LISA: Layerwise Importance Sampling for Memory-Efficient Large Language Model Fine-Tuning

Rui Pan[♡][*], **Xiang Liu**[♣][*], **Shizhe Diao**[♦], **Renjie Pi**[♡], **Jipeng Zhang**[♡],
Chi Han[♠], **Tong Zhang**[♠]

[♡]The Hong Kong University of Science and Technology
[♣]The Hong Kong University of Science and Technology(Guangzhou)
[♦]NVIDIA [♠]University of Illinois Urbana-Champaign
{rpan, sdiaoaa, rpi, jzhanggr}@ust.hk xliu886@connect.hkust-gz.edu.cn
{chihan3, tozhang}@illinois.edu

Abstract

The machine learning community has witnessed impressive advancements since large language models (LLMs) first appeared. Yet, their massive memory consumption has become a significant roadblock to large-scale training. For instance, a 7B model typically requires at least 60 GB of GPU memory with full parameter training, which presents challenges for researchers without access to high-resource environments. Parameter efficient fine-tuning techniques such as Low-Rank Adaptation (LoRA) have been proposed to alleviate this problem. However, in most large-scale fine-tuning settings, their performance does not reach the level of full parameter training because they confine the parameter search to a low-rank subspace. Attempting to complement this deficiency, we investigate the layerwise properties of LoRA on fine-tuning tasks and observe an unexpected but consistent skewness of weight norms across different layers. Utilizing this key observation, a surprisingly simple training strategy is discovered, which outperforms both LoRA and full parameter training in a wide range of settings with memory costs as low as LoRA. We name it **L**ayerwise **I**mportance **S**ampled **A**damW (**LISA**), a promising alternative for LoRA, which applies the idea of importance sampling to different layers in LLMs and randomly freeze most middle layers during optimization. Experimental results show that with similar or less GPU memory consumption, LISA surpasses LoRA or even full parameter tuning in downstream fine-tuning tasks, where LISA consistently outperforms LoRA by over 10%-35% in terms of MT-Bench score while achieving on-par or better performance in MMLU, AGIEval and WinoGrande. On large models, specifically LLaMA-2-70B, LISA surpasses LoRA on MT-Bench, GSM8K, and PubMedQA, demonstrating its effectiveness across different domains.

1 Introduction

Large language models (LLMs) like ChatGPT excel in tasks such as writing documents, generating complex code, answering questions, and conducting human-like conversations [1]. With LLMs being increasingly applied in diverse task domains, domain-specific fine-tuning has emerged as a critical strategy to enhance their downstream capabilities [2, 3, 4, 5]. Nevertheless, these methods are typically time-intensive and consume substantial computational resources, posing significant challenges to the development of large-scale models [6]. For example, continual pre-training typically

---

[*]Equal Contribution.

requires several weeks even with multiple 80 GB GPUs. To reduce costs, Parameter-Efficient Fine-Tuning (PEFT) techniques have been proposed to minimize the number of trainable parameters.

These techniques include adapter weights [7], prompt weights [8], and LoRA [9]. Among these, LoRA stands out as one of the most widely adopted due to its unique ability to merge the adaptor back into the base model parameters, significantly enhancing efficiency. However, LoRA's superior performance in fine-tuning tasks has yet to reach a point that universally surpasses full parameter fine-tuning in all settings [10, 11]. In particular, it has been observed that LoRA tends to falter on large-scale datasets during continual pre-training [12], which raises doubts about the effectiveness of LoRA under those circumstances. We attribute this to LoRA's much fewer trainable parameters compared to the base model, which limits the representation power of LoRA training.
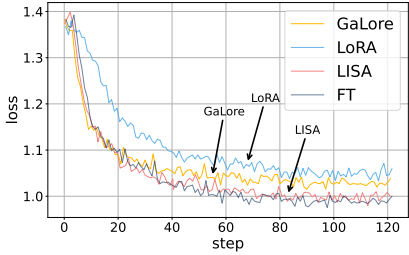

Figure 1: Training loss of LLaMA-2-7B on Alpaca GPT-4.

To overcome this shortcoming, we delve into LoRA's training statistics in each layer, aspiring to bridge the difference between LoRA and full-parameter fine-tuning. Surprisingly, we discover that LoRA's layerwise weight norms have an uncommonly skewed distribution, where the bottom layer and/or the top layer occupy the majority of weights during the update. In contrast, the other self-attention layers only account for a small amount, which means different layers have different importance when updating. This key observation inspires us to "sample" different layers by their importance, which matches the idea of importance sampling [13, 14].

As a natural consequence, this strategy brings forth our **L**ayerwise **I**mportance **S**ampled **A**dam (**LISA**) algorithm, where by selectively updating only essential LLM layers and leaving others untouched, LISA enables training large-scale language models ($\geq$ 65B parameters) with less or similar memory consumption as LoRA. Furthermore, fine-tuned on downstream tasks, LISA outperformed both LoRA and conventional full-parameter fine-tuning approaches by a large margin, indicating the large potential of LISA as a promising alternative to LoRA.

We summarize our key contributions as follows,

- We discover the phenomenon of skewed weight-norm distribution across layers in LoRA, which implies the varied importance of different layers in large-scale LLM training.

- We propose the Layerwise Importance Sampled AdamW (LISA), a simple optimization method capable of scaling up to over 70B LLMs with less or similar memory cost as LoRA.

- We demonstrate LISA's effectiveness in fine-tuning tasks for modern LLMs, where it outperforms LoRA by 10%-35% in MT-Bench and achieves better performance in multiple benchmarks. In addition, LISA exhibits much better convergence behaviors than LoRA. LISA even outperforms full parameters training under certain settings. Similar performance gain is observed across different sized models (7B-70B) and tasks, including instruction following, medical QA, and math problems.

2 Related Work

2.1 Large Language Models

In the realm of natural language processing (NLP), the Transformer architecture has been a revolutionary technique, initially known for its effectiveness in machine translation tasks [15]. With the inception of models like BERT [16] and GPT-2 [17], the approach shifted towards pre-training on extensive corpora, which led to significant performance enhancements in downstream fine-tuning tasks [2, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]. However, the growing number of parameters in these models results in a huge GPU memory consumption, rendering the fine-tuning of large scale models ($\geq$ 65B) infeasible under low resource scenarios. This has prompted a shift towards more efficient training of LLMs.

2.2 Parameter-Effieient Fine-Tuning

Parameter-efficient fine-tuning (PEFT) methods adapt pre-trained models by fine-tuning only a subset of parameters. In general, PEFT methods can be grouped into three classes: 1) Prompt Learning methods [8, 28, 29, 30, 31, 32, 33], 2) Adapter methods [7, 9, 34, 35, 36, 37, 38], and 3) Selective methods [39, 39, 40, 41]. Prompt learning methods emphasize optimizing the input token or input embedding with frozen model parameters, which generally has the least training cost among all three types. Adapter methods normally introduce an auxiliary module with much fewer parameters than the original model, and updates are only applied to the adapter module during training. Compared with them, selective methods are more closely related to LISA, which focuses on optimizing a fraction of the model's parameters without appending extra modules. Recent advances in this domain have introduced several notable techniques through layer freezing. AutoFreeze [39] offers an adaptive mechanism to identify layers for freezing automatically and accelerates the training process. FreezeOut [42] progressively freezes intermediate layers, significantly reducing training time without notably affecting accuracy. The SmartFRZ [40] framework utilizes an attention-based predictor for layer selection, substantially cutting computation and training time while maintaining accuracy. However, none of these layer-freezing strategies has been widely adopted in the context of Large Language Models due to their inherent complexity or non-compatibility with modern memory reduction techniques [43, 44, 27] for LLMs.

2.3 Low-Rank Adaptation (LoRA)

In contrast, the Low-Rank Adaptation (LoRA) technique is much more prevalent in common LLM training [9]. LoRA reduces the number of trainable parameters by employing low-rank matrices, thereby lessening the computational burden and memory cost. One key strength of LoRA is its compatibility with models featuring linear layers, where the decomposed low-rank matrices can be merged back into the original model. This allows for efficient deployment without changing the model architecture. As a result, LoRA can be seamlessly combined with other techniques, such as quantization [11] or Mixture of Experts [45]. Despite these advantages, LoRA's performance is not universally comparable with full parameter fine-tuning. There have been tasks in [10] that LoRA performs much worse than full parameter training on. This phenomenon is especially evident in large-scale pre-training settings [12], where to the best of our knowledge, only full parameter training was adopted for successful open-source LLMs [21, 22, 23, 46, 47, 26, 27].

2.4 Large-scale Optimization Algorithms

In addition to approaches that change model architectures, there have also been efforts to improve the efficiency of optimization algorithms for LLMs. One such approach is layerwise optimization, a concept with roots extending back several decades. Notably, [48] introduced an effective layer-by-layer pre-training method for Deep Belief Networks (DBN), demonstrating the benefits of sequential layer optimization. This idea was expanded by researchers like [49, 50], who illustrated the advantages of a greedy, unsupervised approach to pre-training each layer of deep networks. In the context of large batch training, [51, 52] developed LARS and LAMB to improve generalization and mitigate the performance declines associated with large batch sizes. Despite these innovations, Adam [53, 54, 55, 27] and AdamW [56] continue to be the predominant optimization methods used in most LLM settings.

Recently, other attempts have also been made to reduce the training cost of LLMs. For example, MeZO [57] adopted zeroth order optimization, bringing significant memory savings during training. However, it also incurred a considerable performance drop in multiple benchmarks, particularly in complex fine-tuning scenarios. Regarding acceleration, Sophia [58] incorporates clipped second-order information into the optimization, obtaining non-trivial speedup on LLM training. The significant downsides are its intrinsic complexity of Hessian estimation and unverified empirical performance in large-size models (e.g., $\geq$ 65B). In parallel to our work, [59] proposed GaLore, a memory-efficient training strategy that reduces memory cost by projecting gradients into a low-rank compact space. Yet the performance has still not surpassed full-parameter training in fine-tuning settings. To sum up, LoRA-variant methods [9, 11, 59] with AdamW [56] is still the dominant paradigm for large-size LLM fine-tuning, the performance of which still demands further improvements.

3 Method

3.1 Motivation

To understand how LoRA achieves effective training with only a few parameters, we conducted empirical studies on multiple models, especially observing the weight norms across various layers. We fine-tune it on the Alpaca-GPT4 dataset [60]. During the training, we meticulously recorded the mean weight norms of each layer ℓ at every step t after updates, i.e.

$$\mathbf{w}^{(\ell)} \triangleq \texttt{mean-weight-norm}(\ell) = \frac{1}{T} \sum_{t=1}^{T} \|\boldsymbol{\theta}_t^{(\ell)}\|_2$$

Figure 2 presents these findings, with the x-axis representing the layer id, from embedding weights to the final layer, and the y-axis quantifying the weight norm. The visualization reveals one key trend:

- The embedding layer or the language modeling (LM) head layer exhibits significantly larger weight norms than intermediary layers in LoRA, often by a factor of hundreds. This phenomenon, however, was not salient under full-parameter training settings.
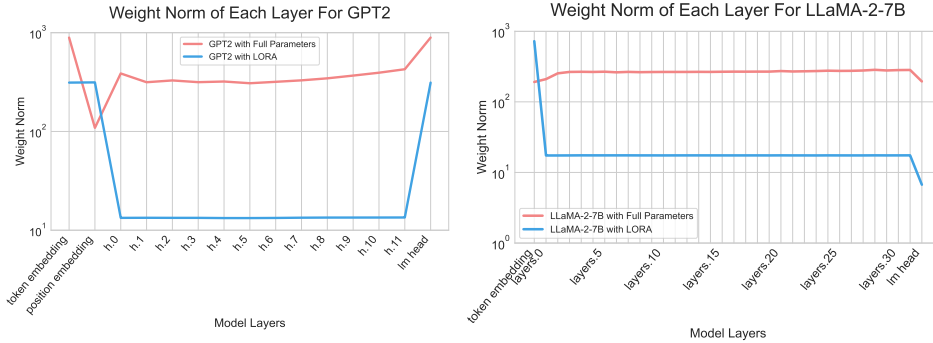

Figure 2: Layer-wise weight norms during training of GPT2 and LLaMA-2-7B Model with LoRA and Full Parameters training.

This observation indicates that the update emphasis of LoRA and full parameter training differ significantly, which can be attributed to the difference in their learned knowledge. For example, in embedding layers, tokens with similar meanings, i.e., synonyms, can be projected into the same embedding space and converted to similar embeddings. LoRA may capture this similarity in language and "group" them in the low-dimension space, allowing frequent features of language meanings to be promptly identified and optimized. The price is LoRA's limited representation power restricted by its intrinsic low-rank space, as we can see from the comparison with LISA in image generation tasks (Appendix A.1), where LoRA memorizes and learns details much slower than LISA. Other possible explanations can also justify this phenomenon. Despite various interpretations of this observation, one fact remains clear: *LoRA values layerwise importance differently from full parameter tuning*.

3.2 Layerwise Importance Sampled AdamW (LISA)

To exploit the discovery above, we aspire to simulate LoRA's updating pattern via sampling different layers to freeze. This way, we can avoid LoRA's inherent deficiency of limited low-rank representation ability and emulate its fast learning process. Intuitively, given the same global learning rates across layers, layers with small weight norms in LoRA should also have small sampling probabilities to unfreeze in full-parameter settings so the expected learning rates across iterations can stay the same. This is exactly the idea of importance sampling [13, 14], where instead of applying layerwise different learning rates $\{\eta_t\}$ in full-parameter settings to emulate LoRA's updates $\{\tilde{\eta}_t\}$, we apply sampling and instead get the same expected parameter update

$$\eta_t^{(\ell)} = \tilde{\eta}_t^{(\ell)} \cdot \frac{\tilde{\mathbf{w}}^{(\ell)}}{\mathbf{w}^{(\ell)}} \quad \Rightarrow \quad \eta_t^{(\ell)} = \eta^{(\ell)}, p^{(\ell)} = \frac{\tilde{\mathbf{w}}^{(\ell)}}{\mathbf{w}^{(\ell)}}$$

4

This gives rise to our Layerwise Importance Sampling AdamW method, as illustrated in Algorithm 1. In practice, since all layers except the bottom and top layer have small weight norms in LoRA, we adopt $\{p_\ell\}_{\ell=1}^{N_L} = \{1.0, \gamma/N_L, \gamma/N_L, \ldots, \gamma/N_L, 1.0\}$ in practice, where γ controls the expected number of unfreeze layers during optimization, and the embedding layer E and head layer H remain active. Intuitively, γ serves as a compensation factor to bridge the difference between LoRA and full parameter tuning, letting LISA emulate a similar layerwise update pattern as LoRA. To further control the memory consumption in practical settings, we instead randomly sample γ layers every time to upper-bound the maximum number of unfrozen layers during training.

Algorithm 1 **L**ayerwise **I**mportance **S**ampling **A**damW (**LISA**)

Require: number of layers N_L, number of iterations T, sampling period K, number of sampled layers γ, initial learning rate η_0
1: **for** $i \leftarrow 0$ to $T/K - 1$ **do**
2:    Freeze all layers except the embedding and language modeling head layer
3:    Randomly sample γ intermediate layers to unfreeze
4:    Run AdamW for K iterations with $\{\eta_t\}_{t=ik}^{ik+k-1}$
5: **end for**

---

4 Experimental Results

Table 1: The chart illustrates peak GPU memory consumption for various model architectures and configurations, highlighting differences across models. The LISA configuration is specifically labeled in the table: "E" denotes the embedding layer, "H" represents the language modeling head layer, and "2L" indicates two additional intermediate layers. *: Model parallelism is applied for the 70B model.

	VANILLA	LoRA RANK			LISA ACTIVATE LAYERS		
MODEL	-	128	256	512	E+H	E+H+2L	E+H+4L
GPT2-SMALL	3.8G	3.3G	3.5G	3.7G	3.3G	3.3G	3.4G
TINYLLAMA	13G	7.9G	8.6G	10G	7.4G	8.0G	8.3G
MISTRAL-7B	59G	23G	26G	28G	21G	23G	24G
LLAMA-2-7B	59G	23G	26G	28G	21G	23G	24G
LLAMA-2-70B*	OOM	79G	OOM	OOM	71G	75G	79G

4.1 Memory Efficiency

We conducted peak GPU memory experiments to demonstrate LISA's memory efficiency and showcase its comparable or lower memory cost than LoRA.

Settings To reasonably estimate the memory cost, we randomly sample prompts from the Alpaca dataset [61] and limit the maximum output token length to 1024. We focus on two key hyperparameters: LoRA's rank and LISA's number of activation layers. For other hyperparameters, a mini-batch size of 1 was consistently used across five LLMs from 120M to 70B parameters, deliberately excluding other GPU memory-saving techniques such as gradient checkpointing [62], offloading [63], and flash attention [64, 65]. All memory-efficiency experiments are conducted on $4\times$ NVIDIA Ampere Architecture GPUs with 80G memory.

Figure 3: GPU memory consumption of LLaMA-2-7B with different methods and batch size 1.

Results Upon examining Table 1, it is evident that the LISA configuration, particularly when enhanced with both the embedding layer (E) and two additional layers (E+H+2L), demonstrates a considerable reduction in GPU memory usage when fine-tuning the LLaMA-2-70B model, as compared to the LoRA method. Specifically, the LISA E+H+2L configuration shows a decrease to 75G of peak GPU memory from the 79G required by the
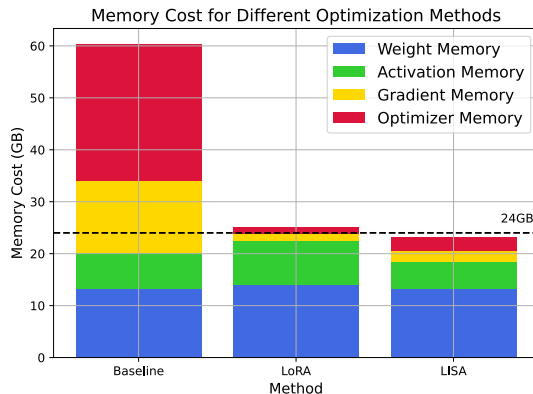
LoRA Rank 128 configuration. This efficiency gain is not an isolated incident; a systematic memory usage decrease is observed across various model architectures, suggesting that LISA's method of activating layers is inherently more memory-efficient.

In Figure 3, it is worth noticing that the memory reduction in LISA allows LLaMA-2-7B to be trained on a single RTX4090 (24GB) GPU, which makes high-quality fine-tuning affordable even on a laptop computer. In particular, LISA requires much less activation memory consumption than LoRA since it does not introduce additional parameters brought by the adaptor. LISA's activation memory is even slightly less than full parameter training since pytorch [66] with deepspeed [44] allows deletion of redundant activations before backpropagation.

On top of that, a reduction in memory footprint from LISA also leads to an acceleration in speed. As shown in Figure 4, LISA provides almost $2.9\times$ speedup when compared with full-parameter training, and $\sim 1.5\times$ speedup against LoRA, partially due to the removal of adaptor structures. It is worth noticing that the reduction of memory footprint in both LoRA and LISA leads to a significant acceleration of forward propagation, emphasizing the importance of memory-efficient training.

Figure 4: Single-iteration time cost of LLaMA-2-7B with different methods and batch size 1.

4.2 Moderate Scale Fine-Tuning

LISA can achieve this significant memory saving while still obtaining competitive performance under the fine-tuning setting.
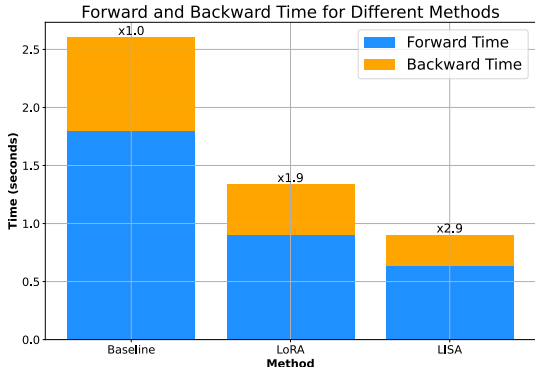
Table 2: Results of different methods on MMLU, AGIEval, and WinoGrande, measured by accuracy.

MODEL	METHOD	MMLU (5-SHOT)	AGIEval (3-SHOT)	WinoGrande (5-SHOT)
TINYLLAMA	VANILLA	25.50	19.55	59.91
	LoRA	25.81 ± 0.07	19.82 ± 0.11	61.33 ± 0.09
	GaLore	25.21 ± 0.06	21.19 ± 0.07	61.09 ± 0.12
	LISA	**26.02 ± 0.13**	**21.71 ± 0.09**	61.48 ± 0.08
	FT	25.62 ± 0.10	21.28 ± 0.07	**62.12 ± 0.15**
MISTRAL-7B	VANILLA	60.12	26.79	79.24
	LoRA	61.78 ± 0.09	27.56 ± 0.07	78.85 ± 0.11
	GaLore	57.87 ± 0.08	26.23 ± 0.05	75.85 ± 0.13
	LISA	**62.09 ± 0.10**	**29.76 ± 0.09**	**78.93 ± 0.08**
	FT	61.70 ± 0.13	28.07 ± 0.12	78.85 ± 0.12
LLaMA-2-7B	VANILLA	45.87	25.69	74.11
	LoRA	45.50 ± 0.07	24.73 ±0.04	74.74 ± 0.09
	GaLore	45.56 ± 0.05	24.39 ± 0.11	73.32 ± 0.12
	LISA	**46.21 ± 0.12**	26.06 ± 0.08	**75.30 ± 0.11**
	FT	45.66 ± 0.09	**27.02 ± 0.10**	75.06 ± 0.13

Settings To demonstrate the superiority of LISA over LoRA, we evaluate them on the instruction-following fine-tuning task with the Alpaca GPT-4 dataset [61], which consists of 52k conversation pairs generated by GPT-4 [5]. The effectiveness of fine-tuning was evaluated on multiple benchmarks: MT-Bench [67] features 80 high-quality, multi-turn questions designed to assess LLMs on multiple aspects; MMLU [68] includes a total of 57 tasks with 14,079 questions covering a broad spectrum of world knowledge; AGIEval [69] serves as a human-centric benchmark for general abilities, comprising 9,316 instances; WinoGrande [70] is a large-scale dataset for commonsense reasoning, consisting of 44,000 instances designed to challenge models' understanding of the context and commonsense knowledge.

In our experiments, we assessed three baseline models: TinyLlama [47], Mistral-7B [46], and LLaMA-2-7B [23]. These models, varying in size ranging from 1B to 7B parameters, provide a

diverse representation of decoder-only models. For hyper-parameters, we adopt a rank of 128 for LoRA and E+H+2L for LISA in this section, with full details available in Appendix B.

Results Table 2 and 3 present a detailed comparison on moderate-scale LLMs. The baselines include Full-parameter Training (FT), Low-Rank Adaptation (LoRA) [9] and Gradient Low-Rank Projection (GaLore) [59]. The results demonstrate that LISA consistently outperforms other fine-tuning methods in most evaluation tracks, indicating its robustness and effectiveness across diverse tasks and model architectures. LISA is particularly effective in instruction following tasks, where a large gap is observed when compared with other baseline methods. LISA even outperforms Full-parameter Training, suggesting that an implicit regularization effect is present when the number of unfrozen layers is restricted, which is similar to dropout [71]. According to more results in stable diffusion and detailed MT-Bench scores, we found that LISA outperforms LoRA mostly in memorization tasks, such as depicting high-resolution image details in image generation, or Writing or Humanities tasks

Table 3: Different methods on MT-Bench.

MODEL	METHOD	MT-BENCH ↑
TINYLLAMA	VANILLA	1.25
	LoRA	1.90 ± 0.14
	GALORE	**2.61 ± 0.17**
	LISA	2.57 ± 0.25
	FT	2.21 ± 0.16
MISTRAL-7B	VANILLA	4.32
	LoRA	4.41 ± 0.09
	GALORE	4.36 ± 0.16
	LISA	**4.85 ± 0.14**
	FT	4.64 ± 0.12
LLAMA-2-7B	VANILLA	3.29
	LoRA	4.45 ± 0.15
	GALORE	4.63 ± 0.09
	LISA	**4.94 ± 0.14**
	FT	4.75 ± 0.16

in instruction following. This implies that LISA's performance improvement may majorly come from the ability to memorize long-tailed patterns, while LoRA is better at multi-hop reasoning with limited knowledge. For more details, please refer to Appendix A.1 and A.2.

4.3 Moderate Scale Continual Pre-training

Continual pre-training is crucial for enabling models to adapt to new data and domains. To evaluate LISA's efficacy in the continual pre-training scenario, we experiment on the mathematics domain in comparison with Full-parameter Training.

Settings We adopt the mathematics corpus OpenWebMath [72] for constructing the continual pre-training dataset. Specifically, we extracted a high-quality subset from it which contains 1.5 billion tokens. Full details are explained in Appendix B.2. After continual pre-

Table 4: Comparison of Moderate Scale Model Continual Pre-training on OpenWebMath Dataset.

MODEL	METHOD	GSM8K ↑	MEM. ↓
TINYLLAMA	VANILLA	2.26	-
	LISA	**3.56**	**8G**
	FT	3.26	13G
LLAMA-2-7B	VANILLA	14.40	-
	LISA	**22.21**	**26G**
	FT	22.21	59G

trainig, we then apply the same fine-tuning procedure on the GSM8K [73] training set, which comprises 7473 instances.

Results Table 4 shows that LISA is capable of achieving on-par or even better performance than full-parameter training with much less memory consumption. Specifically, LISA requires only half of the memory cost compared to full-parameter training. This indicates a better balance between computational efficiency and model performance is achieved by LISA. According to our experience, reducing the number of unfrozen layers to half the original size leads to no worse or even better performance during continual pretraining, while requiring much less memory consumption.

4.4 Large Scale Fine-Tuning

To further demonstrate LISA's scalability on large-sized LLMs, we conduct additional fine-tuning experiments on LLaMA-2-70B [23].

Settings On top of the aforementioned instruction-following tasks in Section 4.2, we use extra domain-specific fine-tuning tasks on mathematics and medical QA benchmarks. The GSM8K dataset [73], comprising 7473 training instances and 1319 test instances, is used for the mathematics

domain. For the medical domain, we select the PubMedQA dataset [74], which includes 211.3K artificially generated QA training instances and 1K test instances.

Evaluation on the PubMedQA dataset [74] is conducted in a 5-shot prompt setting, while the GSM8K dataset [73] assessment was conducted using Chain-of-Thought (CoT) prompting, following recent studies [75, 76, 77]. Regarding hyperparameters, as detailed in the section 4.1, we utilize the rank 256 for LoRA and the configuration E+H+4L for LISA. Further information is available in Appendix B.

Table 5: Different methods on MT-Bench, GSM8K, and PubMedQA score for LLaMA-2-70B.

METHOD	MT-BENCH↑	GSM8K↑	PUBMEDQA↑
VANILLA	5.19	54.8	83.0
LORA	6.10	59.4	90.8
LISA	**6.72**	61.1	**91.6**
FT	6.25	**67.1**	90.8

Results   As shown in Table 5, LISA consistently produces better or on-par performance when compared with LoRA. Furthermore, LISA again surpasses full-parameter training in instruction-tuning tasks, providing strong evidence to support LISA's scalability under large-scale training scenarios. More results are available in Appendix A.2.

4.5 Ablation Studies

Hyperparameters of LISA   The two key hyperparameters of LISA are the number of sampling layers γ and sampling period K. To obtain intuitive and empirical guidance of those hyperparameter choices, we conduct ablation studies using TinyLlama [47] and LLaMA-2-7B [23] models with the Alpaca-GPT4 dataset. The configurations for γ, such as E+H+2L, E+H+8L, were denoted as $\gamma = 2$ and $\gamma = 8$. As for the sampling period $K = T/n$, $T = 122$ representing the maximum training step within our experimental framework. The findings, presented in Table 6, reveal that both γ and K markedly affect the LISA algorithm's performance. Specifically, a higher γ value increases the quantity of trainable parameters, albeit with higher memory costs. On the other hand, an optimal K value facilitates more frequent layer switching, thereby improving performance to a certain threshold, beyond which the performance may deteriorate. Generally, the rule of thumb is: *More sampling layers and higher sampling period lead to better performance.* For a detailed examination of loss curves and MT-Bench results, refer to Appendix A.4.

Table 6: Different LISA hyperparameters combinations. All settings adopt learning rate $\eta_0 = 10^{-5}$. Here γ stands for sampling layers, K stands for sampling period.

MODELS	γ	K	MT-BENCH SCORE
TINYLLAMA	2	$\lceil T/125 \rceil$	2.44
		$\lceil T/25 \rceil$	**2.73**
		$\lceil T/5 \rceil$	2.64
		T	2.26
	8	$\lceil T/125 \rceil$	2.59
		$\lceil T/25 \rceil$	**2.81**
		$\lceil T/5 \rceil$	2.74
		T	2.53
LLAMA-2-7B	2	$\lceil T/125 \rceil$	4.86
		$\lceil T/25 \rceil$	**4.91**
		$\lceil T/5 \rceil$	4.88
		T	4.64
	8	$\lceil T/125 \rceil$	4.94
		$\lceil T/25 \rceil$	**5.11**
		$\lceil T/5 \rceil$	5.01
		T	4.73

Sensitiveness of LISA   As LISA is algorithmically dependent on the sampling sequence of layers, it is intriguing to see how stable LISA's performance is under the effect of randomness. For this purpose, we further investigate LISA's performance variance over three distinct runs, each with a different random seed for layer selection. Here, we adopt TinyLlama, LLaMA-2-7B, and Mistral-7B models with the Alpaca-GPT4 dataset while keeping all other hyperparameters consistent with those used in the instruction following experiments in section 4.2. As shown in Table 7, LISA is quite resilient to different random seeds, where the performance gap across three runs is within 0.13, a small value compared to the

Table 7: The MT-Bench scores derived from varying random seeds for layer selection.

MODEL	SEED 1	SEED 2	SEED 3
TINYLLAMA	2.57	2.55	2.60
MISTRAL-7B	4.85	4.82	4.82
LLAMA-2-7B	4.94	4.92	4.89

performance gains over baseline methods. For more ablation experiment on LISA hyperparameters, please refer to Appendix A.4.

5 Discussion

Theoretical Properties of LISA   Compared with LoRA, which introduces additional parameters and leads to changes in loss objectives, layerwise importance sampling methods enjoy nice convergence guarantees in the original loss. For layerwise importance sampled SGD, similar to gradient sparsification [78, 55], the convergence can still be guaranteed for unbiased estimation of gradients with increased variance. The convergence behavior can be further improved by reducing the variance with appropriately defined importance sampling strategy [14]. For layerwise importance sampled Adam, theoretical results in [79, 55] prove its convergence in convex objectives. If we denote f as the loss function and assume that the stochastic gradients are bounded, then based on [56], we know that AdamW optimizing f aligns with Adam optimizing f with a scaled regularizer, which can be written as

$$f^{\text{reg}}(\mathbf{w}) \triangleq f(\mathbf{w}) + \frac{1}{2}\mathbf{w}^\top \mathbf{S}\mathbf{w},$$

where $\mathbf{S}$ is a finite positive semidefinite diagonal matrix. Following existing convergence results of RBC-Adam (Corollary 1 in [79]), we have the convergence guarantee of LISA in Theorem 1.

Theorem 1 *Let the loss function f be convex and smooth. If the algorithm runs in a bounded convex set and the stochastic gradients are bounded, the sequence $\{\mathbf{w}_t\}_{t=1}^{T}$ generated by LISA admits the following convergence rate:*

$$\frac{1}{T}\sum_{t=1}^{T} f^{\text{reg}}(\mathbf{w}_t) - f_*^{\text{reg}} \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right),$$

where f_*^{reg} denotes the optimum value of f^{reg}.

Memorization and Reasoning   In our instruction following experiments in Appendix A.1 and A.2, we observe that LISA is much better than LoRA at memorization-centered tasks, such as Writing or depicting image details, while this gap is much smaller in reasoning-centered tasks like Code or Math. It is an intriguing observation since LISA emphasizes more on layer-wise width and restricts the depth of learned parameters, while LoRA focuses more on depth and restricts the representation space in each layer. It may suggest that width is crucial for memorization, while depth is important for reasoning, a similar phenomenon that echos the intuition of [80]. Based on the same intuition, it may be possible to combine the benefits of both and bring forth an even better PEFT method.

6 Conclusion

In this paper, we propose Layerwise Importance Sampled AdamW (LISA), an optimization algorithm that randomly freezes layers of LLM based on a given probability. Inspired by observations of LoRA's skewed weight norm distribution, a simple and memory-efficient freezing paradigm is introduced for LLM training. This paradigm achieves significant performance improvements over LoRA on downstream fine-tuning tasks with various models, including LLaMA-2-70B. Further experiments on domain-specific training also demonstrate its effectiveness, showing LISA's huge potential as a promising alternative to LoRA for LLM training.

Limitations

The major bottleneck of LISA is the same as LoRA, where during optimization, the forward pass still requires the model to be presented in the memory, leading to significant memory consumption. This limitation shall be compensated by approaches similar to QLoRA [11], where we intend to conduct further experiments to verify its performance.

In addition, as suggested by the theoretical intuition, the strategy of E+H+2L in Section 4.2 and E+H+4L in Section 4.4 may not be the optimal importance sampling strategy, given it still sampled

intermediate layers in a uniformly random fashion. We anticipate the optimizer's efficiency will be further improved when considering data sources and model architecture in the importance sampling procedure.

References

[1] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

[2] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[3] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113, 2022.

[4] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2023.

[5] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike,

Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2023.

[6] Longteng Zhang, Xiang Liu, Zeyu Li, Xinglin Pan, Peijie Dong, Ruibo Fan, Rui Guo, Xin Wang, Qiong Luo, Shaohuai Shi, and Xiaowen Chu. Dissecting the runtime performance of the training, fine-tuning, and inference of large language models, 2023.

[7] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019.

[8] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, 2021. Association for Computational Linguistics.

[9] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[10] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *ArXiv preprint*, abs/2203.06904, 2022.

[11] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *ArXiv preprint*, abs/2305.14314, 2023.

[12] Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. Relora: High-rank training through low-rank updates, 2023.

[13] Teun Kloek and Herman K Van Dijk. Bayesian estimates of equation system parameters: an application of integration by monte carlo. *Econometrica: Journal of the Econometric Society*, pages 1–19, 1978.

[14] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1–9. JMLR.org, 2015.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[17] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[18] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[19] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *ArXiv preprint*, abs/2205.01068, 2022.

[20] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv preprint*, abs/2211.05100, 2022.

[21] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Maitha Alhammadi, Mazzotta Daniele, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. The falcon series of language models: Towards open frontier models. 2023.

[22] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *ArXiv preprint*, abs/2302.13971, 2023.

[23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288, 2023.

[24] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, 2023.

[25] 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. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

[26] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *ArXiv preprint*, abs/2401.04088, 2024.

[27] Zhenheng Tang, Yuxin Wang, Xin He, Longteng Zhang, Xinglin Pan, Qiang Wang, Rongfei Zeng, Kaiyong Zhao, Shaohuai Shi, Bingsheng He, et al. Fusionai: Decentralized training and deploying llms with massive consumer-level gpus. *arXiv preprint arXiv:2309.01172*, 2023.

[28] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4921–4933, Online, 2021. Association for Computational Linguistics.

[29] Zexuan Zhong, Dan Friedman, and Danqi Chen. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033, Online, 2021. Association for Computational Linguistics.

[30] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. PTR: Prompt Tuning with Rules for Text Classification. *ArXiv preprint*, abs/2105.11259, 2021.

[31] Guanghui Qin and Jason Eisner. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online, 2021. Association for Computational Linguistics.

[32] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *ArXiv preprint*, abs/2103.10385, 2021.

[33] Shizhe Diao, Zhichao Huang, Ruijia Xu, Xuechun Li, LIN Yong, Xiao Zhou, and Tong Zhang. Black-box prompt learning for pre-trained language models. *Transactions on Machine Learning Research*, 2022.

[34] Shizhe Diao, Ruijia Xu, Hongjin Su, Yilei Jiang, Yan Song, and Tong Zhang. Taming pre-trained language models with n-gram representations for low-resource domain adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3336–3349, Online, 2021. Association for Computational Linguistics.

[35] Shizhe Diao, Tianyang Xu, Ruijia Xu, Jiawei Wang, and Tong Zhang. Mixture-of-domain-adapters: Decoupling and injecting domain knowledge to pre-trained language models memories. *ArXiv preprint*, abs/2306.05406, 2023.

[36] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models, 2024.

[37] Haobo SONG, Hao Zhao, Soumajit Majumder, and Tao Lin. Increasing model capacity for free: A simple strategy for parameter efficient fine-tuning. In *The Twelfth International Conference on Learning Representations*, 2024.

[38] Yuheng Ji, Yue Liu, Zhicheng Zhang, Zhao Zhang, Yuting Zhao, Gang Zhou, Xingwei Zhang, Xinwang Liu, and Xiaolong Zheng. Advlora: Adversarial low-rank adaptation of vision-language models, 2024.

[39] Yuhan Liu, Saurabh Agarwal, and Shivaram Venkataraman. Autofreeze: Automatically freezing model blocks to accelerate fine-tuning, 2021.

[40] Sheng Li, Geng Yuan, Yue Dai, Youtao Zhang, Yanzhi Wang, and Xulong Tang. SmartFRZ: An efficient training framework using attention-based layer freezing. In *The Eleventh International Conference on Learning Representations*, 2023.

[41] Qijun Luo, Hengxu Yu, and Xiao Li. Badam: A memory efficient full parameter training method for large language models, 2024.

[42] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Freezeout: Accelerate training by progressively freezing layers. *ArXiv preprint*, abs/1706.04983, 2017.

[43] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[44] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 3505–3506. ACM, 2020.

[45] Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T Kwok, and Yu Zhang. Mixture of cluster-conditional lora experts for vision-language instruction tuning. *ArXiv preprint*, abs/2312.12379, 2023.

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

[47] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model, 2024.

[48] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, page 1527–1554, 2006.

[49] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In Bernhard Schölkopf, John C. Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 153–160. MIT Press, 2006.

[50] Zhenheng Tang, Yonggang Zhang, Shaohuai Shi, Xinmei Tian, Tongliang Liu, Bo Han, and Xiaowen Chu. Fedimpro: Measuring and improving client update in federated learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[51] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *ArXiv preprint*, abs/1708.03888, 2017.

[52] Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[53] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[54] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[55] Zhenheng Tang, Shaohuai Shi, Xiaowen Chu, Wei Wang, and Bo Li. Communication-efficient distributed deep learning: A comprehensive survey. *arXiv preprint arXiv:2003.06307*, 2020.

[56] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[57] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[58] Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations*, 2024.

[59] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *ArXiv preprint*, abs/2403.03507, 2024.

[60] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *ArXiv preprint*, abs/2304.03277, 2023.

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

[62] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *ArXiv preprint*, abs/1604.06174, 2016.

[63] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. Zero-offload: Democratizing billion-scale model training, 2021.

[64] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

[65] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[66] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.

[67] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[68] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[69] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023.

[70] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press, 2020.

[71] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[72] Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text. In *The Twelfth International Conference on Learning Representations*, 2024.

[73] 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. Training verifiers to solve math word problems, 2021.

[74] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China, 2019. Association for Computational Linguistics.

[75] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[76] Kashun Shum, Shizhe Diao, and Tong Zhang. Automatic prompt augmentation and selection with chain-of-thought from labeled data. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12113–12139, 2023.

[77] Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. Active prompting with chain-of-thought for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1330–1350, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[78] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 1306–1316, 2018.

[79] Yangfan Zhou, Mingchuan Zhang, Junlong Zhu, Ruijuan Zheng, and Qingtao Wu. A randomized block-coordinate adam online learning optimization algorithm. *Neural Computing and Applications*, 32(16):12671–12684, 2020.

[80] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[81] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[82] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *ArXiv preprint*, abs/2310.04378, 2023.

[83] Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[84] Lucas Georges Gabriel Charpentier and David Samuel. Not all layers are equally as important: Every layer counts BERT. In Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell, editors, *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 238–252, Singapore, December 2023. Association for Computational Linguistics.

[85] Shizhe Diao, Rui Pan, Hanze Dong, KaShun Shum, Jipeng Zhang, Wei Xiong, and Tong Zhang. LMFlow: An extensible toolkit for finetuning and inference of large foundation models. In Kai-Wei Chang, Annie Lee, and Nazneen Rajani, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)*, pages 116–127, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

A Additional Experiments

A.1 Image Generation

Stable Diffusion has emerged as a powerful approach for generating high-quality images by leveraging the diffusion process in a latent space [81]. This method involves diffusing the data distribution over time and iteratively refining the generated samples to enhance their realism and fidelity. The key innovation lies in operating within the latent space of a pre-trained autoencoder, significantly reducing computational complexity while maintaining high-quality outputs. The Latent Consistency Model (LCM) further improves the efficiency and performance of diffusion models in the latent space by incorporating a consistency loss, which penalizes deviations from the expected latent trajectories [82]. This ensures that the generated samples remain coherent and visually plausible throughout the diffusion steps.

In our experiments, we evaluated the performance of LCM against other fine-tuning methods such as LoRA, and LISA using latent consistency distillation to distill stable diffusion v1.5 and v2.1 model. We utilized the official code* and parameters for both LISA and LoRA from the Diffusers library to ensure consistency and comparability of results. Adjustments were made only to the batch size and accumulation steps to achieve a balanced trade-off between computational efficiency and performance. Figure 5 is the generated image comparison. Observations reveal that LISA can generate higher-quality images in fewer inference steps. The images generated using LISA display more intricate details and sharper clarity, particularly evident in the distinct facial features and environment textures. In contrast, the LoRA-generated images offer a softer, more blended aesthetic with a dream-like quality, emphasizing smooth transitions over precise detail. The prompts for images in figure 5 in the left-to-right are given below

- Self-portrait oil painting, a beautiful cyborg with golden hair, 8k.

- Astronaut in a jungle, cold color palette, muted colors, detailed, 8k.

- A photo of a beautiful mountain with a realistic sunset and blue lake, a highly detailed masterpiece.

A.2 Instruction Following Fine-tuning

Table 8 offers a comprehensive evaluation of three fine-tuning methods—Full Parameter Fine-Tuning (FT), Low-Rank Adaptation (LoRA), Gradient Low-Rank Projection(GaLore), and Layerwise Importance Sampling AdamW (LISA)—across a diverse set of tasks including Writing, Roleplay, Reasoning, Math, Extraction, STEM, and Humanities within the MT-Bench benchmark. The results demonstrate LISA's superior performance, which surpasses LoRA, GaLore, and full parameter tuning in most settings. Notably, LISA consistently outperforms LoRA and full parameter tuning in domains such as Writing, STEM, and Humanities. This implies that LISA can benefit memorization tasks, while LoRA partially favors reasoning tasks.

Table 8: Comparison of Language Model Fine-Tuning Methods on the MT-Bench score.

MODEL & METHOD	WRITING	ROLEPLAY	REASONING	CODE	MATH	EXTRACTION	STEM	HUMANITIES	AVG. ↑
TINYLLAMA (VANILLA)	1.05	2.25	1.25	1.00	1.00	1.00	1.45	1.00	1.25
TINYLLAMA (LoRA)	2.77	4.05	1.35	1.00	**1.40**	1.00	1.55	2.15	1.90
TINYLLAMA (GALORE)	**3.55**	**5.20**	2.40	**1.15**	1.40	**1.85**	2.95	2.40	**2.61**
TINYLLAMA (**LISA**)	3.30	4.40	**2.65**	1.12	1.30	1.75	**3.00**	**3.05**	<u>2.57</u>
TINYLLAMA (FT)	3.27	3.95	1.35	1.04	1.33	1.73	2.69	2.35	2.21
MISTRAL-7B (VANILLA)	5.25	3.20	4.50	1.60	2.70	**6.50**	**6.17**	4.65	4.32
MISTRAL-7B (LoRA)	5.30	4.40	4.65	**2.35**	**3.30**	5.50	5.55	4.30	4.41
MISTRAL-7B (GALORE)	5.05	**5.27**	4.45	1.70	2.50	5.21	5.52	5.20	4.36
MISTRAL-7B (**LISA**)	**6.84**	3.65	**5.45**	2.20	2.75	5.65	5.95	**6.35**	**4.85**
MISTRAL-7B (FT)	5.50	4.45	5.45	2.50	3.25	5.78	4.75	5.45	4.64
LLAMA-2-7B (VANILLA)	2.75	4.40	2.80	1.55	1.80	3.20	5.25	4.60	3.29
LLAMA-2-7B (LoRA)	6.30	5.65	**4.05**	1.60	1.45	4.17	6.20	6.20	4.45
LLAMA-2-7B (GALORE)	5.60	6.40	3.20	1.25	1.95	**5.05**	6.57	7.00	4.63
LLAMA-2-7B (**LISA**)	**6.55**	**6.90**	3.45	**1.60**	**2.16**	4.50	**6.75**	**7.65**	**4.94**
LLAMA-2-7B (FT)	5.55	6.45	3.60	1.75	2.00	4.70	6.45	7.50	4.75

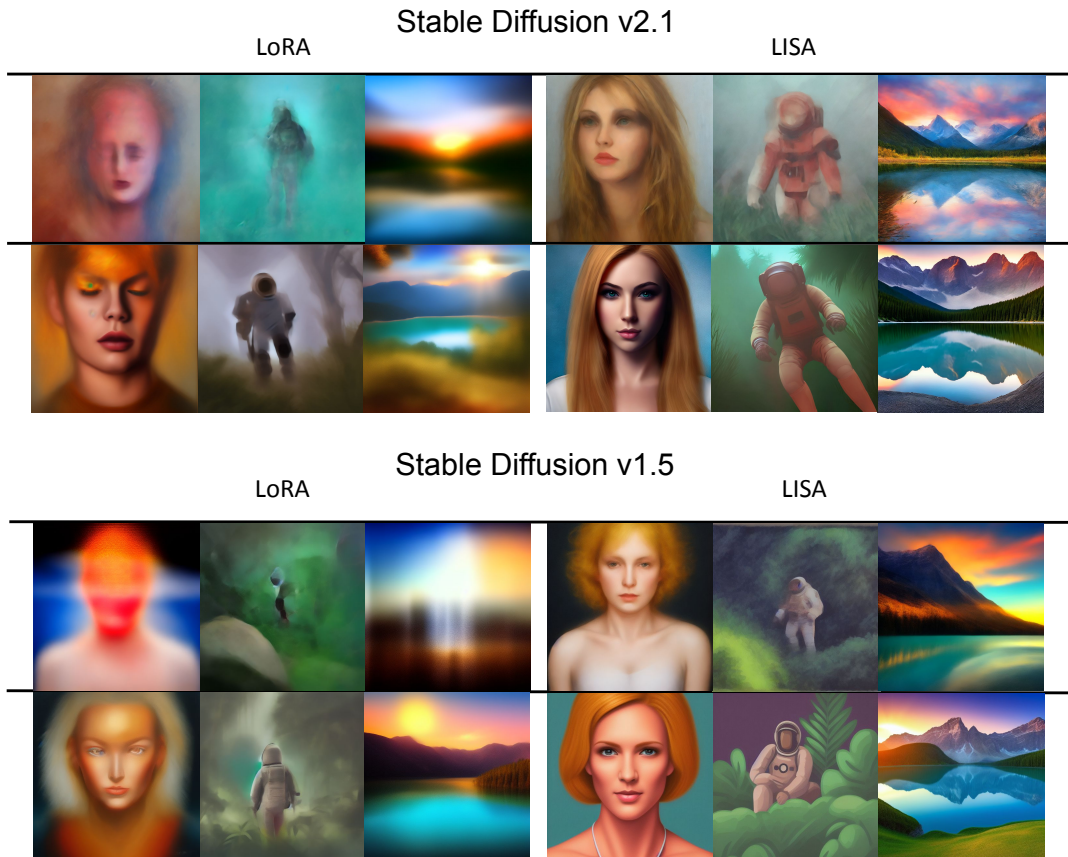*https://github.com/huggingface/diffusers/tree/main/examples/consistency_distillation

Figure 5: Generated images using LoRA (left) and LISA (right) on Stable Diffusion v2.1 model and Stable Diffusion v1.5. **First row**: number of inference step = 2. **Second row**: number of inference step = 10.

Table 9 provides detailed MT-Bench scores for the LLaMA-2-70B model discussed in Section 4.4, demonstrating LISA's superior performance over LoRA in all aspects under large-scale training scenarios. Furthermore, in Figure 6, we observe that LISA consistently exhibits on-par or faster convergence speed than LoRA across different models, which provides strong evidence for LISA's superiority in practice.

Table 9: Mean score of three fine-tuning methods over three seeds for LLaMA-2-70B on the MT-Bench.

MODEL & METHOD	MT-BENCH								
	WRITING	ROLEPLAY	REASONING	CODE	MATH	EXTRACTION	STEM	HUMANITIES	AVG. ↑
LLAMA-2-70B(VANILLA)	7.77	5.52	2.95	1.70	1.70	6.40	7.42	8.07	5.19
LLAMA-2-70B(LORA)	7.55	7.00	5.30	3.15	2.60	6.55	8.00	8.70	6.10
LLAMA-2-70B(**LISA**)	**8.18**	**7.90**	**5.45**	**4.45**	**2.75**	**7.45**	**8.60**	**9.05**	**6.72**
LLAMA-2-70B(FT)	6.45	7.50	5.50	3.40	2.15	7.55	8.10	9.40	6.25

The aforementioned results show that Vanilla LLaMA-2-70B excels in Writing, but full-parameter fine-tuning led to a decline in these areas, a phenomenon known as the "Alignment Tax" [1]. This tax highlights the trade-offs between performance and human alignment in instruction tuning. LISA, however, maintains strong performance across various domains with a lower "Alignment Tax".
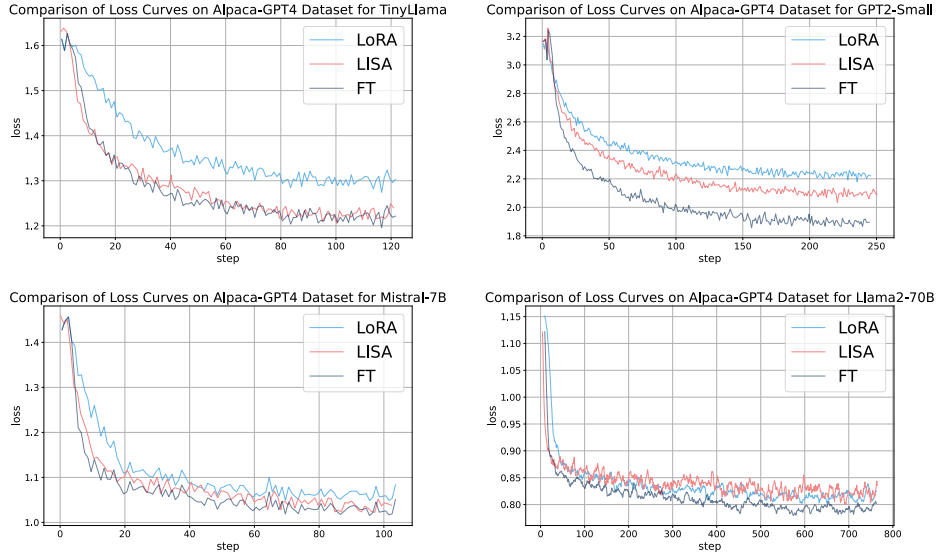
Figure 6: Loss curves for LoRA, LISA, and full-parameter training on the Alpaca-GPT4 dataset across different models.

A.3 Continual Pre-training

To better analyze the performance of LISA in the continual pre-training scenario, we used OpenWebMath for continual pre-training and the GSM8K train split for the fine-tuning stage, varying the number of sampling layers, γ, within LISA, ranging from $2, 4, 8, 16$, compared to the accuracy of the full parameter (FT) continual pre-training. Table 7 details the results for the LLaMA-2-7B model under various continual pre-training configurations. Notably, LISA with eight sampling layers achieves comparable accuracy with full parameters continual pre-training method. Furthermore, LISA with 16 sampling layers passes the accuracy of full parameter training.
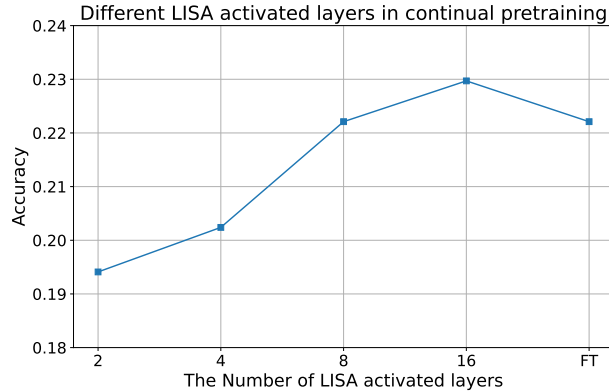

Figure 7: The comparison of full parameter (FT) training and LISA with different sampling layers under continual pre-training scenario. The accuracy is the test set of GSM8K.

A.4 Ablation Experiments

A.4.1 Sampling Layers γ

We conducted an ablation study on the LLaMA-2-7B model trained with the Alpaca-GPT4 dataset, setting the sampling period $K = 13$, so the number of samplings is exactly 10. The study explored different configurations of sampling layers γ including {E+H+2L, E+H+4L, E+H+8L}. Figure 8 depicts the impact of the number of sampling layers γ on the training dynamics of the model. Three scenarios were analyzed: $\gamma = 2$ (blue line), $\gamma = 4$ (green line), and $\gamma = 8$ (red line), throughout 120 training steps. Initially, all three configurations exhibit a steep decrease in loss, signaling rapid initial improvements in model performance. it's clear that the scenario with $\gamma = 8$ consistently maintains a lower loss compared to the $\gamma = 2$ and $\gamma = 4$ configurations, suggesting that a higher γ value leads to better performance in this context.
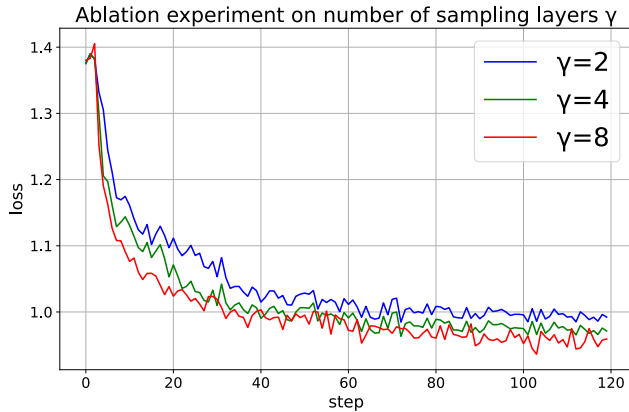
Figure 8: Comparison of loss curves for the γ ablation experiment.

Table 10 demonstrates that LISA with only $E + H$ layers experiences a performance decline when the 0 sampling layers γ are omitted, highlighting the importance of these layers.

Table 10: Results of additional experiments on LISA with only $E + H$ layers.

MODEL	METHOD	MMLU (5-SHOT)	AGIEVAL (3-SHOT)	WINOGRANDE (5-SHOT)	MT-BENCH
	VANILLA	25.50	19.55	59.91	1.25
	LISA (E + H)	25.49 ± 0.14	20.75 ± 0.21	60.43 ± 0.19	2.18 ± 0.31
TINYLLAMA	**LISA**	**26.02 ± 0.13**	**21.71 ± 0.09**	61.48 ± 0.08	**2.57 ± 0.25**
	FT	25.62 ± 0.10	21.28 ± 0.07	**62.12 ± 0.15**	2.21 ± 0.16
	VANILLA	60.12	26.79	79.24	4.32
	LISA (E + H)	61.49 ± 0.12	27.66 ± 0.07	77.93 ± 0.11	4.51 ± 0.27
MISTRAL-7B	**LISA**	**62.09 ± 0.10**	**29.76 ± 0.09**	**78.93 ± 0.08**	**4.85 ± 0.14**
	FT	61.70 ± 0.13	28.07 ± 0.12	78.85 ± 0.12	4.64 ± 0.12
	VANILLA	45.87	25.69	74.11	3.29
	LISA (E + H)	45.88 ± 0.12	25.82 ± 0.15	73.48 ± 0.22	4.63 ± 0.35
LLAMA-2-7B	**LISA**	**46.21 ± 0.12**	26.06 ± 0.08	**75.30 ± 0.11**	**4.94 ± 0.14**
	FT	45.66 ± 0.09	**27.02 ± 0.10**	75.06 ± 0.13	4.75 ± 0.16

To better understand the LISA and hyperparameter Sampling Layers γ, we conducted ablation experiments on Sampling Layers γ and learning rate η. The aim was to investigate the combined impact of these two variables on the LISA. Our experiments utilized the LLaMA-2-7B model, trained on the GSM8K dataset. We examined the effect of increasing the number of Sampling Layers γ while simultaneously decreasing the learning rate η.

Table 11: Compare LISA with different Sampling Layers γ and Learning Rate η, evaluate on GSM8K.

SAMPLING LAYERS γ	LEARNING RATE η			
	5×10^{-5}	2.5×10^{-5}	1.25×10^{-5}	6.25×10^{-6}
2	**15.77**	15.32	15.21	15.01
4	11.29	11.34	**15.87**	15.27
8	13.32	14.39	**16.30**	15.32
16	15.42	15.92	15.78	**16.57**

The Table 11, indicates that a higher number of sampling layers can enhance the model's effectiveness, provided the learning rate is adjusted appropriately. Specifically, the optimal performance is observed when the learning rate η is reduced in proportion to the increase in the number of sampling layers γ, demonstrating the delicate balance required between these two parameters to maximize LISA's efficacy.

A.4.2 Sampling Period K

Figure 9 displays the effects of varying sampling Period K on training a 7B-sized model using the 52K-entry Alpaca-GPT4 dataset. This graph contrasts loss curves for different sampling period K values: $K = 122$ (green line), $K = 25$ (red line), and $K = 13$ (blue line) across 122 training steps. The results indicate that although each K value results in distinct training trajectories, their convergence points are remarkably similar. This finding implies that for a 7B model trained on a 52K instruction conversation pair dataset, a sampling period of $K = 13$ is optimal for achieving the best loss curve and corresponding MT-Bench score radar graph.
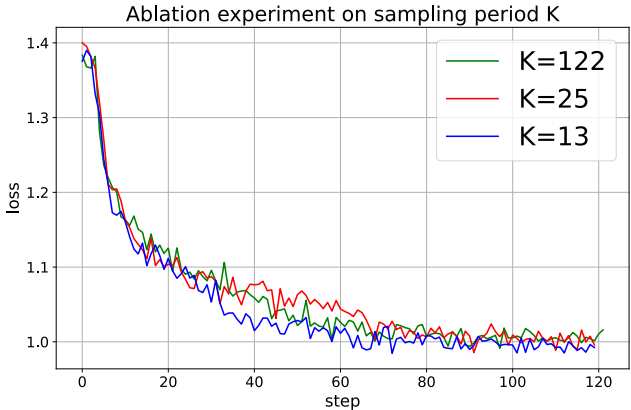

Figure 9: Comparison of loss curves for the sampling period K ablation experiment.

A.4.3 Sensitiveness to Randomness

LLaMA-2-7B on Alpaca-GPT4 with update step per sampling period $K = 13$, and sampling layers $\gamma = 2$, run three times with different random layer pick. Figure 10 shows that different random selections of layers slightly affect the training process but converge similarly. Despite initial fluctuations, the loss trends of three runs—distinguished by blue, green, and red lines—demonstrate that the model consistently reaches a stable state, underscoring the robustness of the training against the randomness in layer selection.
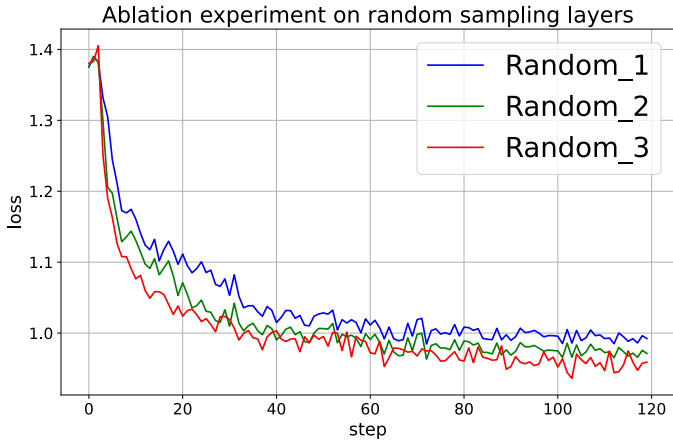

Figure 10: Comparison of loss curves for random variance ablation experiment, indicating the loss metric over steps.

Additionally, we also conduct experiments that analyze the impact of fixing randomly selected layers during training. The suggestion highlighted a need to evaluate the model's stability and performance under such conditions. To address this, we conducted further experiments using the LLaMA-2-7B model, maintaining identical hyperparameters and datasets as in our paper, and repeated the experiments three times to reduce variability from the random selection process.

Table 12: Compare LISA with fixed layers on LLaMA-2-7B, evaluate on MT-Bench.

METHOD	MT-BENCH ↑
LISA	**4.94**
LISA-FIX-1	4.62
LISA-FIX-2	4.60
LISA-FIX-3	4.67

The results are presented as table 12, with 'LISA-fix' denoting the experiments with randomly fixed layers and the appended number indicating the different selected seeds. The results are presented as table 12, with 'LISA-fix' denoting the experiments with randomly fixed layers and the appended number indicating the different selected seeds.

A.5   Performance with Early Exiting

According to previous works [83, 84], the early exiting strategy in LLMs is effective. We are interested in investigating whether the LISA algorithm will have a different impact when combined with the early exiting strategy. To explore this, we conducted a series of experiments using the LLaMA-2-7B model, focusing on various training methods and early exit points. The early exiting method is DoLa [83].

Table 13: GSM8K Scores for LLaMA-2-7B when LISA meets the early exiting strategy DoLa.

METHOD	GSM8K % ↑
VANILLA	15
VANILLA + DOLA	11
FT + DOLA	16
LISA + DOLA	17

We conducted all of our experiments using the LLaMA-2-7B model, selecting layers **[0, 8, 16, 32]** as early exit points for evaluation under four different training conditions: Vanilla (baseline), Vanilla with DoLa, Full Parameter Fine-Tuning (FT) with DoLa, and LISA with DoLa. The model used is the same one trained on the Alpaca-GPT4 dataset, as reported in section 4.2. This comprehensive approach enabled a detailed comparison of the model's performance and layer representation capabilities under various training methodologies. Due to time constraints, our experiments were conducted on a subset of 100 questions from the GSM8K test set, employing the same prompt as in the DoLa paper.

From the table 13, it can be seen that the LISA algorithm does not negatively affect the representation or performance of some layers of the model; instead, it contributes to some improvements in effectiveness.

A.6   Comparison of Evaluation Loss

Besides the training loss, we also care about how LISA performs on the validation dataset. So, we split the Alpaca-GPT4 dataset into train and validation sets, the ratio is $9:1$, and ensure there is no data overlap between these two sets. Then we use the same setting in the sec 4.2, training the LLaMA-2-7B on the Alpaca-GPT4 dataset with full parameter training (FT), LoRA, GaLore, and LISA. As Figure 11 shows, the trend in validation loss mirrors that observed in training loss, with LISA exhibiting no signs of overfitting. This consistency underlines LISA's robustness in maintaining performance across different dataset splits.
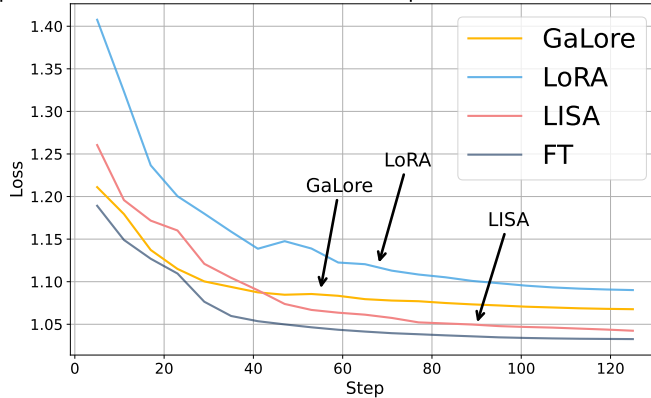

Figure 11: Validation loss comparison on the Alpaca-GPT4 dataset for LLaMA-2-7B, showing LISA, GaLore, LoRA, and FT strategies, with arrows indicating specific observations in the loss trends.

23

Table 15: The statistics of datasets. # TRAIN and # TEST denote the number of training and test samples respectively. The unit for OpenWebMath is the number of documents.

DATASET	# TRAIN	# TEST
ALPACA GPT-4[60]	52,000	-
MT-BENCH [67]	-	80
GSM8K [73]	7,473	1,319
MMLU [68]	-	14,079
AGIEVAL [69]	-	9316
WINOGRANDE [70]	-	44,000
PUBMEDQA [74]	211,269	1,000
OPENWEBMATH [72]	6.3M	-

A.7 Additional Observations of Layerwise Skewness

We conduct further weight norm experiments on Mistral-7 B to support our motivation that the bottom and top layers have a more significant impact on the output. Figure 12 provides similar observations as LLaMA-2-7B, where the bottom layer has a larger weight norm than other layers.
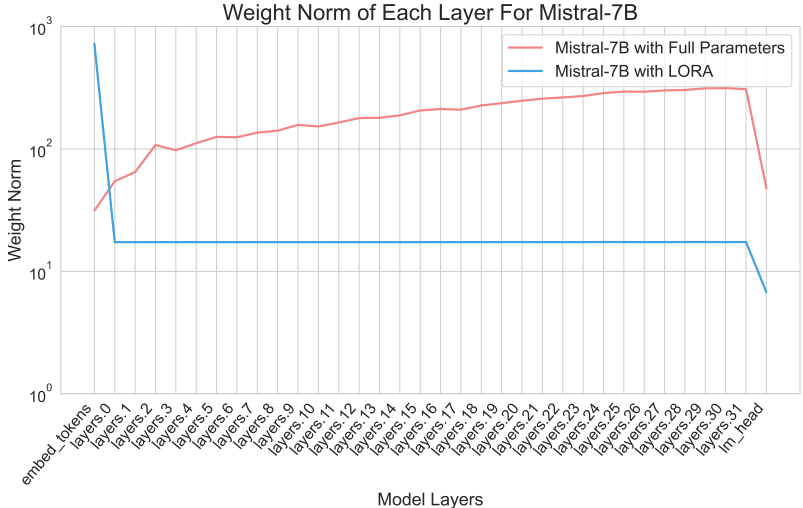

Figure 12: Layer-wise weight norms during training of Mistral-7B with LoRA and Full Parameters training.

B Training Setup and Hyperparameters

B.1 Training Setup

In our experiments, we employ the LMFlow toolkit [85]* for conducting full parameter fine-tuning, LoRA tuning, and LISA tuning. We set the epoch number to 1 for fine-tuning and continual pre-training scenarios. Additionally, we utilized DeepSpeed offload technology [63] to run the LLMs

Table 14: Baseline Model Specifications

MODEL NAME	# PARAMS	# LAYERS	MODEL DIM	# HEADS
TINYLLAMA	1.1 B	22	2048	32
MISTRAL-7B	7 B	32	4096	32
LLAMA-2-7B	7 B	32	4096	32
LLAMA-2-70B	70 B	80	8192	64

efficiently. All experiments were conducted on $8\times$ NVIDIA Ampere Architecture GPU with 48 GB memory. Table 14 and table 15 are the information covered in this paper.

---

*https://github.com/OptimalScale/LMFlow

Table 16: The hyperparameter search identified optimal settings for each method: FP (Full Parameter Training), LoRA, GaLore, and LISA.

	FP	LoRA		LISA		
Model	lr	lr	Rank	lr	γ	K
GPT2-Small	3×10^{-4}	6×10^{-4}	128	6×10^{-4}	2	3
TinyLlama	5×10^{-6}	5×10^{-5}	128	5×10^{-5}	2	10
Mistral-7B	5×10^{-6}	5×10^{-5}	128	5×10^{-5}	2	10
LLaMA-2-7B	5×10^{-6}	5×10^{-5}	128	5×10^{-5}	2	10
LLaMA-2-70B	5×10^{-6}	5×10^{-5}	128	5×10^{-5}	4	10

Our study explored a range of learning rates from 5×10^{-6} to 3×10^{-4}, applying this spectrum to Full Parameter Training, LoRA, and LISA methods. For LoRA, we adjusted the rank r to either 128 or 256 to vary the number of trainable parameters, applying LoRA across all linear layers. Regarding the number of sampling layers γ, our selections were guided by GPU memory considerations as reported in LoRA studies [9]; For the LISA algorithm, we selected $\gamma = 2$, and for experiments involving the 70B model, we opted for $\gamma = 4$. The sampling period (K), defined as the number of update steps per sampling interval, ranges from 1 to 50. This range was influenced by variables such as the size of the dataset, the batch size, and the number of training steps. To manage this effectively, we partitioned the entire training dataset into K segments, thereby enabling precise regulation of the training steps within each sampling period.

B.2 Continual Pre-training Dataset

We extracted a high-quality subset from OpenWebMath [72], using the '*Math_score*' attribute from the metadata as the metric for high-quality instances. The '*Math_Score*' represents the probability that a document is mathematical, and we set the threshold at 0.95. Finally, the number of tokens for this high-quality subset is 1.5 billion.

B.3 Hyperparameter search

We commenced our study with a grid search covering (i) learning rate, (ii) number of sampling layers γ, and (iii) sampling period K. Noting the effective performance of the LoRA method, we set the rank value to $r = 128$ or $r = 256$.

The optimal learning rate was explored within the range $\{5 \times 10^{-6}, 10^{-5}, 5 \times 10^{-5}, 6 \times 10^{-4}, 3 \times 10^{-4}\}$, applicable to full parameter training, LoRA, and LISA. For GaLore, we adhered to the official Transformers implementation[*], utilizing default parameters, with the learning rate matching that of the full parameter training.

Regarding the number of sampling layers γ, in alignment with Table 1, we selected values that matched or were lower than LoRA's GPU memory cost. Consequently, $\gamma = 2$ was predominantly used in the LISA experiments, while $\gamma = 4$ was chosen for the 70B model experiments.

For the sampling period K, we examined values within $1, 3, 5, 10, 50, 80$, aiming to maintain the model's update steps within a range of 10 to 50 per sampling period. This selection was informed by dataset size, batch size, and total training steps.

The comprehensive results of our hyperparameter search, detailing the optimal values for each configuration, are presented in Table 16.

C Licenses

For instruction following and domain-specific fine-tuning tasks, all the datasets, including Alpaca [61], GSM8k [73], MMLU [68], AGIEval [69] and PubMedQA [74] are released under MIT license.

---

[*]https://huggingface.co/blog/galore

WinoGrande [70] and MT-Bench [67] are under Apache-2.0 license. For GPT-4, the generated dataset is only for research purposes, which shall not violate its terms of use.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We proposed **L**ayerwise **I**mportance **S**ampled **A**damW (**LISA**), a promising alternative for LoRA, which applies the idea of importance sampling to different layers in LLMs and randomly freeze most middle layers during optimization. Experimental results show that with similar or less GPU memory consumption, LISA surpasses LoRA or even full parameter tuning in downstream fine-tuning tasks, where LISA consistently outperforms LoRA by over 11%-37% in terms of MT-Bench and multiple benchmarks.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Limitation section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: See Discussion section 5.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: See Experimental Result section 4

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: See Experiment section 4 and Appendix B.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Experiment section 4 and Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: See Experiment section 4 and Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Experiment section 4.1 and Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Yes, the research conducted in the paper conforms in every respect with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: NA.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: NA.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: See Licenses section C.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: NA.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: NA.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: NA.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.