LISA: Layerwise Importance Sampled AdamW

Anonymous ACL submission

Abstract

The machine learning community has witnessed impressive advancements since the first appearance of large language models (LLMs), yet their huge memory consumption has become a major roadblock to large-scale training. Parameter Efficient Fine-Tuning techniques such as Low-Rank Adaptation (LoRA) have been proposed to alleviate this problem, but their performance still fails to match full parameter training in most large-scale fine-tuning settings. Attempting to complement this deficiency, we investigate layerwise properties of LoRA on fine-tuning tasks and notice an uncommon skewness of weight norms across different layers. Utilizing this key observation, a surprisingly simple training strategy is discovered, which can outperform 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**dam (**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 in downstream fine-tuning tasks by a large margin on small models ($\leq 7B$), where LISA consistently outperforms LoRA by over 11%-37% in terms of MT-Bench scores. On large models, specifically Llama-2-70b, LISA achieves on-par or better performance than LoRA on MT-Bench, GSM8K, and PubMedQA, demonstrating its effectiveness across different domains.

1 Introduction

Large pre-trained decoder language models (LLMs), like the newly developed ChatGPT, excel in tasks such as writing documents, generating complex code, answering questions, and conducting human-like conversations (Ouyang et al., 2022). With LLMs being increasingly applied in diverse task domains, domain-specific data fine-tuning have emerged as key strategies to enhance their downstream capabilities (Raffel et al., 2020; Chowdhery et al., 2022; Rozière et al., 2023; OpenAI et al., 2023). Nevertheless, these methods are notably expensive, introducing major obstacles in developing large-scale models. Aiming to reduce the cost, Parameter-Efficient Fine-Tuning (PEFT) techniques, like adapter weights (Houlsby et al., 2019), prompt weights (Li and Liang, 2021), and LoRA (Hu et al., 2022) have been proposed to minimize the number of trainable parameters. Among them, LoRA can be considered one of the most widely adopted PEFT techniques, given its nice property of allowing the adaptor to be merged back to the base model. However, LoRA's superior performance in fine-tuning tasks has yet reached a point of universally surpassing full parameter fine-tuning in all settings (Ding et al., 2022; Dettmers et al., 2023). In particular, it has been observed that LoRA tends to falter with large-scale datasets during continual pre-training (Lialin et al., 2023), which raises into doubts the effectiveness of LoRA under those circumstances.

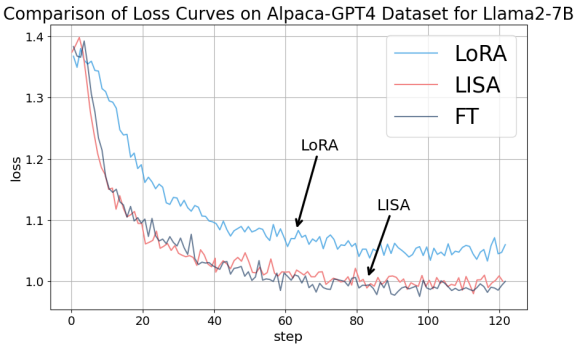

Figure 1: Comparison of training loss among different methods on the Llama2-7B model using the Alpaca GPT-4 dataset, including Full Parameter Training (FT), LoRA, and LISA.

To overcome this shortcoming, we delve into LoRA's training statistics in each layer, aspiring to

tell the major difference between LoRA and full-parameter fine-tuning. Surprisingly, we discover that LoRA's layerwise weight norms have generally a skewed distribution, where the bottom layer and/or the top layer occupy the majority of weights during the update, while the other self-attention layers only account for a small amount. This key observation inspires us to "sample" different layers by their importance, which exactly matches the idea of importance sampling (Kloek and Van Dijk, 1978; Zhao and Zhang, 2015).

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 outrageously large-scale training ($\geq$ 65B) with less or similar memory consumption as LoRA. Furthermore, in downstream fine-tuning 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 both LoRA and full parameter fine-tuning, which implies the varied importance of different layers in large-scale LLM training.
- We propose Layerwise Importance Sampled AdamW (LISA) algorithm, a simple optimization method that is 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 8%-36% in MT-Bench and exhibits much better convergence behaviors. LISA even outperforms full parameters training under certain settings. Similar performance gain is observed across different sized models (7B-70B) and different 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 (Vaswani et al., 2017). With the inception of models like BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019a), the approach shifted towards pre-training on extensive corpora, which led to significant performance enhancements in downstream fine-tuning tasks (Brown et al., 2020a; Raffel et al., 2020; Zhang et al., 2022; Scao et al., 2022; Almazrouei et al., 2023; Touvron et al., 2023a,b; Chiang et al., 2023; Biderman et al., 2023; Jiang et al., 2024). 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, calling for more resource-friendly methods in practice.

2.2 Parameter-Effieient Fine-Tuning

Parameter-Efficient Fine-Tuning (PEFT) methods fall into this category, which 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 (Hambardzumyan et al., 2021; Zhong et al., 2021; Han et al., 2021; Li and Liang, 2021; Qin and Eisner, 2021; Liu et al., 2021a), 2) Adapter methods (Houlsby et al., 2019; Hu et al., 2022; Diao et al., 2023c), and 3) Selective methods (Liu et al., 2021b,b; Li et al., 2023a). Prompt learning methods emphasize optimization of the input token or input embedding with fixed model parameters, which generally has the least training cost among all three types of approaches. Adapter methods normally introduce an auxiliary adapter module with much fewer parameters compared to the original model, where updates will only be applied to the adapter module during training.

Compared with them, selective methods have a closer relation to LISA, which focuses more on optimizing a fraction of the model's parameters without appending extra modules. Recent advances in this domain have introduced several notable techniques for efficient training through layer freezing. AutoFreeze (Liu et al., 2021b) offers an adaptive mechanism to automatically identify layers for freezing and accelerates the training process. FreezeOut (Brock et al., 2017), progressively freezes intermediate layers, resulting in significant training time reductions without notably affecting accuracy. The SmartFRZ (Li et al., 2023a) framework utilizes an attention-based predictor for layer

selection, substantially cutting down on 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 (Rajbhandari et al., 2020; Rasley et al., 2020) for LLMs.

2.3 Low-Rank Adaptation (LoRA)

In contrast, the Low-Rank Adaptation (LoRA) technique is much more popular in common practices of LLM training (Hu et al., 2022). By employing low-rank matrices, LoRA reduces the number of trainable parameters, thereby lessening the computational burden and memory cost. One of the key strengths of LoRA is its compatibility with models featuring linear layers, where the decomposed low-rank matrices can be merged back into the original model, allowing for efficient task-switching without changing the model architecture. As a result, LoRA can be seamlessly combined with other techniques, such as quantization (Dettmers et al., 2023) or Mixture of Experts (Gou et al., 2023). Despite these advantages, LoRA's performance is not universally comparable with full parameter fine-tuning. There have been tasks reported in Ding et al. (2022) that LoRA achieves much worse performance than full parameter training. This phenomenon is especially evident in large-scale pre-training settings (Lialin et al., 2023), which to the best of our knowledge, only full parameter training was adopted for successful open-source LLMs (Almazrouei et al., 2023; Touvron et al., 2023a,b; Jiang et al., 2023; Zhang et al., 2024; Jiang et al., 2024).

2.4 Large-scale Optimization Algorithms

Besides approaches focusing on changing model architectures, there has also been efforts improving the efficiency of optimization algorithm for LLMs. One branch of large-scale optimization algorithms is layerwise optimization, whose origin can be traced back to its first appearance decades ago, where Hinton et al. (2006) pioneered an effective layer-by-layer pre-training method for Deep Belief Networks (DBN), proving the benefits of sequential layer optimization. Bengio et al. (2007) furthered this concept by showcasing the advantages of a greedy, unsupervised approach for pre-training each layer in deep networks. For large batch settings, You et al. (2017, 2019) propose LARS and LAMB, providing improvements in generalization behaviors to avoid performance degradation incurred by large batch sizes. Neverthless, for common settings of LLMs, Adam (Kingma and Ba, 2014; Reddi et al., 2019) and AdamW (Loshchilov and Hutter, 2017) are currently still the dominant optimization methods.

Recently, there have also been other attempts to reduce the training cost of LLMs. For example, MeZO (Malladi et al., 2023) adopted zeroth order optimization that promises significant memory savings during training. However, it also introduced a huge performance drop in multiple benchmarks, particularly in complex fine-tuning scenarios. In terms of acceleration, Sophia (Liu et al., 2023a) incorporates clipped second-order information into the optimization, obtaining non-trivial speedup on LLM training tasks. The major downsides are its intrinsic complexity of Hessian estimation and unexplored performance in large-size models ($\geq$ 65B). To sum up, LoRA (Hu et al., 2022; Dettmers et al., 2023) with AdamW (Loshchilov and Hutter, 2017) is still the dominant paradigm for large-size LLM fine-tuning, which demands further studies and improvements.

3 Method

3.1 Motivation

To understand how LoRA achieves effective training with only a tiny portion of parameters, we carried out empirical studies on multiple models, specifically observing the weight norms across various layers while fine-tuning it on the Alpaca-GPT4 dataset (Peng et al., 2023). 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 (wte and wpe layers for GPT2) or the language model (LM) head layer exhibited significantly larger weight norms compared to intermediary layers in LoRA, often by a factor of hundreds. This phenomenon, however, was not salient under full-parameter training settings.

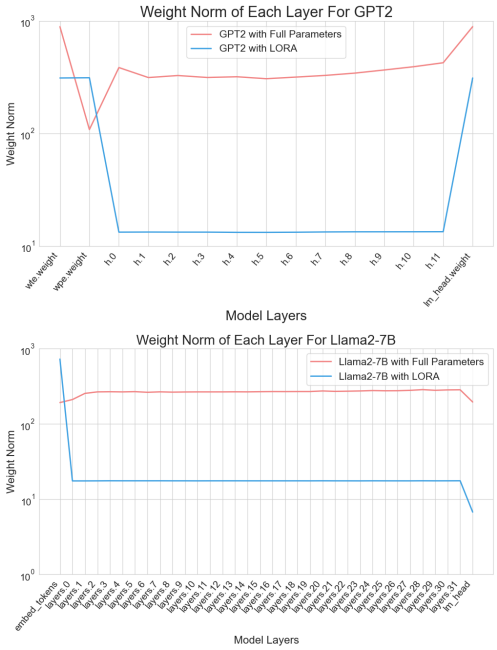

Figure 2: Layer-wise weight norms during Training of GPT2 and Llama2-7B Model with LoRA and Full Parameters training.

This observation indicates that the update emphasis of LoRA and full parameter training differs significantly, which can be caused by the difference in their learned knowledge. For example, in embedding layers, synonyms of tokens can be projected into the same embedding space, leading to similar embeddings. LoRA may capture this similarity and "group" them in the low-dimension space, allowing frequent features to be promptly identified and optimized. The price is that LoRA's intrinsic low-rank space may restrict its representation power. Despite other possible explanations, one fact remains clear: *LoRA values layerwise importance differently from full parameter tuning*.

3.2 Layerwise Importance Sampled AdamW (LISA)

By exploiting the aforementioned discovery, 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 also emulate its fast learning process. Intuitively, given the same magnitude of 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 (Kloek and Van Dijk, 1978; Zhao and Zhang, 2015), 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 instead

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

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 unfreezed layers during optimization. Intuitively, γ serves as a compensation factor to bridge the difference between LoRA and full parameter tuning, letting LISA to 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 bound the maximum number of unfreezed 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, sampling probability $\{p_\ell\}_{\ell=1}^{N_L}$, initial learning rate η_0

```
for i ← 0 to T/K − 1 do
    for ℓ ← 1 to N_L do
        if U(0,1) > p_ℓ then
            Freeze layer ℓ
        end if
    end for
    Run AdamW for K iterations with {η_t}_{t=ik}^{ik+k−1}
end for
```

4 Experimental Settings

4.1 Datasets and Evaluation Metrics

In our experiments, we focused on two primary training scenarios for Large Language Models (LLMs): *Instruction Following Fine-Tuning* and *Domain Specific Fine-Tuning*.

During the Instruction Following Fine-Tuning phase, we utilized the Alpaca GPT-4 dataset (Peng et al., 2023)[1], which consists of 52k instruction-following instances generated by GPT-4 (OpenAI et al., 2023) with inputs from Alpaca (Taori et al., 2023). This dataset aims to improve a model's

[1]https://instruction-tuning-with-gpt-4.github.io/

ability in conversational understanding and instruction following. The effectiveness of fine-tuning was evaluated with MT-Bench (Zheng et al., 2023), featuring 80 high-quality, multi-turn questions designed to assess LLMs. MT-Bench employs GPT-4 to judge the models' responses across a variety of tasks including Writing, Roleplay, Reasoning, Math, Extraction, STEM, and Humanities, thus offering a comprehensive framework for evaluating fine-tuning approaches.

In the Domain-Specific Fine-Tuning phase, our efforts were concentrated on the Llama2-70B model, focusing on mathematics and medicine to gauge the LISA algorithm's performance. The GSM8K dataset (Cobbe et al., 2021), comprising 7473 training instances and 1319 test instances, was utilized for the mathematics domain. For the medical domain, we selected the PubMedQA dataset (Jin et al., 2019), which includes 211.3k artificially generated QA training instances and 1k test instances. The statistics of these datasets are summarized in Table 1.

DATASET	# TRAIN	#TEST
ALPACA GPT-4(PENG ET AL., 2023)	52,000	-
MT-BENCH (ZHENG ET AL., 2023)	-	80
GSM8K (COBBE ET AL., 2021)	7,473	1,319
PUBMEDQA (JIN ET AL., 2019)	211,269	1,000

Table 1: The statistics of datasets used in this paper. # TRAIN and # TEST denote the number of training data and test data, respectively.

4.2 Models and Baseline

We conducted a comparative analysis of the LISA and LoRA methods, focusing on Instruction Following Fine-Tuning and Domain Specific Fine-Tuning scenarios.

Instruction Following Fine-Tuning aims to bridge the gap between the inherent next-word prediction objective of Large Language Models (LLMs) and the practical user objective of following instructions both helpfully and safely. Unlike the traditional training of LLMs, which focuses on minimizing contextual word prediction errors across extensive corpora, this approach directs the model to prioritize user instructions, enhancing its applicability and safety in user interactions (Radford et al., 2019b; Brown et al., 2020b; Fedus et al., 2021; Rae et al., 2021; Thoppilan et al., 2022).

Domain Specific Fine-Tuning offers a computationally efficient method for Large Language Models (LLMs) to quickly adapt to a specific domain, compensating the need for extensive pre-training or significant architectural modifications. This approach effectively constrains the model's outputs to align with the desired response patterns and domain-specific knowledge (Zhang et al., 2023; Li et al., 2023c; Liu et al., 2023b; Toma et al., 2023).

In our experiments, we assessed five baseline models: GPT2-Small (Radford et al., 2019a), TinyLlama (Zhang et al., 2024), Phi-2 (Li et al., 2023b), Mistral-7B (Jiang et al., 2023), Llama2-7B, and Llama2-70B (Touvron et al., 2023b). These models, varying in size with parameters ranging from 124M to 70B, were selected to provide a diverse representation of decoder models. Detailed specifications of each model are provided in Table 2.

Model Name	Total Params	Layers	Model Dim	Heads
GPT2-Small	124 M	12	768	12
TinyLlama	1.1 B	22	2048	32
Phi-2	2.7 B	32	2560	32
Mistral-7B	7 B	32	4096	32
LLaMA2-7B	7 B	32	4096	32
LLaMA2-70B	70 B	80	8192	64

Table 2: Baseline Model Specifications

4.3 GPU Memory Efficiency

To demonstrate the memory efficiency of LISA, showcasing that it has a comparable or lower memory cost than LoRA, we conducted peak GPU memory experiments. When using LISA in the training process, the majority of trainable parameters and activation memory remain frozen, resulting in significant GPU memory savings. This section presents a detailed analysis of GPU memory usage for both LISA and LoRA implementations. Our focus is on two key hyperparameters: LoRA's rank and LISA's number of activation layers. For our memory efficiency experiments, a mini-batch size of 1 was consistently used across all five models, deliberately excluding other GPU memory-saving techniques such as gradient checkpointing, offloading (Ren et al., 2021), and flash attention (Dao et al., 2022; Dao, 2023). Conducted on four A800 GPUs with 80GB of memory each, the outcomes are summarized in Table 3.

Upon examining table 3, it is evident that the LISA configuration, particularly when enhanced with both the embedding layer (E) and two addi-

tional layers (E+H+2L), demonstrates a considerable reduction in GPU memory usage when fine-tuning the Llama2-70B 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 light of these findings, the subsequent experiments within this paper will employ the LISA E+H+2L configuration as a standard, with the aim to leverage its superior memory efficiency.

4.4 Training Setup and Hyperparameters

In our experiments, we employed the LMFlow toolkit (Diao et al., 2023a)[2] for conducting full parameter fine-tuning, LoRA tuning, and LISA tuning, setting the epoch to 1 for both fine-tuning and continual pretraining scenarios. Additionally, we utilized DeepSpeed offload technology (Ren et al., 2021) to efficiently run the LLMs. All experiments were conducted on an NVIDIA A40 GPU*8 server.

In our study, we 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 (Hu et al., 2022); For 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, ranged from 1 to 50. This range was influenced by variables such as the size of the dataset, the batch size, and the total 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.

The comprehensive results of this hyperparameter search, detailing optimal values for each configuration, are thoroughly documented in Appendix A. The hyperparameters identified through this process were subsequently applied across all our fine-tuning experiments.

[2]https://github.com/OptimalScale/LMFlow

5 Experimental Results

In our preliminary experiments, we employ both instruction following and domain specific fine-tuning settings to assess the LISA performance in LLM training.

Instruction Following Fine-Tuning In this section, we present the results of instruction following fine-tuning for the TinyLlama, Mistral-7B, and Llama2-7B models. Table 4 offers a comprehensive evaluation of three fine-tuning methods—Full Parameter Fine-Tuning (FT), Low-Rank Adaptation (LoRA), 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 clearly demonstrate LISA's superior performance, which surpasses LoRA 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 be beneficial for tasks involving memorization, while LoRA partially favors reasoning tasks.

Large scale LLM Fine-Tuning This section outlines the methodology employed for both instruction-following and domain-specific fine-tuning using the Llama2-70B model. The instruction-following fine-tuning leveraged the dataset discussed in Section 5, while for domain-specific fine-tuning, the medical and mathematics domains were adopted. In the medical domain, the model was fine-tuned with the PubMedQA training dataset (Jin et al., 2019) and evaluated against its test set. For the mathematics domain, the GSM8K dataset (Cobbe et al., 2021) served both for training and subsequent performance evaluation on its test set.

Table 5 summarizes the experimental results for the Llama2-70B model across these domains. Evaluation on the PubMedQA dataset (Jin et al., 2019) utilized a 5-shot prompt setting, while the GSM8K dataset (Cobbe et al., 2021) assessment was conducted using a Chain of Thought (CoT) prompt, as suggested by recent studies (Wei et al., 2022; Diao et al., 2023b).

Figure 3 visualizes the MT-Bench results for the Llama2-70B model. The radar chart highlights

	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
PHI-2	28G	14.3G	15.5G	18G	13.5G	14.6G	15G
MISTRAL-7B	59G	23G	26G	28G	21G	23G	24G
LLAMA2-7B	59G	23G	26G	28G	21G	23G	24G
LLAMA2-70B*	OOM	79G	OOM	OOM	-	75G	79G

Table 3: The chart illustrates peak GPU memory consumption for various model architectures and configurations, highlighting differences across models. In the table, the LISA configuration is specifically labeled: "E" denotes the embedding layer, "H" represents the language model head layer, and "2L" indicates two additional standard layers. *: Model parallelism is apply for the 70B model.

	MT-BENCH							
MODEL & METHOD	WRITING	ROLEPLAY	REASONING	MATH	EXTRACTION	STEM	HUMANITIES	AVG. ↑
TINYLLAMA (VANILLA)	1.05	2.25	1.25	1.00	1.00	1.45	1.00	1.28
TINYLLAMA (FT)	3.27	3.95	1.35	1.33	1.73	2.69	2.35	2.38
TINYLLAMA (LORA)	2.77	4.05	1.35	1.40	1.00	1.55	2.15	2.03
TINYLLAMA (**LISA**)	**3.30**	4.40	2.65	1.30	1.75	**3.00**	**3.05**	**2.78**
MISTRAL-7B (VANILLA)	5.25	3.20	4.50	2.70	6.50	6.17	4.65	4.71
MISTRAL-7B (FT)	5.50	4.45	5.45	3.25	5.78	4.75	5.45	4.94
MISTRAL-7B (LORA)	5.30	4.40	4.65	3.30	5.50	5.55	4.30	4.71
MISTRAL-7B (**LISA**)	**6.84**	3.65	5.45	2.75	5.65	**5.95**	**6.35**	**5.23**
LLAMA2-7B (VANILLA)	2.75	4.40	2.80	1.80	3.20	5.25	4.60	3.54
LLAMA2-7B (FT)	5.55	6.45	3.60	2.00	4.70	6.45	7.50	5.10
LLAMA2-7B (LORA)	6.30	5.65	4.05	1.45	4.17	6.20	6.20	4.86
LLAMA2-7B (**LISA**)	**6.55**	6.90	3.45	2.16	4.50	**6.75**	**7.65**	**5.42**

Table 4: Comparison of Language Model Fine-Tuning Methods on the MT-Bench score.

	MT-BENCH	GSM8K	PUBMEDQA
VANILLA	5.69	54.8	83.0
FT	6.66	67.1	90.8
LORA	6.52	59.4	90.8
LISA	**7.05**	**61.1**	**91.6**

Table 5: This table shows MT-Bench, GSM8K, and PubMedQA score for Llama2-70B with three fine-tuning methods.

the model's performance in various domains, particularly LISA's superiority over all methods in aspects of Writing, Roleplay and STEM. On top of that, LISA displays consistently higher performance than LoRA on all subtasks, underscoring LISA's effectiveness across diverse tasks. The chart also contrasts the yellow LoRA line with the purple Vanilla line, revealing that in large models like the 70B, LoRA does not perform as well as expected, showing only marginal improvements on specific aspects. More fine-tuning experiments result in the Appendix B.1.

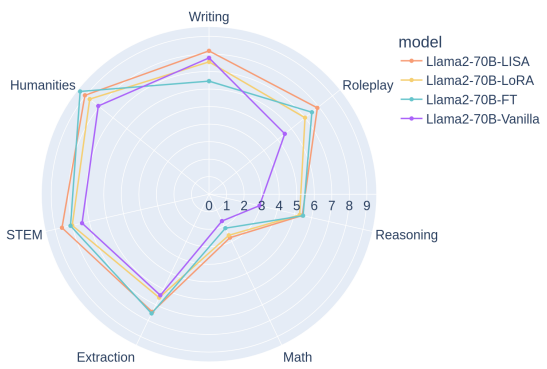

Figure 3: Performance comparison of Llama2-70B models using various tuning methods across MT-Bench.

6 Ablation Studies

6.1 Hyperparameters of LISA

The number of sampling layers γ and sampling period K are the two key hyperparameters of

LISA. To obtain intuitive and empirical guidance of those hyperparameter choices, we conduct ablation studies using TinyLlama (Zhang et al., 2024) and Llama2-7B (Touvron et al., 2023b) 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 — the number of updates per sampling interval — values were chosen from $\{1, 5, 25, 122\}$, with 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 C.

Models	MT-Bench Score	Sampling Layers γ	Sampling Frequency K
TinyLlama	2.59	8	1
	2.81	8	5
	2.73	2	5
	2.64	2	25
	2.26	2	122
Llama2-7B	4.94	8	1
	5.11	8	5
	4.91	2	5
	4.88	2	25
	4.64	2	122

Table 6: The MT-Bench scores derived from different LISA hyperparameters combination. All settings adopt learning rate $\eta_0 = 10^{-5}$.

6.2 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, Llama2-7B, 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 5. 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 when compared with the performance gains over baseline methods.

Seed	TinyLlama	Llama2-7B	Mistral-7B
1	2.65	5.42	5.23
2	2.75	5.31	5.12
3	2.78	5.35	5.18

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

7 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 from observations of LoRA's skewed weight norm distribution, a simple and memory-efficient freezing paradigm is introduced for LLM training, which 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 (Dettmers et al., 2023), where we intend to conduct further experiments to verify its performance.

The potential risk of this approach is that the theoretical convergence guarantees of LISA are still not available, which leads to uncertainty about the algorithm's convergence behavior. We intend to fill the gap in future works.

References

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.

2023. The falcon series of language models: Towards open frontier models.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. *Greedy Layer-Wise Training of Deep Networks*, page 153–160.

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

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020a. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff 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. 2020b. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

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. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

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. 2022. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

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

Tri Dao. 2023. FlashAttention-2: Faster attention with better parallelism and work partitioning.

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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. Association for Computational Linguistics.

Shizhe Diao, Rui Pan, Hanze Dong, Ka Shun Shum, Jipeng Zhang, Wei Xiong, and Tong Zhang. 2023a. Lmflow: An extensible toolkit for finetuning and inference of large foundation models. *arXiv preprint arXiv:2306.12420*.

Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023b. Active prompting with chain-of-thought for large language models.

Shizhe Diao, Tianyang Xu, Ruijia Xu, Jiawei Wang, and Tong Zhang. 2023c. Mixture-of-domain-adapters: Decoupling and injecting domain knowledge to pre-trained language models memories. *arXiv preprint arXiv:2306.05406*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning:

A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, KaShun SHUM, and Tong Zhang. 2023. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*.

William Fedus, Barret Zoph, and Noam M. Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23:120:1–120:39.

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

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. 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. Association for Computational Linguistics.

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

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

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

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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

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. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. 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.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

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

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023b. Textbooks are all you need ii: phi-1.5 technical report.

Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, and You Zhang. 2023c. Chatdoctor: A medical chat model fine-tuned on llama model using medical domain knowledge. *ArXiv*, abs/2303.14070.

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

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. 2023a. Sophia: A scalable stochastic second-order optimizer for language model pretraining. *arXiv preprint arXiv:2305.14342*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021a. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

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

Zheng Liu, Aoxiao Zhong, Yiwei Li, Longtao Yang, Chao Ju, Zihao Wu, Chong Ma, Peng Shu, Cheng Chen, Sekeun Kim, Haixing Dai, Lin Zhao, Dajiang Zhu, Jun Liu, Wei Liu, Dinggang Shen, Xiang Li, Quanzheng Li, and Tianming Liu. 2023b. Radiology-gpt: A large language model for radiology.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

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

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. 2023. Gpt-4 technical report.

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. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

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

Guanghui Qin and Jason Eisner. 2021. 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. Association for Computational Linguistics.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019b. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John

Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

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

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

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. 2019. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*.

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

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. 2023. Code llama: Open foundation models for code.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347.

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

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Augustin Toma, Patrick R Lawler, Jimmy Ba, Rahul G Krishnan, Barry B Rubin, and Bo Wang. 2023. Clinical camel: An open-source expert-level medical language model with dialogue-based knowledge encoding. *arXiv preprint arXiv:2305.12031*.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, AidanN. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Neural Information Processing Systems,Neural Information Processing Systems*.

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

Yang You, Igor Gitman, and Boris Ginsburg. 2017. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*.

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

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

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

Peilin Zhao and Tong Zhang. 2015. Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*, pages 1–9. PMLR.

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. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. 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. Association for Computational Linguistics.

A 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 either $r = 128$ or $r = 256$.

The optimal learning rate was explored within the range $\{5\times10^{-6}, 10^{-5}, 5\times10^{-5}, 6\times10^{-4}, 3\times10^{-4}\}$, applicable to full parameter training, LoRA, and LISA.

Regarding the number of sampling layers γ, in alignment with Table 3, 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 instance. This selection was informed by factors such as 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 8.

	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
Phi-2	5×10^{-6}	5×10^{-5}	128	5×10^{-5}	2	3
Mistral-7B	5×10^{-6}	5×10^{-5}	128	5×10^{-5}	2	3
Llama2-7B	5×10^{-6}	5×10^{-5}	128	5×10^{-5}	2	3
Llama2-70B	5×10^{-6}	5×10^{-5}	128	5×10^{-5}	4	50

Table 8: The hyperparameter search identified optimal settings for each method: FP (Full Parameter Training), LoRA, and LISA. For the sampling period K, these settings differ.

B Additional Experimental Settings

B.1 Fine-tuning

Table 9 provides detailed MT-Bench scores for the Llama2-70B model as discussed in Section 5.

Initially, we conducted fine-tuning experiments on the GPT2-Small model. As illustrated in Figure 4, the loss curve distinctly shows that LISA exhibits superior convergence capabilities compared to LoRA, highlighting its effectiveness in the fine-tuning process.

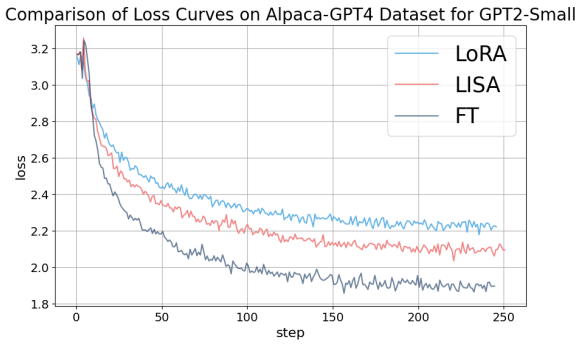

Figure 4: Comparison of loss curves for three different training approaches on the Alpaca-GPT4 dataset for GPT2-Small.

In this extended study, we focused on the TinyLlama model, specifically the TinyLlama-1.1B-intermediate-step-1431k-3T variant [3]. This model variant, pretrained on 3 trillion tokens, has not undergone training on any instruction-following dataset nor been aligned with Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022; Schulman et al., 2017; Dong et al., 2023). We standardized the learning rate, LoRA rank, and LISA activation layer settings to match those of Llama2-7B. The loss graph in the Figure 5, the graph depicts a sharper decline in loss for FT, suggesting a more rapid convergence compared to the other methods. LISA and LoRA display a similar trend initially, but LISA achieves a better loss reduction over time. The radar graph in Figure 5 presents a comparative analysis of instruction-following capabilities. This chart reveals that LISA achieves performance nearly equivalent to full parameter fine-tuning across various domains, with notable enhancements in specific areas like STEM and Reasoning. Conversely, LoRA tuning demonstrated subpar performance in several domains, including Writing, STEM, and Roleplay.

[3] https://huggingface.co/TinyLlama/TinyLlama-1.1B-intermediate-step-1431k-3T

Model & Method	MT-Bench							
	Writing	Roleplay	Reasoning	Math	Extraction	STEM	Humanities	Avg. ↑
Llama2-70B(Vanilla)	7.77	5.52	2.95	1.70	6.40	7.42	8.07	5.69
Llama2-70B(FT)	6.45	7.50	5.50	2.15	7.55	8.10	9.40	6.66
Llama2-70B(LoRA)	7.55	7.00	5.30	2.60	6.55	8.00	8.70	6.52
Llama2-70B(**LISA**)	**8.18**	**7.90**	**5.45**	**2.75**	**7.45**	**8.60**	**9.05**	**7.05**

Table 9: Comparison of three Fine-Tuning Methods for Llama2-70B on the MT-Bench score. Mean score over three seeds are reported.

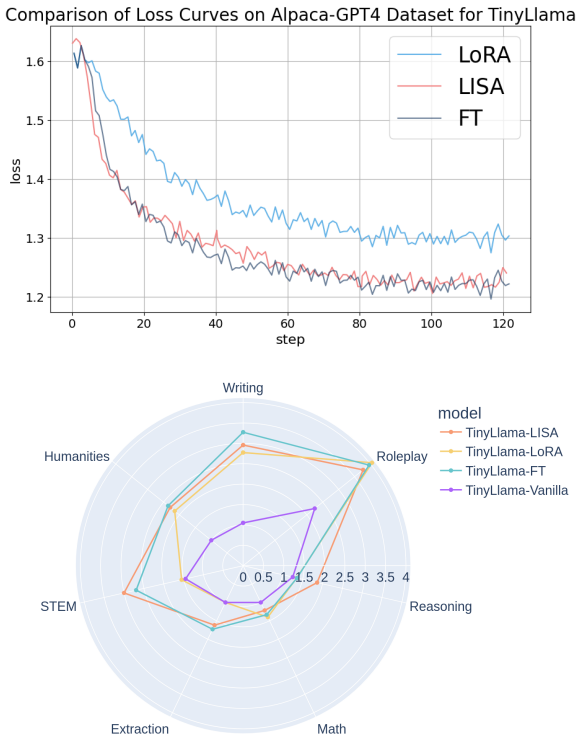

Figure 5: Comparison of loss curves and MT-Bench radar graphs for three different training approaches on the Alpaca-GPT4 dataset for TinyLlama.

We then shifted our focus to the Phi2 model. The loss curves presented in Figure 6 compare the fine-tuning, LoRA, and LISA methods. Here, LISA demonstrates a convergence rate comparable to full parameter fine-tuning, surpassing LoRA.

Our experiments extended to the Mistral-7B-v0.1 model [4], which is based on the Llama architecture but trained on a high-quality corpus. The loss curve and MT-Bench results for this model, fine-tuned on the Alpaca-GPT4 dataset, are shown in Figure 7. We observed a sharper decrease in loss for the FT method, indicating a faster convergence. LISA, in comparison to LoRA, shows better performance

[4]https://huggingface.co/mistralai/Mistral-7B-v0.1

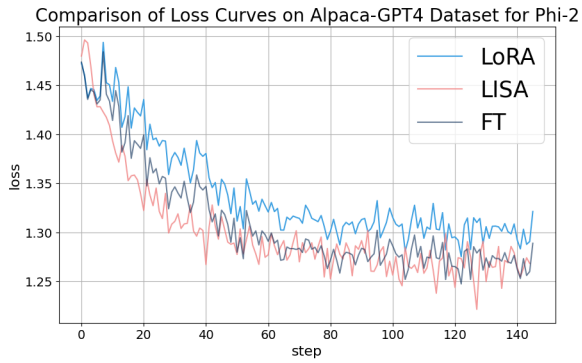

Figure 6: Comparison of loss curves and MT-Bench radar graphs for three different training approaches on the Alpaca-GPT4 dataset for Phi-2.

over time. The radar graph reveals that the vanilla Mistral-7B excelled in STEM and Extraction, but fine-tuning led to a decline in these areas, a phenomenon known as the "Alignment Tax" (Ouyang et al., 2022). 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," whereas LoRA shows a significant decrease in STEM performance, albeit excelling in Roleplay.

Finally, our experiments included the Llama2-70B model. The training loss curve versus step graph for this model is depicted in Figure 8. This part of the study aimed to further understand the fine-tuning dynamics on an even larger scale model.

C Ablation Experiments

C.1 Sampling Layers γ

We conducted an ablation study on the Llama2-7B 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 9 depicts

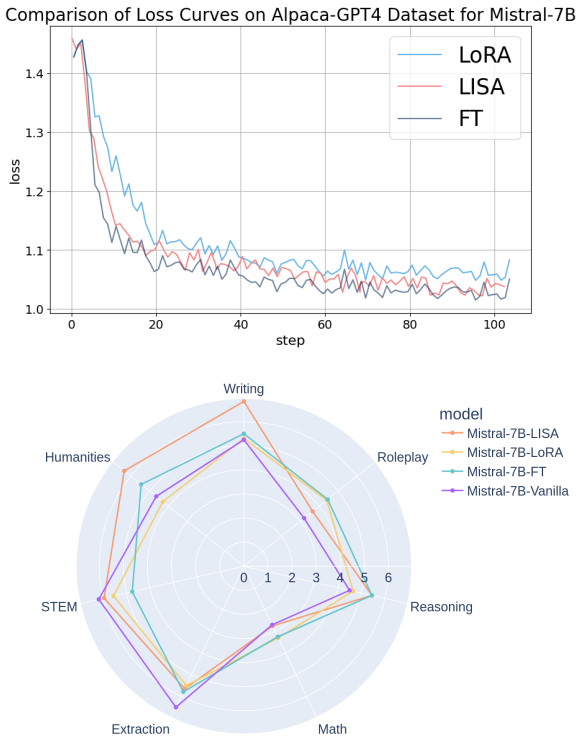

Figure 7: Comparison of loss curves and MT-Bench radar graphs for three different training approaches on the Alpaca-GPT4 dataset for Mistral-7B.

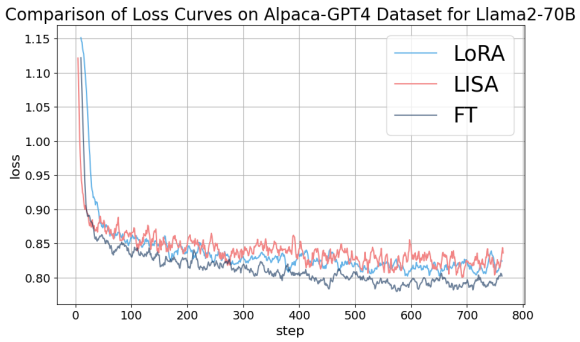

Figure 8: Comparison of loss curves for three different training approaches on the Alpaca-GPT4 dataset for Llama2-70B, indicating the loss metric over steps.

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), over a span of 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. The radar graph of the MT-Bench score also indicates that the $\gamma 8$ configuration yields the best conversational ability of the model in this ablation experiment.

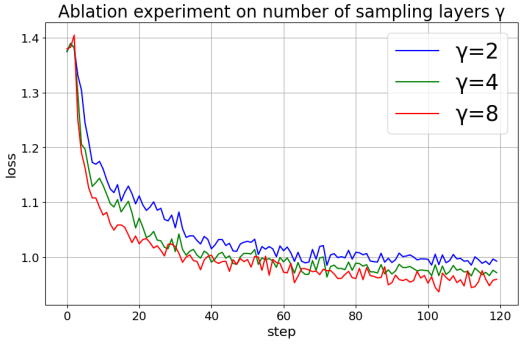

Figure 9: Comparison of loss curves for the γ (number of sampling layers) ablation experiment.

C.2 Sampling Period K

Figure 10 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 dataset of 52K instruction conversation pairs, a sampling period of $K = 13$ is optimal for achieving the best loss curve and corresponding MT-Bench score radar graph.

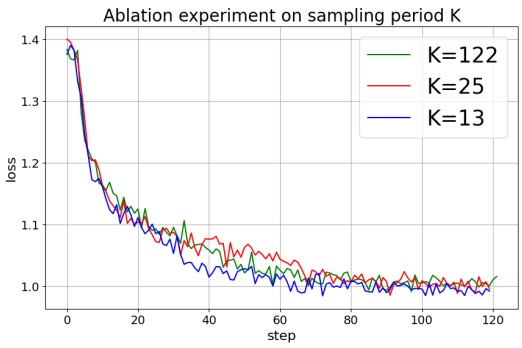

Figure 10: Comparison of loss curves for the sampling period K ablation experiment.

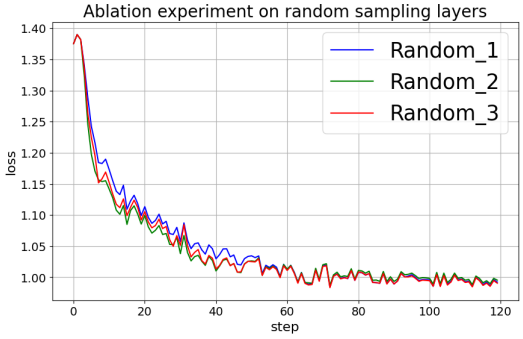

Figure 11: Comparison of loss curves for random variance ablation experiment, indicating the loss metric over steps.

C.3 Sensitiveness to Randomness

Llama2-7B on Alpaca-GPT4 with update step per sampling period $K = 13$, and sampling layers $\gamma = 2$, run 3 times with different random layer pick. The figure 11 showing 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.

D Licenses

For instruction following and domain-specific fine-tuning tasks, all the datasets including Alpaca (Taori et al., 2023), GSM8k (Cobbe et al., 2021) and PubMedQA (Jin et al., 2019) are released under MIT license. For GPT-4, the generated dataset is only for research purposes, which shall not violate its terms of use.