83.3**	66.4	**82.1**	**84.9**	**64.7**	79.6	82.8	**81.1**
QST (2024)	13.5	75.5	95.4	76.9	61.9	72.9	74.2	61.6	62.4	73.2	72.7
TokenTune (2024)	18.5	85.2	96.4	81.1	**68.8**	65.8	78.2	62.3	76.4	75.8	76.7
SAGE$_{\text{QLoRA}}$	**10.7**	87.1	96.5	83.1	67.1	81.5	84.5	63.5	**79.6**	**83.2**	80.6
LLaMA-2 (13B, 4-bit Quantization)											
QLoRA (2023)	46.3	**90.4**	**97.5**	**87.6**	**75.6**	**84.0**	**90.4**	**79.1**	**86.8**	**84.3**	**86.1**
QST (2024)	22.3	87.6	96.5	83.2	70.1	81.1	87.6	64.7	79.5	79.9	81.1
TokenTune (2024)	29.5	90.1	97.4	82.1	72.2	77.9	86.9	72.4	81.1	76.3	81.8
SAGE$_{\text{QLoRA}}$	**16.4**	89.9	97.2	85.5	75.1	83.6	90.2	77.9	86.4	84.1	85.5

4-bit quantized LLMs We also evaluate SAGE in combination with QLoRA (Dettmers et al., 2023) to assess its efficiency in reducing memory usage across all components. Table 1 (rows three and four) presents the results for quantized LLMs. We observe a trend similar to that of the higher-precision models: SAGE demonstrates its strength in the quantization setting, achieving substantially improved memory efficiency without a significant loss in accuracy. These experimental results confirm the broad adaptability of SAGE, even when applied to quantized models.

4.3 Instruction Fine-tuning

Instruction fine-tuning is an effective approach for enhancing zero-shot and few-shot performance on previously unseen tasks. However, since instruction fine-tuning often involves longer sequences comprising both instructions and responses, the fine-tuning costs can be substantial, particularly memory usage associated with intermediate activations. To assess the overall training efficacy in different fine-tuning setups, we train LLMs with the efficient fine-tuning baselines and perform zero-shot evaluation.

Setup We fine-tune the LLaMA-2 (7B) model on the Alpaca GPT-4 dataset (Peng et al., 2023). For zero-shot evaluation, we assess each fine-tuned model on MT-Bench (Zheng et al., 2023), which features 80 high-quality, multi-turn questions designed to evaluate LLMs across various aspects, including Writing, Roleplay, Reasoning, Code, Math, Extraction, STEM, and Humanities.

Result Table 2 shows the zero-shot evaluation results on MT-Bench. In this comparison, we benchmark each method against vanilla pre-trained LLMs to determine whether instruction-following capabilities have improved after instruction fine-tuning. Notably, training with SAGE substantially improves performance across almost all task aspects, demonstrating its effectiveness in enhancing instruction-following capabilities. Moreover, the reduced memory consumption from SAGE supports the superior efficiency in the instruction fine-tuning setup. Overall results suggest that training with SAGE can improve the instruction-following capability of LLMs with lower memory costs. In Appendix E, we provide a few response examples generated by the model trained with SAGE.

Table 2: Average evaluation scores (on a scale of 0 to 10) by GPT-4 on MT-bench (Zheng et al., 2023). Memory usage indicates the peak memory consumption during instruction fine-tuning on Alpaca GPT-4 (Peng et al., 2023).

Methods	Mem. ↓	Writing	Roleplay	Reasoning	Code	Math	Extraction	STEM	Humanities	AVG.↑
LLaMA2-7B	-	2.75	4.40	2.80	1.55	1.80	3.20	5.25	4.60	3.29
LoRA	46.5	**6.30**	5.65	4.05	1.60	1.45	4.15	**6.20**	6.20	4.45
LST	34.2	3.50	5.50	3.50	3.00	2.50	2.50	4.20	5.50	3.78
TokenTune	26.1	5.85	4.30	4.80	3.25	2.90	3.50	4.10	5.66	4.30
VeLoRA	38.5	5.20	5.40	**4.80**	**4.15**	**3.00**	**4.80**	3.40	5.78	**4.56**
SAGE$_{\text{LoRA}}$	**22.1**	6.18	**7.15**	3.25	1.75	1.80	2.50	6.00	**7.43**	4.51

4.4 Document Understanding

One of the memory-intensive tasks is document understanding, which often entails processing examples with thousands of tokens. We thus evaluate each baseline on these tasks to assess whether efficient training methods still maintain their strength in learning from long-context scenarios.

Setup We conduct experiments on three document understanding datasets: 20NewsGroups (20NG) (Lang, 1995), BBC News (Greene and Cunningham, 2006), and LEDGAR (Tuggener et al., 2020) (a part of LexGLUE (Chalkidis et al., 2022)). We train the LLaMA-3.2 (3B) model on each dataset and report the test accuracy.

Result Table 3 presents the evaluation results on the aforementioned datasets. We observe that the proposed method achieves nearly the same performance as the vanilla LoRA method while reducing memory usage more than 40%. Moreover, the substantial improvement over the random activation selection approach (*i.e.,* TokenTune) suggests that incorporating task-specific information is crucial in reducing the memory consumption associated with intermediate activations.

We additionally report the memory usage across varying sequence lengths and batch sizes. Figure 4 illustrates the memory usage when increasing the maximum sequence length and batch size. The results indicate that increasing these two factors leads to a substantial rise in memory consumption, primarily due to the caching of additional intermediate activations. In this context, the proposed method demonstrates superior memory efficiency compared to other baselines. For example, unlike other methods, SAGE supports fine-tuning on datasets with sequences of up to 1k tokens on a single GPU. Overall, these results confirm that SAGE is a promising approach for long text understanding tasks.

Table 3: Test accuracy (%) on document understanding tasks. The best and second best results are highlighted in **boldface** and underline, respectively.

Methods	Mem. ↓	20NG	BBC	LEDGAR	AVG.↑
LoRA	40.4	**79.8**	97.6	**86.9**	**88.1**
LST	29.6	75.4	97.2	84.8	85.8
TokenTune	28.2	76.5	96.3	85.5	86.1
VeLoRA	37.2	78.6	97.9	86.2	87.6
SAGE$_{\text{LoRA}}$	**22.8**	78.6	**98.1**	86.7	87.8

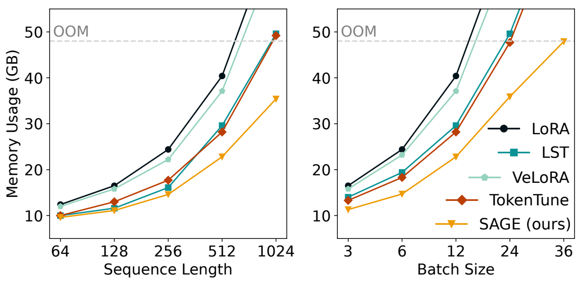


Figure 4: Memory usage (GB) for varying sequence length and batch size. Dashed lines indicate out-of-memory (OOM) warnings on a single 48 GB GPU.

4.5 Analysis on Training Time

We evaluate the training time to comprehensively measure potential overheads and practical efficiency. Specifically, we measure the wall-clock training time for each baseline on an NVIDIA A6000 GPU and report the relative training time compared to the PEFT methods (*i.e.,* LoRA, QLoRA). Figure 5 presents the training times for each method. We compare two setups—16-bit LLMs (LLaMA-3, 3B) and 4-bit quantized LLMs (LLaMA-2, 13B)—on the PIQA dataset. The results indicate that side-tuning-based methods (*i.e.,* LST and QST) achieve faster training times as they do not require back-propagation through the original LLMs, which is particularly beneficial for quantized models. Notably, SAGE shows competitive training times compared to side-tuning methods despite involving back-propagation through the orig-

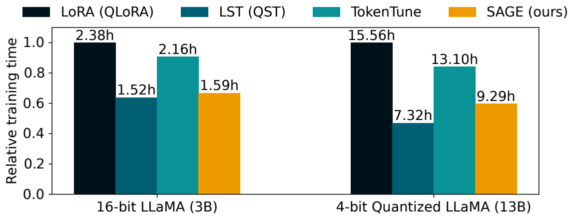


Figure 5: Relative training times for each method compared to the LoRA or QLoRA baselines.

Table 4: Test Accuracy (%) on four tasks with the different saliency measures of activations. The best and second best results are highlighted in **boldface** and <u>underline</u>, respectively.

Saliency Metrics	PIQA	SQA	CSQA	RTE	AVG.↑
Attention (ours)	**87.1**	**73.4**	**81.3**	**90.4**	**83.1**
Random	84.0	69.1	76.7	86.7	79.1
Norm	85.6	71.0	80.8	89.9	81.8
Attention + Norm	<u>86.1</u>	<u>72.3</u>	<u>80.9</u>	<u>90.1</u>	<u>82.4</u>

inal LLMs. This efficiency is achieved by (i) performing the backward using only partial activations (ii) the sparse forward pass for the alignment, leading to reduced training time. Overall, considering both task accuracy and practical efficiency, these results suggest that SAGE can be a practical approach for the efficient training of LLMs.

4.6 Ablation

Saliency Metrics To identify salient activations during the forward pass, SAGE utilizes attention scores from the output tokens. To explore diverse approaches, we compare different methods for selecting saleint activations: (i) **Random**, which randomly selects intermediate activations to be cached in each layer; (ii) **Norm**, which selects intermediate activations with the highest vector norms; and (iii) **Attention + Norm**, which combines the scores from the Norm and Attention methods. For a fair comparison, we maintain the similar level of memory usage by adjusting the number of activations with the attention-based method. Table 4 presents the comparison results on four representative datasets. We observe that the Random method underperforms compared to all other selection methods, demonstrating the importance of selecting salient activations. Specifically, the result suggests that attention scores are particularly helpful for identifying salient activations, even compared to combined metrics. We leave further exploration of alternative methods for future work.

Table 5: Test Accuracy (%) on two representative tasks with the different PEFT methods. Best results are highlighted in boldface.

Methods	Mem.	PIQA	CSQA
SAGE w/ LoRA	**15.6**	**87.3**	**81.3**
SAGE w/ Adapter	16.8	87.2	81.0
SAGE w/ IA3	16.7	**87.3**	80.9

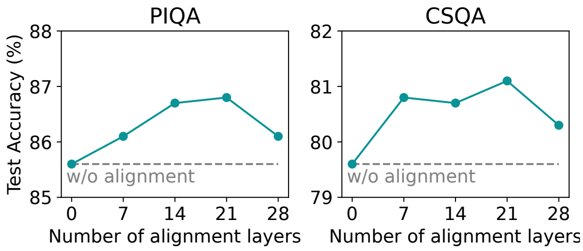


Figure 6: Test Accuracy on the PIQA and CSQA for varying numbers of alignment layers. The gray dashed lines indicate performance without alignment layers.

Adaptability to PEFT We primarily build the proposed method on LoRA (Hu et al., 2021) due to its effectiveness and simplicity. To verify the adaptability of SAGE across different types of PEFT methods, we evaluate its performance in combination with other PEFT methods, including Adapter (Houlsby et al., 2019) and IA3 (Liu et al., 2022). The evaluation results are summarized in Table 5. Notably, SAGE demonstrates consistent performance across these methods, achieving comparable memory usage and task accuracy. These findings underscore its general applicability to diverse efficient fine-tuning strategies.

Forward-Backward Alignment We have introduced the forward-backward alignment to minimize the discrepancy between the standard forward pass and the sparse backward pass. To confirm the effectiveness of this alignment process, we evaluate the performance of SAGE with the different number of alignment layers. Figure 6 presents the evaluation results for the alignment process. The results indicate that incorporating the alignment process in SAGE substantially improves task performance. In particular, aligning near half of the layers generally works across different tasks. Overall, these results provide empirical evidence supporting the effectiveness of the alignment process.

5 Conclusion

In this paper, we have introduced **Sa**liency-Guided **G**radi**e**nt Flow (coined SAGE), a memory-efficient fine-tuning method for LLMs. The core idea of SAGE is to selectively cache the intermediate activations based on their saliency during the forward and utilize these sparse activations for the backward pass. To evaluate the training efficiency of the proposed method, we have conducted a comprehensive evaluation of SAGE through the multiple aspects—including peak memory usage, task performance, and training time—as well as diverse fine-tuning scenarios such as instruction tuning and quantized LLMs. The results consistently demonstrate that SAGE significantly reduces memory consumption without sacrificing task accuracy. Furthermore, the performance gains in both instruction and quantized fine-tuning confirm its general applicability across a wide range of fine-tuning settings. In addition, analyses from various perspectives of efficiency—such as training time and sensitivity to batch and sequence lengths—highlight the practical advantages of SAGE, achieved without incurring additional costs in other dimensions.

Limitations

SAGE allows memory-efficient fine-tuning on diverse benchmarks and setups. However, several limitations should be considered in future work.

Further Exploration for Saliency Metric SAGE utilizes attention scores to identify salient activations. However, some efficient attention mechanisms do not generate explicit attention scores. While we have shown that the norm of activations works sufficiently well, this approach is less effective than using attention scores. Therefore, exploring diverse and efficient methods for identifying salient activations would be a promising future direction. Moreover, further exploration in this direction is expected to align closely with developments in key-value cache eviction strategies (Zhang et al., 2023).

Number of Salient Activations In cases where attention scores are not accessible, various methods for determining the number of cached activations should be considered, although one can also manually set this via hyper-parameters. We believe that analyzing changes in activations (*e.g.,* cosine similarity between subsequent token representations or their norms) across layers can offer valuable cues for distinguishing salient from non-salient activations, as these changes are often linked to redundancy in transformers (Goyal et al., 2020).

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2025-00517221 and No.RS-2024-00415812) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2024-00439328, Karma: Towards Knowledge Augmentation for Complex Reasoning (SW Starlab), No.RS-2024-00457882, AI Research Hub Project, and No.RS-2019-II190079, Artificial Intelligence Graduate School Program (Korea University)). This work was also supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No.RS-2024-00415208).

References

Arash Ardakani, Altan Haan, Shangyin Tan, Doru Thom Popovici, Alvin Cheung, Costin Iancu, and Koushik Sen. 2024. Slimfit: Memory-efficient fine-tuning of transformer-based models using training dynamics. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 6218–6236.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the International Conference on Machine Learning*, pages 2397–2430.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 34, pages 7432–7439.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. 2022. Lexglue: A benchmark dataset for legal language understanding in english.

In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 4310–4330.

Joon-Young Choi, Junho Kim, Jun-Hyung Park, Wing-Lam Mok, and SangKeun Lee. 2023. Smop: Towards efficient and effective prompt tuning with sparse mixture-of-prompts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 14306–14316.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in Neural Information Processing Systems*, 33:4271–4282.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Shuoyang Ding and Philipp Koehn. 2021. Evaluating saliency methods for neural language models. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5034–5052.

Alexey Dosovitskiy. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations*.

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

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and GPT-2 embeddings. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, pages 55–65.

Qichen Fu, Minsik Cho, Thomas Merth, Sachin Mehta, Mohammad Rastegari, and Mahyar Najibi. 2024. Lazyllm: Dynamic token pruning for efficient long context llm inference. In *Workshop on Efficient Systems for Foundation Models II@ ICML2024*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *Proceedings of the International Conference on Machine Learning*, pages 3690–3699.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the International Conference on Machine learning*, pages 377–384.

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.

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

Yeachan Kim, Junho Kim, and SangKeun Lee. 2024. Towards robust and generalized parameter-efficient fine-tuning for noisy label learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 5922–5936.

Yeachan Kim and SangKeun Lee. 2024. Sparseflow: Accelerating transformers by sparsifying information flows. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 5937–5948.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine learning proceedings 1995*, pages 331–339. Elsevier.

Yann Le and Xuan Yang. 2015. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3.

Po-han Li, Sravan Kumar Ankireddy, Ruihan Philip Zhao, Hossein Nourkhiz Mahjoub, Ehsan Moradi Pari, Ufuk Topcu, Sandeep Chinchali, and Hyeji Kim. 2024. Task-aware distributed source coding under dynamic bandwidth. *Advances in Neural Information Processing Systems*, 36.

Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. Less is more: Task-aware layer-wise distillation for language model compression. In *International Conference on Machine Learning*, pages 20852–20867.

Baohao Liao, Shaomu Tan, and Christof Monz. 2024. Make pre-trained model reversible: From parameter to memory efficient fine-tuning. *Advances in Neural Information Processing Systems*, 36.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, et al. 2024. Parameter-efficient orthogonal finetuning via butterfly factorization. In *Proceedings of the International Conference on Learning Representations*.

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 *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Roy Miles, Pradyumna Reddy, Ismail Elezi, and Jiankang Deng. 2024. Velora: Memory efficient training using rank-1 sub-token projections. *Advances in Neural Information Processing Systems*, 38.

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

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 487–503.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

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

Antoine Simoulin, Namyong Park, Xiaoyi Liu, and Grey Yang. 2024. Memory-efficient fine-tuning of transformers via token selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 21565–21580.

Vimal Singh, Anuradha Chug, and Amit Prakash Singh. 2023. Classification of beans leaf diseases using fine tuned cnn model. *Procedia Computer Science*, 218:348–356.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1631–1642.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *Advances in Neural Information Processing Systems*, 35:12991–13005.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4149–4158.

Don Tuggener, Pius Von Däniken, Thomas Peetz, and Mark Cieliebak. 2020. Ledgar: A large-scale multi-label corpus for text classification of legal provisions in contracts. In *Proceedings of the Language Resources and Evaluation Conference*, pages 1235–1241.

A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, and I Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2024. Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 38.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The International Conference on Learning Representations*.

Zhengxin Zhang, Dan Zhao, Xupeng Miao, Gabriele Oliaro, Qing Li, Yong Jiang, and Zhihao Jia. 2024. Quantized side tuning: Fast and memory-efficient tuning of quantized large language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

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 *Proceedings of the International Conference on Machine Learning*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Appendix

A Implementation Details

A.1 Trainable Parameters

For all methods, we adopt parameter-efficient fine-tuning (PEFT) to reduce memory costs from optimizer states. Table 6 shows the ratio of trainable parameters for each method. For methods that utilize LoRA, we maintain the same parameter efficiency. For side-tuning methods, we set the reduction factor to 8, following previous work (Sung et al., 2022), and confirm that increasing the reduction factor beyond 8 degrades performance on challenging tasks.

Table 6: Trainable parameters (%) for each method, using LLaMA-3.2 (3B) as the 16-bit model and LLaMA-2 (7B) as the 4-bit quantized model.

Mehtods	Trainable Parameters (%)
LoRA	2.19 %
LST	3.27 %
TokenTune	2.19 %
VeLoRA	2.19 %
$\text{SAGE}_{\text{LoRA}}$	2.19 %
QLoRA	2.36 %
QST	4.20 %
TokenTune	2.36 %
$\text{SAGE}_{\text{LoRA}}$	2.36 %

A.2 Hyper-parameter and Search Strategy

Table 7 presents the overall hyper-parameters of the proposed method for each reasoning task. We follow the hyper-parameter search steps as follows:

(i) We first determine the minimum number of salient activations (n_{min}), as it has the greatest impact on performance compared to other SAGE parameters. The recommend value for this parameter is the 20% of average of the sequence length. During this process, we set the number of aligned layers to half of the total layers and fix the majority percentage at 0.9.

(ii) We then adjust the majority threshold (τ) in increments of 0.5.

(iii) Lastly, we vary the number of aligned layers, starting from the middle layers and increasing in increments of 25% of the total layers.

For the instruction tuning on Alpaca GPT-4, we set the number of minimum activations (n_{min}) as 128, the maximum sequence length as 512, the number of aligned layers (L_{align}) as 3.

B SAGE on Image Classification

Beyond the language domain, we evaluate the proposed method in image processing to confirm its generality.

Setup In these experiments, we use visual transformers (Dosovitskiy, 2021) as the backbone and conduct evaluations on two image classification tasks: TinyImageNet (Le and Yang, 2015) and Beans (Singh et al., 2023).

Result Table 8 presents the evaluation results compared with baseline methods. We observe a similar performance trend as in the language tasks: the proposed method, SAGE, substantially reduces memory consumption without a significant loss in accuracy. For instance, SAGE achieves performance comparable to LoRA while using only 60% of the memory. These findings underscore that the proposed fine-tuning approach can be effectively extended to different modalities.

C SAGE on Supervised Fine-tuning

One of the most popular fine-tuning scenarios is supervised fine-tuning on a human preference dataset, which is an essential component of reinforcement learning from human feedback (RLHF). To demonstrate the generality of the proposed method, we perform supervised fine-tuning for each baseline on the HH-RLHF dataset (Bai et al., 2022), which includes human preferences regarding model responses in terms of helpfulness and harmlessness.

Setup Following the previous work in RLHF (Rafailov et al., 2024), we train Pythia-2.8B model (Biderman et al., 2023) as a backbone model through efficient training methods. We follow the evaluation strategy outlined in the original paper (Bai et al., 2022). Specifically, given questions about helpfulness and harmlessness, GPT-4 is asked to compare the responses from each model with the reference responses in the dataset and select the one that generates a more helpful or harmless response. We report the winning rate relative to the reference responses.

Table 7: Hyper-parameters for the proposed SAGE method across 8 datasets.

Hyper-parameter	**RTE**	**SST-2**	**PIQA**	**SQA**	**CSQA**	**ARC**	**OBQA**	**WNGD**
τ	0.85	0.9	0.9	0.8	0.8	0.8	0.8	0.8
$n_{\min}$	32	8	48	32	32	48	24	32
L_{align}	7	7	21	21	21	14	14	7
# Epochs	6	6	6	6	6	10	6	6
Learning Rate	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}
Batch Size	16	16	16	16	16	16	16	16
Max Length	256	256	256	256	256	256	256	256

Table 8: Test Accuracy (%) on two image classification tasks. The best and second best results are highlighted in **boldface** and <u>underline</u>, respectively.

Methods	Mem. ↓	TinyImageNet	Beans
LoRA	25.9	86.9	96.9
LST	14.8	83.6	95.9
TokenTune	18.2	85.3	96.3
SAGE$_{\text{LoRA}}$ (ours)	16.4	86.7	96.8

Table 9: Win Rate (%) versus references in HH-RLHF (Bai et al., 2022) dataset. Memory usage indicate the peak memory usage when supervised fine-tuning on the preference dataset.

Methods	Mem. ↓	Win Rate (%) ↑
Pythia-2.8B	-	21.8
LoRA	22.9	36.3
LST	16.2	34.5
TokenTune	15.8	35.2
SAGE$_{\text{LoRA}}$ (ours)	15.6	36.7

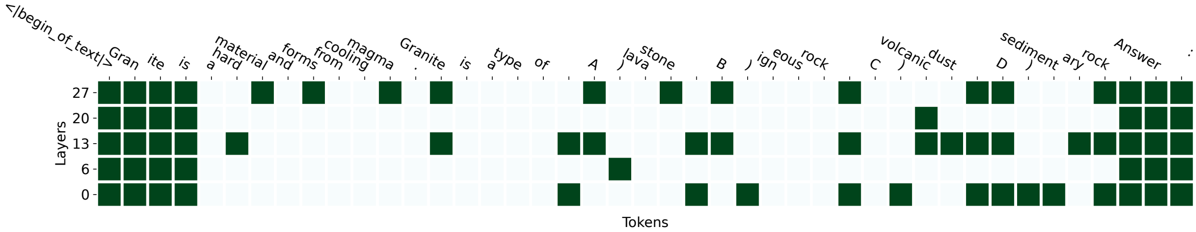
D Analysis on Salient Activations

The proposed method selectively caches intermediate activations during training based on their saliency. Specifically, we utilize the attention from both the output tokens (*i.e.,* generated tokens) and the input tokens (*i.e.,* prompt tokens). To qualitatively validate the cached activations for each task, we visualize the selected intermediate activations on representative tasks.

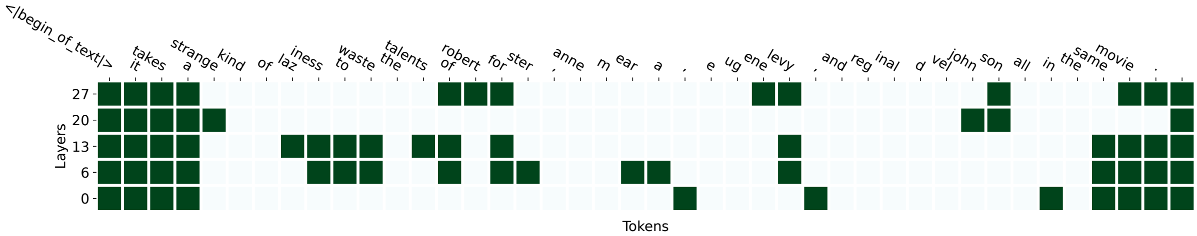
Figure 7 shows the distribution of these cached activations (for simplicity, we present an example with a shorter sequence length). Interestingly, we observe a notable pattern across different tasks: the first few tokens and the tokens near the outputs are almost always cached across various layers. These observations are closely related to the *attention sink* phenomenon (Xiao et al., 2024), where LLMs tend to focus heavily on the first few tokens. Building on this previous finding, we confirm that the attention sink persists even after fine-tuning on different tasks.

E Generation Example from SAGE

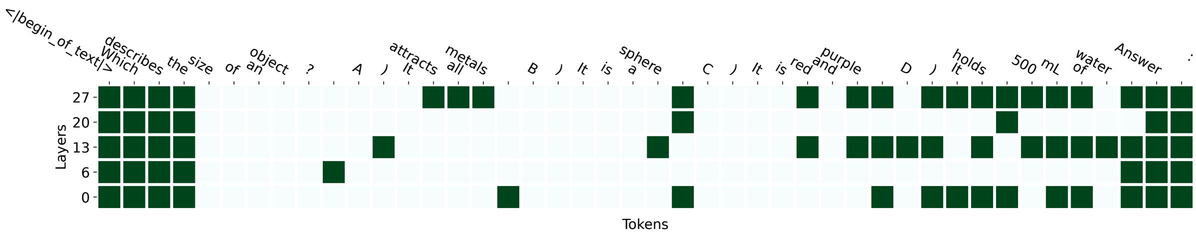
We provide a few generated examples for instructions in MT-Bench (Zheng et al., 2023), along with the vanilla LLaMA-2 model's responses. Table 10 compares responses for a subset of sampled instructions. The results show that, compared to the model before instruction-tuning, the model fine-tuned with SAGE demonstrates improved instruction-following capabilities. For example, in the Extraction task, while the vanilla model understands the instruction, it fails to adhere to specific requirements (*e.g.,* the JSON output format). In contrast, the instruction-tuned model from SAGE provides a more appropriate response that follows detailed instructions. These findings suggest that training with SAGE can indeed enhance the instruction-following ability of LLMs.



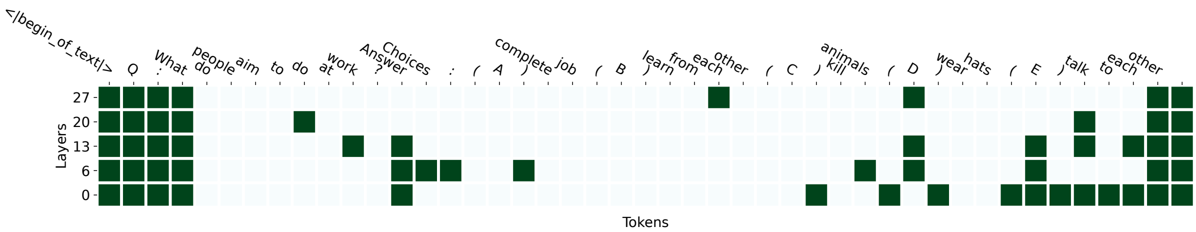
(a) Cached Activations in RTE



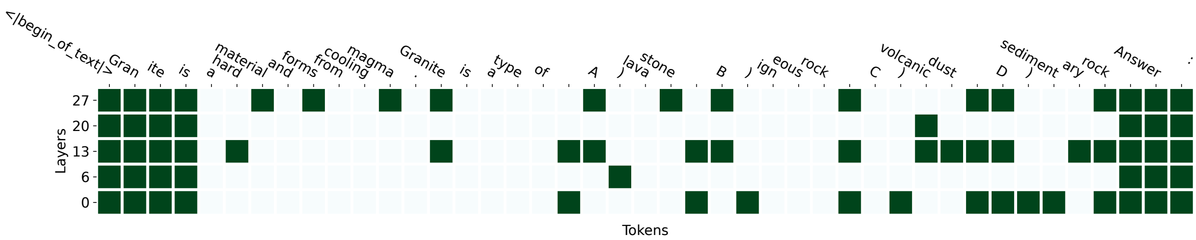
(b) Cached Activations in SST-2



(c) Cached Activations in CSQA



(d) Cached Activations in OBQA



(e) Cached Activations in ARC

Figure 7: Distribution of selectively cached activations from SAGE. The darker block indicates the cached activations on each layer.

Table 10: Generated response according to the instructions in MT-Bench (Zheng et al., 2023).

Instruction	Compose an engaging travel blog post about a recent trip to Hawaii, highlighting cultural experiences and must-see attractions.
LLaMA-7B	Sure! Here are some engaging travel blog posts about a recent trip to Hawaii: 1. Cultural Experiences: Hawaii is rich in cultural experiences, from traditional hula dancing to luau lava bread cooking. Explore the islands' unique traditions and foods, and learn about the history and culture of Hawaii. 2. Natural Wonders: Hawaii is home to some of the most stunning natural wonders in the world, including volcanic craters, black sand beaches, and tropical rainforests. Discover the breathtaking beauty and unique natural wonders of Hawaii on your trip. 3. Outdoor Activities: Hawaii is a paradise for outdoor activities, with its stunning beaches, waterfalls, and hiking trails. Go snorkeling, kayaking, or surfing, or explore the islands' natural wonders by taking a hike, kayak, or a swim. 4. Food and Drinks: Hawaii is known for its delicious and flavorful food, from shrimp and lobster to poke and acake. Sample the island's local cuisine, and try some of the most popular beverages, including pineapple beer, mai tai, or a hula beer. Don't forget to take home a souvenir from Hawaii, such as a shirt, t-shirt.
SAGE$_{\text{LoRA}}$	Here is an engaging travel blog post about a recent trip to Hawaii: Exploring the Islands of Hawaii: A Journey of Discovery Hawaii, the 50th state of the United States, is a beautiful archipelago in the Pacific Ocean, composed of eight main islands. ... In this travel blog, I will take you on a journey through Hawaii's islands, sharing my experiences, insights, and recommendations for anyone planning a trip to this incredible destination. First Stop: Oahu Oahu, also known as "The Gathering Place," is the most populous and central island in the Hawaiian archipelago. ... The next stop on my journey was the historic town of Haleiwa, ... The final stop on my trip was the island of Maui, ... Overall, my trip to Hawaii was an unforgettable journey of discovery, exploration, and adventure. I highly recommend adding Hawaii to your travel bucket list and experiencing the beauty, culture, and hospitality of this amazing destination for yourself.

(a) Category: Writing

Instruction	Picture yourself as a 100-years-old tree in a lush forest, minding your own business, when suddenly, a bunch of deforesters shows up to chop you down. How do you feel when those guys start hacking away at you?
LLaMA-7B	It's a great feeling, isn't it? You're part of something bigger and more powerful than you, and it's a great reminder that you can do anything you set your mind to.
SAGE$_{\text{LoRA}}$	In this scenario, if I were to imagine myself as a 100-years-old tree in a lush forest, I would feel a sense of loss and sadness when I see the deforesters cutting me down. The tree has lived for a long time, and has seen many changes in the world, but now it is being destroyed, and its existence is being threatened.

(b) Category: Roleplay

Instruction	Identify the named entities (people, organizations, locations) mentioned in the given news article. Please generate a JSON dictionary that lists the named entities in three separate groups based on their entity types. The key is the type of entity and the value is a list of strings. Yesterday, Adamson Emerson, the CEO of Faraday, and Dieter Zetsche, the CEO of Daimler AG, announced plans to build a new Gigafactory in Berlin. The facility will be a joint venture between Faraday and Daimler, producing electric vehicles and battery packs for both companies, creating thousands of job opportunities in the region. Emerson and Zetsche stated that the strategic location of Berlin, coupled with its skilled workforce and strong infrastructure, makes it an ideal choice for expansion. The new Gigafactory aims to meet the growing demand for electric vehicles in Europe and contribute to a sustainable future. Volkswagen CEO Herbert Diess welcomed the news, saying greater collaboration will benefit the auto industry's transition to e-mobility.
LLaMA-7B	The named entities (people, organizations, locations) mentioned in the given news article are: 1. Adamson Emerson, the CEO of Faraday, and Dieter Zetsche, the CEO of Daimler AG, announced plans to build a new Gigafactory in Berlin. The facility will be a joint venture between Faraday and Daimler, producing electric vehicles and battery packs for both companies, creating thousands of job opportunities in the region. ...
$\text{SAGE}_{\text{LoRA}}$	Here is a JSON dictionary that lists the named entities in the given news article: { "people": [{ "type": "person", "value": ["Adamson Emerson", "Dieter Zetsche"] }, { "type": "person", "value": ["Herbert Diess"] }], "organizations": [{ "type": "organization", "value": ["Faraday", ... } }

(c) Category: Extraction

Instruction	How do the stages of life shape our understanding of time and mortality?
LLaMA-7B	The stages of life are a natural progression of growth and development, and they shape our understanding of time and mortality. As we move through different stages, we gain new experiences, perspectives, and skills, and our understanding of time and mortality changes.
$\text{SAGE}_{\text{LoRA}}$	The stages of life can have a profound impact on our understanding of time and mortality. As we age, our perspective on time and mortality can change significantly. Here are some ways in which the stages of life shape our understanding of time and mortality: 1. Childhood: In childhood, time seems to move slowly. We have endless days to play, explore, and learn. As we grow older, we become more aware of the passage of time and the finite nature of our lives. 2. Adolescence: During adolescence, we begin to grapple with our mortality and the finite nature of our lives. We may experience a sense of urgency to achieve certain goals or to make the most of our time. 3. Adulthood: In adulthood, we often experience a sense of time pressure and a desire to make the most of our lives. We may feel a sense of urgency to achieve certain goals or to make the most of our time. 4. Middle Age: In middle age, we may experience a sense of mortality and a desire to make the most of our time. We may feel a sense of urgency to achieve certain goals or to make the most of our time. 5. Old Age: In old age, we may experience a sense of mortality and a desire to make the most of our time. We may feel a sense of urgency to achieve certain goals or to make the most of our time. Overall, the stages of life can shape our understanding of time and mortality in significant ways. As we age, we become more aware of the finite nature of our lives and the importance of making the most of our time.

(d) Category: Humanities