Efficient LLM Adaptation with Forward-Only Passes

Baichuan Huang[*] Lund University, Sweden

Ananth Balashankar[†] Google DeepMind, USA

Amir Aminifar Lund University, Sweden

Abstract

Large language model (LLM) inference serving is undergoing rapid growth and large-scale deployment, motivating us to rethink how the inference process itself can be leveraged to enable efficient task-specific LLM adaptation. In this paper, we propose a forward-only approach for efficient LLM adaptation with forward-only passes in LLM inference serving. We exploit the angle concentration of activations induced by each singular value decomposition (SVD) component to measure its contribution—*dispersed* or *concentrated*—to the angle concentration of the hidden states. Based on this, our forward-only search then efficiently identifies the weight matrix with the highest dispersion that merits rank reduction. We then selectively remove the higher-order components and retain the lower-order components in SVD. Empirical results across diverse datasets demonstrate the competitive accuracy of our forward-only approach, while theoretical analysis shows lower peak memory and greater speedup than the gradient-based approach and greater speedup than the exhaustive search approach. The experiments further present the robustness and generalization of our forward-only approach to various LLMs with up to 57B parameters. The code is included in the supplementary material.

1 Introduction

Artificial intelligence (AI) serving is emerging as a key driver of growth in energy consumption as AI systems are deployed at scale [Bashir et al., 2024, Maslej et al., 2025, Thunder Said Energy, 2026], particularly for Transformer-based large language models (LLMs) inference serving [Elsworth et al., 2025, Epoch AI, 2026]. Large-scale LLM inference generated from real user interactions during LLM serving motivates us to rethink how the inference process itself can be leveraged to improve the serving system, particularly enabling efficient task-specific LLM adaptation.

The mainstream LLM adaptation is based on tuning techniques (for instance, parameter-efficient fine-tuning (PEFT) [Hu et al., 2022, Houlsby et al., 2019], prompt tuning [Lester et al., 2021], etc.) and knowledge augmentation techniques (for instance, retrieval-augmented generation (RAG) [Lewis et al., 2020], prompt engineering [White et al., 2023], etc.). The knowledge augmentation techniques primarily operate by injecting knowledge and are orthogonal to modifying the internal weights during LLM inference. The tuning techniques usually rely on backward passes, incurring extra resource overhead compared to inference itself. On the contrary, the emerging forward-only learning [Hinton, 2022, Ren et al., 2023, Zhang et al., 2024, Huang and Aminifar, 2025b, Huang et al., 2026b], which relies solely on forward passes, is becoming a promising approach for adaptation in LLM inference.

In this paper, we aim to achieve efficient LLM adaptation with forward-only passes during LLM inference serving. We exploit the singular value decomposition (SVD) in LLM compression techniques, based on the empirical observation that selectively removing the higher-order components of weight matrices in SVD can improve downstream accuracy [Sharma et al., 2024, Sreeram et al.,

[*]baichuan.huang@eit.lth.se

[†]contributed in an advisory role.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

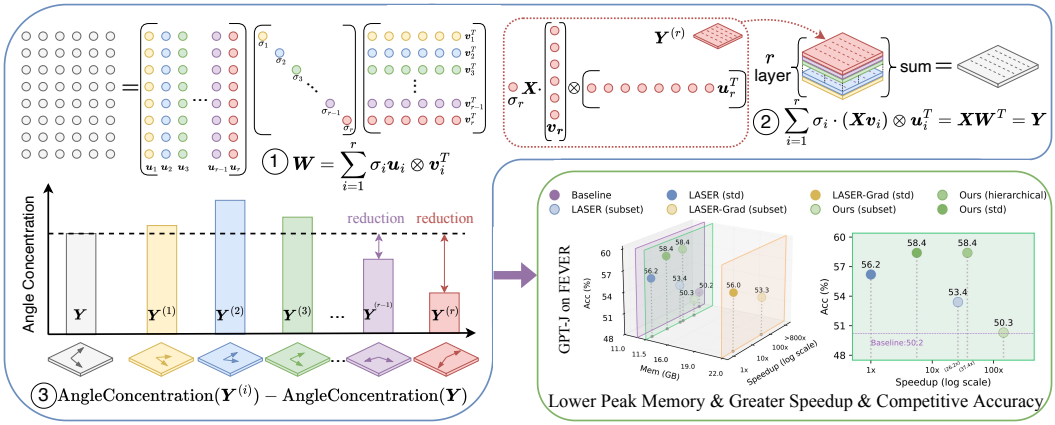


Figure 1: Efficient LLM adaptation in LLM inference serving. **Our forward-only search** contains ① SVD with multiple SVD components; ② The sum of activations from all SVD components recovers the standard forward pass; ③ Forward-only passes measure the angle concentration to identify which weight matrix has the highest dispersion that should be applied with rank reduction. **Results** show that our forward-only approach achieves lower peak memory and greater speedup than the gradient-based approach (LASER-Grad [Sreeram et al., 2025]), as well as greater speedup and competitive accuracy compared to the exhaustive search approach (LASER [Sharma et al., 2024]).

2025].[3] We propose the forward-only search to speed up the identification of the selected weight matrix, by the difference in angle concentration between activations induced by SVD components and the hidden states. The angle concentration of activations induced by each SVD component is to measure its contribution—*dispersed* or *concentrated*—to the angle concentration of the hidden states, as shown in Fig. 1. If the angle concentration of activations induced by one SVD component is more dispersed than the angle concentration of the hidden states, the corresponding SVD component tends to be removed. Conversely, if more concentrated, the SVD component tends to be retained. Our forward-only search relies solely on forward passes of LLM inference, hence eliminating the need for backpropagated gradients in Sreeram et al. [2025] and avoiding the computational burden of exhaustive search that sweeps all the weights in Sharma et al. [2024]. Then, we investigate the rank-reduction strategy across different sizes of validation datasets to further improve the computational efficiency. The contributions of our paper are shown below:

- We propose a forward-only approach for efficient LLM adaptation in LLM inference. By exploiting the difference in angle concentration between activations induced by SVD components and the hidden states, our forward-only search identifies which SVD components should be reduced, and further identifies which weight matrix merits rank reduction.
- We demonstrate that our forward-only approach achieves lower peak memory and greater speedup than the gradient-based approach, as well as greater speedup and competitive accuracy compared to the exhaustive search approach, across diverse datasets (including knowledge reasoning, knowledge intervention, multi-hop reasoning, fact verification, fairness and bias evaluation, hallucination detection, and information retrieval). The extended experiments further show the effectiveness and robustness of our forward-only approach, as well as its ability to enhance the fine-tuning process. Additionally, we conduct extensive experiments on various LLMs with up to 57B parameters, including LLaMA2, DeepSeekV1, Mistral, Qwen2 and its mixture-of-experts (MoE) version, and the multimodal Gemma 4. The results show the strong generalization of our forward-only approach.

2 Our Forward-Only Approach

Our forward-only approach aims to identify which weight matrix in LLMs should be applied with rank reduction and how to apply it using SVD, for effective adaptation in LLM inference serving.

[3]Unlike other LLM compression techniques [Sanh et al., 2019, Dettmers et al., 2023, Frantar and Alistarh, 2023, Lin et al., 2025] designed to maintain downstream accuracy, the standard SVD is expected to enhance downstream accuracy by removing the higher-order components (smaller singular values) and retaining lower-order components (bigger singular values) as suggested by recent studies [Sharma et al., 2024, Sreeram et al., 2025].

Preliminary. Given a weight matrix $\boldsymbol{W} \in \mathbb{R}^{h\times d}$, at a given layer of a pre-trained LLM, we denote the hidden states of input as $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, the hidden states of output as $\boldsymbol{Y} \in \mathbb{R}^{n\times h}$, where n is the sequence length, d is the size of input, and h is the size of output. Therefore, the forward pass of this given layer is $\boldsymbol{X}\boldsymbol{W}^T = \boldsymbol{Y}$. Meanwhile, as shown in Fig. 1 ①, the SVD of the weight matrix $\boldsymbol{W} \in \mathbb{R}^{h\times d}$ is $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, where $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_r] \in \mathbb{R}^{h\times r}$, $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, ..., \sigma_r) \in \mathbb{R}^{r\times r}$, and $\boldsymbol{V}^T = [\boldsymbol{v}_1^T; \boldsymbol{v}_2^T; ...; \boldsymbol{v}_r^T] \in \mathbb{R}^{r\times d}$ as well as the rank $r \leq \min(h, d)$. Equivalently, the forward pass of this given layer can be reformulated as the element-wise sum of activations induced by each SVD component $\boldsymbol{Y}^{(i)} \in \mathbb{R}^{n\times h}$, in Fig. 1 ②:

Algorithm 1 Our Proposed Forward-Only Approach

Given: Weight matrices $\boldsymbol{W}$ with the SVD $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}^T)$, and the corresponding input $\boldsymbol{X}$ with output $\boldsymbol{Y}$ at a given layer of LLM, subset $\mathcal{D}_{\text{subset}}$, Candidate Set $\mathbb{S}$
for multiple $\boldsymbol{W}$ matrices in all layers **do**
  # Step 1: Forward Passes Averaged Over $\mathcal{D}_{\text{subset}}$
  $\boldsymbol{Y} = \sum_{i=1}^{r} \sigma_i \cdot (\boldsymbol{X}\boldsymbol{v}_i) \otimes \boldsymbol{u}_i^T = \sum_{i=1}^{r} \boldsymbol{Y}^{(i)}$
  # Step 2: Search for $\widehat{\boldsymbol{W}}$ with High Dispersion
  # Angle Concentration is simplified as $\mathrm{AC}(\cdot)$
  $\Delta_i = \mathrm{AC}(\boldsymbol{Y}^{(i)}) - \mathrm{AC}(\boldsymbol{Y})$
  $\mathbb{S} \leftarrow \mathbb{S} \cup \left\{ \left(\boldsymbol{W}, -\sum_{i=r-m+1}^{r} \min(\Delta_i, 0) \right) \right\}$
end for
# Step 3: Rank Reduction for Evaluation
within top-j $\boldsymbol{W}$ in $\mathbb{S}$
  $\widehat{\boldsymbol{\Sigma}} \leftarrow$ keep top-$k\%$ ranks in $\boldsymbol{\Sigma}$
  $\widehat{\boldsymbol{W}} = \boldsymbol{U}\widehat{\boldsymbol{\Sigma}}\boldsymbol{V}^T$
Best $\widehat{\boldsymbol{W}} \leftarrow \{\text{evaluate over validation datasets}\}$
Return: LLM with the rank-reduced $\widehat{\boldsymbol{W}}$

$$\boldsymbol{X}\boldsymbol{W}^T = \sum_{i=1}^{r} \sigma_i \cdot (\boldsymbol{X}\boldsymbol{v}_i) \otimes \boldsymbol{u}_i^T = \sum_{i=1}^{r} \boldsymbol{Y}^{(i)} = \boldsymbol{Y}, \tag{1}$$

where the SVD of $\boldsymbol{W}$ is reformulated as multiple rank-1 components, i.e., $\boldsymbol{W} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \otimes \boldsymbol{v}_i^T$.

2.1 Forward-Only Search

We introduce SVD components and their induced activations, where the activations are naturally accessible in the forward pass. More specifically, we propose to exploit the angle concentration of activations induced by each SVD component. The angle concentration of activations induced by the i-th SVD component, i.e., $\boldsymbol{Y}^{(i)}$, is denoted as follows:

$$\text{Angle Concentration}(\boldsymbol{Y}^{(i)}) = \frac{1}{n \times (n-1)} \sum_{q=1}^{n} \sum_{p=1}^{n} \frac{\boldsymbol{Y}_{q,:}^{(i)} \boldsymbol{Y}_{p,:}^{(i)T}}{\left\|\boldsymbol{Y}_{q,:}^{(i)}\right\| \cdot \left\|\boldsymbol{Y}_{p,:}^{(i)}\right\|} \quad (q \neq p) \tag{2}$$

where $\boldsymbol{Y}_{q,:}^{(i)} \in \mathbb{R}^{1\times h}$ is the q-th row of $\boldsymbol{Y}^{(i)}$ and $\boldsymbol{Y}_{p,:}^{(i)} \in \mathbb{R}^{1\times h}$ is the p-th row of $\boldsymbol{Y}^{(i)}$. Similarly, the angle concentration of the hidden states $\boldsymbol{Y}$ is given by $\text{Angle Concentration}(\boldsymbol{Y})$, where $\boldsymbol{Y}$ is defined as the sum of activations induced by all SVD components.

We exploit the angle concentration of activations induced by each SVD component $\boldsymbol{Y}^{(i)}$, as a heuristic indicator, to measure its contribution—*dispersed* or *concentrated*—to the angle concentration of the hidden states $\boldsymbol{Y}$. We aim to identify the weight matrix with the highest dispersion that benefits most from rank reduction to promote greater concentration. Empirically, we show that LLMs with angles becoming more concentrated, through selective rank reduction, achieve higher accuracy (Appendix A.5). Moreover, Wang et al. [2025b] also demonstrate that the angle between hidden states concentrates progressively when fine-tuning. Accordingly, making the angle between hidden states more concentrated may have the same effect as fine-tuning.[4] As shown in Fig. 1 ③, if the angle concentration of activations induced by one SVD component $\boldsymbol{Y}^{(i)}$ is more dispersed than the angle concentration of $\boldsymbol{Y}$, namely, $\text{Angle Concentration}(\boldsymbol{Y}^{(i)}) - \text{Angle Concentration}(\boldsymbol{Y}) < 0$, **the corresponding SVD component tends to make $\boldsymbol{Y}$ more *dispersed*, and can be removed**. In contrast, if the angle concentration of activations induced by one SVD component $\boldsymbol{Y}^{(i)}$ is more concentrated than the angle concentration of $\boldsymbol{Y}$, namely, $\text{Angle Concentration}(\boldsymbol{Y}^{(i)}) - \text{Angle Concentration}(\boldsymbol{Y}) > 0$, the corresponding SVD component tends to make $\boldsymbol{Y}$ more *concentrated*, and needs to be retained.

[4] We also provide the intuition for the relationship between angle concentration of hidden states (available in the forward pass) and the gradients of singular values (computed in the backward pass) in Appendix I. The gradients of singular values can be used to pinpoint which matrices merit reduction [Sreeram et al., 2025], which avoids the exhaustive search that sweeps all the weights [Sharma et al., 2024] but Sreeram et al. [2025] requires backward passes.

Next, we calculate the difference in angle concentration Δ_i between activations induced by the last-m-rank SVD component and the hidden states $\boldsymbol{Y}$, as shown in Lines 7–8 of Algorithm 1. Then we sum these Δ_i only if $\Delta_i < 0$ (i.e., the corresponding i-th SVD component has the *dispersed* effect). Different weight matrices have distinct levels of dispersion, in other words, distinct levels of necessity for rank reduction. We rank all weight matrices by the levels of dispersion and apply low-rank decomposition only for the top-j weight matrices that have a high level of dispersion, avoiding exhaustive search. In the end, the rank-reduction strategy for retaining the top-$k\%$ ranks is determined by the accuracy evaluated on validation datasets, as presented in Appendix C.2.1.

2.2 Efficient Forward-Only Approach

We further analyze our proposed forward-only approach in terms of memory efficiency and computational efficiency. For memory efficiency, given peak memory consumption, we refer to the baseline (direct inference) as all layers' weight matrices $\sum_1^L |\boldsymbol{W}_l|$ plus the dynamic memory for maximum activations among L layers, i.e., $\max_{l\in\{1,\dots,L\}} \{\max(|\boldsymbol{X}_l|, |\boldsymbol{Y}_l|)\}$, based on the memory analysis in Zhang et al. [2024]. Moreover, in dynamic memory, $|\boldsymbol{W}_l|$ also captures the memory consumption of the corresponding weight gradient, and $|\boldsymbol{\Sigma}_l|$ is for the gradient of singular values. Then we can have the peak memory consumption for the baseline (direct inference), LASER [Sharma et al., 2024], LASER-Grad [Sreeram et al., 2025][5], and our proposed forward-only search in Table 1. Our proposed forward-only search is more memory-efficient than LASER-Grad [Sreeram et al., 2025] by avoiding the backward pass. The detailed analysis for LLMs is derived in Appendix E.

Table 1: The peak memory consumption includes memory for the model (Weight Memory) and the dynamic allocation (Dynamic Memory) for the gradient (if applicable), the hidden states, the SVD components and their induced activations. Here, we consider the baseline (direct inference), LASER [Sharma et al., 2024], LASER-Grad [Sreeram et al., 2025], and our forward-only search.

Method	Weight Memory	Dynamic Memory
Baseline	$\sum_1^L \lvert\boldsymbol{W}_l\rvert$	$\max_{l\in\{1,\dots,L\}} \left(\max\left\{\lvert\boldsymbol{X}_l\rvert, \lvert\boldsymbol{Y}_l\rvert\right\}\right)$
LASER	$\sum_1^L \lvert\boldsymbol{W}_l\rvert$	$\max_{l\in\{1,\dots,L\}} \left(\max(\lvert\boldsymbol{X}_l\rvert, \lvert\boldsymbol{Y}_l\rvert) + \lvert\boldsymbol{U}_l\rvert + \lvert\boldsymbol{\Sigma}_l\rvert + \left\lvert\boldsymbol{V}_l^T\right\rvert\right)$
LASER-Grad	$\sum_1^L \lvert\boldsymbol{W}_l\rvert$	$\sum_{l=1}^{l=L} \max\left\{\lvert\boldsymbol{Y}_l\rvert + \lvert\boldsymbol{U}_l\rvert + \lvert\boldsymbol{\Sigma}_l\rvert + \left\lvert\boldsymbol{V}_l^T\right\rvert, \lvert\boldsymbol{W}_l\rvert + \lvert\boldsymbol{\Sigma}_l\rvert\right\}$
Ours	$\sum_1^L \lvert\boldsymbol{W}_l\rvert$	$\max_{l\in\{1,\dots,L\}} \left(\max\left\{\lvert\boldsymbol{X}_l\rvert, \lvert\boldsymbol{Y}_l\rvert + m\cdot\lvert\boldsymbol{Y}_l\rvert\right\} + \lvert\boldsymbol{U}_l\rvert + \lvert\boldsymbol{\Sigma}_l\rvert + \left\lvert\boldsymbol{V}_l^T\right\rvert\right)$

For computational efficiency, to avoid the extra computational complexity of Equation (1) and Equation (2), we further propose an efficient and analytically equivalent forward-only search that includes an efficient forward pass and an efficient angle concentration, as documented in Appendix C.1. The computational complexity of our efficient forward pass for the last-m-rank SVD component is negligible compared to the standard forward pass. Our efficient angle concentration also does not incur extra computational complexity. Overall, the computational complexity of our efficient forward-only search is on par with that of the standard forward pass (see Appendix C.1.3). To further speed up our forward-only approach, we propose three variants of our rank-reduction strategy, namely, ours (subset), ours (std), and ours (hierarchical) by efficient hierarchical search (see Appendix C.2.1).

3 Experimental Results

3.1 Experimental Setting

Experiments are evaluated on CounterFact [Meng et al., 2022], HotPotQA [Yang et al., 2018], FEVER [Thorne et al., 2018], Bios Gender and Bios Profession from Bias in Bios [De-Arteaga et al., 2019], TruthfulQA [Lin et al., 2022], BigBench-Epistemic Reasoning [Bowman et al., 2015], and BigBench-WikidataQA [BIG-bench Contributors, 2022, Srivastava et al., 2023]. We exploit GPT-J [Wang and Komatsuzaki, 2021] (GPT-J-6B) and Roberta [Liu et al., 2019] (RoBERTa$_{\text{BASE}}$) with the weight matrices in the feed-forward network of Transformer architecture ($\boldsymbol{W}_{in/out}$) and the retained

[5]Note that the backward pass typically consumes about twice the memory of the forward pass in convolutional neural network (CNN)-based networks, where activation memory is the primary bottleneck [Cai et al., 2020, Tenison et al., 2026]. For parameter-dominated networks, e.g., Fully connected networks (FCs) [Huang and Aminifar, 2025c] and LLMs [Zhang et al., 2024] with Adam optimizer, the memory overhead of the backward pass is typically more than twice that of the forward pass.

Table 2: GPT-J evaluation with reported accuracy (Acc, %), Speedup, and peak memory (Mem) in Gigabyte (GB). Our approach achieves the best average improvement in Acc and the best average Speedup. Moreover, our approach shows comparable Mem to Baseline and LASER [Sharma et al., 2024], and $1.9\times$ lower Mem to LASER-Grad [Sreeram et al., 2025] due to our forward-only passes.

Dataset	Baseline		LASER (std)		LASER-Grad (std)			Ours (hierarchical)		
	Acc	Mem↓	Acc	Mem↓	Acc	Speedup↑	Mem↓	Acc	Speedup↑	Mem↓
CounterFact	13.1	11.3 GB	24.0	11.5 GB	**24.4**	1.99×	21.8 GB	24.0	47.1×	11.5 GB
HotPotQA	19.6	11.3 GB	19.5	11.5 GB	19.9	1.99×	21.9 GB	**20.6**	38.4×	11.5 GB
FEVER	50.2	11.3 GB	56.2	11.5 GB	56.0	1.99×	21.9 GB	**58.4**	37.4×	11.5 GB
Bios Gender	70.9	11.3 GB	**97.5**	11.5 GB	88.4	1.99×	22.2 GB	**97.5**	45.2×	11.5 GB
Bios Profession	75.6	11.3 GB	**82.1**	11.5 GB	80.5	1.99×	22.2 GB	79.1	40.7×	11.5 GB
TruthfulQA	54.9	11.3 GB	55.6	11.5 GB	**56.1**	1.99×	21.9 GB	55.7	28.4×	11.5 GB
Epistemic Reasoning	37.1	11.3 GB	38.3	11.5 GB	62.3	1.98×	22.1 GB	**62.8**	15.4×	11.6 GB
WikidataQA	51.8	11.3 GB	65.9	11.5 GB	**66.5**	1.99×	21.8 GB	65.3	41.2×	11.5 GB
Average ΔAcc↑	0.00		8.24		10.1			**11.3**		
Average Speedup↑	-		1×		1.99×			**36.7**×		
Average Mem↓	11.3 GB		11.5 GB		22.0 GB			11.5 GB		

top-$k\%$ ranks are $0.5\%, 1\%, 5\%, 10\%, 20\%, 40\%, 60\%, 80\%$, and 90%, similar to the previous study [Sreeram et al., 2025]. For the variants of our rank-reduction strategy, we adopt 20% of the whole dataset as $\mathcal{D}_{\text{std}}$, and 100 samples from $\mathcal{D}_{\text{std}}$ as $\mathcal{D}_{\text{subset}}$, to evaluate the weight matrices with rank reduction. For ours (hierarchical), we exploit the computational efficiency over $\mathcal{D}_{\text{subset}}$ and then only evaluate the top-j weight matrices over $\mathcal{D}_{\text{std}}$ (see Appendix C.2.1). The accuracy is reported on 80% of the whole dataset by the final candidate with rank reduction. There is no randomness for deterministic SVD and we provide the means and standard deviations with multiple random seeds for the enhanced fine-tuning process. The main experiments utilize 1 NVIDIA Tesla A100 graphics processing unit (GPU) with 40GB random-access memory (RAM). The details of datasets and LLMs used in the main experiments are documented in Appendix C.2, Table 11 and Appendix E, Table 12.

3.2 Results

We compare the baseline (direct inference), LASER [Sharma et al., 2024], LASER-Grad [Sreeram et al., 2025], and our forward-only approach, in terms of accuracy (Acc), computational complexity (Speedup), and peak memory (Mem). We analyze the computational complexity for these methods in Appendix C.2 and peak memory for these methods in Appendix E. The final candidates found by our forward-only approach on validation dataset, along with the justification for considering top-j and last m for GPT-J are presented in Appendix A, while those for Roberta are presented in Appendix B.

3.2.1 Lower Peak Memory & Greater Speedup & Competitive Accuracy

We present the empirical results for GPT-J in Table 2 across various datasets. For GPT-J, our forward-only approach with efficient hierarchical search achieves higher average improvement in Acc, and greater average Speedup compared to LASER (std) and LASER-Grad (std). In terms of average Speedup, our approach (hierarchical) is $36.7\times$ faster than LASER (std), and $18.4\times$ faster than LASER-Grad (std). Meanwhile, our forward-only approach with efficient hierarchical search consumes comparable average Mem to Baseline (direct inference) and LASER, as well as $1.9\times$ less average Mem to LASER-Grad. Moreover, our approach (hierarchical) achieves the best Acc on HotPotQA, FEVER, Bios Gender, and Epistemic Reasoning, i.e., four out of eight datasets. Especially for Epistemic Reasoning, our approach (hierarchical) achieves 62.8% in Acc with $15.4\times$ Speedup, which is a significant improvement than LASER. Taking Bios Gender as an example, our approach (hierarchical) has higher Acc, $22.7\times$ Speedup, and $1.9\times$ lower Mem, than LASER-Grad (std).

For Roberta, our proposed forward-only approach achieves greater average Speedup and comparable Acc compared to LASER (std) and LASER-Grad (std) (see Appendix B.2, Table 9). In terms of average Speedup, our approach is $26.6\times$ faster than LASER (std), while LASER-Grad (std) is actually slower than LASER (std). For the Acc, our approach achieves comparable Acc to LASER (std) and LASER-Grad (std) for the majority of datasets, namely, six out of eight datasets. We notice a substantial difference for Epistemic Reasoning, a pattern that is similarly reflected in LASER-Grad

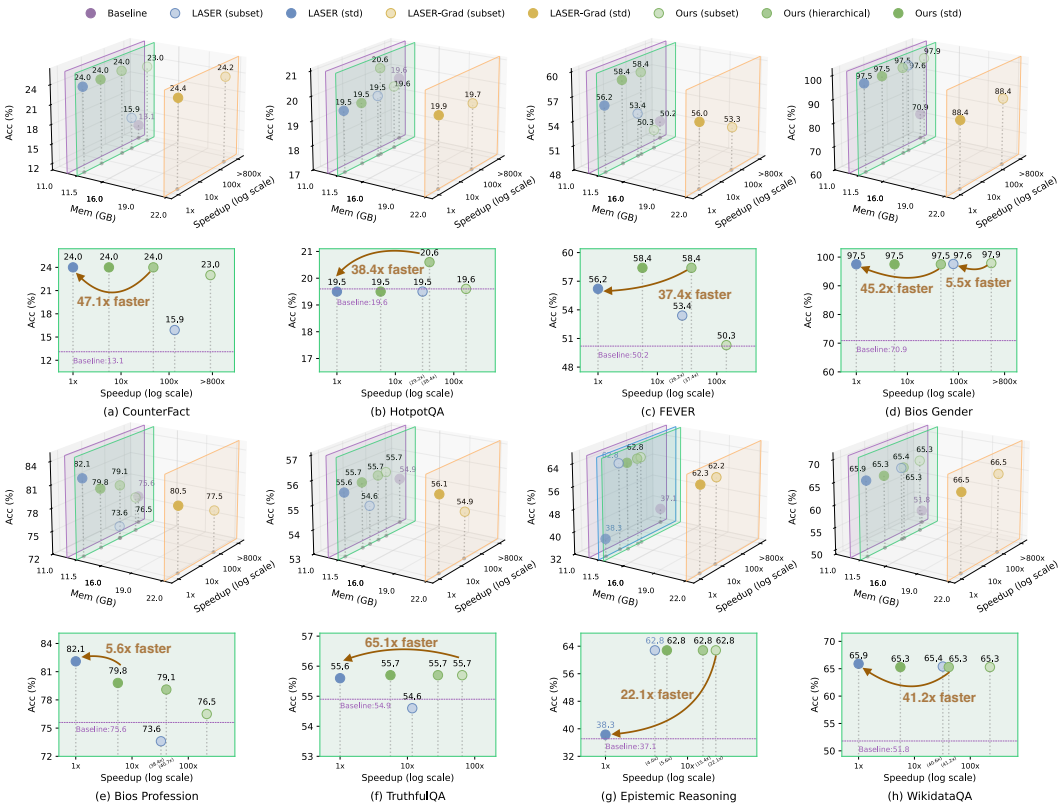


Figure 2: Ablation study on GPT-J: our forward-only approach consumes lower peak memory and greater speedup than LASER-Grad [Sreeram et al., 2025], and achieves faster computation than LASER [Sharma et al., 2024], while maintaining comparable or even superior downstream accuracy.

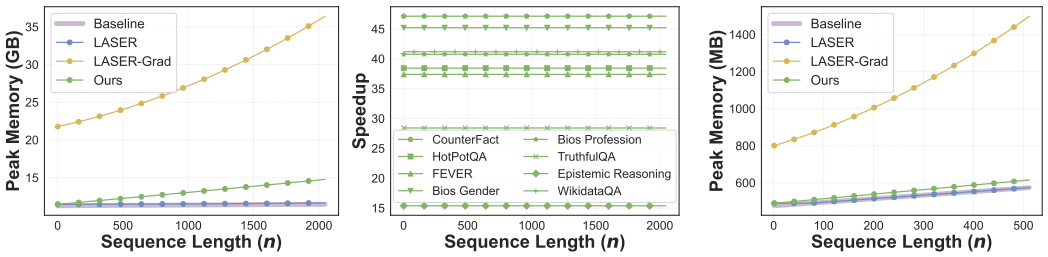
(std) [Sreeram et al., 2025]. Therefore, we reproduce the Acc† for LASER (std). Regarding Bios Profession, our approach achieves 40.5× Speedup and 1.7× lower Mem with a slight decrease in Acc than LASER-Grad (std). Despite this, our approach still demonstrates the effective adaptation on Bios Profession due to the improved Acc from the Baseline. Overall, our proposed forward-only approach achieves lower Mem and greater Speedup than LASER-Grad (std) [Sreeram et al., 2025], as well as greater Speedup and competitive Acc compared to LASER (std) [Sharma et al., 2024].

3.2.2 Ablation Study

We conduct empirical experiments for LASER (subset), LASER-Grad (subset), ours (subset), and ours (std) on GPT-J, to investigate the rank-reduction strategy for LASER [Sharma et al., 2024], LASER-Grad [Sreeram et al., 2025], and our forward-only approach across different sizes of validation datasets and under the same compute. As shown in Fig. 2, LASER-Grad (subset) and LASER-Grad (std) are located on the orange plane, which consumes more Mem than LASER and our approach. Moreover, our approach (subset) achieves faster average Speedup and greater average improvement in Acc than LASER-Grad (subset) (see Appendix A, Table 7). Therefore, we unfold the green plane into two dimensions to compare LASER and our approach. Our approach (subset) always has faster Speedup than LASER (subset), and our approach (std) always has faster Speedup than LASER (std). Besides, our approach (subset) achieves better Acc than LASER (subset) on CounterFact, Bios Profession, TruthfulQA, and Bios Gender, namely, four out of eight datasets. Our approach (std) also achieves comparable and even better Acc than LASER (std) on CounterFact, HotPotQA, FEVER, Bios Gender, TruthfulQA, Epistemic Reasoning and WikidataQA, namely, seven out of eight datasets. Overall, our forward-only approach achieves lower peak memory and greater speedup compared to LASER-Grad [Sreeram et al., 2025], and achieves faster computation compared to LASER [Sharma et al., 2024], with comparable and sometimes even superior downstream performance. In addition, we also present the ablation study on Roberta in Table 3 (Mem in MegaByte (MB)) and the visualization in Appendix B, Fig. 7. Similarly, our forward-only approach achieves lower peak memory and

Table 3: Roberta evaluation with reported accuracy (Acc, %), Speedup, and peak memory (Mem).

Dataset	LASER (subset)			Ours (std)		Ours (hierarchical)		Ours (subset)		
	Acc	Speedup↑	Mem↓	Acc	Speedup↑	Acc	Speedup↑	Acc	Speedup↑	Mem↓
CounterFact	17.7	131×	489 MB	**19.2**	2.4×	17.7	40.4×	17.7	617×	493 MB
HotPotQA	5.9	29.2×	489 MB	**6.3**	4.8×	**6.3**	32.9×	5.5	137×	496 MB
FEVER	52.4	26.2×	489 MB	**52.5**	4.8×	**52.5**	31.9×	**52.5**	123×	496 MB
Bios Gender	92.2	79.3×	495 MB	**93.4**	4.8×	**93.4**	38.7×	93.2	372×	514 MB
Bios Profession	70.9	38.4×	495 MB	**71.9**	4.8×	**71.9**	34.8×	70.9	180×	514 MB
TruthfulQA	**56.2**	11.8×	490 MB	55.6	4.8×	56.0	24.2×	56.0	55.2×	498 MB
Epistemic Reasoning	37.1	4.0×	492 MB	37.1	4.8×	37.1	13.1×	37.1	18.8×	510 MB
WikidataQA	30.7	40.6×	489 MB	30.8	4.8×	30.8	35.2×	**30.9**	191×	492 MB
Average ΔAcc↑		2.05		**2.51**		2.38		2.14		
Average Speedup↑		45.1×		4.5×		31.4×		**211.8**×		
Average Mem↓		491 MB				502 MB				



(a) GPT-J (b) Roberta

Figure 3: The peak memory with different sequence lengths n for Baseline, LASER [Sharma et al., 2024], LASER-Grad [Sreeram et al., 2025], and our approach, as well as the Speedup with different sequence lengths n for our approach (hierarchical). The Speedup with different sequence lengths n for Roberta is in Appendix C.2.1, Fig. 9. Our forward-only approach requires substantially lower peak memory with greater Speedup than LASER-Grad across different sequence lengths n.

greater speedup compared to LASER-Grad [Sreeram et al., 2025], and achieves faster computation compared to LASER [Sharma et al., 2024], with comparable downstream performance.

3.2.3 Extended Experiments

The effectiveness and robustness of efficient hierarchical search. The final candidates for GPT-J in Appendix A, Table 5 found by variants of our approach demonstrate that, there are five datasets having the *Hit*, i.e., the final candidates found in ours (hierarchical) are the same as in ours (std). For the remaining three datasets without *Hit*, our approach (hierarchical) achieves comparable accuracy to ours (std), as shown in Appendix A.3, Table 6. The similar pattern is also observed for Roberta in Appendix B.3. These results show that the top-j weight matrices with rank reduction achieving the top accuracies on $\mathcal{D}_{\text{subset}}$ also tend to rank among the top performers on $\mathcal{D}_{\text{std}}$. Furthermore, we try different sizes of $\mathcal{D}_{\text{subset}}$ (from default 100 to 10, 5, and 1) and the results show the robustness of our efficient hierarchical search and that our proposed efficient hierarchical search can speed up the forward-only search at the extremely small size of $\mathcal{D}_{\text{subset}}$, yielding the same final candidates as the default $\mathcal{D}_{\text{subset}}$ (see Appendix B.4, Table 10). Therefore, our proposed efficient hierarchical search is effective and robust in speeding up the computation while keeping the downstream accuracy.

Resource efficiency across different sequence lengths. To further investigate computational efficiency and memory efficiency of our forward-only approach, we evaluate the peak memory consumption with different sequence lengths n for various methods, and show the Speedup with different sequence lengths n for ours (hierarchical) in Fig. 3. For GPT-J in Fig. 3 (a), the peak memory consumption for our approach increases with a small slope, whereas LASER-Grad exhibits substantial growth. The similar pattern is also observed for Roberta in Fig. 3 (b) and Appendix C.2.1,

Fig. 9. Furthermore, our forward-only approach achieves a greater Speedup as the dataset size of $\mathcal{D}$ increases (see Appendix C.2.1, Fig. 10). Overall, our forward-only approach has significantly lower peak memory and greater Speedup than LASER-Grad across different sequence lengths.

Considering more types of weight matrices. We add the weight matrices of key, query, value, and out projection to default $\boldsymbol{W}_{in/out}$. The corresponding results in Appendix F show that our forward-only search can obtain the same top-j weight matrices as when only considering default $\boldsymbol{W}_{in/out}$, for Roberta. In other words, the weight matrices in attention modules can be excluded from consideration to speed up the computation. Furthermore, the same top-j weight matrices incur the same accuracy as ours (hierarchical) in Table 3, which further justifies the effectiveness of our forward-only search. Moreover, Sharma et al. [2024] present that the weight matrices of key, query, value, and out projection in the attention layers contribute to very little performance increase (GPT-J). Moreover, Sreeram et al. [2025] demonstrate that the weight matrices of key and query can only reach the accuracy of Baseline (GPT-J). Therefore, considering that more weight matrices will increase the computation, it is typical to adopt only the weight matrices in the feed-forward network.

Enhancing Fine-Tuning. Our forward-only approach enhances fine-tuning rather than directly competing with it. Taking Roberta on Bios Gender as an example (Appendix G), fine-tuning the final candidate found by our approach (only 1.89% of the total parameters are trained) achieves the accuracy of 95.1±0.01%, while fine-tuning all the $\boldsymbol{W}_{in}$ across twelve-layer Transformer modules (22.7% of the total parameters are trained) achieves the accuracy of 95.6±0.02%. Similarly, fine-tuning all the $\boldsymbol{W}_{out}$ achieves the accuracy of 95.6±0.01%. Meanwhile, fine-tuning the worst candidate found by our approach (*lower bound*) only gets the accuracy of 89.3±0.03% in Appendix G, Fig. 11. These results show that the final candidate is effective for fine-tuning and can achieve comparable accuracy with 12× fewer trainable parameters than fine-tuning all $\boldsymbol{W}_{in}$ or all $\boldsymbol{W}_{out}$. Notably, obtaining the final candidate without rank reduction from our approach does not require labels. Furthermore, in Fig. 4, the final candidate gives a warm start for effective fine-tuning, where the *warm start* is fine-tuning the final candidate with rank reduction. Especially at the early fine-tuning epochs, the warm start performs better than the cold start (*cold start* is directly fine-tuning the final candidate without rank reduction). More empirical results across datasets in Appendix G consistently show that our forward-only approach enhances the fine-tuning process. In Appendix J, we **transmute weight updates to steering prompts**, analytically and empirically showing that this transmutation achieves nearly identical accuracy without modifying LLM parameters.

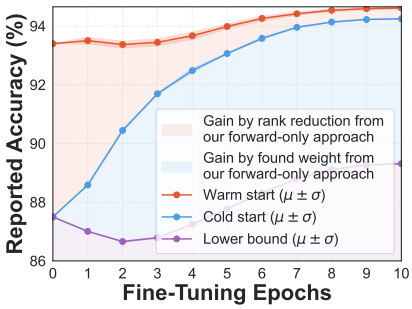


Figure 4: Final candidate with rank reduction from our forward-only approach enhances the fine-tuning process.

Extension to various LLMs. In addition to Roberta (an encoder-only Transformer with bidirectional self-attention) and GPT-J (parallel scheme and rotary positional embedding (RoPE)), we also test LLaMA2 [Touvron et al., 2023] with gated feed-forward networks (Llama2-7B), Mistral [Jiang et al., 2023] with grouped query attention (GQA) and sliding-window attention (Mistral-7B), Qwen2 [Yang et al., 2024a] with instruction-tuning and multilingual capabilities (Qwen2-7B-Instruct), Qwen2 with MoE (Qwen2-57B-A14B-Instruct), Gemma [Team et al., 2024] (Gemma-7B), DeepSeek [Bi et al., 2024] (DeepSeekV1-7B-Chat) with data engineering, as well as the latest Gemma 4 [Google DeepMind, 2026] (Gemma4-E2B-it) with the hybrid attention and per-layer embeddings. These extended LLMs' performance on Epistemic Reasoning (knowledge and belief reasoning) is presented in Table 4 with the found final candidates in Appendix H, Table 18. Our

Table 4: The strong generalization of our forward-only approach across various LLMs up to 57B.

LLMs	Baseline	LASER	Ours	
	Acc	Acc	Acc	Speedup
LLaMA2-7B	44.8	63.4	**63.5**	6.4×
DeepseekV1-7B*	55.1	61.4	**61.8**	29.6×
Mistral-7B	54.8	**61.4**	**61.4**	9.1×
Qwen2-7B*	51.4	**58.3**	**58.3**	27.6×
Gemma-7B	38.3	60.2	**61.4**	11.1×
Gemma4-E2B-it	48.5	**60.5**	60.2	11.6×
Qwen2-57B-A14B*	39.9	**62.1**	**62.1**	13.9×
Average ΔAcc↑	0.00	13.5	**13.7**	
Average Speedup↑	-	1×	**15.6**×	

The ∗ indicates the chat version, i.e., DeepSeekV1-7B-Chat, Qwen2-7B-Instruct and Qwen2-57B-A14B-Instruct.

approach (std) achieves greater speedup and competitive accuracy compared to LASER. Additionally, the results across datasets, including CounterFact (knowledge intervention), HotPotQA (multi-hop reasoning), FEVER (fact verification), Bios Gender and Bios Profession (fairness and bias evaluation), TruthfulQA (hallucination detection), and BigBench-WikidataQA (information retrieval) with LLaMA2 (Appendix H, Table 20) further verify the generalization of our forward-only approach.

Additional Experiments and Analyses. The measurement of realistic memory consumption, computational complexity, and execution time of our approach and LASER-Grad Sreeram et al. [2025] is presented in Appendix K.1. In addition, other simpler forward-only baselines are evaluated in Appendix K.3. Moreover, more baselines in LLM adaptation are presented, including tuning-based techniques (e.g., PEFT in Appendix G, Tables 15 and 16 but such methods require the backward pass, i.e., more memory), and the tuning-free method (e.g., in-context learning (ICL) in Appendix K.2). Together with the enhanced fine-tuning in Fig. 4 and steering prompts in Appendix J, the corresponding results show the effectiveness of our approach compared with tuning-based and tuning-free adaptation methods.

4 Related Work

LLM Compression. Model compression techniques are typically designed to reduce model storage without affecting the downstream accuracy [Han et al., 2016]. To date, LLM compression, including quantization [Dettmers et al., 2023, Xiao et al., 2023, Lin et al., 2025], distillation [Hinton et al., 2015, Sanh et al., 2019], tensor decomposition [Maalouf et al., 2021], pruning [Frantar and Alistarh, 2023, Ashkboos et al., 2024], and KV cache compression [Zhang et al., 2023, Sheng et al., 2023, Hooper et al., 2024], also aim to maintain the downstream accuracy, but to some extent require extra steps such as fine-tuning [Dettmers et al., 2023, Ma et al., 2023], alignment [Ma et al., 2023], or enhancing the downstream accuracy by reasoning [Hsieh et al., 2023] after compression. Notably, the recent studies [Sharma et al., 2024, Sreeram et al., 2025] demonstrate that selectively removing higher-order components of weight matrices in SVD can improve downstream accuracy.

LLM Adaptation. Adapting LLMs to downstream tasks typically involves tuning techniques and knowledge augmentation techniques. Tuning techniques include supervised fine-tuning (SFT) [Radford et al., 2019, Brown et al., 2020], PEFT (such as low-rank adaptation (LoRA) [Hu et al., 2022], adapter tuning [Houlsby et al., 2019], etc.), prompt tuning [Lester et al., 2021], prefix-tuning [Li and Liang, 2021], bias-efficient fine-tuning (BEFT) [Huang et al., 2026a], instruction tuning [Wang et al., 2023] and the following alignment techniques [Ouyang et al., 2022, Rafailov et al., 2023, Guo et al., 2025a], as well as test-time training [Sun et al., 2020, Gandelsman et al., 2022, Tandon et al., 2025]. However, these tuning techniques usually rely on backward passes. Even the training-free alignment of LLMs [Pan et al., 2025] still incurs extra resource overhead compared to inference itself. Besides, knowledge augmentation techniques, including ICL [Brown et al., 2020], prompt instruction [Ye et al., 2025], retrieval-augmented generation [Lewis et al., 2020], prompt engineering [White et al., 2023], activation steering [Li et al., 2023], and the tool-based approach [Schick et al., 2023] primarily operate by augmenting the prompt or injecting information and are orthogonal to modifying the internal weights during LLM inference.

Forward-Only Learning. The emerging forward-only algorithms [Frenkel et al., 2021, Hinton, 2022, Dellaferrera and Kreiman, 2022, Journé et al., 2023, Jiang et al., 2024, Aminifar et al., 2024, Huang and Aminifar, 2025a,b,c, Sun et al., 2025, Huang et al., 2026b] reveal a promising trend in model adaptation from relying on backward passes to more directly leaning on forward passes, which is especially relevant in this era of unprecedented scale of LLMs. The scalability of forward-only learning to deeper models (e.g., LLMs) is still a challenging problem [Jiang et al., 2024, Mei et al., 2024]. Among forward-only techniques, the forward gradient [Baydin et al., 2022] can be scaled to deeper models by either averaging over multiple perturbations or guided from local losses for a better gradient estimation [Ren et al., 2023, Xu et al., 2024], while the zeroth-order optimization can be scaled to deeper models [Chen et al., 2024] and LLMs [Malladi et al., 2023, Zhang et al., 2024, Guo et al., 2025b]. Besides, the evolutionary strategies scaled to LLMs either need to tune the input prompt/embedding [Sun et al., 2022, Prasad et al., 2023] or need several iterative (tuning) forward passes [Yang et al., 2024b], similarly for Niu et al. [2024].

5 Conclusion and Future Work

Amid the rapid growth and large-scale deployment of LLM inference serving, we are motivated to rethink how the inference process itself can be leveraged to improve the serving system, particularly enabling efficient task-specific LLM adaptation. In this paper, we propose a forward-only approach that efficiently identifies the weight matrix with the highest dispersion based on the angle concentration, and subsequently applies rank reduction for efficient LLM adaptation with forward-only passes in LLM inference serving. The results across diverse datasets demonstrate that our forward-only approach achieves lower peak memory and greater speedup than the gradient-based approach, as well as greater speedup and competitive accuracy compared to the exhaustive search approach. The experiments further present the robustness and generalization of our forward-only approach to various LLMs with up to 57B parameters. This paper presents a new perspective on leveraging the inference process itself to improve LLM serving by enabling efficient task-specific LLM adaptation.

Future work includes investigating activation truncation rather than weight truncation [Qinsi et al., 2025], exploring dynamic adaptation [Mi et al., 2025, Li et al., 2025] within the routed experts in MoE [Yang et al., 2024a, Dai et al., 2024], decomposing stacked weight matrices [Luo et al., 2025], combining forward-only adaptation with the post fine-tuning [Li et al., 2024, Wang et al., 2025c,a, Bałazy et al., 2025, Sun et al., 2026], and studying the potential joint subspaces [Kaushik et al., 2026].

Acknowledgments and Disclosures

This research has been partially supported by the Swedish Wallenberg AI, Autonomous Systems and Software Program (WASP), Swedish Research Council, Swedish Foundation for Strategic Research, ELLIIT Strategic Research Environment, and an unrestricted gift from Google. Computational resources were provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), funded by the Swedish Research Council.

Ananth Balashankar is an employee of Google DeepMind. The remaining authors declare no competing interests.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Amin Aminifar, Baichuan Huang, Azra Abtahi Fahliani, and Amir Aminifar. Lightff: Lightweight inference for forward-forward algorithm. In *27th European Conference on Artificial Intelligence, ECAI-2024*, pages 1728–1735. IOS Press, 2024.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=vXxardq6db`.

Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. Lora-xs: Low-rank adaptation with extremely small number of parameters. In *28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy–Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025)*. IOS press, 2025.

Noman Bashir, Priya Donti, James Cuff, Sydney Sroka, Marija Ilic, Vivienne Sze, Christina Delimitrou, and Elsa Olivetti. The Climate and Sustainability Implications of Generative AI. *An MIT Exploration of Generative AI*, mar 27 2024. https://mit-genai.pubpub.org/pub/8ulgrckc.

Atılım Güneş Baydin, Barak A Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without backpropagation. *arXiv preprint arXiv:2202.08587*, 2022.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *OpenAI*, 2023. `https://cdn.openai.com/papers/dall-e-3.pdf`.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

BIG-bench Contributors. BIG-bench: Qa wikidata task, 2022. `https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/qa_wikidata` [Accessed: 26-Feb-2026].

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 632–642, 2015.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. Tinytl: Reduce memory, not parameters for efficient on-device learning. *Advances in Neural Information Processing Systems*, 33:11285–11297, 2020.

Aochuan Chen, Yimeng Zhang, Jinghan Jia, James Diffenderfer, Konstantinos Parasyris, Jiancheng Liu, Yihua Zhang, Zheng Zhang, Bhavya Kailkhura, and Sijia Liu. Deepzero: Scaling up zeroth-order optimization for deep model training. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=qBWhjsNPEY`.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, 2024.

Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 120–128, 2019.

Giorgia Dellaferrera and Gabriel Kreiman. Error-driven input modulation: solving the credit assignment problem without a backward pass. In *International Conference on Machine Learning*, pages 4937–4955. PMLR, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Benoit Dherin, Michael Munn, Hanna Mazzawi, Michael Wunder, and Javier Gonzalvo. Learning without training: The implicit dynamics of in-context learning. *arXiv preprint arXiv:2507.16003*, 2025.

Guodong Du and Wanyu Lin. Dynamic model merging made slim. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026a. URL `https://arxiv.org/abs/2605.18904`.

Guodong Du and Wanyu Lin. Nestedvla: Learning to consolidate and generate skills for vision-language-action models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026b. URL `https://arxiv.org/abs/2610.23271`.

Cooper Elsworth, Keguo Huang, David Patterson, Ian Schneider, Robert Sedivy, Savannah Goodman, Ben Townsend, Parthasarathy Ranganathan, Jeff Dean, Amin Vahdat, Ben Gomes, and James Manyika. Measuring the environmental impact of delivering ai at google scale, 2025. URL `https://arxiv.org/abs/2508.15734`.

Epoch AI. "data on ai models", 2026. URL `https://epoch.ai/data/ai-models`. Accessed: 25 Mar 2026.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pages 10323–10337. PMLR, 2023.

Charlotte Frenkel, Martin Lefebvre, and David Bol. Learning without feedback: Fixed random learning signals allow for feedforward training of deep neural networks. *Frontiers in neuroscience*, 15:629892, 2021.

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022.

Adrian Goldwaser, Michael Munn, Javier Gonzalvo, and Benoit Dherin. Equivalence of context and parameter updates in modern transformer blocks. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=aZ5Di8Ii4X.

Google DeepMind. Gemma 4: Byte for byte, the most capable open models. https://blog.google/innovation-and-ai/technology/developers-tools/gemma-4/, April 2026. Accessed: 2026-04-04.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025a.

Wentao Guo, Jikai Long, Yimeng Zeng, Zirui Liu, Xinyu Yang, Yide Ran, Jacob R. Gardner, Osbert Bastani, Christopher De Sa, Xiaodong Yu, Beidi Chen, and Zhaozhuo Xu. Zeroth-order fine-tuning of LLMs with transferable static sparsity. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=myYzr50xBh.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

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 *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.507. URL https://aclanthology.org/2023.findings-acl.507/.

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 *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Baichuan Huang and Amir Aminifar. Binary forward-only algorithms. *IEEE Design & Test*, 42(5): 55–62, 2025a. doi: 10.1109/MDAT.2025.3528366.

Baichuan Huang and Amir Aminifar. Efficient on-device machine learning with a biologically-plausible forward-only algorithm. *Proceedings of Machine Learning and Systems*, 7, 2025b.

Baichuan Huang and Amir Aminifar. Tinyfoa: Memory efficient forward-only algorithm for on-device learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 17377–17385, 2025c.

Baichuan Huang, Azra Abtahi, and Amir Aminifar. Energy-aware integrated neural architecture search and partitioning for distributed internet of things (iot). *IEEE Transactions on Circuits and Systems for Artificial Intelligence*, 1(2):257–271, 2024. doi: 10.1109/TCASAI.2024.3493036.

Baichuan Huang, Ananth Balashankar, and Amir Aminifar. Beft: Bias-efficient fine-tuning of language models in low-data regimes. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026a.

Baichuan Huang, Alexander Ororbia, and Amir Aminifar. Backpropagation-free learning: On the emergence of forward-only algorithms. *TechRxiv*, 2026(0103), 2026b. doi: 10.36227/techrxiv.176739859.93468234/v1. URL https://www.techrxiv.org/doi/abs/10.36227/techrxiv.176739859.93468234/v1.

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. URL https://arxiv.org/abs/2310.06825.

Jinyang Jiang, Zeliang Zhang, Chenliang Xu, Zhaofei Yu, and Yijie Peng. One forward is enough for neural network training via likelihood ratio method. In *International Conference on Learning Representations*, volume 2024, pages 52253–52276, 2024.

Adrien Journé, Hector Garcia Rodriguez, Qinghai Guo, and Timoleon Moraitis. Hebbian deep learning without feedback. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=8gd4M-_Rj1.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Prakhar Kaushik, Shravan S Chaudhari, Ankit Vaidya, Rama Chellappa, and Alan Yuille. Do neural networks learn similar subspaces? an empirical exploration of joint parametric subspaces in deep neural networks, 2026. URL https://openreview.net/forum?id=xEcUIn81Fp.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, 2021.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.

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, 2021.

Yang Li, Daniel Agyei Asante, Changsheng Zhao, Ernie Chang, Yangyang Shi, and Vikas Chandra. Basis selection: low-rank decomposition of pretrained large language models for target applications. *arXiv preprint arXiv:2405.15877*, 2024.

Zhiteng Li, Mingyuan Xia, Jingyuan Zhang, Zheng Hui, Haotong Qin, Linghe Kong, Yulun Zhang, and Xiaokang Yang. Adasvd: Adaptive singular value decomposition for large language models. *arXiv preprint arXiv:2502.01403*, 2025.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 3214–3252, 2022.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *Proceedings of Machine Learning and Systems*, 7, 2025.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Yiran Luo, Het Patel, Yu Fu, Dawon Ahn, Jia Chen, Yue Dong, and Evangelos E Papalexakis. Trawl: Tensor reduced and approximated weights for large language models. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 402–413. Springer, 2025.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Alaa Maalouf, Harry Lang, Daniela Rus, and Dan Feldman. Deep learning meets projective clustering. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=EQfpYwF3-b`.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

Nestor Maslej, Loredana Fattorini, Raymond Perrault, Yolanda Gil, Vanessa Parli, Njenga Kariuki, Emily Capstick, Anka Reuel, Erik Brynjolfsson, John Etchemendy, et al. Artificial intelligence index report 2025. *arXiv preprint arXiv:2504.07139*, 2025.

Hanna Mazzawi, Benoit Dherin, Michael Munn, Michael Wunder, and Javier Gonzalvo. Transmuting prompts into weights. *arXiv preprint arXiv:2510.08734*, 2025.

Hanzi Mei, Dongqi Cai, Yaozong Wu, Shangguang Wang, and Mengwei Xu. A survey of backpropagation-free training for llms. *TechRxiv*, 2024. DOI: 10.36227/techrxiv.171172909.97532161/v1.

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

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Zhendong Mi, Bian Sun, Grace Li Zhang, and Shaoyi Huang. Layer-wise dynamic rank for compressing large language models. *arXiv preprint arXiv:2509.25622*, 2025.

Shuaicheng Niu, Chunyan Miao, Guohao Chen, Pengcheng Wu, and Peilin Zhao. Test-time model adaptation with only forward passes. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=qz1Vx1v9iK`.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Birong Pan, Yongqi Li, Weiyu Zhang, Wenpeng Lu, Mayi Xu, Shen Zhou, Yuanyuan Zhu, Ming Zhong, and Tieyun Qian. A survey on training-free alignment of large language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 4445–4461, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.238. URL `https://aclanthology.org/2025.findings-emnlp.238/`.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. Grips: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3845–3864, 2023.

Wang Qinsi, Jinghan Ke, Masayoshi Tomizuka, Kurt Keutzer, and Chenfeng Xu. Dobi-SVD: Differentiable SVD for LLM compression and some new perspectives. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kws76i5XB8.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Hui Ren, Joanna Materzynska, Rohit Gandikota, Giannis Daras, David Bau, and Antonio Torralba. Opt-in art: Learning art styles only from few examples. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Creative AI Track: Humanity*, 2025. URL https://openreview.net/forum?id=pHKEHWu1Yr.

Mengye Ren, Simon Kornblith, Renjie Liao, and Geoffrey Hinton. Scaling forward gradient with local losses. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JxpBP1JM15-.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Pratyusha Sharma, Jordan T. Ash, and Dipendra Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ozX92bu8VA.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR, 2023.

Shiva Sreeram, Alaa Maalouf, Pratyusha Sharma, and Daniela Rus. Compress to impress: Efficient LLM adaptation using a single gradient step on 100 samples. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*, 2023.

Chengwei Sun, Jiwei Wei, Yujia Wu, Yiming Shi, Shiyuan He, Zeyu Ma, Ning Xie, and Yang Yang. Unleashing the power of singular values for parameter-efficient fine-tuning of large pre-trained models. *IEEE Transactions on Multimedia*, 2026.

Liang Sun, Yang Zhang, Weizhao He, Jiajun Wen, Linlin Shen, and Weicheng Xie. Deeperforward: Enhanced forward-forward training for deeper and better performance. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kOYnXVQCtA.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pages 20841–20855. PMLR, 2022.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR, 2020.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, et al. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Irene Tenison, Soumyajit Chatterjee, Fahim Kawsar, and Mohammad Malekzadeh. AdaBet: Gradient-free layer selection for efficient training of deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, 2018.

Thunder Said Energy. Internet energy consumption: data, models, forecasts?, 2026. URL `https://thundersaidenergy.com/downloads/internet-energy-consumpion-data-models-forecasts/`. Accessed: 2026-04-06.

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 arXiv:2307.09288*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4823–4836, 2025a.

Qinsi Wang, Jinghan Ke, Hancheng Ye, Yueqian Lin, Yuzhe Fu, Jianyi Zhang, Kurt Keutzer, Chenfeng Xu, and Yiran Chen. Angles don't lie: Unlocking training-efficient RL through the model's own signals. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Shibo Wang and Pankaj Kanwar. BFloat16: The secret to high performance on cloud TPUs. `https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus`, August 2019. Google Cloud Blog. Accessed: 2026-05-03.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *The Thirteenth International Conference on Learning Representations*, 2025c. URL `https://openreview.net/forum?id=LNYIUouhdt`.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508, 2023.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. In *Proceedings of the 30th Conference on Pattern Languages of Programs*, pages 1–31, 2023.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR, 2023.

Mengwei Xu, Dongqi Cai, Yaozong Wu, Xiang Li, and Shangguang Wang. Fwdllm: Efficient federated finetuning of large language models with perturbed inferences. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 579–596, 2024.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024a. URL `https://arxiv.org/abs/2407.10671`.

Jiaxi Yang, Binyuan Hui, Min Yang, Bailin Wang, Bowen Li, Binhua Li, Fei Huang, and Yongbin Li. Iterative forward tuning boosts in-context learning in language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15460–15473, 2024b.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380, 2018.

Zesheng Ye, Chengyi Cai, Ruijiang Dong, Jianzhong Qi, Lei Feng, Pin-Yu Chen, and Feng Liu. Neural network reprogrammability: A unified theme on model reprogramming, prompt tuning, and prompt instruction. *arXiv preprint arXiv:2506.04650*, 2025.

Junjie Oscar Yin and Alexander Rush. Compute-constrained data selection. In *International Conference on Learning Representations*, volume 2025, pages 36708–36729, 2025.

Yihua Zhang, Pingzhi Li, Junyuan Hong, Jiaxiang Li, Yimeng Zhang, Wenqing Zheng, Pin-Yu Chen, Jason D Lee, Wotao Yin, Mingyi Hong, et al. Revisiting zeroth-order optimization for memory-efficient llm fine-tuning: A benchmark. In *International Conference on Machine Learning*, pages 59173–59190. PMLR, 2024.

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

Appendix

Contents

1 Introduction **1**

2 Our Forward-Only Approach **2**

2.1 Forward-Only Search 3

2.2 Efficient Forward-Only Approach 4

3 Experimental Results **4**

3.1 Experimental Setting 4

3.2 Results 5

3.2.1 Lower Peak Memory & Greater Speedup & Competitive Accuracy 5

3.2.2 Ablation Study 6

3.2.3 Extended Experiments 7

4 Related Work **9**

5 Conclusion and Future Work **10**

Acknowledgments and Disclosures **10**

Appendix **18**

A Final Candidates for Our Forward-Only Approach on GPT-J **19**

A.1 Justification for considering top-j in Algorithm 1 19

A.2 Justification for considering last m SVD components in Algorithm 1 20

A.3 The effectiveness of efficient hierarchical search 20

A.4 Various Rank-Reduction Strategies 21

A.5 Angle Concentration and LLM Adaptation 21

B Final Candidates for Our Forward-Only Approach on Roberta **22**

B.1 Justification for considering top-j and last m SVD components in Algorithm 1 . . 22

B.2 Performance on Roberta 22

B.3 The effectiveness of efficient hierarchical search 23

B.4 The robustness of efficient hierarchical search across different $\mathcal{D}_{\text{subset}}$ 23

B.5 Various Rank-Reduction Strategies 24

C Speedup in Computational Complexity **24**

C.1 Efficient Forward-Only Search 24

C.1.1 Speedup in Efficient Forward Pass 25

C.1.2 Speedup in Efficient Angle Concentration 25

C.1.3 Computational Complexity of Efficient Forward-Only Search 26

C.2 Computational Complexity Comparison Across Methods 28
C.2.1 Efficient Hierarchical Search 28

D The Declaration of AI Usage **30**

E Peak Memory Consumption **31**
E.1 Roberta 31
E.2 GPT-J 33

F Considering More Weight Matrices in the Transformer Architecture **35**

G Our Forward-Only Approach Enhances Fine-Tuning **36**

H Extended Experiments on Various LLMs **36**

I Intuitive Explanation **39**

J Transmuting Weight Updates to Steering Prompts **43**

K Additional Experiments and Analyses **44**

L NeurIPS Paper Checklist **46**

A Final Candidates for Our Forward-Only Approach on GPT-J

Table 5: Final candidates for our forward-only approach on GPT-J in Table 2 and 7, including $\boldsymbol{W}_{in/out}$ in the feed-forward network of Transformer architecture, the layer index l ($0 \leq l \leq 27$) of stacked Transformer modules, and the retained top-$k\%$ ranks ($0.5\% \leq k\% \leq 90\%$).

Dataset	Ours (subset)	Ours (std)	Ours (hierarchical)	Hit?
CounterFact	$[\boldsymbol{W}_{in}, 27, 10\%]$	$[\boldsymbol{W}_{in}, 27, 1\%]$	$[\boldsymbol{W}_{in}, 27, 1\%]$	Yes
HotPotQA	$[\boldsymbol{W}_{in}, 10, 80\%]$	$[\boldsymbol{W}_{in}, 22, 80\%]$	$[\boldsymbol{W}_{out}, 27, 1\%]$	No
FEVER	$[\boldsymbol{W}_{in}, 10, 0.5\%]$	$[\boldsymbol{W}_{in}, 10, 10\%]$	$[\boldsymbol{W}_{in}, 10, 10\%]$	Yes
Bios Gender	$[\boldsymbol{W}_{in}, 14, 0.5\%]$	$[\boldsymbol{W}_{in}, 14, 1\%]$	$[\boldsymbol{W}_{in}, 14, 1\%]$	Yes
Bios Profession	$[\boldsymbol{W}_{in}, 25, 40\%]$	$[\boldsymbol{W}_{in}, 14, 1\%]$	$[\boldsymbol{W}_{in}, 14, 10\%]$	No
TruthfulQA	$[\boldsymbol{W}_{out}, 27, 5\%]$	$[\boldsymbol{W}_{out}, 27, 1\%]$	$[\boldsymbol{W}_{out}, 27, 5\%]$	No
Epistemic Reasoning	$[\boldsymbol{W}_{in}, 4, 0.5\%]$	$[\boldsymbol{W}_{in}, 4, 1\%]$	$[\boldsymbol{W}_{in}, 4, 1\%]$	Yes
WikidataQA	$[\boldsymbol{W}_{in}, 27, 0.5\%]$	$[\boldsymbol{W}_{in}, 27, 0.5\%]$	$[\boldsymbol{W}_{in}, 27, 0.5\%]$	Yes

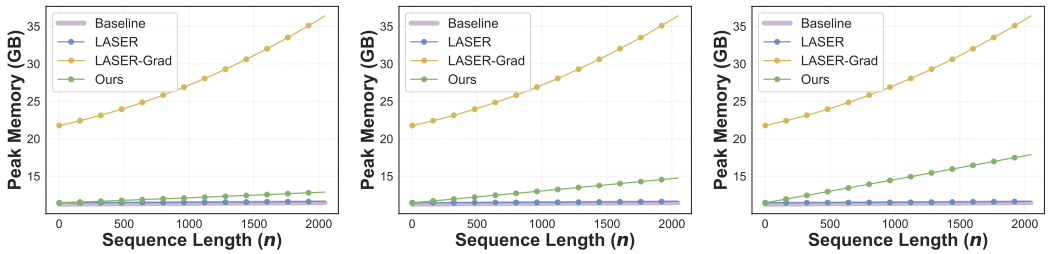
A.1 Justification for considering top-j in Algorithm 1

The final candidates for ours (hierarchical) with $j = 10$ are presented in Table 5. Here, we also test other top-j values. When $j = 5$, for all eight datasets except Epistemic Reasoning, the Acc for ours (hierarchical) with $j = 5$ is lower than the Acc for ours (hierarchical) in Table 2. Only for Epistemic Reasoning, the final candidate for ours (hierarchical) is the same as in Table 5. Therefore, we do not adopt $j = 5$. Moreover, when $j = 7$, the final candidate for ours (hierarchical) on CounterFact is $[\boldsymbol{W}_{in}, 26, 1\%]$ with the Acc of 17.3%, which is worse than 24.0% for ours (hierarchical) in Table 2. Similarly, for WikidataQA, the final candidate for ours (hierarchical) is $[\boldsymbol{W}_{out}, 27, 0.5\%]$ with the Acc of 63.2%, which is worse than 65.3% for ours (hierarchical) in Table 2. In addition, the final candidate for ours (hierarchical) on HotPotQA is $[\boldsymbol{W}_{in}, 25, 80\%]$ with the Acc of 19.5%, and on TruthfulQA is $[\boldsymbol{W}_{out}, 27, 5\%]$ with the Acc of 55.7%, which are comparable to ours (hierarchical) in

Table 2. For the other four datasets (FEVER, Bios Gender, Bios Profession, Epistemic Reasoning), the final candidates for ours (hierarchical) are the same as in Table 5. Overall, we adopt $j = 10$ for GPT-J, while noting that smaller values of j can yield higher Speedup (Equation (7)) with comparable Acc on certain datasets.

A.2 Justification for considering last m singular value decomposition (SVD) components in Algorithm 1

First, we evaluate the peak memory consumption with different last-m-rank ($m = 20, 50, 100$) SVD components. As presented in Fig. 5, higher m will increase the peak memory consumption of our approach, especially with the longer sequence length. Therefore, the smaller m, the better choice as long as the Acc is competitive. Note that the $m = 20, 50, 100$ will not cause great differences in the ratio of our forward-only approach compared to the standard forward pass based on Equation (6); meanwhile, the j value of top-j weight matrices in Algorithm 1 will not influence the peak memory consumption of our approach.



(a) GPT-J ($m = 20$) (b) GPT-J ($m = 50$) (c) GPT-J ($m = 100$)

Figure 5: The peak memory consumption for Baseline, LASER Sharma et al. [2024], LASER-Grad Sreeram et al. [2025], and our forward-only approach with different sequence lengths n and different last-m-rank SVD components for GPT-J. Our forward-only approach requires significantly lower peak memory than LASER-Grad. The smaller m, the lower peak memory consumption.

Next, we evaluate different $m = 20, 50, 100$ with $j = 10$. When $m = 20$, the Acc for ours (hierarchical) on five datasets (CounterFact, HotPotQA, FEVER, Epistemic Reasoning, and WikidataQA) are worse than the Acc for ours (hierarchical) in Table 2. Meanwhile, the Acc for ours (hierarchical) on Bios Gender is 97.5%, which is same as the Acc of LASER (std) Sharma et al. [2024] in Table 2. Therefore, $m = 20$ is enough for Bios Gender. For Bios Profession, the ACC for ours (hierarchical) from $m = 20$ to $m = 50$ is comparable; therefore, $m = 20$ is adopted for Bios Profession. For the other six datasets, we evaluate different $m = 50, 100$ and compare their final candidates for ours (hierarchical). For CounterFact, FEVER, Epistemic Reasoning, and WikidataQA, the final candidates for ours (hierarchical) with $m = 50, 100$ are the same. While for HotPotQA and TruthfulQA, although the final candidates for ours (hierarchical) are not the same among $m = 50, 100$, the Acc are comparable. Therefore, $m = 50$ is adopted for these six datasets.

A.3 The effectiveness of efficient hierarchical search

As presented in Table 5, we present the final candidates $[\boldsymbol{W}, l, k\%]$ for variants of our approach. We select the top-j $[\boldsymbol{W}, l]$ from our forward-only search, and then evaluate top-j $[\boldsymbol{W}, l]$ with 9 different retained top-$k\%$ ranks in $\mathcal{D}_{\text{subset}}$. Next, only the top-$j$ $[\boldsymbol{W}, l, k\%]$ from $\mathcal{D}_{\text{subset}}$ are directly evaluated on $\mathcal{D}_{\text{std}}$. The best $[\boldsymbol{W}, l, k\%]$ from $\mathcal{D}_{\text{std}}$ is the final candidate of ours (hierarchical). Therefore, if the top-j $[\boldsymbol{W}, l, k\%]$ from $\mathcal{D}_{\text{subset}}$ include the best $[\boldsymbol{W}, l, k\%]$ from $\mathcal{D}_{\text{std}}$, we call it *Hit*. In GPT-J, there are 5 datasets (CounterFact, FEVER, Bios Gender, Epistemic Reasoning, and WikidataQA) that have *Hit*, which means that ours (hierarchical) can get exactly the same final candidates with improving the evaluation efficiency (i.e., increases the Speedup as shown in Table 2 and Table 7) compared to ours (std). On the contrary, for the datasets (HotPotQA, Bios Profession, TruthfulQA) without *Hit*, the final candidates for ours (std) and ours (hierarchical) are different. However, as presented in Table 6, ours (hierarchical) can get comparable accuracy to ours (std) with greater Speedup.

Table 6: Acc (%) and Speedup of ours (std) and ours (hierarchical) for datasets without *Hit*.

Dataset	Ours (std)		Ours (hierarchical)	
	Acc	Speedup↑	Acc	Speedup↑
HotPotQA	19.5	5.6×	20.6	38.4×
Bios Profession	79.8	5.6×	79.1	40.7×
TruthfulQA	55.7	5.6×	55.7	28.4×

A.4 Various Rank-Reduction Strategies

Table 7: GPT-J evaluation with reported accuracy (Acc, %), Speedup, and peak memory (Mem) in Gigabyte (GB). Ours (std) achieves the best average improvement in Acc; ours (subset) achieves the best average Speedup. Ours shows 1.9× lower Mem to LASER-Grad due to our forward-only passes.

Dataset	LASER (subset)			LASER-Grad (subset)			Ours (std)			Ours (subset)	
	Acc	Speedup↑	Mem↓	Acc	Speedup↑	Mem↓	Acc	Speedup↑	Mem↓	Acc	Speedup↑
CounterFact	15.9	131×	11.5 GB	24.2	349×	21.8 GB	24.0	5.6×	11.5 GB	23.0	728×
HotPotQA	19.5	29.2×	11.5 GB	19.7	77.5×	21.9 GB	19.5	5.6×	11.5 GB	19.6	162×
FEVER	53.4	26.2×	11.5 GB	53.3	69.4×	21.9 GB	58.4	5.6×	11.5 GB	50.3	145×
Bios Gender	97.6	79.3×	11.5 GB	88.4	210×	22.2 GB	97.5	5.6×	11.5 GB	97.9	439×
Bios Profession	73.6	38.4×	11.5 GB	77.5	102×	22.2 GB	79.8	5.6×	11.5 GB	76.5	213×
TruthfulQA	54.6	11.8×	11.5 GB	54.9	31.2×	21.9 GB	55.7	5.6×	11.5 GB	55.7	65.1×
Epistemic Reasoning	62.8	4.0×	11.5 GB	62.2	10.6×	22.1 GB	62.8	5.6×	11.6 GB	62.8	22.1×
WikidataQA	65.4	40.6×	11.5 GB	66.5	108×	21.8 GB	65.3	5.6×	11.5 GB	65.3	225×
Average ΔAcc↑		8.7			9.2			**11.2**			9.7
Average Speedup↑		45.1×			120×			5.6×			**250**×
Average Mem↓		11.5 GB			22.0 GB				11.5 GB		

A.5 Angle Concentration and LLM Adaptation

We empirically measure the angle concentration of the hidden states in the corresponding layer, before rank reduction (purple dots) and after rank reduction (green dots) for the final candidate found by ours (hierarchical), where the maximum value of angle concentration is 1.0 based on Equation 2. The results for GPT-J on eight datasets are presented in Fig. 6, showing that the angle concentration of the hidden states after rank reduction is higher than that of the hidden states before rank reduction. Meanwhile, large language models (LLMs) with a higher angle concentration perform better with accuracy across datasets. Moreover, as mentioned in Section 2.1, the study [Wang et al., 2025b] demonstrates that the angle between hidden states concentrates progressively when fine-tuning. Similarly to the fine-tuning–like effect, LLMs with rank-reduced final candidate from our approach achieve higher downstream accuracy, with hidden-state angles becoming more concentrated.

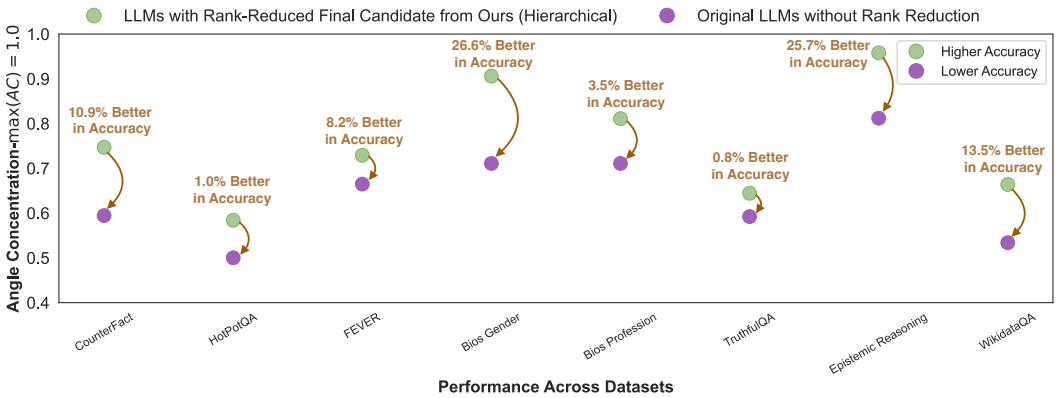


Figure 6: LLMs with rank-reduced final candidate from our forward-only approach achieve higher downstream accuracy, with hidden-state angles becoming more concentrated.

B Final Candidates for Our Forward-Only Approach on Roberta

Table 8: Final candidates for our forward-only approach on Roberta in Table 3, including $\boldsymbol{W}_{in/out}$ in the feed-forward network of Transformer architecture, the layer index l ($0 \leq l \leq 11$) of stacked Transformer modules, and the retained top-$k\%$ ranks ($0.5\% \leq k\% \leq 90\%$).

Dataset	Ours (subset)	Ours (std)	Ours (hierarchical)	Hit?
CounterFact	$[\boldsymbol{W}_{in}, 1, 60\%]$	$[\boldsymbol{W}_{in}, 8, 80\%]$	$[\boldsymbol{W}_{in}, 1, 60\%]$	No
HotPotQA	$[\boldsymbol{W}_{out}, 11, 60\%]$	$[\boldsymbol{W}_{in}, 1, 90\%]$	$[\boldsymbol{W}_{in}, 1, 90\%]$	Yes
FEVER	$[\boldsymbol{W}_{in}, 3, 40\%]$	$[\boldsymbol{W}_{in}, 3, 40\%]$	$[\boldsymbol{W}_{in}, 3, 40\%]$	Yes
Bios Gender	$[\boldsymbol{W}_{in}, 9, 80\%]$	$[\boldsymbol{W}_{in}, 9, 90\%]$	$[\boldsymbol{W}_{in}, 9, 90\%]$	Yes
Bios Profession	$[\boldsymbol{W}_{in}, 1, 80\%]$	$[\boldsymbol{W}_{in}, 3, 90\%]$	$[\boldsymbol{W}_{in}, 3, 90\%]$	Yes
TruthfulQA	$[\boldsymbol{W}_{in}, 2, 80\%]$	$[\boldsymbol{W}_{in}, 3, 80\%]$	$[\boldsymbol{W}_{in}, 2, 80\%]$	No
Epistemic Reasoning	$[\boldsymbol{W}_{in}, 5, 0.5\%]$	$[\boldsymbol{W}_{in}, 9, 1\%]$	$[\boldsymbol{W}_{in}, 9, 0.5\%]$	No
WikidataQA	$[\boldsymbol{W}_{out}, 11, 1\%]$	$[\boldsymbol{W}_{out}, 11, 20\%]$	$[\boldsymbol{W}_{out}, 11, 20\%]$	Yes

B.1 Justification for considering top-j and last m SVD components in Algorithm 1

The final candidates for ours (hierarchical) with $j = 5$ are presented in Table 8. Here, we also test other top-j values. When $j = 7$, for all eight datasets except FEVER and TruthfulQA, the final candidates are the same as $j = 5$. While for FEVER ($[\boldsymbol{W}_{in}, 2, 40\%]$) and TruthfulQA ($[\boldsymbol{W}_{in}, 5, 10\%]$), $j = 7$ for ours (hierarchical) does not gain any improvement in Acc compared to $j = 5$, but decreases the Speedup due to Equation (7). When $j = 10$, for all eight datasets except HotPotQA and Epistemic Reasoning, the final candidates are the same as $j = 7$. Similarly, while for HotPotQA ($[\boldsymbol{W}_{in}, 1, 80\%]$) and Epistemic Reasoning ($[\boldsymbol{W}_{in}, 3, 5\%]$), $j = 10$ for ours (hierarchical) does not gain any improvement in Acc compared to $j = 7$. Overall, we do not adopt $j = 7, 10$. Instead, $j = 5$ is enough for ours (hierarchical).

We evaluate different $m = 20, 50, 100$ with $j = 5$. The results of $m = 20$ are in Table 8. When $m = 50$, the final candidates for ours (hierarchical) on six datasets (CounterFact, HotPotQA, Bios Profession, Epistemic Reasoning, TruthfulQA, and WikidataQA) are the same as $m = 20$ in Table 8. While for FEVER and Bios Gender, $m = 50$ for ours (hierarchical) does not gain any improvement in Acc. Similarly, there is no gain in Acc when $m = 100$. Therefore, $m = 20$ is adopted for Roberta.

B.2 Performance on Roberta

Table 9: Roberta evaluation with reported accuracy (Acc, %), Speedup, and peak memory (Mem) in MegaByte (MB). Ours achieves an average speedup of $26.6\times$, with comparable Acc and Mem to reproduced LASER†. Moreover, ours shows around $1.7\times$ lower Mem to LASER-Grad due to our forward-only passes.

Dataset	Baseline		LASER-Grad (std)			LASER (std)			Ours		
	Acc	Mem↓	Acc	Speedup↑	Mem↓	Acc	Mem↓	Acc†	Acc	Speedup↑	Mem↓
CounterFact	17.3	479 MB	19.3	0.86×	814 MB	19.3	489 MB	19.2	19.2	2.4×	493 MB
HotPotQA	6.1	481 MB	6.5	0.86×	824 MB	6.7	489 MB	6.8	6.3	32.9×	496 MB
FEVER	50.0	481 MB	52.7	0.86×	822 MB	52.3	489 MB	51.9	52.5	31.9×	496 MB
Bios Gender	87.5	495 MB	93.1	0.86×	893 MB	93.7	495 MB	93.5	93.4	38.7×	514 MB
Bios Profession	64.5	495 MB	75.1	0.86×	891 MB	72.5	495 MB	71.6	71.9	34.8×	514 MB
TruthfulQA	56.2	483 MB	56.3	0.85×	830 MB	56.2	490 MB	55.7	56.0	24.2×	498 MB
Epistemic Reasoning	37.1	492 MB	37.2	0.85×	877 MB	41.8	492 MB	37.3	37.1	13.1×	510 MB
WikidataQA	28.0	478 MB	32.7	0.86×	809 MB	30.7	489 MB	29.8	30.8	35.2×	492 MB
Average Speedup↑	-			0.86×			1×			**26.6**×	
Average Mem↓	486 MB			845 MB			491 MB			502 MB	

The reproduced accuracy† indicates a substantial difference for Epistemic Reasoning, a pattern similarly reflected in LASER-Grad (std) Sreeram et al. [2025].

B.3 The effectiveness of efficient hierarchical search

As presented in Table 8, we present the final candidates $[\boldsymbol{W}, l, k\%]$ for variants of our approach. We select the top-j $[\boldsymbol{W}, l]$ from our forward-only search, and then evaluate top-j $[\boldsymbol{W}, l]$ with 9 different retained top-$k\%$ ranks in $\mathcal{D}_{\text{subset}}$. Next, only the top-$j$ $[\boldsymbol{W}, l, k\%]$ from $\mathcal{D}_{\text{subset}}$ are directly evaluated on $\mathcal{D}_{\text{std}}$. The best $[\boldsymbol{W}, l, k\%]$ from $\mathcal{D}_{\text{std}}$ is the final candidate of ours (hierarchical). Therefore, if the top-j $[\boldsymbol{W}, l, k\%]$ from $\mathcal{D}_{\text{subset}}$ include the best $[\boldsymbol{W}, l, k\%]$ from $\mathcal{D}_{\text{std}}$, we call it *Hit*. In Roberta, there are 5 datasets (HotPotQA, FEVER, Bios Gender, Bios Profession, and WikidataQA) that have *Hit*, which means that ours (hierarchical) can get exactly the same final candidates with improving the evaluation efficiency (i.e., increases the Speedup as shown in Table 3) compared to ours (std). On the contrary, for the datasets (CounterFact, TruthfulQA, and Epistemic Reasoning) without *Hit*, the final candidates for ours (std) and ours (hierarchical) are different. However, as presented in Table 3, ours (hierarchical) can overall get comparable accuracy to ours (std) with greater Speedup.

B.4 The robustness of efficient hierarchical search across different $\mathcal{D}_{\text{subset}}$

Consider the forward-only search for weight matrices, namely, $\underset{\text{forward-only}}{\text{ratio}} \times \mathcal{D}_{\text{subset}}$ in Equation (7), smaller size of $\mathcal{D}_{\text{subset}}$ will lead to greater Speedup. Here, we try $\mathcal{D}_{\text{subset}} = 10, 5, 1$ in addition to the default 100. As presented in Table 10, ✔ means the top-j weight matrices are identical and preserve the same ordering as the weight matrices by default. $\checkmark_{\text{reordered}}$ means the top-j weight matrices are identical but with a different order from the weight matrices by default. Taking CounterFact with $\mathcal{D}_{\text{subset}} = 1$ as an example, the weight matrices are $\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 2], [\boldsymbol{W}_{in}, 3]\}$. Although the order is different from the default $\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 3], [\boldsymbol{W}_{in}, 2]\}$, the final candidate will not be influenced. Moreover, $\text{✘}\rightarrow \checkmark_{\text{ours}}$ means that the top-j weight matrices are not identical to the weight matrices by default, but get the same final candidates as in Table 8, by ours (hierarchical). Taking Bios Gender with $\mathcal{D}_{\text{subset}} = 1$ as an example, the weight matrices are $\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 9], [\boldsymbol{W}_{in}, 2]\}$. Although these weight matrices are different from the default $\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 9], [\boldsymbol{W}_{in}, 3]\}$, ours (hierarchical) still selects the $[\boldsymbol{W}_{in}, 9, 90\%]$ as the final candidate (same as in Table 8). Overall, ours (hierarchical) can speed up the forward-only search at the extreme case at $\mathcal{D}_{\text{subset}} = 1$, yielding the same final candidates at $\mathcal{D}_{\text{subset}} = 100$. Our forward-only search with efficient hierarchical search is robust to the extremely small size of $\mathcal{D}_{\text{subset}}$.

Another interesting observation is that from Table 10, the first two weight matrices, i.e., $\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0]\}$, are the same for different datasets with Roberta. However, as presented in Table 8, only WikidataQA selects one of these two weight matrices as the final candidate for ours (hierarchical). Instead, for other seven datasets, different final candidates are selected for the best Acc. Inspired by the recent study [Kaushik et al., 2026], the potential joint-candidates (such as $[\boldsymbol{W}_{out}, 11]$ or $[\boldsymbol{W}_{in}, 0]$) for Roberta on various datasets are left for future investigation.

Table 10: Weight matrices for our forward-only search on Roberta with $j = 5$ and $m = 20$, with different size of $\mathcal{D}_{\text{subset}} = 100\,(\text{default}), 10, 5, 1$. Our forward-only search with efficient hierarchical search is robust to the extremely small size of $\mathcal{D}_{\text{subset}}$.

Size of $\mathcal{D}_{\text{subset}}$	100	10	5	1
CounterFact	$\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 3], [\boldsymbol{W}_{in}, 2]\}$	✔	✔	$\checkmark_{\text{reordered}}$
HotPotQA	$\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 3], [\boldsymbol{W}_{in}, 2]\}$	✔	✔	✔
FEVER	$\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 3], [\boldsymbol{W}_{in}, 6]\}$	$\checkmark_{\text{reordered}}$	✔	✔
Bios Gender	$\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 9], [\boldsymbol{W}_{in}, 3]\}$	$\text{✘}\rightarrow \checkmark_{\text{ours}}$	$\text{✘}\rightarrow \checkmark_{\text{ours}}$	$\text{✘}\rightarrow \checkmark_{\text{ours}}$
Bios Profession	$\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 9], [\boldsymbol{W}_{in}, 3]\}$	$\checkmark_{\text{reordered}}$	✔	$\text{✘}\rightarrow \checkmark_{\text{ours}}$
TruthfulQA	$\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 2], [\boldsymbol{W}_{in}, 3]\}$	$\checkmark_{\text{reordered}}$	$\checkmark_{\text{reordered}}$	$\checkmark_{\text{reordered}}$
Epistemic Reasoning	$\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 9], [\boldsymbol{W}_{in}, 5]\}$	✔	✔	$\text{✘}\rightarrow \checkmark_{\text{ours}}$
WikidataQA	$\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 3], [\boldsymbol{W}_{in}, 2]\}$	✔	$\text{✘}\rightarrow \checkmark_{\text{ours}}$	$\text{✘}\rightarrow \checkmark_{\text{ours}}$

B.5 Various Rank-Reduction Strategies

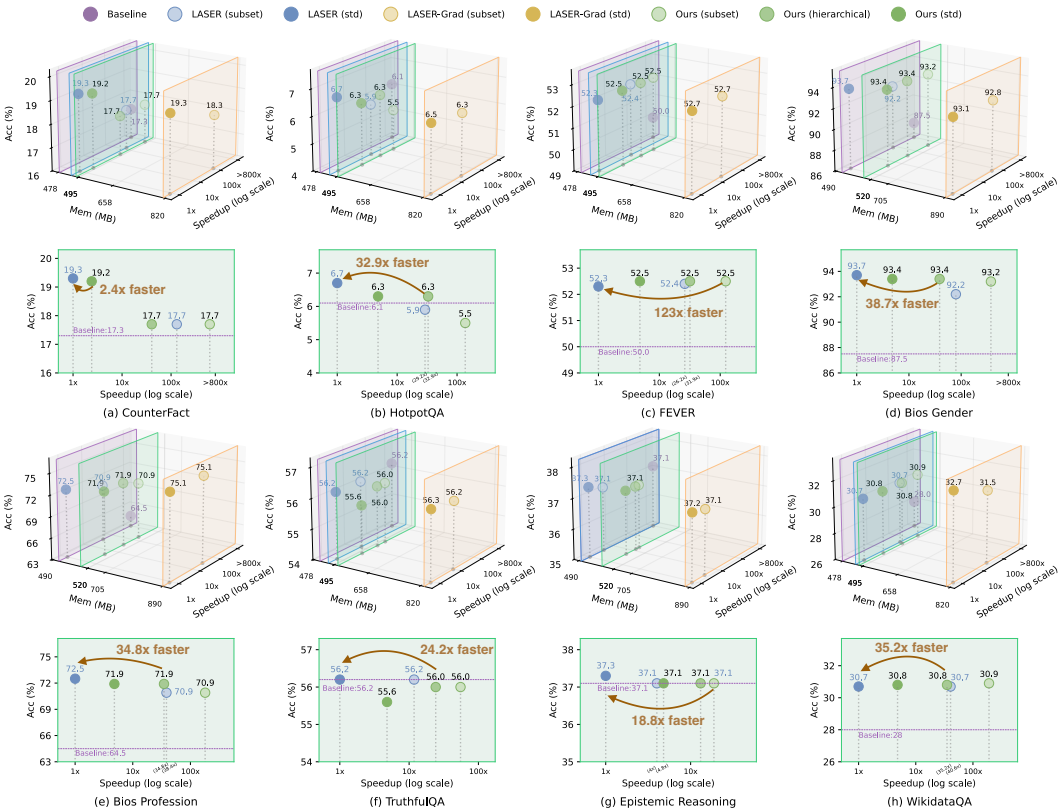


Figure 7: Ablation study on Roberta: our forward-only approach consumes lower peak memory compared to LASER-Grad [Sreeram et al., 2025], and achieves faster computation compared to LASER [Sharma et al., 2024][6], with comparable downstream accuracy.

C Speedup in Computational Complexity

In this section, we will calculate the ratio of our efficient and analytically equivalent forward-only search compared to the standard forward pass in terms of computational complexity. We refer to floating point operations (FLOPs) as a measure of theoretical computational complexity. Since the inference with large batch sizes in LLMs, especially during the prefill stage (similarly with next-token prediction for GPT-J and masked language modeling for Roberta), is dominated by compute-bound operations (e.g., large matrix multiplications), FLOPs are expected to correlate well with computational time in practice. Accordingly, the FLOPs of standard forward pass ($\boldsymbol{X}\boldsymbol{W}^T = \boldsymbol{Y}$) are $2nhd$.

C.1 Efficient Forward-Only Search

Efficient forward pass. For Equation (1) in Algorithm 1, instead of considering all SVD components in one forward pass, we can only consider activations induced by the last-m-rank SVD components ($\boldsymbol{Y}^{(r-m+1)}$ to $\boldsymbol{Y}^{(r)}$) in one efficient forward pass, plus the hidden states $\boldsymbol{Y}$ in one standard forward pass. In the efficient forward pass, the last-m-rank SVD components can be operated with batched outer product, as shown below:

$$(\boldsymbol{X}\boldsymbol{V}_{:,r-m+1:r}\boldsymbol{\Sigma}_{r-m+1:,r-m+1:})[:, \text{newaxis}, :] \odot \boldsymbol{U}_{:,r-m+1:r}[\text{newaxis}, :, :] = [\boldsymbol{Y}^{(r-m+1)}, ..., \boldsymbol{Y}^{(r)}], \quad (3)$$

where $\boldsymbol{\Sigma}_{r-m+1:,r-m+1:} = \text{diag}(\sigma_{r-m+1}, \sigma_{r-m+2}, ..., \sigma_r) \in \mathbb{R}^{m\times m}$, $\boldsymbol{V}_{:,r-m+1:r} = [\boldsymbol{v}_{r-m+1}, \boldsymbol{v}_{r-m+2}, ..., \boldsymbol{v}_r] \in \mathbb{R}^{d\times m}$, and $\boldsymbol{U}_{:,r-m+1:r} = [\boldsymbol{u}_{r-m+1}, \boldsymbol{u}_{r-m+2}, ..., \boldsymbol{u}_r] \in \mathbb{R}^{h\times m}$. $\odot$

[6]For Epistemic Reasoning, the reproduced accuracy† in Table 9, is exploited.

denotes the Hadamard (element-wise) product, where the elements are broadcasted before/after $\odot$, and therefore $[\boldsymbol{Y}^{(r-m+1)}, ..., \boldsymbol{Y}^{(r)}] \in \mathbb{R}^{n \times h \times m}$. For each SVD component, this element-wise product with broadcasting is equivalent to the Kronecker product in Equation (1).

Efficient angle concentration. For Equation (2), given $\widetilde{\boldsymbol{Y}}^{(i)}_{q,:} = \frac{\boldsymbol{Y}^{(i)}_{q,:}}{\left\|\boldsymbol{Y}^{(i)}_{q,:}\right\|}$ and $\widetilde{\boldsymbol{Y}}^{(i)}_{p,:} = \frac{\boldsymbol{Y}^{(i)}_{p,:}}{\left\|\boldsymbol{Y}^{(i)}_{p,:}\right\|}$, by exploiting the identity that the sum of pairwise cosine similarities (excluding self-pairs) equals the squared norm of the summed normalized directions minus n, as well as considering $\left\|\widetilde{\boldsymbol{Y}}^{(i)}_{q,:}\right\|^2 = 1$, the angle concentration for the hidden states of each SVD component can be reformulated as follows ($p \neq q$ and $n \geq 2$):

$$\text{Angle Concentration}(\boldsymbol{Y}^{(i)}) = \frac{\left\|\sum_{q=1}^{n} \widetilde{\boldsymbol{Y}}^{(i)}_{q,:}\right\|^2 - n}{n \times (n-1)}; \sum_{q=1}^{n}\sum_{p=1}^{n} \widetilde{\boldsymbol{Y}}^{(i)}_{q,:} \widetilde{\boldsymbol{Y}}^{(i)^T}_{p,:} = \left\|\sum_{q=1}^{n} \widetilde{\boldsymbol{Y}}^{(i)}_{q,:}\right\|^2 - \sum_{q=1}^{n}\left\|\widetilde{\boldsymbol{Y}}^{(i)}_{q,:}\right\|^2, \tag{4}$$

which reduces the computational complexity from $\mathcal{O}(hn^2)$ in Equation (2) to $\mathcal{O}(hn)$.

C.1.1 Speedup in Efficient Forward Pass

We recall Equation (3) for the last-m-rank SVD components:

$$(\boldsymbol{X}\boldsymbol{V}_{:,r-m+1:r}\boldsymbol{\Sigma}_{r-m+1:,r-m+1:})[:, \text{newaxis}, :] \odot \boldsymbol{U}_{:,r-m+1:r}[\text{newaxis}, :, :] = [\boldsymbol{Y}^{(r-m+1)}, ..., \boldsymbol{Y}^{(r)}].$$

The FLOPs for $\boldsymbol{X}\boldsymbol{V}_{:,r-m+1:r}\boldsymbol{\Sigma}_{r-m+1:,r-m+1:}$ is $2ndm + nm$ and the FLOPs for element-wise product by $\boldsymbol{U}_{:,r-m+1:r}[\text{newaxis}, :, :]$ is nhm. Considering only the input matrix $\boldsymbol{W}_{in}$ and the output matrix $\boldsymbol{W}_{out}$ in the feed-forward network with $r = \min(h, d)$, and $\max(h, d) = 4 \times \min(h, d)$,[7] the ratio of the efficient forward pass compared to the standard forward pass is as follows:

$$\frac{2ndm + nm + nhm}{2nhd} \approx \begin{cases} \frac{6m}{2\times\max(h,d)} \approx 0 & \text{for } \boldsymbol{W}_{in} \\ \frac{9m}{2\times\max(h,d)} \approx 0 & \text{for } \boldsymbol{W}_{out}, \end{cases}$$

where $m \ll \max(h, d)$ as presented in Appendix A.2, B.1, and C.2. Therefore, the calculation of the last-m-rank SVD components does not introduce any significant extra computational complexity. The FLOPs of the efficient forward passes are almost zero to the standard forward pass.

Note that alternatively, we can use only-one-forward pass to get the hidden states $\boldsymbol{Y}$ and the activations induced by the last-m-rank SVD components, i.e., $\boldsymbol{Y}^{(r-m+1)}$ to $\boldsymbol{Y}^{(r)}$ at the same time, as shown below:

$$(\boldsymbol{X}\boldsymbol{V}\boldsymbol{\Sigma})[:, \text{newaxis}, :] \odot \boldsymbol{U}[\text{newaxis}, :, :] = [\boldsymbol{Y}^{(1)}, \boldsymbol{Y}^{(2)}, ..., \boldsymbol{Y}^{(r)}],$$

where the FLOPs for $\boldsymbol{X}\boldsymbol{V}\boldsymbol{\Sigma}$ is $2ndr+nr$ and the FLOPs for element-wise product by $\boldsymbol{U}[\text{newaxis}, :, :]$ is nhr. In addition, for the hidden states $\boldsymbol{Y} = \text{sum}([\boldsymbol{Y}^{(1)}, \boldsymbol{Y}^{(2)}, ..., \boldsymbol{Y}^{(r)}])$, the FLOPs is nhr. Considering only the input matrix $\boldsymbol{W}_{in}$ and the output matrix $\boldsymbol{W}_{out}$ in the feed-forward network with $r = \min(h, d)$, and $\max(h, d) = 4 \times \min(h, d)$, the ratio of only-one-forward pass compared to the standard forward pass is as follows:

$$\frac{2ndr + nr + 2nhr}{2nhd} \approx \frac{2nr(d+h)}{2nhd} = 1.25.$$

The ratio of 1.25 means that only-one-forward pass introduces an extra 25% of computational complexity compared to the standard forward pass. At the same time, this alternative, namely, only-one-forward pass, potentially introduces extra dynamic memory consumption for our approach as shown in Table 1 because m is increased to r, where $m \ll r$.

C.1.2 Speedup in Efficient Angle Concentration

We recall Equation (2) for the angle concentration:

$$\text{Angle Concentration}(\boldsymbol{Y}^{(i)}) = \frac{1}{n \times (n-1)} \sum_{q=1}^{n}\sum_{p=1}^{n} \frac{\boldsymbol{Y}^{(i)}_{q,:}\boldsymbol{Y}^{(i)^T}_{p,:}}{\left\|\boldsymbol{Y}^{(i)}_{q,:}\right\| \cdot \left\|\boldsymbol{Y}^{(i)}_{p,:}\right\|} \quad (q \neq p)$$

[7] The ratio of 4 is applied to GPT-J and Roberta. Although for other LLMs such as LLaMA Touvron et al. [2023] this ratio is different, the trend is still kept, i.e., the efficient forward pass does not incur extra computational complexity.

where $\boldsymbol{Y}^{(i)}_{q,:} \in \mathbb{R}^{1\times h}$ is the q-th row of $\boldsymbol{Y}^{(i)}$ and $\boldsymbol{Y}^{(i)}_{p,:} \in \mathbb{R}^{1\times h}$ is the p-th row of $\boldsymbol{Y}^{(i)}$. Similarly, the angle concentration of the hidden states $\boldsymbol{Y}$, defined as the sum of activations induced by all SVD components, is given by $\text{Angle Concentration}(\boldsymbol{Y})$.

For the FLOPs of $\left\|\boldsymbol{Y}^{(i)}_{q,:}\right\|$, the square of n rows of $\boldsymbol{Y}^{(i)}$ is nh on Multiplication (Mul) and the sum is $n(h-1)$ on Addition (Add). The square root only executes 1 time for each row of $\boldsymbol{Y}^{(i)}$, which is much smaller than the square and the sum in terms of FLOPs. Next, the $\boldsymbol{Y}^{(i)}_{q,:}{\boldsymbol{Y}^{(i)}_{p,:}}^T$ needs h Mul and $h-1$ Add, as well as 1 Mul in $\left\|\boldsymbol{Y}^{(i)}_{q,:}\right\| \cdot \left\|\boldsymbol{Y}^{(i)}_{p,:}\right\|$. The division also only executes 1 time for each q,p pair, which is much smaller than $\boldsymbol{Y}^{(i)}_{q,:}{\boldsymbol{Y}^{(i)}_{p,:}}^T$ in terms of FLOPs. Then, the 1 Add is consumed for being added to the sum for each q,p pair. Moreover, the FLOPs of square root and division can be neglected. Therefore, the FLOPs of the standard calculation for one angle concentration is $2n^2(h+1)-2n$. For last-m-rank SVD components and the hidden states $\boldsymbol{Y}$, the ratio of this standard angle concentration compared to the standard forward pass is as follows:

$$\frac{(m+1)\times\left(2n^2(h+1)-2n\right)}{2nhd} = \frac{(m+1)\times(nh+n-1)}{hd} \approx \underset{\text{with } n\ll h}{\frac{(m+1)\times n}{d}}.$$

In addition, using the symmetry property in Equation (2), i.e., only calculating unique $q-p$ pairs, will reduce the above ratio to half, i.e., $\frac{(m+1)\times n}{2d}$. But still, this introduced computational complexity scales linearly with the sequence lengths n, which can not be neglected, e.g., when $m=20$, $n=100$, and $d=768$ for Roberta with $\boldsymbol{W}_{in}$ on Bios Profession, $\frac{(m+1)\times n}{2d} = 1.37$.

Instead, as shown in Equation (4) below, for $\widetilde{\boldsymbol{Y}}^{(i)}_{q,:} = \frac{\boldsymbol{Y}^{(i)}_{q,:}}{\left\|\boldsymbol{Y}^{(i)}_{q,:}\right\|}$ or $\widetilde{\boldsymbol{Y}}^{(i)}_{p,:} = \frac{\boldsymbol{Y}^{(i)}_{p,:}}{\left\|\boldsymbol{Y}^{(i)}_{p,:}\right\|}$, the FLOPs is $nh + n(h-1) + nh$. In addition, the FLOPs for $\left\|\sum_{q=1}^{n}\widetilde{\boldsymbol{Y}}^{(i)}_{q,:}\right\|^2$ is $h(n-1)+h+h$.

$\text{Angle Concentration}(\boldsymbol{Y}^{(i)}) = \frac{\left\|\sum_{q=1}^{n}\widetilde{\boldsymbol{Y}}^{(i)}_{q,:}\right\|^2 - n}{n\times(n-1)}$; $\sum_{q=1}^{n}\sum_{p=1}^{n}\widetilde{\boldsymbol{Y}}^{(i)}_{q,:}{\widetilde{\boldsymbol{Y}}^{(i)}_{p,:}}^T = \left\|\sum_{q=1}^{n}\widetilde{\boldsymbol{Y}}^{(i)}_{q,:}\right\|^2 - \sum_{q=1}^{n}\left\|\widetilde{\boldsymbol{Y}}^{(i)}_{q,:}\right\|^2$.

Overall, the FLOPs of the efficient calculation for one angle concentration is $4hn+h-n$, which reduces the computational complexity from $\mathcal{O}(hn^2)$ in Equation (2) to $\mathcal{O}(hn)$. For last-m-rank SVD components and the hidden states $\boldsymbol{Y}$, the ratio of efficient angle concentration compared to the standard forward pass is as follows:

$$\frac{(m+1)\times(4hn+h-n)}{2nhd} = \frac{2(m+1)}{d} + \frac{(m+1)}{2nd} - \frac{(m+1)}{2hd} \approx \underset{\text{with } m\ll d}{0}.$$

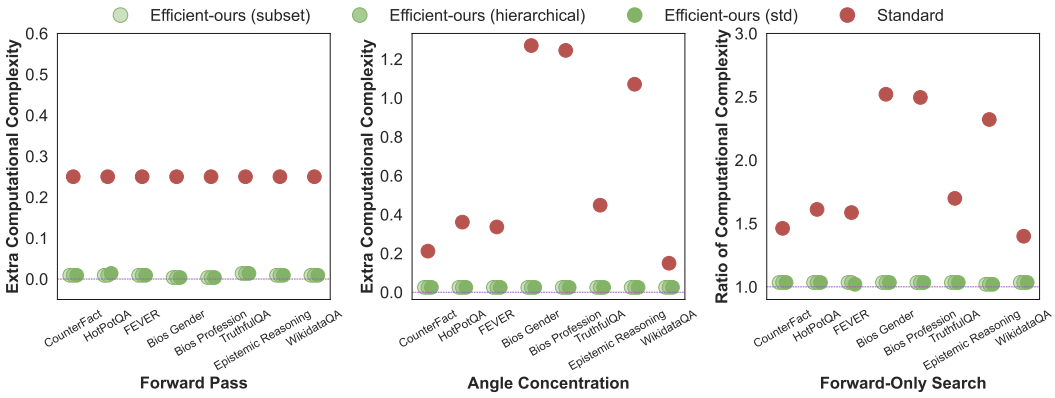
C.1.3 Computational Complexity of Efficient Forward-Only Search

Based on the analysis of speedup in efficient forward pass in Appendix C.1.1 and efficient angle concentration in Appendix C.1.2, we have the ratio of the efficient forward pass for the last-m-rank SVD components compared to the standard forward pass is $\begin{cases}\frac{6m}{2\times\max(h,d)} \approx 0 & \text{for } \boldsymbol{W}_{in}\\ \frac{9m}{2\times\max(h,d)} \approx 0 & \text{for } \boldsymbol{W}_{out},\end{cases}$; and the ratio of efficient angle concentration for the last-m-rank SVD components and the hidden states $\boldsymbol{Y}$ compared to the standard forward pass is $\frac{2(m+1)}{d} + \frac{(m+1)}{2nd} - \frac{(m+1)}{2hd} \approx \underset{\text{with } m\ll d}{0}$. Therefore, the ratio of our efficient forward-only search compared to the standard forward pass in terms of computational complexity is:

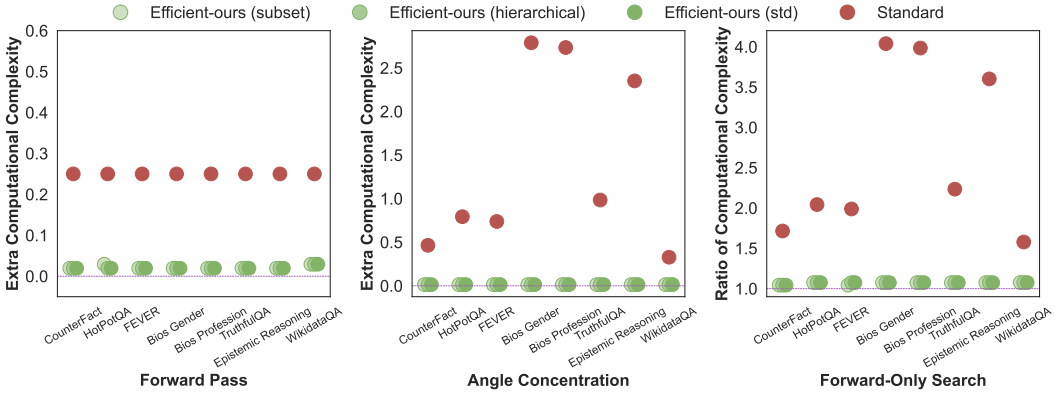
$$\text{ratio} = 1 + \begin{cases}\frac{6m}{2\times\max(h,d)} \approx 0 & \text{for } \boldsymbol{W}_{in}\\ \frac{9m}{2\times\max(h,d)} \approx 0 & \text{for } \boldsymbol{W}_{out},\end{cases} + \underset{\text{with } m\ll d}{0} \approx 1. \tag{5}$$

Moreover, Fig. 8 shows that the only-one-forward pass and the standard angle concentration both introduce non-negligible extra computational complexity. Furthermore, the ratio of our efficient forward-only search compared to the standard forward pass in computational complexity is approximately 1 for both Roberta and GPT-J on eight datasets.[8]

[8]For ● Standard in angle concentration, we consider $\boldsymbol{W}_{in}$ for presentation. While for $\boldsymbol{W}_{out}$, the trend is still kept, i.e., the standard angle concentration introduces non-negligible extra computational complexity.



(a) GPT-J



(b) Roberta

Figure 8: Compared to the standard forward pass, our efficient forward pass introduces negligible extra computational complexity. Compared to the standard angle concentration, our efficient angle concentration does not incur extra computational complexity. Overall, the ratio of our efficient forward-only search compared to the standard forward pass in computational complexity is approximately 1 for both (a) GPT-J and (b) Roberta on eight datasets.

Furthermore, our experimental setting in Section 3.1 only considers the weight matrices $\boldsymbol{W}_{in/out}$ in the feed-forward network of the Transformer architecture, rather than all the weight matrices in the Transformer architecture. **Therefore, the actual computational complexity of our efficient forward pass plus our efficient angle concentration compared to the standard forward pass in LLMs is significantly smaller than** $\begin{cases} \frac{6m}{2\times\max(h,d)} \approx 0 & \text{for } \boldsymbol{W}_{in} \\ \frac{9m}{2\times\max(h,d)} \approx 0 & \text{for } \boldsymbol{W}_{out}, \end{cases} + \underset{\text{with } m \ll d}{0}$. The ratio of our efficient forward-only search compared to the standard forward pass in terms of computational complexity for LLMs in our paper's experimental setting is almost 1.

C.2 Computational Complexity Comparison Across Methods

Table 11: The size of $\mathcal{D}_{\text{subset}}$, $\mathcal{D}_{\text{std}}$, $\mathcal{D}$, and the average length of sequence n for evaluated datasets.

Dataset	$\mathcal{D}_{\text{subset}}$ (100 from $\mathcal{D}_{\text{std}}$)	Sequence Length n	$\mathcal{D}_{\text{std}}$ (20% of $\mathcal{D}$)	$\mathcal{D}$
CounterFact	100	17	13151	65757
HotPotQA	100	29	2923	14618
FEVER	100	27	2617	13086
Bios Gender	100	102	7928	39642
Bios Profession	100	100	3844	19223
TruthfulQA	100	36	1176	5882
Epistemic Reasoning	100	86	400	2000
WikidataQA	100	12	4064	20321

When referring to the comparison of computational complexity, we consider the speedup in search steps, excluding the computation of the test stage, i.e., over 80% of the whole dataset $\mathcal{D}$. The sequence length n is the averaged value across $\mathcal{D}_{\text{subset}}$.

For **LASER** Sharma et al. [2024],

$$\text{LASER (Std)} : \underset{\text{layers}}{L} \times \underset{\boldsymbol{W}_{\text{in/out}}}{2} \times \underset{\text{rates}}{9} \times \mathcal{D}_{\text{std}}$$
$$\text{LASER (subset)} : \underset{\text{layers}}{L} \times \underset{\boldsymbol{W}_{\text{in/out}}}{2} \times \underset{\text{rates}}{9} \times \mathcal{D}_{\text{subset}},$$

where $L = 12$ for Roberta and $L = 28$ for GPT-J.

For **LASER-Grad** Sreeram et al. [2025], LASER-Grad (std) has $j = 7$ and LASER-Grad (subset) $j = 7$ for Roberta and $j = 5$ for GPT-J.

$$\text{LASER-Grad (std)} : \underset{\text{forward and backward}}{2.5 \times 4 \times \mathcal{D}_{\text{subset}}} + \underset{\text{top-j}}{j} \times \underset{\text{clustering}}{4} \times \underset{\text{rates}}{9} \times \mathcal{D}_{\text{std}}$$
$$\text{LASER-Grad (subset)} : \underset{\text{forward and backward}}{2.5 \times 4 \times \mathcal{D}_{\text{subset}}} + \underset{\text{top-j}}{j} \times \underset{\text{clustering}}{4} \times \underset{\text{rates}}{9} \times \mathcal{D}_{\text{subset}}.$$

For our proposed forward-only approach (**ours**), the computational complexity of our efficient forward-only search compared to the standard forward pass is denoted as in Equation (5) and Fig. 8, where m is the last-m-rank SVD components and d is the dimension in a weight matrix $\boldsymbol{W} \in \mathbb{R}^{h \times d}$,

$$\text{ratio} = 1 + \begin{cases} \frac{6m}{2\times\max(h,d)} + \frac{2(m+1)}{d} + \frac{(m+1)}{2nd} - \frac{(m+1)}{2hd} & \text{for } \boldsymbol{W}_{in} \\ \frac{9m}{2\times\max(h,d)} + \frac{2(m+1)}{d} + \frac{(m+1)}{2nd} - \frac{(m+1)}{2hd} & \text{for } \boldsymbol{W}_{out}, \end{cases} \approx 1 \tag{6}$$

where $m = 20$ for Roberta and $m = 50$ for GPT-J.[9] Next, we will analyze the computational complexity for variants of rank-reduction strategy in our forward-only approach.

C.2.1 Efficient Hierarchical Search

As we mentioned in Section 2.1, the rank-reduction strategy is determined based on the accuracy evaluated across the validation datasets. In other words, the rank-reduction strategy varies across different sizes of datasets. As shown in Algorithm 2, Algorithm 3, and Algorithm 4, we present the variants of rank-reduction strategy in our forward-only approach, i.e., ours (subset), ours (std), ours (hierarchical) correspondingly. For ours (subset) in Algorithm 2, the validation dataset is $\mathcal{D}_{\text{subset}}$, which is only 100 samples selected from $\mathcal{D}_{\text{std}}$, as shown in Table 11. For ours (std), the validation dataset is $\mathcal{D}_{\text{std}}$ in Algorithm 3, which is 20% samples from the whole dataset $\mathcal{D}$, as shown in Table 11. For ours (hierarchical) in Algorithm 4, we exploit the computational efficiency over $\mathcal{D}_{\text{subset}}$ and then only evaluate the top-j weight matrices $\widehat{\boldsymbol{W}}$ over $\mathcal{D}_{\text{std}}$, avoiding the costly evaluation with all weight matrices $\widehat{\boldsymbol{W}}$ over $\mathcal{D}_{\text{std}}$.

[9] $m = 20$ is enough for Bios Gender and Bios Profession.

Algorithm 2 Ours (subset)

within candidate $\widehat{\boldsymbol{W}}$ from Algorithm 1
  # efficient evaluation
  Best $\widehat{\boldsymbol{W}} \leftarrow \{\text{evaluate over } \mathcal{D}_{\text{subset}}\}$
Return: LLM with the best $\widehat{\boldsymbol{W}}$

Algorithm 3 Ours (std)

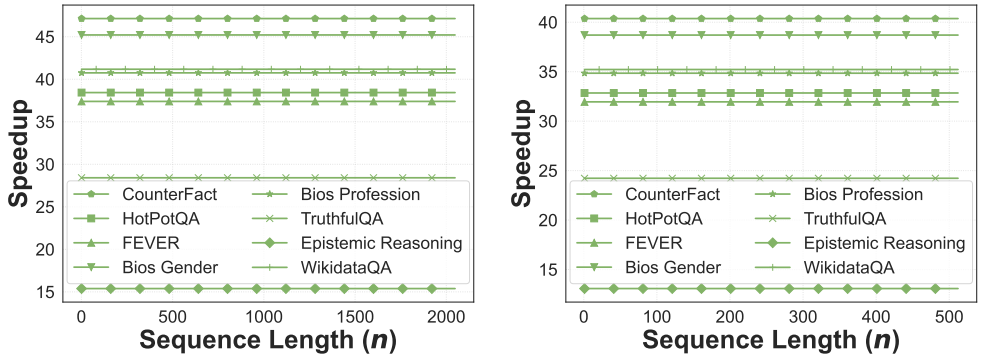
within candidate $\widehat{\boldsymbol{W}}$ from Algorithm 1
  # costly evaluation
  Best $\widehat{\boldsymbol{W}} \leftarrow \{\text{evaluate over } \mathcal{D}_{\text{std}}\}$
Return: LLM with the best $\widehat{\boldsymbol{W}}$

Algorithm 4 Ours (hierarchical)

for each $\widehat{\boldsymbol{W}}$ from Algorithm 1 **do**
  # efficient evaluation with Set $\mathbb{S}'$
  $\mathbb{S}' \leftarrow \mathbb{S}' \cup \left\{ \left(\widehat{\boldsymbol{W}}, \text{ evaluate over } \mathcal{D}_{\text{subset}} \right) \right\}$
end for
within top-j $\widehat{\boldsymbol{W}}$ in $\mathbb{S}'$
  # efficient evaluation with only top-j
  Best $\widehat{\boldsymbol{W}} \leftarrow \{\text{evaluate over } \mathcal{D}_{\text{std}}\}$
Return: LLM with the best $\widehat{\boldsymbol{W}}$

Consequently, we have the computational complexity for variants of the rank-reduction strategy in our forward-only approach, where $j = 5$ for Roberta[10] and $j = 10$ for GPT-J:

$$\begin{aligned}
\text{Ours (hierarchical)} &: \underset{\text{forward-only}}{\text{ratio}} \times \mathcal{D}_{\text{subset}} + \underset{\text{top-j}}{j} \times \underset{\text{rates}}{9} \times \mathcal{D}_{\text{subset}} + \underset{\text{top-j}}{j} \times \mathcal{D}_{\text{std}} \\
\text{Ours (std)} &: \underset{\text{forward-only}}{\text{ratio}} \times \mathcal{D}_{\text{subset}} + \underset{\text{top-j}}{j} \times \underset{\text{rates}}{9} \times \mathcal{D}_{\text{std}} \\
\text{Ours (subset)} &: \underset{\text{forward-only}}{\text{ratio}} \times \mathcal{D}_{\text{subset}} + \underset{\text{top-j}}{j} \times \underset{\text{rates}}{9} \times \mathcal{D}_{\text{subset}}.
\end{aligned} \tag{7}$$

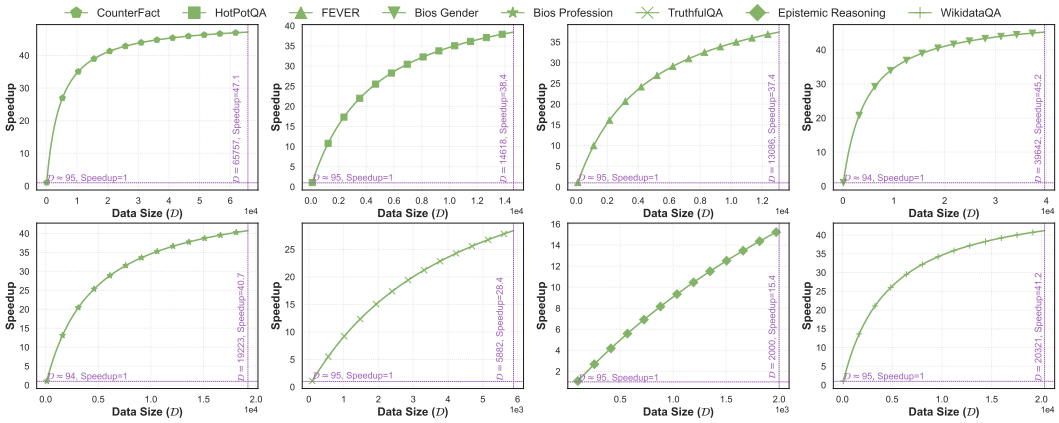


(a) GPT-J (b) Roberta

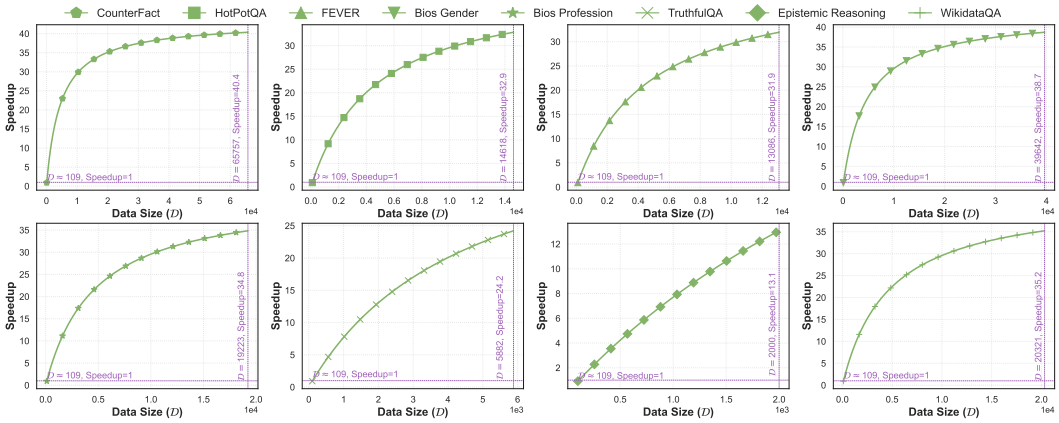
Figure 9: The Speedup with different sequence lengths n for ours (hierarchical). Our forward-only approach achieves greater Speedup across different n.

[10]Only $j = 10$ for ours (std) with CounterFact on Roberta.

Besides, we also show the Speedup with assumed different data size $\mathcal{D}$ for ours (hierarchical). Our forward-only approach achieves greater Speedup as the data size $\mathcal{D}$ increases for both (a) GPT-J and (b) Roberta on eight datasets, as demonstrated in Fig. 10



(a) GPT-J



(b) Roberta

Figure 10: The Speedup with assumed different data size $\mathcal{D}$ for ours (hierarchical). Our forward-only approach achieves greater Speedup as the data size $\mathcal{D}$ increases for both (a) GPT-J and (b) Roberta.

D The Declaration of AI Usage

The icon (cheetah) is generated with ChatGPT [Betker et al., 2023, Achiam et al., 2023]. In addition, we use LLMs only for polishing the writing, and it does not impact the core methodology, scientific rigorousness, or originality of the research.

E Peak Memory Consumption

From Table 1, we analyze the peak memory consumption for conceptualized fully connected layers across baseline (direct inference), LASER [Sharma et al., 2024], LASER-Grad [Sreeram et al., 2025], and our forward-only approach. In this section, we examine in detail the peak memory consumption specifically for Transformer-based Vaswani et al. [2017] LLMs, with the default precision in Float32 (4 bytes per data point) for Roberta and in Float16 (2 bytes per data point) for GPT-J.

Table 12: The settings for Roberta Liu et al. [2019] and GPT-J Wang and Komatsuzaki [2021], where vocab_size (the size of vocabulary), hidden_size (the dimension of the hidden layer), L (the number of stacked Transformer modules), attention_head (the number of attention heads).

Setting	vocab_size	hidden_size	attention_head	L	Max Sequence Length
Roberta	50265	768	12	12	512
GPT-J	50400	4096	16	28	2048

E.1 Roberta

The input sequence is firstly processed with an input embedding and a positional encoding, and then passed into L stacked Transformer modules, finally with a fully-connected-layer block called the LLM head. Moreover, each Transformer module has a multi-head self-attention and a feed-forward network. We assume that the input sequence is $\mathbb{R}^{n\times\text{hidden_size}}$ and the batch size is 1. In the multi-head self-attention,

$$\text{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{\text{head_dimension}}})\boldsymbol{V},$$

where $\text{head_dimension} = \frac{\text{hidden_size}}{\text{attention_head}}$ and the generated matrices $\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}$ are all in $\mathbb{R}^{n\times\text{hidden_size}}$. In this multi-head self-attention, the attention matrix $\text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{\text{head_dimension}}})$ consumes memory quadratically, i.e., in $\mathbb{R}^{n\times n\times\text{attention_head}}$. After multiplication with $\boldsymbol{V}$, the attention output ($\mathbb{R}^{n\times\text{hidden_size}}$) with a linear projection, is passed into the feed-forward network. In the feed-forward network, we have the input matrix $\boldsymbol{W}_{in}\in\mathbb{R}^{(4\times\text{hidden_size})\times\text{hidden_size}}$ and the output matrix $\boldsymbol{W}_{out}\in\mathbb{R}^{\text{hidden_size}\times(4\times\text{hidden_size})}$. In this paper, we also only search for $\boldsymbol{W}_{in/out}$ as did in LASER [Sharma et al., 2024] and LASER-Grad [Sreeram et al., 2025]. After the stacked Transformer modules, the LLM head generates the output, in $\mathbb{R}^{n\times\text{vocab_size}}$, over the vocabulary.

For **Baseline** (direct inference), the LLM is loaded with parameters (mainly weight memory). For $\underset{\text{Roberta}}{}$ dynamic memory, mainly the generated activations in Transformer modules, we consider the maximum peak memory in one Transformer module because this part of the maximum peak memory is the same for L stacked Transformer modules. Then, we need to compare the maximum peak memory for Transformer modules and the peak memory in LLM head. Usually, the attention matrix occupies the peak memory in Transformer modules, and the output in LLM head also occupies lots of memory. For simplicity, we exclude the effect of the input embedding and the positional encoding, skip connection, and LayerNorm, because they can be ignored when compared to the attention matrix and the output in LLM head. Furthermore, for the layers with $\boldsymbol{W}_{in}$ or $\boldsymbol{W}_{out}$, and activation function (GELU Activation for Roberta Liu et al. [2019]), in the feed-forward network, the maximum activations are the same, i.e., in $\mathbb{R}^{n\times(4\times\text{hidden_size})}$.

$$\underset{\text{one_Transformer_module}}{\text{max_baseline}} \leftarrow \max\Big\{\underset{\text{attention matrix}}{n\times n\times\text{attention_head}}, \underset{\text{feed-forward network}}{n\times(4\times\text{hidden_size})}\Big\},$$

$$\underset{\text{Roberta}}{\textbf{Baseline}} : \underset{\text{LLM}}{\text{Parameters}} + \max\Big\{\underset{\text{one_Transformer_module}}{\text{max_baseline}}, \underset{\text{LLM head}}{n\times\text{vocab_size}}\Big\},$$

where the $\underset{\text{one_Transformer_module}}{\text{max_baseline}}$ is simplified because the input to each Transformer module, the output of multi-head self-attention, and the output of the feed-forward network are all in $\mathbb{R}^{n\times\text{hidden_size}}$. Moreover, the memory for $\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}$ are in $\mathbb{R}^{n\times(3\times\text{hidden_size})}$, which is smaller than $\underset{\text{feed-forward network}}{n\times(4\times\text{hidden_size})}$.

For **LASER** Sharma et al. [2024], the LLM is also loaded with parameters (mainly weight memory). For dynamic memory, we also only need to consider the maximum peak memory of one Transformer module. In addition to the generated activations as in **Baseline**, the SVD decomposition $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is introduced, which changes the dynamic memory. Because of only considering $\boldsymbol{W}_{in/out}$ and taking $\boldsymbol{W}_{in}$ for example, the introduced extra dynamic memory for $\boldsymbol{U}$ is in $\mathbb{R}^{(4\times\text{hidden_size})\times r}$, for $\boldsymbol{\Sigma}$ is in $\mathbb{R}^{r\times r}$, and for $\boldsymbol{V}^T$ is in $\mathbb{R}^{r\times\text{hidden_size}}$, where $r = \text{hidden_size}$. Totally, the introduced extra dynamic memory is in $\mathbb{R}^{(4\times\text{hidden_size})\times r+r\times r+r\times\text{hidden_size}}$ for both $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_{out}$.

$$\begin{aligned}\underset{\text{one_Transformer_module}}{\text{max_LASER}} \leftarrow &\max\Big\{\underset{\text{attention matrix}}{n\times n\times\text{attention_head}},\\ &\underset{\text{feed-forward network}}{n\times(4\times\text{hidden_size})}+\underset{\text{SVD}}{(5\times\text{hidden_size}+r)\times r}\Big\},\\ \underset{\text{Roberta}}{\textbf{LASER}}:\underset{\text{LLM}}{\text{Parameters}}+\max\Big\{&\underset{\text{one_Transformer_module}}{\text{max_LASER}},\underset{\text{LLM head}}{n\times\text{vocab_size}}\Big\}.\end{aligned}$$

Similarly, we exclude the effect of the input embedding and the positional encoding, skip connection, LayerNorm, and activation function, as in $\underset{\text{Roberta}}{\textbf{Baseline}}$. The $\underset{\text{one_Transformer_module}}{\text{max_LASER}}$ is also simplified by excluding the input to each Transformer module, the output of multi-head self-attention, and the output of the feed-forward network, as well as $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$.

For $\underset{\text{Roberta}}{\textbf{Ours}}$, the parameters (mainly weight memory) are the same as $\underset{\text{Roberta}}{\textbf{Baseline}}$ and $\underset{\text{Roberta}}{\textbf{LASER}}$. For dynamic memory, in addition to the generated activations as in $\underset{\text{Roberta}}{\textbf{Baseline}}$ and the SVD decomposition $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ as in $\underset{\text{Roberta}}{\textbf{LASER}}$, the extra m SVD components change the dynamic memory. When only considering the maximum generated activations within $\boldsymbol{W}_{in/out}$ in $\mathbb{R}^{n\times(4\times\text{hidden_size})}$, the introduced extra dynamic memory of m SVD components is in $\mathbb{R}^{m\times n\times(4\times\text{hidden_size})}$. Therefore, the introduced maximum dynamic memory here is totally in $\mathbb{R}^{(m+1)\times n\times(4\times\text{hidden_size})}$.

$$\begin{aligned}\underset{\text{one_Transformer_module}}{\text{max_ours}} \leftarrow &\max\Big\{\underset{\text{attention matrix}}{n\times n\times\text{attention_head}},\\ &\underset{\substack{\text{feed-forward network with}\\ m\text{ singular components}}}{(m+1)\times n\times(4\times\text{hidden_size})}+\underset{\text{SVD}}{(5\times\text{hidden_size}+r)\times r}\Big\},\\ \underset{\text{Roberta}}{\textbf{Ours}}:\underset{\text{LLM}}{\text{Parameters}}+\max\Big\{&\underset{\text{one_Transformer_module}}{\text{max_ours}},\underset{\text{LLM head}}{n\times\text{vocab_size}}\Big\}.\end{aligned}$$

Again, we not only exclude the effect of the input embedding and the positional encoding, skip connection, LayerNorm, activation function, as in $\underset{\text{Roberta}}{\textbf{Baseline}}$, but also simplify the $\underset{\text{one_Transformer_module}}{\text{max_ours}}$ by excluding the input to each Transformer module, the output of multi-head self-attention, and the output of the feed-forward network, as well as $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$, as in $\underset{\text{Roberta}}{\textbf{LASER}}$.

For $\underset{\text{Roberta}}{\textbf{LASER-Grad}}$ Sreeram et al. [2025], the generated activations in the forward pass and the SVD decomposition $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, as well as LLM parameters are required to be stored in memory for getting the gradient of weights $\boldsymbol{W}$ and the corresponding singular values $\boldsymbol{\Sigma}$ in the backward pass. In other words, we need to store the generated activations that are prepared to backpropagate the gradients. In this case, we also need to include the intermediate generated activations in the analysis of dynamic memory. More specifically, when the input sequence is in $\mathbb{R}^{n\times\text{hidden_size}}$, for each Transformer module, there are $12\times$ (considering $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$, LayerNorm, activation function, the input to each Transformer module, and the output of multi-head self-attention) the memory of the input sequence, adding with $2\times$ the memory of the attention weights (due to Softmax). For the feed-forward network with $\boldsymbol{W}_{in/out}$, based on the memory analysis in Zhang et al. [2024], we assume that the generated activations in the forward pass and the SVD decomposition could be released once the gradient of weights $\boldsymbol{W}$ and the corresponding singular values $\boldsymbol{\Sigma}$ in the backward pass are available. Therefore, we get the maximum memory for the layers with $\boldsymbol{W}_{in/out}$ in the feed-forward network, which are

denoted as follows,

$$\max_\text{LASER-Grad}_{\boldsymbol{W}_{in}} \leftarrow \max\Big\{ \underbrace{n \times (4 \times \text{hidden_size})}_{\text{feed-forward network}} + \underbrace{(5 \times \text{hidden_size} + r) \times r}_{\text{SVD}},$$
$$\underbrace{(4 \times \text{hidden_size}) \times \text{hidden_size}}_{\text{gradient of } \boldsymbol{W}_{in}} + \underbrace{r \times r}_{\text{gradient of } \boldsymbol{\Sigma}} \Big\},$$

$$\max_\text{LASER-Grad}_{\boldsymbol{W}_{out}} \leftarrow \max\Big\{ \underbrace{n \times \text{hidden_size}}_{\text{feed-forward network}} + \underbrace{(5 \times \text{hidden_size} + r) \times r}_{\text{SVD}},$$
$$\underbrace{\text{hidden_size} \times (4 \times \text{hidden_size})}_{\text{gradient of } \boldsymbol{W}_{out}} + \underbrace{r \times r}_{\text{gradient of } \boldsymbol{\Sigma}} \Big\}.$$

Overall, the generated activations and the SVD decomposition, as well as the required gradients for each Transformer module, are:

$$\text{sum_LASER-Grad}_{\text{one_Transformer_module}} \leftarrow \text{sum}\Big\{ \underbrace{12 \times n \times \text{hidden_size}}_{\text{intermediate activations}} + \underbrace{2 \times n \times n \times \text{attention_head}}_{\text{attention matrix with softmax}} +$$
$$\max_\text{LASER-Grad}_{\boldsymbol{W}_{in}} + \max_\text{LASER-Grad}_{\boldsymbol{W}_{out}} \Big\},$$

Furthermore, the embedding layer generates $3\times$ the memory of the input sequence (because of input embedding, positional encoding, and LayerNorm for Roberta). We also need to consider the generated activations in the LLM head, which is $2\times$ the memory of the input sequence, adding with the output over the vocabulary.

$$\textbf{LASER-Grad}_{\text{Roberta}} : \text{Parameters}_{\text{LLM}} + \underbrace{3 \times n \times \text{hidden_size}}_{\text{embedding layer}} + L \times \Big\{ \text{sum_LASER-Grad}_{\text{one_Transformer_module}} \Big\}$$
$$+ \underbrace{(2 \times n \times \text{hidden_size} + n \times \text{vocab_size})}_{\text{LLM head}}.$$

E.2 GPT-J

For GPT-J, the peak memory consumption is different from that of Roberta. The input sequence is first processed with an input embedding, and then passed into L stacked Transformer modules, finally with an LLM head. Each Transformer module has a multi-head self-attention (rotary positional embeddings (RoPEs) are applied to $\boldsymbol{Q}$ and $\boldsymbol{K}$) and a feed-forward network in parallel. We assume that the input sequence is $\mathbb{R}^{n \times \text{hidden_size}}$ and the batch size is 1.

For $\textbf{Baseline}_{\text{GPT-J}}$ (direct inference), the LLM is loaded with parameters (mainly weight memory). For dynamic memory, mainly the generated activations, we first only need to consider the maximum peak memory of one Transformer module. Then, we need to compare the maximum peak memory for Transformer modules and the peak memory in LLM head.

For the maximum peak memory of one Transformer module, we add the maximum peak memory in the multi-head self-attention and the maximum peak memory in the feed-forward network together. In the multi-head self-attention, we need to compare the attention matrix and $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$. In the feed-forward network, the layers with $\boldsymbol{W}_{in}$ or $\boldsymbol{W}_{out}$, as well as the activation function (GELU Activation for GPT-J), have the same maximum activations, i.e., in $\mathbb{R}^{n \times (4 \times \text{hidden_size})}$. Therefore, we have

$$\text{max_baseline}_{\text{one_Transformer_module}} \leftarrow \max\Big\{ \underbrace{n \times n \times \text{attention_head}}_{\text{attention matrix}}, 3 \times \underbrace{(n \times \text{hidden_size})}_{\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}} \Big\}$$
$$+ \underbrace{n \times (4 \times \text{hidden_size})}_{\text{feed-forward network}},$$

$$\textbf{Baseline}_{\text{GPT-J}} : \text{Parameters}_{\text{LLM}} + \max\Big\{ \text{max_baseline}_{\text{one_Transformer_module}}, \underbrace{n \times \text{vocab_size}}_{\text{LLM head}} \Big\}.$$

For $\textbf{LASER}_{\text{GPT-J}}$ Sharma et al. [2024], in dynamic memory, we need to consider the SVD decomposition $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ in feed-forward network. The introduced extra dynamic memory is in $\mathbb{R}^{(5 \times \text{hidden_size} + r) \times r}$

for both $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_{out}$ in feed-forward network.

$$\begin{aligned}\underset{\text{one_Transformer_module}}{\text{max_LASER}} \leftarrow &\max\Big\{n \times n \times \underset{\text{attention matrix}}{\text{attention_head}}, 3 \times (n \times \underset{\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}}{\text{hidden_size}})\Big\} \\ &+ n \times (4 \times \underset{\text{feed-forward network}}{\text{hidden_size}}) + (5 \times \underset{\text{SVD}}{\text{hidden_size}} + r) \times r,\end{aligned}$$

$$\underset{\text{GPT-J}}{\textbf{LASER}} : \underset{\text{LLM}}{\text{Parameters}} + \max\Big\{\underset{\text{one_Transformer_module}}{\text{max_LASER}}, n \times \underset{\text{LLM head}}{\text{vocab_size}}\Big\}.$$

For $\underset{\text{GPT-J}}{\textbf{Ours}}$, in dynamic memory, the m SVD components introduced extra memory in $\mathbb{R}^{m \times n \times (4 \times \text{hidden_size})}$. Therefore, we have

$$\begin{aligned}\underset{\text{one_Transformer_module}}{\text{max_ours}} \leftarrow &\max\Big\{n \times n \times \underset{\text{attention matrix}}{\text{attention_head}}, 3 \times (n \times \underset{\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}}{\text{hidden_size}})\Big\} \\ &+ (m+1) \times n \times (4 \times \underset{\substack{\text{feed-forward network with} \\ m \text{ singular components}}}{\text{hidden_size}}) + (5 \times \underset{\text{SVD}}{\text{hidden_size}} + r) \times r,\end{aligned}$$

$$\underset{\text{GPT-J}}{\textbf{Ours}} : \underset{\text{LLM}}{\text{Parameters}} + \max\Big\{\underset{\text{one_Transformer_module}}{\text{max_ours}}, n \times \underset{\text{LLM head}}{\text{vocab_size}}\Big\}.$$

For $\underset{\text{GPT-J}}{\textbf{LASER-Grad}}$ Sreeram et al. [2025], the generated activations, the SVD decomposition $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, and LLM parameters are required to be stored in memory to get the gradient. Although parallelism of the multi-head self-attention and the feed-forward network is attainable together, we need to include both of them. In this case, we also need to include the intermediate generated activations. More specifically, when the input sequence is in $\mathbb{R}^{n \times \text{hidden_size}}$, for each Transformer module, there are $12\times$ (considering $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$, rotary position embeddings, LayerNorm, activation function, the input to each Transformer module, and the output of multi-head self-attention) the memory of the input sequence, adding with $2\times$ the memory of the attention weights (due to Softmax). For the feed-forward network with $\boldsymbol{W}_{in/out}$, based on the memory analysis in Zhang et al. [2024], we assume that the generated activations in the forward pass and the SVD decomposition could be released once the gradient of weights $\boldsymbol{W}$ and the corresponding singular values $\boldsymbol{\Sigma}$ in the backward pass are available. Therefore, we get the maximum memory for the layers with $\boldsymbol{W}_{in/out}$ in the feed-forward network, as denoted as follows,

$$\begin{aligned}\underset{\boldsymbol{W}_{in}}{\text{max_LASER-Grad}} \leftarrow &\max\Big\{n \times (4 \times \underset{\text{feed-forward network}}{\text{hidden_size}}) + (5 \times \underset{\text{SVD}}{\text{hidden_size}} + r) \times r, \\ &(4 \times \underset{\text{gradient of } \boldsymbol{W}_{in}}{\text{hidden_size}}) \times \text{hidden_size} + \underset{\text{gradient of } \boldsymbol{\Sigma}}{r \times r}\Big\}, \\ \underset{\boldsymbol{W}_{out}}{\text{max_LASER-Grad}} \leftarrow &\max\Big\{n \times \underset{\text{feed-forward network}}{\text{hidden_size}} + (5 \times \underset{\text{SVD}}{\text{hidden_size}} + r) \times r, \\ &\text{hidden_size} \times (4 \times \underset{\text{gradient of } \boldsymbol{W}_{out}}{\text{hidden_size}}) + \underset{\text{gradient of } \boldsymbol{\Sigma}}{r \times r}\Big\}.\end{aligned}$$

Overall, the generated activations and the SVD decomposition, as well as the required gradients for each Transformer module, are:

$$\begin{aligned}\underset{\text{one_Transformer_module}}{\text{sum_LASER-Grad}} \leftarrow &\text{sum}\Big\{12 \times n \times \underset{\text{intermediate activations}}{\text{hidden_size}} + 2 \times n \times n \times \underset{\text{attention matrix with softmax}}{\text{attention_head}} + \\ &\underset{\boldsymbol{W}_{in}}{\text{max_LASER-Grad}} + \underset{\boldsymbol{W}_{out}}{\text{max_LASER-Grad}}\Big\},\end{aligned}$$

Furthermore, the embedding layer generates $1\times$ the memory of the input sequence (because of only input embedding for GPT-J). We also need to consider the generated activations in the LLM head, which is the output over the vocabulary.

$$\begin{aligned}\underset{\text{GPT-J}}{\textbf{LASER-Grad}} : &\underset{\text{LLM}}{\text{Parameters}} + 1 \times n \times \underset{\text{embedding layer}}{\text{hidden_size}} + L \times \Big\{\underset{\text{one_Transformer_module}}{\text{sum_LASER-Grad}}\Big\} \\ &+ n \times \underset{\text{LLM head}}{\text{vocab_size}}.\end{aligned}$$

F Considering More Weight Matrices in the Transformer Architecture

Sharma et al. [2024] and Sreeram et al. [2025] have presented that the weight matrices in the attention layers contribute to very little performance increase compared to the weight matrices in the feed-forward networks on GPT-J. To consider more weight matrices in the Transformer architecture, we also add the weight matrices of key projection $\boldsymbol{W}_{\text{attention_}k}$, query projection $\boldsymbol{W}_{\text{attention_}q}$, value projection $\boldsymbol{W}_{\text{attention_}v}$, and out projection $\boldsymbol{W}_{\text{attention_}out}$ in the attention layers to $\boldsymbol{W}_{in/out}$.

We apply our proposed forward-only search across eight datasets on Roberta and only select the top-5 weight matrices for the subsequent rank-reduction strategy. Now, we have 6 types of weight matrices, and there are 12 layers of Transformer modules on Roberta. Therefore, there are a total of $6 \times 12 - 5 = 67$ weight matrices that are abandoned, which are not considered for the subsequent rank-reduction strategy. We take the Bios Profession dataset as an example, and the top-5 weight matrices are $\{[\boldsymbol{W}_{out}, 11], [\boldsymbol{W}_{in}, 0], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 9], [\boldsymbol{W}_{in}, 3]\}$, which is the same as the weight matrices in Table 10. All the weight matrices for $\boldsymbol{W}_{\text{attention_}k}$, $\boldsymbol{W}_{\text{attention_}q}$, $\boldsymbol{W}_{\text{attention_}v}$, $\boldsymbol{W}_{\text{attention_}out}$ are in the abandoned 67 weight matrices. This example shows that when considering more weight matrices in the Transformer architecture, our forward-only approach can still get the same weight matrices as the case of only considering $\boldsymbol{W}_{in/out}$.

```
#Top-5 W selected by our Algorithm 1
{[W_out,11], [W_in,0], [W_in,1], [W_in,9], [W_in,3]}

#The abandoned 67 W that are not considered
{[W_in,2], [W_in,5], [W_in,6], [W_in,4], [W_in,8], [W_attention_out,0], [W_in
    ,7], [W_attention_out,1], [W_in,10], [W_attention_out,11], [W_attention_out,10], [
    W_attention_k,9], [W_in,11], [W_attention_out,9], [W_attention_k,3], [W_attention_out
    ,8], [W_attention_out,2], [W_attention_out,5], [W_attention_out,7], [W_attention_k
    ,10], [W_attention_k,5], [W_attention_k,8], [W_attention_out,6], [W_attention_q,3], [
    W_attention_k,7], [W_attention_k,6], [W_attention_out,3], [W_attention_q,9], [
    W_attention_q,11], [W_out,5], [W_attention_out,4], [W_attention_q,4], [W_attention_k
    ,4], [W_out,7], [W_attention_k,11], [W_attention_q,6], [W_out,6], [W_out,10],
    [W_out,3], [W_attention_q,8], [W_attention_q,1], [W_out,8], [W_attention_k,0], [
    W_out,1], [W_out,2], [W_attention_q,0], [W_out,0], [W_out,4], [W_attention_q
    ,2], [W_attention_q,7], [W_attention_q,5], [W_attention_k,1], [W_attention_k,2], [
    W_attention_v,9], [W_attention_v,10], [W_out,9], [W_attention_q,10], [W_attention_v
    ,8], [W_attention_v,11], [W_attention_v,3], [W_attention_v,1], [W_attention_v,6], [
    W_attention_v,2], [W_attention_v,4], [W_attention_v,0], [W_attention_v,7],[W_attention_v
    ,5]}
```

We further extend these experiments of considering more weight matrices for all eight datasets, as presented in Table 13. ✔ means the top-j weight matrices are identical and preserve the same ordering as the weight matrices when only considering $\boldsymbol{W}_{in/out}$. The results show that our forward-only search can get the same top-j weight matrices for all eight datasets. $\boldsymbol{W}_{\text{attention_}k}$, $\boldsymbol{W}_{\text{attention_}q}$, $\boldsymbol{W}_{\text{attention_}v}$, $\boldsymbol{W}_{\text{attention_}out}$ in the attention layers can be excluded from the search. Furthermore, the same top-j weight matrices incur the same accuracy over eight datasets, which further justifies the effectiveness of our forward-only search.

Table 13: Weight matrices of considering the weight matrices $\boldsymbol{W}_{\text{attention_}k}$, $\boldsymbol{W}_{\text{attention_}q}$, $\boldsymbol{W}_{\text{attention_}v}$, and $\boldsymbol{W}_{\text{attention_}out}$ in the attention layers, as well as $\boldsymbol{W}_{in/out}$. Our forward-only search can get the same top-j weight matrices as the case of only considering $\boldsymbol{W}_{in/out}$.

Considering $\boldsymbol{W}_{\text{attention_}q/k/v/out}$, and $\boldsymbol{W}_{in/out}$	Same as only $\boldsymbol{W}_{in/out}$?
CounterFact	✔
HotPotQA	✔
FEVER	✔
Bios Gender	✔
Bios Profession	✔
TruthfulQA	✔
Epistemic Reasoning	✔
WikidataQA	✔

G Our Forward-Only Approach Enhances Fine-Tuning

To investigate the relationship between our forward-only approach and fine-tuning methods, we conducted various fine-tuning experiments with Roberta on FEVER and Bios Gender. These variants include fine-tuning the final candidate found by our forward-only approach, the worst candidate found by our forward-only approach, and all the $\boldsymbol{W}_{in}$ and all the $\boldsymbol{W}_{out}$ across twelve-layer Transformer modules. In the final candidate and the worst candidate found by our forward-only approach, we further test the cold start (fine-tuning the weight matrix) and warm start (fine-tuning the weight matrix with the corresponding rank reduction). We exploit $\mathcal{D}_{\text{std}}$ (20% of $\mathcal{D}$) as training data, while the rest 80% of $\mathcal{D}$ as test data. AdamW optimizer and learning rate scheduler are exploited. Hyperparameters are listed in Table 14.

As shown in Table 15, for FEVER, fine-tuning the final candidate found by our forward-only approach (only 1.89% of the total parameters are trained) achieves the accuracy of 75.2$\pm$0.14%, while fine-tuning all the $\boldsymbol{W}_{in}$ (22.7% of the total parameters are trained) achieves the accuracy of 76.5$\pm$0.16%. Similarly, fine-tuning all the $\boldsymbol{W}_{out}$ achieves the accuracy of 76.5$\pm$0.16%. For Bios Gender, fine-tuning the final candidate achieves the accuracy of 95.1$\pm$0.01%, while fine-tuning all the $\boldsymbol{W}_{in}$ achieves the accuracy of 95.6$\pm$0.02%, and fine-tuning all the $\boldsymbol{W}_{out}$ achieves the accuracy of 95.6$\pm$0.01%. These results show that the final candidate found by our approach is effective for fine-tuning and can achieve comparable accuracy with 12$\times$ fewer trainable parameters than fine-tuning all $\boldsymbol{W}_{in}$ or all $\boldsymbol{W}_{out}$.

Table 14: Hyperparameters for fine-tuning experiments in Table 15 and Table 16.

Dataset	Setting	Batch Size	Fine-Tuning Epoch	Learning Rate	Weight Decay
FEVER	Cold Start/Warm Start (Table 16)	256	1	6e-5	0.01
	Fine-Tuning‡(Table 15)	256	50	6e-5	0.01
Bios Gender	Cold Start/Warm Start (Table 16)	256	1	2e-6	0.01
	Fine-Tuning‡(Table 15)	256	10	2e-6	0.01

Note that in Fine-Tuning‡, fine-tuning the final candidate has 12$\times$ less trainable parameters compared to all $\boldsymbol{W}_{in}$ or all $\boldsymbol{W}_{out}$. More fine-tuning epochs tend to improve the accuracy of fine-tuning with relatively fewer parameters, e.g., 20 epochs for Bios Gender with fine-tuning the final candidate achieving the accuracy of 95.1$\pm$0.01%.

Table 15: Fine-Tuning Roberta with multiple epochs across multiple random seeds.

Dataset	FEVER				Bios Gender			
	Fine-Tuning				Fine-Tuning			
	Final Candidate (50 epochs) Final Candidate (100 epochs)	All $\boldsymbol{W}_{in}$	All $\boldsymbol{W}_{out}$	Ours (std)	Final Candidate (10 epochs) Final Candidate (20 epochs)	All $\boldsymbol{W}_{in}$	All $\boldsymbol{W}_{out}$	Ours (std)
Trainable Params↓	1.89%	22.7%		-	1.89%	22.7%		-
Acc	74.6$\pm$0.24 (75.2$\pm$0.14)	76.5$\pm$0.16	76.5$\pm$0.16	52.5	94.2$\pm$0.02 (95.1$\pm$0.01)	95.6$\pm$0.02	95.6$\pm$0.01	93.4

As presented in Table 16, the warm start with the final candidate performs better than the cold start in one-epoch fine tuning (one forward pass and one backward pass), while our forward-only search (two forward passes) performs comparably. For Roberta on Bios Gender, the final candidate is $[\boldsymbol{W}_{in}, 9, 90\%]$ and the worst candidate is $[\boldsymbol{W}_{in}, 1, 1\%]$.

Besides, as shown in Fig. 11, fine-tuning the final candidate found by our forward-only approach with the warm start performs better than its cold start, especially at the early fine-tuning epochs. While for the worst candidate found by our forward-only approach, fine-tuning the worst candidate with the cold start only gets the accuracy of 89.3$\pm$0.03%, which is significantly worse than the final candidate with the cold start. At the same time, fine-tuning the worst candidate with the warm start even performs significantly worse than its cold start, which demonstrates the effectiveness of our forward-only approach. Overall, these results consistently present that our forward-only approach can enhance the fine-tuning process.

H Extended Experiments on Various LLMs

We extend experiments to various LLMs including LLaMA2 Touvron et al. [2023] with gated feed-forward networks (Llama2-7B), Mistral [Jiang et al., 2023] with grouped query attention (GQA)

Table 16: Fine-Tuning Roberta with one epoch across multiple random seeds.

Dataset	FEVER			Bios Gender		
	Fine-Tuning Final Candidate		Ours (std)	Fine-Tuning Final Candidate		Ours (std)
	Cold Start	Warm Start		Cold Start	Warm Start	
Acc	50.00±0.00	**53.35±0.17**	52.5	88.58±0.03	**93.50±0.11**	93.4

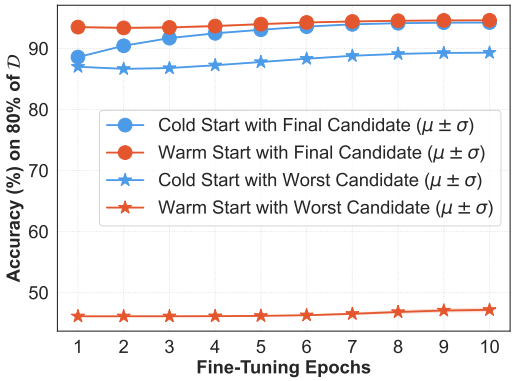


Figure 11: Fine-tuning the final candidate with the warm start performs the best.

and sliding-window attention (Mistral-7B), Qwen2 [Yang et al., 2024a] with instruction-tuning and multilingual capabilities (Qwen2-7B-Instruct), Qwen2 with mixture-of-experts (MoE) (Qwen2-57B-A14B-Instruct), Gemma [Team et al., 2024] (Gemma-7B), and DeepSeek [Bi et al., 2024] (DeepSeekV1-7B-Chat) with data engineering, as well as the latest Gemma 4 [Google DeepMind, 2026] (Gemma4-E2B-it) with the hybrid attention and per-layer embeddings (PLE). For Qwen2-57B-A14B-Instruct[11] with NormalFloat4 quantization [Dettmers et al., 2023] and BFloat16 computation [Wang and Kanwar, 2019], we consider $\boldsymbol{W}_{in/out}$ in the shared experts of MoE. The number of stacked Transformer modules is listed in Table 17. The final candidates for LASER [Sharma et al., 2024] and our forward-only approach on Epistemic Reasoning are presented in Table 18.

Table 17: The number of stacked Transformer modules for various LLMs.

LLMs	LLaMA2-7B Mistral-7B	Qwen2-7B-Instruct/Gemma-7B Qwen2-57B-A14B-Instruct	DeepSeekV1-7B-Chat	Gemma4-E2B-it
L	32	28	30	35

Furthermore, we also extend the experiments across various datasets. The final candidates for LASER [Sharma et al., 2024] and our forward-only approach on LLaMA2 are presented in Tables 18 and 19. The corresponding results in Acc and Speedup are presented in Table 20. Our forward-only approach achieves a greater speedup, while ours (std) and ours (hierarchical) achieve greater average improvement in Acc compared to LASER. Overall, ours (hierarchical) achieves a balanced trade-off between average Speedup and average improvement in Acc across ours (std), ours (hierarchical), and ours (subset).

[11] For Qwen2-57B-A14B-Instruct, we use 2 NVIDIA Tesla A40 graphics processing unit (GPU) with 48GB random-access memory (RAM) for each GPU.

Table 18: Final candidates for LASER and our forward-only approach with various LLMs on Epistemic Reasoning in Table 4, including $\boldsymbol{W}_{in/out}$ in the feed-forward network of Transformer architecture, the layer index l of stacked Transformer modules, and the retained top-$k\%$ ranks ($0.5\% \leq k\% \leq 90\%$), as well as considering top-j Weight Matrices and last-m-rank SVD component.

LLMs	LASER (std)	Ours (std)	top-j	last-m
LLaMA2-7B	$[\boldsymbol{W}_{out}, 28, 1\%]$	$[\boldsymbol{W}_{in}, 31, 40\%]$	top-10	last-50
DeepseekV1-7B-Chat	$[\boldsymbol{W}_{out}, 11, 0.5\%]$	$[\boldsymbol{W}_{in}, 7, 0.5\%]$	top-2	last-20
Mistral-7B	$[\boldsymbol{W}_{in}, 20, 0.5\%]$	$[\boldsymbol{W}_{in}, 20, 0.5\%]$	top-7	last-20
Qwen2-7B-Instruct	$[\boldsymbol{W}_{in}, 2, 1\%]$	$[\boldsymbol{W}_{in}, 2, 1\%]$	top-2	last-20
Gemma-7B	$[\boldsymbol{W}_{out}, 3, 5\%]$	$[\boldsymbol{W}_{in}, 7, 20\%]$	top-5	last-20
Gemma4-E2B-it	$[\boldsymbol{W}_{in}, 2, 1\%]$	$[\boldsymbol{W}_{out}, 25, 0.5\%]$	top-6	last-50
Qwen2-57B-A14B-Instruct	$[\boldsymbol{W}_{in}, 9, 5\%]$	$[\boldsymbol{W}_{in}, 9, 5\%]$	top-4	last-20

Table 19: Final candidates for our forward-only approach on LLaMA2 in Table 20, including $\boldsymbol{W}_{in/out}$ in the feed-forward network of Transformer architecture, the layer index l ($0 \leq l \leq 31$) of stacked Transformer modules, and the retained top-$k\%$ ranks ($0.5\% \leq k\% \leq 90\%$).

Dataset	Ours (subset)	Ours (std)	Ours (hierarchical)	Hit?
CounterFact	$[\boldsymbol{W}_{in}, 9, 10\%]$	$[\boldsymbol{W}_{in}, 31, 60\%]$	$[\boldsymbol{W}_{in}, 31, 60\%]$	Yes
HotPotQA	$[\boldsymbol{W}_{in}, 6, 5\%]$	$[\boldsymbol{W}_{in}, 10, 0.5\%]$	$[\boldsymbol{W}_{in}, 9, 5\%]$	No
FEVER	$[\boldsymbol{W}_{in}, 9, 90\%]$	$[\boldsymbol{W}_{in}, 30, 20\%]$	$[\boldsymbol{W}_{in}, 9, 0.5\%]$	No
Bios Gender	$[\boldsymbol{W}_{in}, 7, 0.5\%]$	$[\boldsymbol{W}_{in}, 7, 0.5\%]$	$[\boldsymbol{W}_{in}, 7, 0.5\%]$	Yes
Bios Profession	$[\boldsymbol{W}_{in}, 16, 5\%]$	$[\boldsymbol{W}_{in}, 16, 0.5\%]$	$[\boldsymbol{W}_{in}, 16, 0.5\%]$	Yes
TruthfulQA	$[\boldsymbol{W}_{out}, 0, 5\%]$	$[\boldsymbol{W}_{in}, 30, 10\%]$	$[\boldsymbol{W}_{in}, 30, 1\%]$	No
Epistemic Reasoning	$[\boldsymbol{W}_{in}, 16, 1\%]$	$[\boldsymbol{W}_{in}, 31, 40\%]$	$[\boldsymbol{W}_{in}, 31, 40\%]$	Yes
WikidataQA	$[\boldsymbol{W}_{out}, 0, 80\%]$	$[\boldsymbol{W}_{in}, 30, 5\%]$	$[\boldsymbol{W}_{in}, 30, 5\%]$	Yes

Table 20: LLaMA2 evaluation with reported accuracy (Acc, %) and Speedup. Ours (hierarchical) achieves a balanced trade-off between average Speedup and average improvement in Acc across ours (std), ours (hierarchical), and ours (subset) settings.

Dataset	Baseline	LASER	Ours (std)		Ours (hierarchical)		Ours (subset)	
	Acc	Acc	Acc	Speedup↑	Acc	Speedup↑	Acc	Speedup↑
CounterFact	35.6	**37.6**	35.9	6.4×	35.9	53.9×	34.9	832×
HotPotQA	16.5	17.2	19.7	6.4×	**20.5**	43.9×	17.5	185×
FEVER	59.3	**64.5**	**64.5**	6.4×	62.9	42.7×	59.9	165×
Bios Gender	75.5	88.4	**91.5**	6.4×	**91.5**	51.7×	**91.5**	502×
Bios Profession	85.0	86.7	**87.0**	6.4×	**87.0**	46.6×	85.7	243×
TruthfulQA	50.5	**56.2**	**56.2**	6.4×	**56.2**	32.5×	53.7	74.5×
Epistemic Reasoning	44.8	63.4	**63.5**	6.4×	**63.5**	17.6×	62.1	25.3×
WikidataQA	59.5	**62.0**	60.9	6.4×	60.9	47.1×	59.8	257×
Average ΔAcc↑	0.00	6.16	**6.56**		6.46		4.80	
Average Speedup↑	-	1×	6.4×		42.0×		**285**×	

I Intuitive Explanation

In Equation (1), with $\boldsymbol{W} \in \mathbb{R}^{h\times d} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \otimes \boldsymbol{v}_i^T$ in full SVD, we have

$$\boldsymbol{X}\boldsymbol{W}^T = \sum_{i=1}^{r} \sigma_i \cdot (\boldsymbol{X}\boldsymbol{v}_i) \otimes \boldsymbol{u}_i^T = \sum_{i=1}^{r} \boldsymbol{Y}^{(i)} = \boldsymbol{Y}, \text{with} \quad r = \min(h, d).$$

The gradient of the loss function $\mathcal{L}$ with respect to $\boldsymbol{W}$ ($\nabla_{\boldsymbol{W}}\mathcal{L}$, simplified as $\nabla\boldsymbol{W}$), is backpropagated from the gradient of the loss function $\mathcal{L}$ with respect to $\boldsymbol{Y}$ ($\nabla_{\boldsymbol{Y}}\mathcal{L}$, simplified as $\nabla\boldsymbol{Y}$).

$$\nabla\boldsymbol{W} = (\nabla\boldsymbol{Y})^T\boldsymbol{X} = \sum_{p=1}^{n} (\nabla\boldsymbol{Y}_{p,:})^T\boldsymbol{X}_{p,:},$$

where n is the sequence length and p is the p-th token in the sequence. Next, we have the gradient of i-th singular value $\nabla\sigma_i$ as follows,

$$\nabla\sigma_i = \boldsymbol{u}_i^T\nabla\boldsymbol{W}\boldsymbol{v}_i = \boldsymbol{u}_i^T(\nabla\boldsymbol{Y})^T\boldsymbol{X}\boldsymbol{v}_i = \sum_{p=1}^{n} \boldsymbol{u}_i^T(\nabla\boldsymbol{Y}_{p,:})^T\boldsymbol{X}_{p,:}\boldsymbol{v}_i,$$

where $\nabla\boldsymbol{Y}^{(i)} = \nabla\boldsymbol{Y}$. Moreover, we exploit the Frobenius norm of $\nabla\sigma_i$ and the linearity of the Frobenius inner product, and therefore we have:

$$\begin{aligned}
\|\nabla\sigma_i\|_F^2 &= \left\langle \sum_{p=1}^{n} \boldsymbol{u}_i^T(\nabla\boldsymbol{Y}_{p,:})^T\boldsymbol{X}_{p,:}\boldsymbol{v}_i, \sum_{q=1}^{n} \boldsymbol{u}_i^T(\nabla\boldsymbol{Y}_{q,:})^T\boldsymbol{X}_{q,:}\boldsymbol{v}_i \right\rangle_F \\
&= \sum_{p=1}^{n}\sum_{q=1}^{n} \left\langle \boldsymbol{u}_i^T(\nabla\boldsymbol{Y}_{p,:})^T\boldsymbol{X}_{p,:}\boldsymbol{v}_i, \boldsymbol{u}_i^T(\nabla\boldsymbol{Y}_{q,:})^T\boldsymbol{X}_{q,:}\boldsymbol{v}_i \right\rangle_F,
\end{aligned} \tag{8}$$

where p and q denote the different tokens in the sequence. Moreover, utilizing the compatibility between the Frobenius inner product and the matrix outer product, for any $\boldsymbol{a}$ and $\boldsymbol{c}$ ($\boldsymbol{b}$ and $\boldsymbol{d}$) having the same dimension, we have the formula $\left\langle \boldsymbol{a}\boldsymbol{b}^T, \boldsymbol{c}\boldsymbol{d}^T \right\rangle_F = \mathrm{tr}\left((\boldsymbol{a}\boldsymbol{b}^T)^T(\boldsymbol{c}\boldsymbol{d}^T)\right) = (\boldsymbol{a}^T\boldsymbol{c})(\boldsymbol{b}^T\boldsymbol{d})$, where tr is the matrix trace operator. Then, Equation (8) can be simplified as follows:

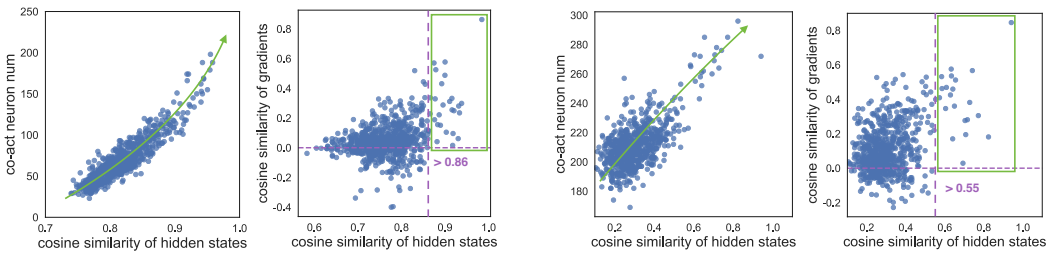
$$\|\nabla\sigma_i\|_F^2 = \sum_{p=1}^{n}\sum_{q=1}^{n} \left((\boldsymbol{X}_{p,:}\boldsymbol{v}_i)(\boldsymbol{X}_{q,:}\boldsymbol{v}_i)^T\right) \cdot \left((\nabla\boldsymbol{Y}_{p,:}\boldsymbol{u}_i)(\nabla\boldsymbol{Y}_{q,:}\boldsymbol{u}_i)^T\right).$$

When considering $\boldsymbol{V} \in \mathbb{R}^{d\times d}$ in full SVD as the orthogonal matrix (i.e., $\boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}$), for full ranks, the angle between $\boldsymbol{X}_{p,:}$ and $\boldsymbol{X}_{q,:}$ contributes as follows:

$$\begin{aligned}
\sum_{p=1}^{n}\sum_{q=1}^{n}\sum_{i=1}^{d} (\boldsymbol{X}_{p,:}\boldsymbol{v}_i)(\boldsymbol{X}_{q,:}\boldsymbol{v}_i)^T &= \sum_{p=1}^{n}\sum_{q=1}^{n} \boldsymbol{X}_{p,:}\left(\sum_{i=1}^{d} \boldsymbol{v}_i\boldsymbol{v}_i^T\right)(\boldsymbol{X}_{q,:})^T \\
&= \sum_{p=1}^{n}\sum_{q=1}^{n} \boldsymbol{X}_{p,:}(\boldsymbol{X}_{q,:})^T \\
&= \sum_{p=1}^{n}\sum_{q=1}^{n} \|\boldsymbol{X}_{p,:}\|\|\boldsymbol{X}_{q,:}\| \cos\theta_{p,q},
\end{aligned} \tag{9}$$

where $\cos\theta_{p,q}$ denotes the cosine similarity of the angle between $\boldsymbol{X}_{p,:}$ and $\boldsymbol{X}_{q,:}$. Therefore, given the LLM with the deterministic SVD, $\cos\theta_{p,q}$ will exert an influence on the gradient of each singular value among $(\nabla\sigma_1, \nabla\sigma_2, ..., \nabla\sigma_r)$ by decomposing the angle between $\boldsymbol{X}_{p,:}$ and $\boldsymbol{X}_{q,:}$ into individual singular components with $(\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_d)$. This decomposition also applies for the backpropagated gradients with the $(\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_h)$ and the orthogonal matrix $\boldsymbol{U} \in \mathbb{R}^{h\times h}$ (i.e., $\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}$) [12].

$$\sum_{p=1}^{n}\sum_{q=1}^{n}\sum_{i=1}^{h} (\nabla\boldsymbol{Y}_{p,:}\boldsymbol{u}_i)(\nabla\boldsymbol{Y}_{q,:}\boldsymbol{u}_i)^T = \sum_{p=1}^{n}\sum_{q=1}^{n} \nabla\boldsymbol{Y}_{p,:}(\nabla\boldsymbol{Y}_{q,:})^T. \tag{10}$$



(a) $\boldsymbol{W}_{in}$ (b) $\boldsymbol{W}_{out}$

Figure 12: Visualization of the relationship between cosine similarity of hidden states ($\cos\theta_{p,q}$) and the number of co-activated neurons, as well as $\cos\theta_{p,q}$ and cosine similarity of gradients ($\cos\gamma_{p,q}$) for (a) $\boldsymbol{W}_{in}$ (Layer 6) and (b) $\boldsymbol{W}_{out}$ (Layer 4) with Roberta on the FEVER dataset. More layers for $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_{out}$ are detailed in Fig. 13 (a) and Fig. 14 (a).

To analyze Equation (9) and Equation (10), we evaluate how the angle between $\boldsymbol{X}_{p,:}$ and $\boldsymbol{X}_{q,:}$ (which is available in the forward pass) influences the $\nabla\boldsymbol{Y}_{p,:}(\nabla\boldsymbol{Y}_{q,:})^T$ (which is computed in the backward pass). We exploit the $\nabla\boldsymbol{Y}_{p,:}(\nabla\boldsymbol{Y}_{q,:})^T = \|\nabla\boldsymbol{Y}_{p,:}\|\|\nabla\boldsymbol{Y}_{q,:}\|\cos\gamma_{p,q}$, where γ denotes the angle between $\nabla\boldsymbol{Y}_{p,:}$ and $\nabla\boldsymbol{Y}_{q,:}$. Taking the evaluation with Roberta on the FEVER dataset, we present the relationship between cosine similarity of hidden states ($\cos\theta_{p,q}$) and the number of co-activated neurons, as well as $\cos\theta_{p,q}$ and cosine similarity of gradients ($\cos\gamma_{p,q}$) in Fig. 12. The results show that a higher $\cos\theta_{p,q}$ tends to activate the similar patterns among neurons, i.e., implying an increased number of co-activated neurons[13]. At the same time, when $\cos\theta_{p,q}$ exceeds a certain threshold, $\cos\gamma_{p,q}$ tends to become positive. Therefore, a sufficiently large $\cos\theta_{p,q}$ tends to increase the $\|\nabla\boldsymbol{W}\|_F^2$[14] because $\|\nabla\boldsymbol{W}\|_F^2 = \sum_{p=1}^{n}\sum_{q=1}^{n}\|\boldsymbol{X}_{p,:}\|\|\boldsymbol{X}_{q,:}\|\cos\theta_{p,q}\|\nabla\boldsymbol{Y}_{p,:}\|\|\nabla\boldsymbol{Y}_{q,:}\|\cos\gamma_{p,q}$.

Therefore, this change in $\nabla\boldsymbol{W}$ induced by $\cos\theta_{p,q}$ tends to influence the $\nabla\sigma_i$ and therefore tends to re-allocate the updating power within different $\|\nabla\sigma_i\|_F^2$, as shown below,

$$\|\nabla\boldsymbol{W}\|_F^2 = \left\|\boldsymbol{U}^T\nabla\boldsymbol{W}\boldsymbol{V}\right\|_F^2 \geq \sum_{i=1}^{r}\|\nabla\sigma_i\|_F^2 \quad \text{with} \quad \nabla\sigma_i = \boldsymbol{u}_i^T\nabla\boldsymbol{W}\boldsymbol{v}_i = \langle\nabla\boldsymbol{W}, \boldsymbol{u}_i\boldsymbol{v}_i^T\rangle_F,$$

where $\langle\boldsymbol{u}_i\boldsymbol{v}_i^T, \boldsymbol{u}_j\boldsymbol{v}_j^T\rangle_F = 0$ for $i \neq j$ and $i, j \in \{1, 2, \ldots, r\}$ because $\langle\boldsymbol{u}_i\boldsymbol{v}_i^T, \boldsymbol{u}_j\boldsymbol{v}_j^T\rangle_F = (\boldsymbol{u}_i^T\boldsymbol{u}_j)(\boldsymbol{v}_i^T\boldsymbol{v}_j)$ and the orthogonality of $\boldsymbol{U}$ and $\boldsymbol{V}^T$ matrices in full SVD. Besides, this study [Kaushik et al., 2026] shows that deep neural networks converge to shared subspaces and this study [Sun et al., 2026] shows that fine-tuning the top singular values is effective, as well as this study [Li et al., 2024] that fine-tunes the singular values and keeps the most important components/bases. Potentially, given this heuristic explanation, the increased $\|\nabla\boldsymbol{W}\|_F^2$ tends to amplify/attenuate the updating power of different $\|\nabla\sigma_i\|_F^2$. In this case, a higher $\cos\theta_{p,q}$ tends to have more effective updating power, improving adaptation to downstream tasks.

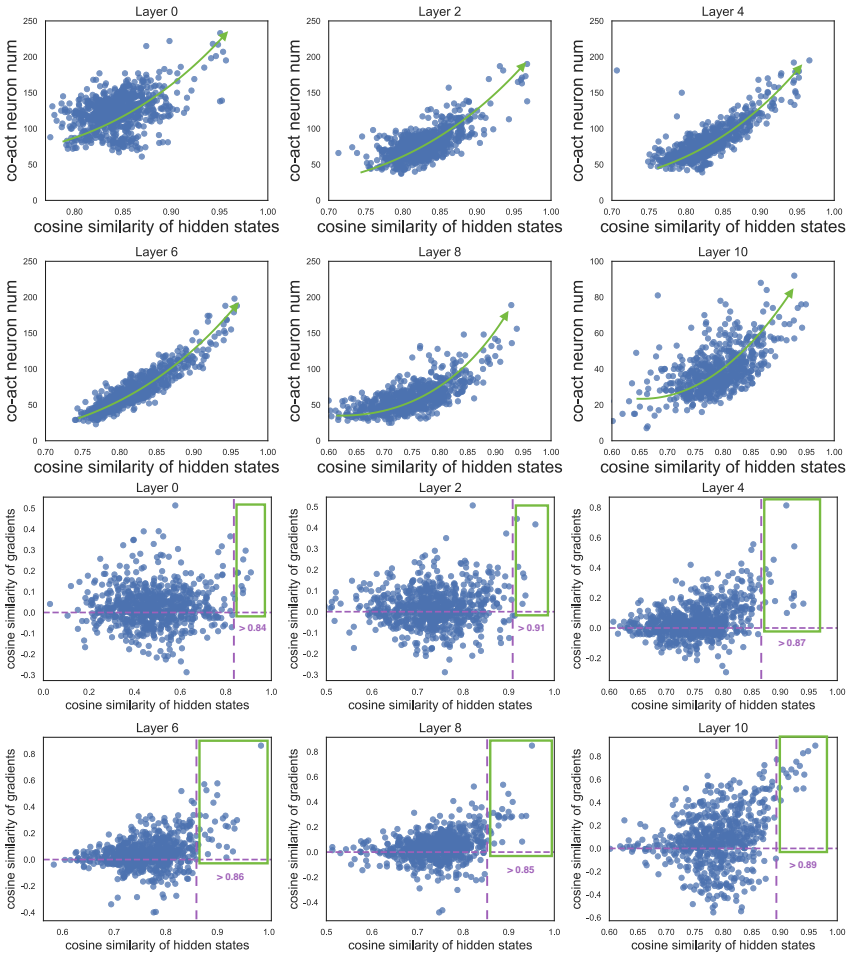
Overall, we provide some intuition for the relationship between angle concentration and the gradients of singular values. Angle concentration is available during the forward pass, unlike gradients computed in the backward pass, making it well suited for LLM adaptation in LLM serving.

Additionally, due to the approximate orthogonality of $\boldsymbol{W}_{in/out}$ as shown in Fig. 13 (b) and Fig. 14 (b), the angle between $\boldsymbol{X}_{p,:}$ and $\boldsymbol{X}_{q,:}$ is similar to that between $\boldsymbol{Y}_{p,:}$ and $\boldsymbol{Y}_{q,:}$. Therefore, we exploit the angle concentration of hidden states $\boldsymbol{Y}$ in our paper. Accordingly, we exploit the angle concentration of activations induced by each SVD component, as a heuristic indicator, to measure its contribution—*dispersed* or *concentrated*—to the angle concentration of the hidden states $\boldsymbol{Y}$. Consequently, this contribution (*dispersed* or *concentrated*) implies which singular components dominate the informative versus noisy contributions to the update and subsequent effective adaptation.

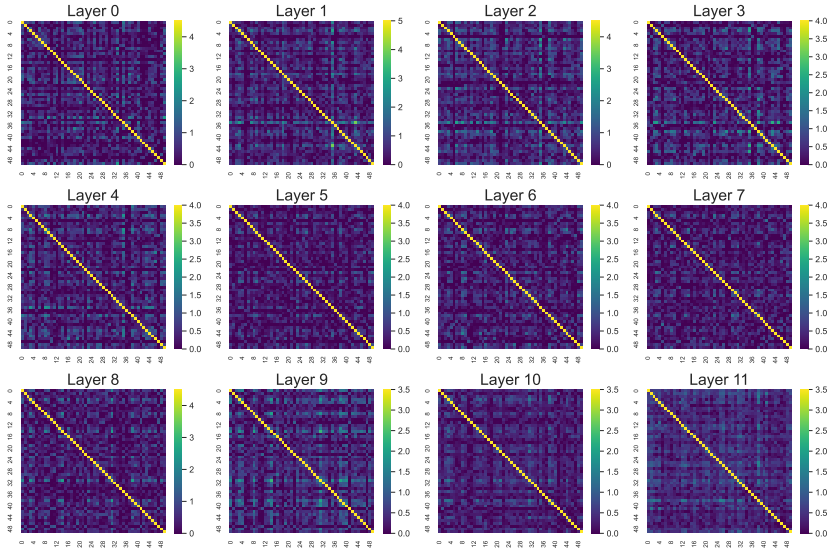
[12]Wang et al. [2025b]:"*The number of gradient components received by a neuron is proportional to the frequency of its activation*" and "*Tokens with higher angle concentration activate similar neurons due to shared value patterns*". Moreover, "*the more concentrated the angles between tokens, the larger the gradient norm*".

[13]This is also observed in Wang et al. [2025b].

[14]This is also observed in Wang et al. [2025b].

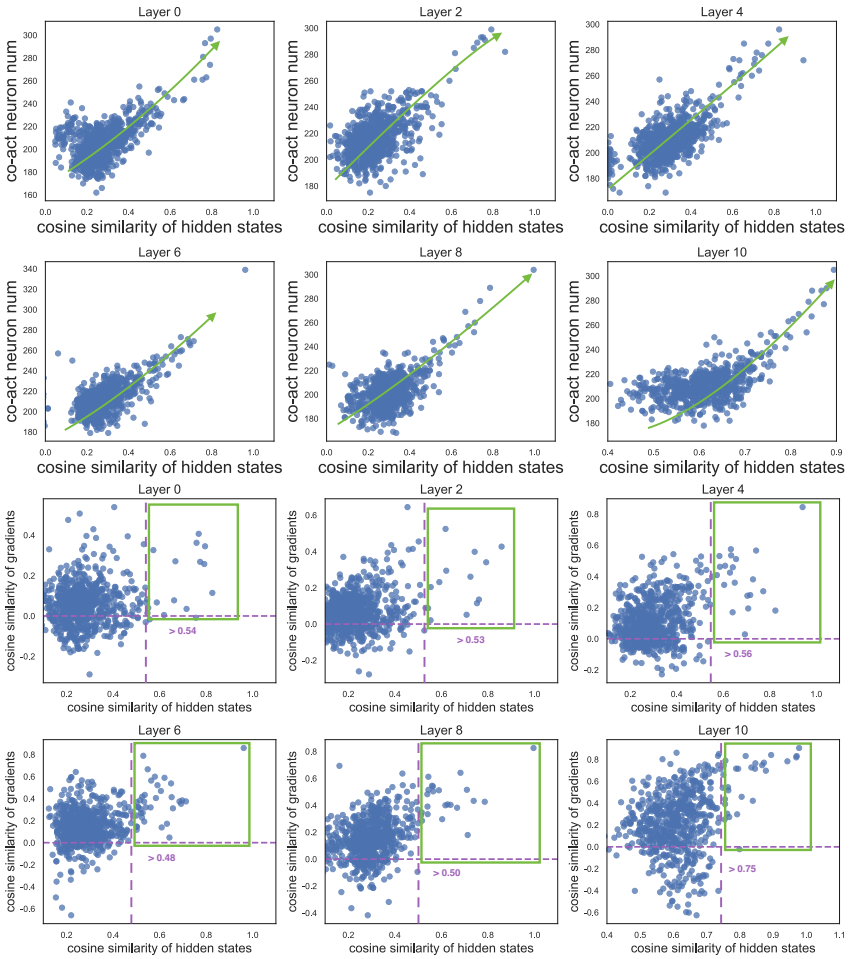


(a) Visualization of the relationship between cosine similarity of hidden states ($\cos\theta_{p,q}$) and the co-activated neurons, as well as the relationship between cosine similarity of hidden states ($\cos\theta_{p,q}$) and cosine similarity of gradients ($\cos\gamma_{p,q}$) for $\boldsymbol{W}_{in}$ with Roberta on the FEVER dataset.

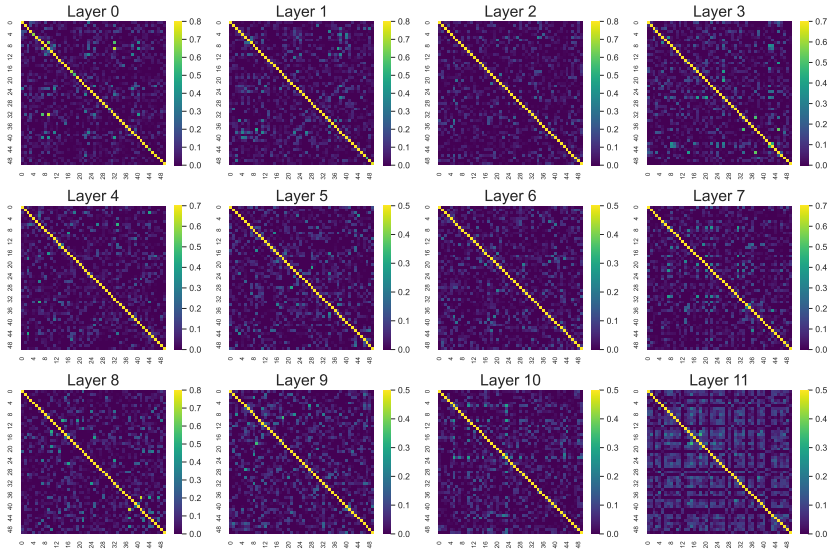


(b) Visualization of submatrix of $\boldsymbol{W}_{in}^T\boldsymbol{W}_{in}$ at different layers in Roberta. The approximately orthogonal matrix of $\boldsymbol{W}_{in}$ holds $\boldsymbol{W}_{in}^T\boldsymbol{W}_{in} \approx \lambda\boldsymbol{I}$, where λ is a constant.

Figure 13: Visualization for $\boldsymbol{W}_{in}$.



(a) Visualization of the relationship between cosine similarity of hidden states ($\cos\theta_{p,q}$) and the co-activated neurons, as well as the relationship between cosine similarity of hidden states ($\cos\theta_{p,q}$) and cosine similarity of gradients ($\cos\gamma_{p,q}$) for $\boldsymbol{W}_{out}$ with Roberta on the FEVER dataset.



(b) Visualization of submatrix of $\boldsymbol{W}_{out}^T\boldsymbol{W}_{out}$ at different layers in Roberta. The approximately orthogonal matrix of $\boldsymbol{W}_{out}$ holds $\boldsymbol{W}_{out}^T\boldsymbol{W}_{out} \approx \lambda\boldsymbol{I}$, where λ is a constant.

Figure 14: Visualization for $\boldsymbol{W}_{out}$.

J Transmuting Weight Updates to Steering Prompts

Recent studies [Dherin et al., 2025, Mazzawi et al., 2025, Goldwaser et al., 2026] reveal the implicit weight updates within the context, namely, transmuting added/modified prompts into weight updates. In our paper, we transmute weight updates into steering prompts so that the parameters of LLMs can stay unmodified. In Equation (1), we have the standard forward pass, i.e., $\boldsymbol{X}\boldsymbol{W}^T = \boldsymbol{Y}$. Given the final candidate with deterministic rank reduction from our forward-only approach, we can get the weight update $\Delta\boldsymbol{W}$. Considering the weights without updates and with updates, we have

$$\begin{cases} \boldsymbol{X}\boldsymbol{W}^T = \boldsymbol{Y} \\ \boldsymbol{X}(\boldsymbol{W} + \Delta\boldsymbol{W})^T = \boldsymbol{Y}' \end{cases} \implies \boldsymbol{Y}' - \boldsymbol{Y} = \boldsymbol{X}\Delta\boldsymbol{W}^T,$$

where $\boldsymbol{Y}'$ are the hidden states of the output with the weight-updated LLMs. Therefore, for a specific weight update $\Delta\boldsymbol{W}$ and a specific input $\boldsymbol{X}$, we transmute this weight update into a change in output.

To extend this input-wise transmutation to the unified transmutation for downstream tasks, we derive the steering prompts by averaging over validation datasets $\mathcal{D}_{\text{std}}$. The steering prompts $\Delta\boldsymbol{Y}$, which broadcasts across token positions, are denoted:

$$\Delta\boldsymbol{Y} = \mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}_{\text{std}}}\left[\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{X}_{i,:}\Delta\boldsymbol{W}^T\right],$$

where $\boldsymbol{X}_{i,:}$ is the i-th token with the sequence length n of $\boldsymbol{X}$.

Taking the evaluation with Roberta on the Bios Profession dataset as the example, the final candidate with rank reduction from our forward-only approach is $[\boldsymbol{W}_{in}, 3, 90\%]$ from Table 8. We transmute weight updates to steering prompts, with different sizes of validation datasets, as presented in Table 21. The results show that the steering prompts achieve nearly identical accuracy to the weight updates, which justifies that we can transmute weight updates to steering prompts with unmodified parameters.

Table 21: The reported accuracy (Acc, %) for Roberta on Bios Profession. We can transmute weight updates to steering prompts with unmodified parameters in LLMs.

Baseline	Ours (with $\Delta\boldsymbol{W}$)	Steering Prompts (transmute $\Delta\boldsymbol{W}$ to $\Delta\boldsymbol{Y}$)				
		20% of $\mathcal{D}_{\text{std}}$	40% of $\mathcal{D}_{\text{std}}$	60% of $\mathcal{D}_{\text{std}}$	80% of $\mathcal{D}_{\text{std}}$	$\mathcal{D}_{\text{std}}$
64.5	71.9	72.0	71.9	71.8	72.0	72.0

Besides, we analyze the SVD of $\Delta\boldsymbol{W}$, as demonstrated in Fig. 15. The effective ranks of $\Delta\boldsymbol{W}$ are equal to the removed ranks (removing $1-k\%$ of ranks with $k\% = 90\%$), i.e., $10\% \times \underset{\text{hidden_size}}{768}$.

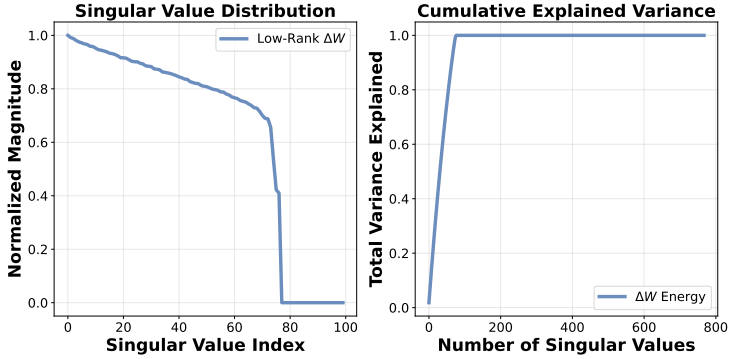


Figure 15: The SVD of the weight updates $\Delta\boldsymbol{W}$.

Therefore, we can further decompose the steering prompts $\Delta\boldsymbol{Y}$ into $\Delta\boldsymbol{Y}^{(i)}$ in different SVD components, $\Delta\boldsymbol{Y} = \sum_{i=1}^{e}\Delta\boldsymbol{Y}^{(i)}$, where e is the effective ranks and $e < r$. Moreover, given this study [Kaushik et al., 2026] showing that deep neural networks converge to shared subspaces and these studies [Meng et al., 2022, Li et al., 2024, Meng et al., 2024, Sun et al., 2026] showing the capacity of low-rank update in SVD components (even the singular value itself), future work could investigate

how shared and generated expert weight matrices [Du and Lin, 2026a,b] and steering prompts $\Delta \boldsymbol{Y}^{(i)}$ in different ranks are related to adaptation across downstream tasks and LLMs [Touvron et al., 2023, Jiang et al., 2023, Yang et al., 2024a, Huang et al., 2024, Bi et al., 2024, Ren et al., 2025, Google DeepMind, 2026].

K Additional Experiments and Analyses

K.1 Empirical Results to Show the Efficiency Compared to LASER-Grad

We did an initial empirical measurement of realistic memory consumption, computational complexity, and execution time. We mainly measure the memory change during the core running, i.e., calculating angle concentration for our approach (only forward passes) and calculating gradients for LASER-Grad Sreeram et al. [2025] (forward pass and backward pass). The empirical results are shown in Table 22. For Roberta on the FEVER dataset, LASER-Grad Sreeram et al. [2025] has around 2.6x more memory change than our approach; for Roberta on the Bios Profession dataset with a longer average sequence length, LASER-Grad Sreeram et al. [2025] has around 4.5x more memory change than our approach. The results are expected because our forward-only approach avoids the backward pass. Moreover, our forward-only search achieves lower computational complexity and shorter execution time when compared to LASER-Grad Sreeram et al. [2025].

Table 22: Measurement of realistic memory consumption, computational complexity, and execution time. Note that the average complexity and average execution time are averages over all samples in $\mathcal{D}_{\text{subset}}$. When calculating FLOPs for a specific sample, the backward pass has approximately twice the FLOPs of the forward pass Kaplan et al. [2020], Yin and Rush [2025].

Roberta	Our Approach	Laser-Grad	Our Approach	Laser-Grad
Dataset	FEVER		Bios Profession	
Average Sequence Length	27		100	
Memory Change↓	130 MB	343 MB	235 MB	1062 MB
Average Complexity↓	3G FLOPs	10G FLOPs	13G FLOPs	38G FLOPs
Average Execution Time↓	1.00 second	3.34 seconds	1.31 seconds	5.55 seconds

K.2 Comparison to In-context Learning

We did extra experiments with LLaMA2 [Touvron et al., 2023] on the FEVER dataset using in-context learning (ICL). In these experiments, few-shot prompts improve performance compared to zero-shot prompts (i.e., Baseline), as shown in Table 23. For example, the accuracy of 4-shot prompts is 76.2%, and the accuracy of 16-shot prompts is 77.2%, while the accuracy of zero-shot prompts is 59.3%. Therefore, ICL as a tuning-free method, is also expected to achieve better accuracy but at a higher cost (for example, adding more examples from $\mathcal{D}_{\text{subset}}$ with longer tokens, incurring more inference cost).

Table 23: The higher the cost in ICL computation, the better the accuracy overall.

LLaMA2 on FEVER	Baseline	Ours (std)	ICL (1-shot)	ICL (2-shot)	ICL (4-shot)	ICL (8-shot)	ICL (16-shot)
Accuracy↑	59.3	64.5	74.6	75.4	76.2	74.3	77.2
Additional Tokens↓	0	0	33	61	114	237	461

K.3 Comparison to Simpler Forward-Only Baselines

To have the new forward-only baseline, we directly replace the module of angle concentration (cosine similarity) by activation norms and activation variance, cosine dispersion without SVD decomposition, as well as the random top-layer selection, while the harness is the same as our approach. Note that cosine dispersion without SVD decomposition directly measures the dispersion of different weights. Overall, the results are shown in Table 24.

Table 24: Our forward-only approach is more effective and competitive than other simpler forward-only baselines. ⋆Note that for activation norms (Roberta on Bios Profession), the top-5 weight matrices are $[\boldsymbol{W}_{out}, 8], [\boldsymbol{W}_{out}, 9], [\boldsymbol{W}_{in}, 1], [\boldsymbol{W}_{in}, 2], [\boldsymbol{W}_{in}, 3]$. Therefore, by the harness of our efficient hierarchical search, the final candidate is $[\boldsymbol{W}_{in}, 3]$.

Roberta	Bios Gender		Bios Profession	
	Final Candidate	Accuracy↑(%)	Final Candidate	Accuracy↑(%)
Direct Inference	-	87.5	-	64.5
Activation Norms	$[\boldsymbol{W}_{in}, 2, 80\%]$	90.0	$[\boldsymbol{W}_{in}, 3, 90\%]\star$	71.9
Activation Variance	$[\boldsymbol{W}_{out}, 11, 5\%]$	91.8	$[\boldsymbol{W}_{out}, 7, 1\%]$	67.2
Cosine Dispersion Without SVD	$[\boldsymbol{W}_{out}, 0, 40\%]$	89.2	$[\boldsymbol{W}_{out}, 1, 1\%]$	70.9
Random Top-Layer Selection	$[\boldsymbol{W}_{in}, 11, 1\%]$	91.8	$[\boldsymbol{W}_{out}, 11, 40\%]$	65.1
Random Top-Layer Selection	$[\boldsymbol{W}_{out}, 10, 60\%]$	89.9	$[\boldsymbol{W}_{in}, 11, 80\%]$	64.9
Ours (hierarchical)	$[\boldsymbol{W}_{in}, 9, 90\%]$	93.4	$[\boldsymbol{W}_{in}, 3, 90\%]$	71.9

The results first show that random top-layer selection underperforms our approach in the evaluated settings. For Roberta on Bios Profession, the activation variance cannot compete with our approach. While the activation norms search the same final candidate as ours, resulting in 71.9% accuracy, activation norms cannot be generalized to other cases, such as Roberta on Bios Gender. For Roberta on Bios Gender, the activation norms and activation variance are both worse than our approach. Moreover, the cosine dispersion without SVD decomposition also cannot compete with our approach over Bios Gender and Bios Profession. Overall, our approach, as the forward-only approach, is more effective and competitive than other simpler forward-only baselines.

L 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: The claims made match the theoretical results in Section 2 and Appendix C, Section E. The claims made match the experimental results in Section 3.2.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: We discuss the limitation that our proposed forward-only approach does not aim to directly compete with fine-tuning approaches, but rather aims to enhance the fine-tuning process, as presented in Section 3.2.3 and Fig. 4.

   Guidelines:

   - The answer [N/A] 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: For the theoretical result for speedup, we provide the assumption and proof in Appendix C. For the theoretical results for peak memory consumption, we provide the assumption and proof in Appendix E. For the relationship between angle concentration of hidden states and the gradients of singular values, we provide the intuition in Appendix I.

Guidelines:

- The answer [N/A] 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: We provide the final candidates for various LLMs in Appendix A, B, and H. Moreover, for the enhancing fine-tuning experiments, we also provide the hyperparameters in Appendix G. We also provide the implementation code in the supplementary material.

Guidelines:

- The answer [N/A] 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: We provide the implementation code with the README in the supplementary material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the experimental details in Section 3.1 with the found final candidates in Appendix A, B, and H. Furthermore, for the enhancing fine-tuning experiments, we also provide the hyperparameters in Appendix G.

   Guidelines:

   - The answer [N/A] 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: We report the means and standard deviations with multiple random seeds for **Enhancing Fine-Tuning** in Section 3.2.3 and Fig. 4. Besides, there is no need to have means and standard deviations for deterministic SVD decomposition.

   Guidelines:

   - The answer [N/A] 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: We provide the number and the type of GPU and the RAM requirements in Section 3.1 and Section H.

   Guidelines:

   - The answer [N/A] 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: All data used in this paper are from public, standard benchmarks that comply with their respective licenses.

   Guidelines:

   - The answer [N/A] 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: [N/A]

Justification: This paper discusses the potential positive societal impacts in Section 5, such as *This paper opens a new perspective on how the inference process itself can be leveraged to improve LLM serving systems by enabling efficient task-specific LLM adaptation*. Moreover, this paper is foundational research and not tied to particular applications, let alone deployments.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper poses no such risks because this paper is a foundational methodology. All the models and datasets used in this paper are public and open-access.

Guidelines:

- The answer [N/A] 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: We cite the references for datasets and LLMs used in Section 3.1 and Appendix H.

Guidelines:

- The answer [N/A] 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: [Yes]

    Justification: We provide the implementation code in the supplementary material.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] 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.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We declare the LLM usage in Appendix D.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.