*A*dvantageous *P*arameter *EX*pansion Training Makes Better Large Language Models

Anonymous ACL submission

Abstract

Although scaling up the number of trainable parameters can effectively improve the training performance of large language models, it also leads to increased computational overhead. When delving into the parameter difference, we find that a subset of parameters, termed advantageous parameters, plays a crucial role in determining model performance. Further analysis reveals that stronger models tend to possess more such parameters. In this paper, we propose **A**dvantageous **P**arameter **EX**pansion Training (APEX), a method that progressively expands advantageous parameters into the space of disadvantageous ones, thereby increasing their proportion and enhancing training effectiveness, while keeping the total parameter count unchanged. Extensive experiments on both instruction tuning and continued pre-training across five base models demonstrate that, in instruction tuning, APEX outperforms full-parameter tuning while using only 52% of the trainable parameters. In continued pre-training, APEX achieves the same perplexity level as conventional training with just 33% of the training data, and yields significant improvements on downstream tasks.

1 Introduction

Large language models (LLMs) have demonstrated remarkable performance across various natural language processing tasks (OpenAI et al., 2024; Touvron et al., 2023a,b). A prevailing strategy to enhance the model's performance is to scale up trainable parameters, as evidenced in both pre-training (DeepSeek-AI et al., 2025; Cai et al., 2024; Yang et al., 2024a) and fine-tuning (Ivison et al., 2023; Meng et al., 2024): more parameters tend to improve effectiveness. However, despite its success, this scaling paradigm entails substantial computational cost, posing significant challenges for both training and deployment.

Intriguingly, large language models are known to exhibit substantial parameter redundancy (Frankle and Carbin, 2019; Ma et al., 2023; Li et al., 2024). Recent studies have revealed that this redundancy is highly structured: parameters associated with high-magnitude activations disproportionately control information flow and model outputs (Yu et al., 2025; Wang et al., 2024; An et al., 2024). For example, removing up to 50% of parameters with low activation magnitudes causes negligible performance loss (Sun et al., 2024b), suggesting that the model capability concentrates in a subset of parameters. Herein, we refer to these influential parameters as **advantageous parameters**, while noting that a substantial portion of capacity remains underutilized. This observation raises a pivotal research question: *Can we optimize training strategies to increase the proportion of advantageous parameters, thereby maximizing model capabilities without merely scaling up the parameters?*

To answer this question, we first analyze activation distributions across different components in modern LLMs, confirming that high-activation parameters consistently dominate model outputs. On this basis, we investigate the relationship between activation distribution and model capability. By comparing several groups of strong and weak models (i.e., higher vs. lower benchmark performance) of the same scale, we observe that weak models tend to exhibit sharper activation distributions, suggesting that model output is dominated by a subset of parameters while the majority remain underutilized. In contrast, strong models display flatter, more balanced distributions, indicating a broader set of advantageous parameters and more uniform contributions. Based on this observation, we hypothesize that model performance is not solely dictated by the total parameter budget, but by how effectively that budget is utilized.

In this paper, we present **A**dvantageous **P**arameter **EX**pansion Training (APEX), a novel training strategy that expands the proportion of ad-

vantageous parameters without altering the model scale and architecture. Specifically, to progressively expand the advantageous parameter space, APEX divides training into multiple stages. At the beginning of each stage, APEX selects the sets of advantageous and disadvantageous parameters based on the assessment results from the previous stage. The assessment is performed by tracking the activations of heads in the Multi-Head Attention (MHA) and channels in the Feed-Forward Network (FFN) during each forward pass. Building upon the assessment, APEX introduces expansion operators into each MHA and FFN module. In MHA, the head expansion operators expand the parameter space from advantageous heads to disadvantageous ones, and similarly for channels in the FFN, thereby increasing the proportion of advantageous parameters. By initializing these operators to zero at the start and fusing them back into the model at the end of each stage, APEX preserves the original architecture and integrates smoothly into standard training procedures.

We conduct extensive experiments in two common training scenarios: instruction tuning and continued pre-training, using five base models of different sizes (1B to 13B). In the instruction tuning scenario, experiments show that APEX consistently outperforms 9 strong baselines, including parameter-efficient tuning, half-parameter tuning, and full-parameter tuning methods. Remarkably, it achieves better performance than full-parameter tuning using only 52% of the trainable parameters. In the continued pre-training scenario, APEX achieves the perplexity level of conventional training using only 33% of the data budget, demonstrating a convergence speedup. Furthermore, evaluations on 10 representative downstream tasks demonstrate that APEX significantly outperforms conventional training. These empirical results are further supported by both observations of activation distribution shifts and a simplified theoretical model (App. C), indicating that APEX improves the utilization of previously under-used subspaces. Overall, APEX demonstrates a strong capability to enhance model training performance through advantageous parameter expansion.

2 Observations

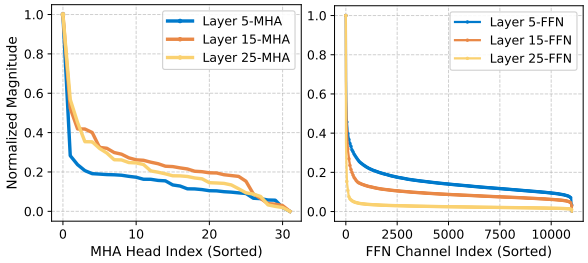
Background and Notations. In Transformer-based models, each layer mainly consists of a Multi-Head Attention (MHA) module and a Feed-Forward Network (FFN) module. Each MHA module comprises H attention heads parameterized by projection matrices $\mathbf{W}_Q^h, \mathbf{W}_K^h, \mathbf{W}_V^h \in \mathbb{R}^{d_{\text{model}} \times d_{\text{head}}}$ and $\mathbf{W}_O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$, where $d_{\text{head}} = d_{\text{model}}/H$. Given an input $\mathbf{X}$, we have:

$$\text{MHA}(\mathbf{X}) = \text{Concat}(\text{head}_1, ..., \text{head}_H)\mathbf{W}_O, \quad (1)$$

Most LLMs employ a Gated Linear Units structure (Dauphin et al., 2017) in FFNs, and each module involves three weight matrices: $\mathbf{W}_U, \mathbf{W}_G \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ffn}}}$ for upward projection and gating, and $\mathbf{W}_D \in \mathbb{R}^{d_{\text{ffn}} \times d_{\text{model}}}$. For the input $\mathbf{X}$, we have:

$$\text{FFN}(\mathbf{X}) = ((\mathbf{X}\mathbf{W}_U) \odot f(\mathbf{X}\mathbf{W}_G))\mathbf{W}_D, \quad (2)$$

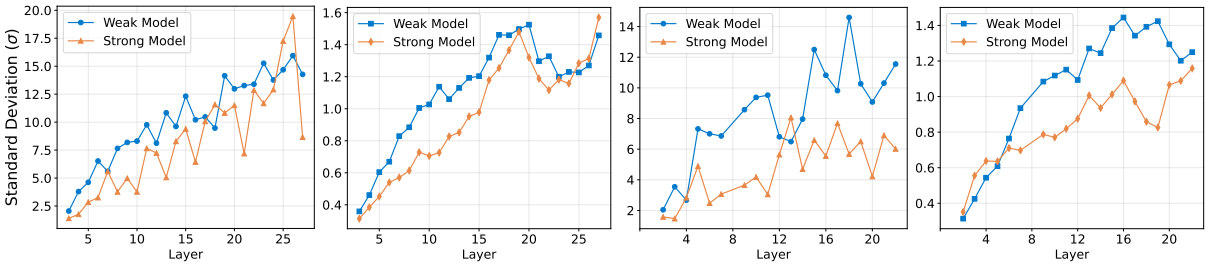
where $f(\cdot)$ denotes the activation function and $\odot$ denotes the element-wise multiplication. Motivated by previous findings that only a subset of parameters is crucial for model performance, we are curious about the prevalence of such advantageous parameters across different components, as well as the relationship between the proportion of advantageous parameters and model performance. Following prior work (Sun et al., 2024b; An et al., 2024), we examine the activation magnitudes of MHA heads and FFN channels on a calibration dataset (Ivison et al., 2023). Specifically, we compute the Frobenius norm of the output for each head h and the post-activation output for each channel c: $\mathbf{A}_{\text{MHA}}^h = \|\text{head}_h\|_F^2$ and $\mathbf{A}_{\text{FFN}}^c = \|f(\mathbf{X}\mathbf{W}_G^{[:,c]})\|_F^2$.



(a) MHA Activations (b) FFN Activations

Figure 1: Normalized activation magnitudes of MHA heads and FFN channels. Both are based on LLaMA2-7B and sorted in descending order.

Differences in Parameter Advantages. Figures 1a and 1b reveal a distinct long-tailed pattern in activation magnitudes across both MHA and FFN modules. This phenomenon is prevalent across various layers. We further conduct an experiment (App. B.1) and find that removing parameters with high activations leads to catastrophic degradation in model performance, whereas removing pa-



(a) 7B models-MHA (b) 7B models-FFN (c) 3B models-MHA (d) 3B models-FFN

Figure 2: Activation distribution comparison between models with similar architectures but different performance (i.e., strong vs. weak models). One group consists of standard 7B models trained from scratch (Touvron et al., 2023a; Jiang et al., 2023). Another group includes 3B models obtained via model compression (Xia et al., 2024; Grattafiori et al., 2024). See Appendix A for details.

rameters with low activations has little impact. This indicates that across both MHA and FFN modules, a subset of parameters dominates model capability, while the remainder contributes less.

The Relationship Between Advantageous Parameters and Performance. We further explore the relationship between activation distribution and model capability. We employ standard deviation σ to quantify distribution sharpness, where a higher σ indicates a more skewed distribution dominated by fewer parameters. As illustrated in Figure 2, we compare models of similar architectures but different benchmark performance (i.e., strong vs. weak). We observe that weaker models tend to exhibit sharper activation distributions, suggesting that they possess fewer advantageous parameters while the majority remain underutilized. In contrast, stronger models display more uniform patterns, indicating a broader set of advantageous parameters that contribute more evenly. This motivates our hypothesis that model performance depends not only on parameter count, but on how effectively parameters are utilized.

3 Method

In this section, we introduce **A**dvantageous **P**arameter **EX**pansion Training (APEX) to increase the proportion of advantageous parameters (Fig. 3). Sec. 3.1 details how we assess advantageous parameters. Sec. 3.2 introduces expansion operators for MHA and FFN modules. Sec. 3.3 describes the stage-wise APEX implementation, and App. C provides a theoretical perspective on its mechanism.

3.1 Advantage Assessment

As observed in Sec. 2, parameter contributions to model performance are highly uneven. To effectively expand the advantages of parameters with higher contributions to those with lower contributions, the primary issue to be addressed is how to accurately assess the advantages of various parameters. A naive approach is to calculate the average output activations, but it faces two problems, (1) the activation magnitudes vary to a relatively large extent across different input samples, and (2) some parameters may exhibit abnormally high activations in specific data (Sun et al., 2024a), leading to incorrect estimation of advantages.

Based on the Frobenius norm used in observations, we propose a robust metric with relative ranking. For each MHA and FFN module, we calculate the difference between the number of times it appears in the Top-K and Min-K by activation magnitude across the given dataset $\mathcal{D}$. Specifically, for the h-th head, the score is computed as:

$$\begin{aligned} \mathbf{s}^h_{\text{MHA}} = &\sum_{\mathcal{D}} \mathbb{I}[\mathbf{A}^h_{\text{MHA}} \in \text{Top-K}(\mathbf{A}_{\text{MHA}})] \\ &- \sum_{\mathcal{D}} \mathbb{I}[\mathbf{A}^h_{\text{MHA}} \in \text{Min-K}(\mathbf{A}_{\text{MHA}})], \end{aligned} \tag{3}$$

where $\mathbb{I}[\cdot]$ is the indicator function, it returns 1 if the condition inside holds true and 0 otherwise. Unlike simple averaging, this voting-based mechanism filters out the noise from sporadic outliers. Based on the scores, the sets of advantageous parameters and disadvantageous parameters can be selected:

$$d^P_{\text{MHA}} = \{h \mid \mathbf{s}^h_{\text{MHA}} \in \text{Top-K}(\mathbf{s}_{\text{MHA}})\}, \tag{4}$$

$$d^N_{\text{MHA}} = \{h \mid \mathbf{s}^h_{\text{MHA}} \in \text{Min-K}(\mathbf{s}_{\text{MHA}})\}, \tag{5}$$

where d^P_{MHA} and d^N_{MHA} denote the dimension indices of the advantageous and disadvantageous parameters set, respectively[1]. Similarly, we select the

[1] P and N stand for positive (advantageous) and negative (disadvantageous).

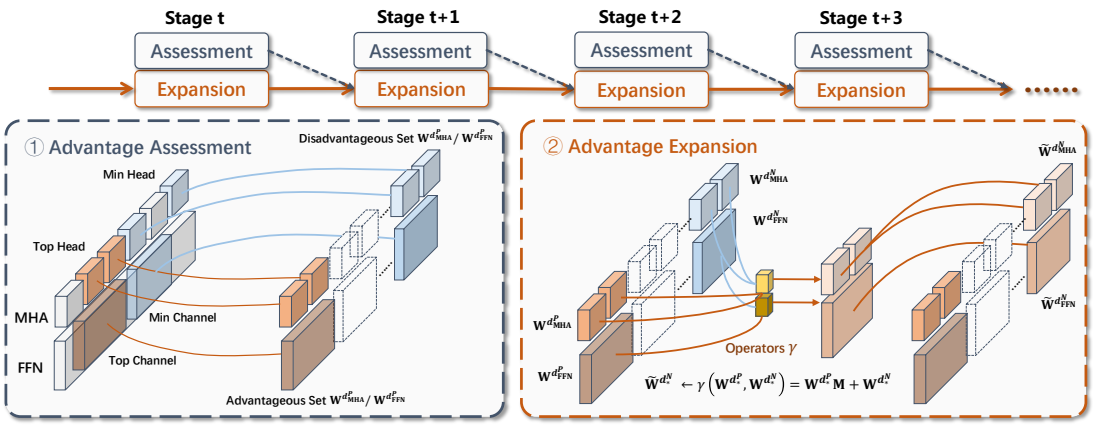


Figure 3: Overview of APEX Training. To progressively expand the advantageous parameter space, APEX divides the entire training process into multiple stages. At the beginning of each stage, we reselect the sets of advantageous and disadvantageous parameters, and then initialize the new operators based on assessment results collected from the previous stage. In the expansion training of each stage, we continuously perform advantage assessment during each forward pass for the parameter sets selection of the next stage. At the end of each stage, we fuse the operators into the model to ensure that the model architecture remains unchanged.

sets of advantageous parameters d^P_{FFN} and disadvantageous parameters d^N_{FFN} in the FFN module:

$$\begin{aligned} \mathbf{s}^c_{\text{FFN}} = &\sum_{\mathcal{D}} \mathbb{I}[\mathbf{A}^c_{\text{FFN}} \in \text{Top-K}(\mathbf{A}_{\text{FFN}})] \\ &- \sum_{\mathcal{D}} \mathbb{I}[\mathbf{A}^c_{\text{FFN}} \in \text{Min-K}(\mathbf{A}_{\text{FFN}})], \end{aligned} \tag{6}$$

$$d^P_{\text{FFN}} = \{c \mid \mathbf{s}^c_{\text{FFN}} \in \text{Top-K}(\mathbf{s}_{\text{FFN}})\}, \tag{7}$$

$$d^N_{\text{FFN}} = \{c \mid \mathbf{s}^c_{\text{FFN}} \in \text{Min-K}(\mathbf{s}_{\text{FFN}})\}. \tag{8}$$

For simplicity, we use a unified threshold K for both MHA and FFN throughout all experiments.

3.2 Advantage Expansion Operators

MHA Expansion Operators. Given an advantageous set head_P and a disadvantageous set head_N, we introduce an expansion operator γ_{MHA} to expand the advantages of head_P to head_N:

$$\widetilde{\text{head}}_N \leftarrow \gamma_{\text{MHA}}(\text{head}_P, \text{head}_N), \tag{9}$$

where the operator γ_{MHA} is realized by applying a learnable transformation to parameter sets head_P and head_N, resulting in a new set of parameters $\widetilde{\text{head}}_N$ that substitutes for the output of the original parameter set head_N. Afterwards, considering $\mathbf{W}_O$ receives the concatenated outputs from all heads, its dimensions correspond to those of each head. Therefore, when expanding the heads, the corresponding input dimensions in $\mathbf{W}_O$ should also be expanded to align with $\widetilde{\text{head}}_N$ accordingly:

$$\tilde{\mathbf{W}}_O^{[d^N_{\text{MHA}},:]} \leftarrow \gamma_{\text{MHA}}(\mathbf{W}_O^{[d^P_{\text{MHA}},:]}, \mathbf{W}_O^{[d^N_{\text{MHA}},:]}), \tag{10}$$

FFN Expansion Operators. Similar to head expansion in the MHA module, we expand advantageous channels onto disadvantageous ones in the FFN module. Given the set of dimension indices d^P_{FFN} and d^N_{FFN}, we introduce the expansion operator γ_{FFN} to perform this expansion:

$$\tilde{\mathbf{W}}_U^{[:,d^N_{\text{FFN}}]} \leftarrow \gamma_{\text{FFN}}(\mathbf{W}_U^{[:,d^P_{\text{FFN}}]}, \mathbf{W}_U^{[:,d^N_{\text{FFN}}]}), \tag{11}$$

$$\tilde{\mathbf{W}}_G^{[:,d^N_{\text{FFN}}]} \leftarrow \gamma_{\text{FFN}}(\mathbf{W}_G^{[:,d^P_{\text{FFN}}]}, \mathbf{W}_G^{[:,d^N_{\text{FFN}}]}), \tag{12}$$

Next, the corresponding input dimensions in $\mathbf{W}_D$ also need to be operated for alignment:

$$\tilde{\mathbf{W}}_D^{[d^N_{\text{FFN}},:]} \leftarrow \gamma_{\text{FFN}}(\mathbf{W}_D^{[d^P_{\text{FFN}},:]}, \mathbf{W}_D^{[d^N_{\text{FFN}},:]}). \tag{13}$$

Implementation of Operators. Recall that the goal of expansion is to improve the utilization of the under-utilized (disadvantageous) subspace by propagating directions spanned by advantageous parameters to it, thereby increasing the number of parameters that behave advantageously during training. To this end, we introduce a transformation matrix $\mathbf{M} \in \mathbb{R}^{d^P_* \times d^N_*}$ for the matrix $\mathbf{W}^{[:,d^P_*]}$ to be expanded into the space of $\mathbf{W}^{[:,d^N_*]}$, where $*$ denotes either MHA or FFN, and d^P_*, d^N_* are the corresponding dimension indices:

$$\begin{aligned} \tilde{\mathbf{W}}^{[:,d^N_*]} &\leftarrow \gamma(\mathbf{W}^{[:,d^P_*]}, \mathbf{W}^{[:,d^N_*]}) \\ &= \mathbf{W}^{[:,d^P_*]}\mathbf{M} + \mathbf{W}^{[:,d^N_*]}, \end{aligned} \tag{14}$$

where the transformation matrix $\mathbf{M}$ is initialized with zeros to ensure that the starting point of the expansion remains consistent with the original model.

As analyzed in App. C, the expansion creates an auxiliary optimization path: updates to $\mathbf{M}$ induce an additional structured term in the effective update of $\tilde{\mathbf{W}}^{[:,d_*^N]}$ that is aligned with the span of $\mathbf{W}^{[:,d_*^P]}$ (see Lemma 1). Moreover, since projection does not increase the expected energy of the gradient noise (Lemma 2), this correction can improve the effective signal-to-noise ratio of updates in under-utilized directions, enabling them to participate more during training.

After the expansion is completed, in order to preserve the original architecture, we fuse the expansion operators into the model: $\mathbf{W}^{[:,d_*^N]} \leftarrow \tilde{\mathbf{W}}^{[:,d_*^N]}$. Such designs allow APEX to be incorporated into standard training procedures seamlessly. In addition, to decrease the computational complexity of the transformation matrix, we leverage Monarch matrices (Dao et al., 2022) to reconstruct it:

$$\mathbf{M} = \begin{bmatrix} \mathbf{D}_{1,1} & \cdots & \mathbf{D}_{1,d} \\ \vdots & \ddots & \vdots \\ \mathbf{D}_{d,1} & \cdots & \mathbf{D}_{d,d} \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 & & \\ & \ddots & \\ & & \mathbf{R}_d \end{bmatrix}, \tag{15}$$

where $d = \sqrt{\min\left(d_*^P, d_*^N\right)}$ and each $\mathbf{D}$,$\mathbf{R}$ is a $d \times d$ diagonal matrix, thus the mapping complexity is reduced from $O(d^4)$ to $O(d^2)$. After reconstructing the matrix, in order to perform zero initialization, we initialize each $\mathbf{R}$ matrix to zeros and randomly initialize each $\mathbf{D}$ matrix. App. B.3 shows the parameter efficiency benefits of the operator.

3.3 Stage-wise Training

To progressively expand the advantageous parameter space, we divide the entire training process into multiple stages. At the beginning of each stage, we reselect the sets of advantageous and disadvantageous parameters and initialize the new operators based on statistics collected from the previous stage. During the training of each stage, we continuously record metric scores $\mathbf{s}_{\text{MHA}}$ and $\mathbf{s}_{\text{FFN}}$ during each forward pass for the parameter sets selection of the next stage[2]. At the end of each stage, we fuse the operators into the model:

$$\mathbf{W}_t^{[:,d_*^N]_{\text{final}}} = \mathbf{W}_t^{[:,d_*^P]}\mathbf{M}_t + \mathbf{W}_t^{[:,d_*^N]}, \text{ where } t \in T, \tag{16}$$

where T denotes the total number of stages. $\mathbf{W}_t^{[:,d_*^N]_{\text{final}}}$ denotes the parameter values corresponding to the disadvantageous parameter sets after the operators have been fused at stage t. Regarding training efficiency, since the assessment is fused into the forward pass and the Monarch-based expansion operator is lightweight, APEX maintains training overhead comparable to other sparse fine-tuning methods (see App. B.3 for details).

[2]For the first stage, we identify advantageous parameters by performing forward passes on a small subset of the training set prior to training. In each stage, $\mathcal{D}$ actually refers to the training data of the previous stage.

4 Empirical Evaluation

We conduct experiments under two scenarios: instruction tuning and continued pre-training. For instruction tuning, we compare methods across different trainable parameter settings. For continued pre-training, we assess APEX's improvements in convergence speed and performance on downstream tasks over conventional training.

4.1 Instruction Tuning

Models and Data. To comprehensively evaluate the performance of different methods, following Hui et al. (2025), we use TÜLU V2 instruction-tuning dataset (Ivison et al., 2023) as the training set, which contains a variety of high-quality data. We employ LLaMA2-7B (Touvron et al., 2023b), LLaMA3-8B (Grattafiori et al., 2024), LLaMA2-13B, and Mistral-7B-v0.1 (Jiang et al., 2023) as base models. The evaluation covers several aspects: MMLU (Hendrycks et al., 2021) (factual knowledge), GSM8K (Cobbe et al., 2021) and BBH (Suzgun et al., 2023) (reasoning), TyDiQA (Clark et al., 2020) (multilinguality), TruthfulQA (Lin et al., 2022) (truthfulness), HumanEval (Chen et al., 2021) (coding).

Baselines and Settings. We compare APEX with two baseline categories. The first includes parameter-efficient fine-tuning (PEFT) methods with a minimal number of trainable parameters. Specifically, we compare against widely used approaches LoRA (Hu et al., 2022), as well as its variants DoRA (Liu et al., 2024), ReLoRA (Lialin et al., 2024), PiSSA (Meng et al., 2024), and S^{2}FT (Yang et al., 2024b). The second category consists of half parameter fine-tuning methods that enable approximately half of the model parameters to be trainable, including HFT (Hui et al., 2025), which randomly selects half of the weight matrices to be trainable; RMT and GMT (Li et al., 2025), which masks parameters randomly or based on gradient magnitude. APEX controls the number of trainable parameters by allowing only the selected parameter set and expansion operators to be trainable. More details are presented in App. A.

Method	#Param.	MMLU 0-shot, EM	GSM8K 8-shot CoT, EM	BBH 3-shot CoT, EM	TyDiQA 1-shot, F1	TruthfulQA 0-shot, MC2	HumanEval 0-shot, Pass@10	Average
Pre-trained†	-	41.6	12.0	39.9	48.4	38.5	26.2	34.4
Full-FT	100%	49.6	30.8	45.2	49.4	46.9	36.5	43.1
Parameter-Efficient Fine-tuning								
LoRA	2.4%	44.1	26.5	44.1	54.0	43.7	28.8	40.2
DoRA	2.4%	47.7	25.5	44.0	55.0	45.4	31.8	41.6
PiSSA	2.4%	48.2	25.5	42.5	50.6	45.5	33.6	41.0
ReLoRA	2.4%	47.4	25.5	43.4	55.1	43.9	31.4	41.1
S^2FT	2.4%	47.6	25.0	45.5	52.8	43.7	33.9	41.4
APEX	2.4%	48.1	28.5	45.7	54.5	45.5	33.2	**42.6**
Half Parameter Fine-tuning								
HFT†	52%	50.8	30.5	43.6	52.3	45.4	34.6	42.9
APEX	52%	49.5	33.5	46.5	54.0	46.3	34.2	**44.0**
RMT‡	60%	47.4	34.5	44.4	52.9	47.9	33.7	43.4
GMT‡	60%	47.6	38.0	43.3	53.1	47.5	34.6	44.0
APEX	60%	50.0	34.5	46.4	53.9	45.9	36.6	**44.6**

Table 1: Instruction Tuning results for LLaMA2-7B. #Param. denotes trainable parameters. The results are averaged over three runs. All results with † are taken from HFT (Hui et al., 2025) and those marked with ‡ are taken from GMT (Li et al., 2025). Our experiments are conducted under the exact same settings as these works. More results on LLaMA3-8B, LLaMA2-13B, and Mistral-7B-v0.1 are provided in Appendix B.2.

Results. Table 1 reports the results of instruction tuning. Compared with other PEFT methods and half-parameter tuning approaches under various trainable parameter settings, APEX consistently achieves superior average performance. Specifically, in the PEFT setting, APEX achieves an average improvement of over 1% across multiple tasks compared to other methods. Notably, it uses only 2.4% of trainable parameters, yet approaches the performance of HFT, which requires training half of the model parameters. In the half-parameter tuning methods, APEX (with 52% parameters) clearly outperforms HFT and Full-FT, while matching the performance of GMT, which requires 60%. Furthermore, when the budget is increased to 60%, APEX pushes the model capability to a new performance peak. These results demonstrate the effectiveness of advantage expansion. More results on other models are provided in App. B.2. Overall, APEX consistently demonstrates superior performance across various models.

4.2 Continued Pre-Training

Models and Data. We utilize TinyLLaMA-v1-1.1B (Zhang et al., 2024) as the base model and use the RedPajama dataset (Weber et al., 2024) for continued pre-training. The training corpus consists of 10 billion tokens, accompanied by a validation set with 2 million tokens. For evaluation, we follow the procedures established by LLaMA2 and report accuracy across 10 representative downstream tasks: ARC-Easy/Challenge (Clark et al., 2018), LAMBADA (Paperno et al., 2016), LogiQA (Liu et al., 2020), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), WinoGrande (Sakaguchi et al., 2020), HellaSwag (Zellers et al., 2019), MMLU, and BoolQ (Clark et al., 2019).

Baselines and Settings. We adopt the conventional continued pre-training of TinyLLaMA (Vanilla) as our primary baseline. For APEX, we first assess parameter advantages using 30 million tokens, then perform continued pre-training on 10 billion tokens. The baseline is trained on the same 10 billion tokens for fair comparison.

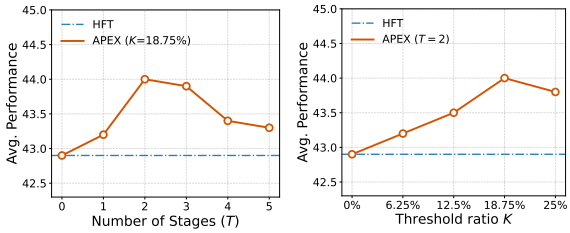
Results. Table 2 presents the performance after continued pre-training with APEX across multiple stages. The results show that APEX achieves a significant reduction in language modeling perplexity (PPL). Specifically, compared to Vanilla CPT, which attains a perplexity of 5.83 after training on 10B tokens, APEX reduces the perplexity to 5.74 after training on only 3B tokens, achieving a lower perplexity with just 33% of the training data. After training on 10B tokens, APEX further reduces the perplexity to 5.44, representing a decrease of approximately 0.39. Notably, the improvement becomes more pronounced as training progresses: the PPL gap between APEX and Vanilla widens from 0.12 at 3B, to 0.30 at 6B, and to 0.39 at 10B, suggesting that advantage expansion provides cumulative benefits. Moreover, the APEX model consistently outperforms the baseline on most of the

Method	#Param.	PPL(↓)	Commonsense & Reading Comprehension						Continued		LM	Knowledge	Avg.
			SciQ	PIQA	WG	ARC-E	ARC-C	Hella.	LogiQA	BoolQ	Lam.	MMLU	
Training budget from $0B$ *to* $3B$													
Vanilla	100%	5.87	87.9	71.8	57.1	58.8	28.0	44.1	20.0	63.9	58.2	24.9	51.5
APEX	100%	5.75	88.4	71.9	57.5	60.8	29.1	44.7	21.4	63.9	58.6	25.5	**52.2**
Training budget from $3B$ *to* $6B$													
Vanilla	100%	5.85	88.1	72.2	58.6	58.5	28.6	44.0	20.7	62.7	57.8	25.1	51.6
APEX	100%	5.55	88.3	72.4	58.8	61.0	29.4	45.7	21.0	64.2	60.2	26.1	**52.7**
Training budget from $6B$ *to* $10B$													
Vanilla	100%	5.83	86.6	71.9	58.6	56.8	33.2	44.4	23.2	62.1	58.8	25.3	52.1
APEX	100%	5.44	88.5	73.1	59.6	61.6	29.9	45.9	20.9	62.9	61.4	26.9	**53.1**

Table 2: Continued pre-training results for APEX and conventional training (Vanilla). The results are averaged over three runs. In APEX, we reselect the advantage and disadvantage parameter sets every 3B data. And the initial point of the next stage is based on the checkpoint of the previous stage.

Method	MML.	GSM.	BBH	TyD.	Tru.	Hum.	Avg.
HFT-52%	50.8	30.5	43.6	52.3	45.4	34.6	42.9
APEX-52%	49.5	33.5	46.5	54.0	46.3	34.2	44.0
w/o Exp.	48.7	31.0	44.4	54.6	46.0	32.7	42.9
w/o Ass.	49.9	30.5	46.3	53.2	45.3	34.2	43.2
w/o Mon.	49.5	32.0	46.8	53.1	45.3	36.0	43.8

Table 3: Ablation experiments of APEX. *w/o Exp.* denotes removing expansion and only finetuning the advantageous parameters; *w/o Ass.* denotes removing assessment and performing random expansion; *w/o Mon.* denotes replacing the Monarch layer with a dense layer.



(a) Effect of stages T. (b) Effect of threshold K.

Figure 4: Effect of the number of stages T and the threshold ratio K. The threshold K is swept up to 25% under the 50% half-parameter tuning budget.

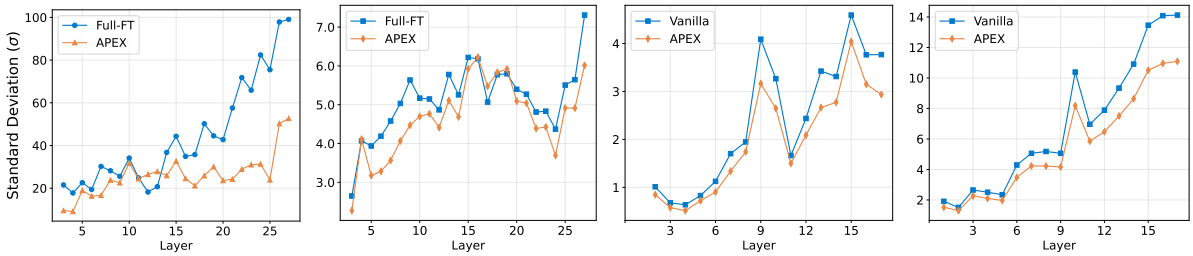
downstream tasks. These results indicate that advantage expansion effectively enhances the model's knowledge storage and reasoning capabilities.

4.3 Analysis

Ablation Study. Table 3 presents the ablation study of our method. Removing either Advantage Expansion or Advantage Assessment significantly degrades performance, confirming that both components are essential. Furthermore, replacing the Monarch matrix with a fully dense linear operator also reduces performance. We attribute this to the dense matrix's excessive degrees of freedom, which introduce noise and instability during finetuning with limited steps. In contrast, the structured Monarch form provides a more stable reparameterization. See App. B.4 for extended analysis.

Effect of the number of stages T. Figure 4a shows that performance improves as T increases from 0 to 2, indicating that expansion–merge cycles effectively enhance learning. The performance peaks at $T = 2$–3, which is close to the number of training epochs, while larger T yields no further benefit. We attribute this to the fact that, under a fixed total step budget, excessive stages result in overly brief training intervals. This leads to insufficient data exposure per stage, which hampers the accurate estimation of advantageous directions for parameter expansion. Consequently, we set T equal to the number of training epochs, which naturally aligns the expansion frequency with the data exposure in our experiments. Corresponding analysis for continued pre-training is provided in App. B.4.

Effect of the threshold K. Figure 4b shows the impact of increasing the selection threshold K on half-parameter tuning performance, where we sweep K up to 25% since half-parameter tuning caps the trainable budget at 50% (i.e., $2K \leq 50\%$). When $K = 0$, no expansion occurs, and APEX performs similarly to HFT. As K increases, performance improves steadily, peaking at $K = 18.75\%$. Further increasing K yields diminishing returns, likely because expanding too many dimensions enlarges the transformation subspace and makes optimization harder without introducing additional useful advantageous directions. Accordingly, for settings with a trainable budget below 50% (e.g., PEFT), we set K to half of the trainable ratio, while for budgets



(a) MHA (SFT) (b) FFN (SFT) (c) MHA (CPT) (d) FFN (CPT)

Figure 5: Comparison of activation distributions between models trained with APEX and those trained with conventional methods under both instruction tuning (SFT) and continued pre-training (CPT) scenarios.

above 50%, we fix $K = 18.75\%$.

Observations. Figure 5 visualizes the standard deviations of MHA and FFN activations in both instruction tuning and continued pre-training scenarios. Models trained with APEX consistently exhibit lower standard deviations compared to conventional training, indicating more balanced activation distributions and broader parameter utilization. This aligns with our observation in Section 2 that stronger models exhibit flatter distributions, confirming that APEX effectively increases the proportion of advantageous parameters.

5 Related Work

Parameter-Efficient Training. The goal of Parameter-Efficient Fine-Tuning (PEFT) is to adapt large models with minimal trainable parameters. Existing approaches effective for LLMs generally fall into two categories: addition-based and partial parameter (or sparse) fine-tuning methods. Among addition-based methods, LoRA (Hu et al., 2022) stands out for its efficiency. Recent variants like DoRA (Liu et al., 2024) and PiSSA (Meng et al., 2024) further optimize the mechanism. Partial parameter methods identify and update only a subset of the original parameters. While early works (Guo et al., 2021; Sung et al., 2021) focused on small models, recent approaches like S^2FT (Yang et al., 2024b) have scaled to LLMs. Despite these advances, some studies (Ivison et al., 2023; Dou et al., 2024) show that PEFT techniques may underperform full-parameter fine-tuning on complex tasks. To bridge this gap, high-ratio tuning methods have emerged: HFT(Hui et al., 2025) randomly fine-tunes half of the parameters, while GMT(Li et al., 2025) employs gradient-based selection. Different from existing approaches, APEX introduces a novel paradigm. By training only operators for expansion and the expanded advantageous and disadvantageous parameters, APEX achieves superior efficacy across various parameter budgets.

Activations in Large Language Models. Recent studies have demonstrated a close correlation between activation magnitudes and parameter importance in large language models (Voita et al., 2023; Sun et al., 2024a; Zhao et al., 2024; Lin et al., 2024; Yu et al., 2025), revealing which parameters contribute most significantly to model outputs. This insight has primarily driven pruning strategies. For instance, Wanda (Sun et al., 2024a) and FLAP (An et al., 2024) prune up to 50% of parameters based on activation metrics without retraining. Collectively, these studies reveal inherent redundancy in LLMs, implying substantial underutilized capacity. Unlike pruning methods that discard "dormant" parameters, APEX adopts a constructive approach by treating them as available resources. By expanding advantageous weights to these parameters, APEX maximizes parameter efficiency and performance without altering the model architecture.

6 Conclusion

In this paper, we introduce **A**dvantageous **P**arameter **EX**pansion Training (APEX), a novel paradigm designed to maximize parameter utilization without scaling up model size. Specifically, APEX identifies advantageous parameters and progressively expands them into the space of disadvantageous parameters, thereby increasing their proportion and improving model capabilities. Extensive experiments in both instruction tuning and continued pre-training scenarios show that APEX can achieve the performance of models with more trainable parameters, thus reducing computational costs. Crucially, APEX preserves the original architecture of the model and can be seamlessly integrated into standard training procedures, offering research value and broad application potential.

Limitations

APEX demonstrates its effectiveness in training scenarios that build upon pre-existing model parameters, such as continued pre-training or fine-tuning processes. However, for scenarios where pre-training is conducted entirely from scratch, it is necessary to first conduct conventional training for a period before transitioning to APEX training. Our future work will investigate how to better extend APEX to support this scenario.

Ethical Considerations

In this study, we employ publicly available datasets and techniques to avoid privacy concerns. These resources are used solely to investigate APEX, our proposed efficiency-focused algorithm, with the aim of reducing training costs for a wide range of model users, thus offering a positive social impact. To minimize the potential impact of data bias on our experimental results, we select large-scale and diverse datasets for our experiments. In addition, we strictly adhered to relevant laws, regulations, and ethical guidelines throughout the research process. No sensitive or personal information is involved, nor is any unfair treatment directed toward specific groups. All results and methodologies are reported transparently to facilitate reproducibility and further ethical evaluation by the research community.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-based adaptive structured pruning for large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 10865–10873. AAAI Press.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. *Preprint*, arXiv:2304.01373.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, and 81 others. 2024. Internlm2 technical report. *Preprint*, arXiv:2403.17297.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *Preprint*, arXiv:1905.10044.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

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

Tri Dao, Beidi Chen, Nimit Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. 2022. Monarch: Expressive structured matrices for efficient and accurate training. *Preprint*, arXiv:2204.00595.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941. PMLR.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945, Bangkok, Thailand. Association for Computational Linguistics.

Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. 2024. Open llm leaderboard v2. `https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard`.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. 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 4884–4896, Online. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

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

Tingfeng Hui, Zhenyu Zhang, Shuohuan Wang, Weiran Xu, Yu Sun, and Hua Wu. 2025. HFT: Half fine-tuning for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12791–12819, Vienna, Austria. Association for Computational Linguistics.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *Preprint*, arXiv:2311.10702.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations – democratizing large language model alignment. *Preprint*, arXiv:2304.07327.

Haoling Li, Xin Zhang, Xiao Liu, Yeyun Gong, Yifan Wang, Qi Chen, and Peng Cheng. 2025. Enhancing large language model performance with gradient-based parameter selection. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 24431–24439. AAAI Press.

Lujun Li, Peijie Dong, Zhenheng Tang, Xiang Liu, Qiang Wang, Wenhan Luo, Wei Xue, Qifeng Liu, Xiaowen Chu, and Yike Guo. 2024. Discovering sparsity allocation for layer-wise pruning of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. 2024. ReloRA: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for llm compression and acceleration. *Preprint*, arXiv:2306.00978.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. 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, Dublin, Ireland. Association for Computational Linguistics.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *Preprint*, arXiv:2306.02707.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

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

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

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. 2024a. Massive activations in large language models. In *First Conference on Language Modeling*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024b. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.

Yi-Lin Sung, Varun Nair, and Colin Raffel. 2021. Training neural networks with fixed sparse masks. In *Advances in Neural Information Processing Systems*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

The Mosaic ML Team. 2021. composer. `https://github.com/mosaicml/composer/`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. 2023. Neurons in large language models: Dead, n-gram, positional. *Preprint*, arXiv:2309.04827.

Yudong Wang, Damai Dai, and Zhifang Sui. 2024. Exploring activation patterns of parameters in language models. *Preprint*, arXiv:2405.17799.

Maurice Weber, Daniel Y Fu, Quentin Gregory Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Re, Irina Rish, and Ce Zhang. 2024. Redpajama: an open dataset for training large language models. In *The Thirty-eight Conference on*

Neural Information Processing Systems Datasets and Benchmarks Track.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *Preprint*, arXiv:2304.12244.

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, and 43 others. 2024a. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Xinyu Yang, Jixuan Leng, Geyang Guo, Jiawei Zhao, Ryumei Nakada, Linjun Zhang, Huaxiu Yao, and Beidi Chen. 2024b. S2ft: Efficient, scalable and generalizable llm fine-tuning by structured sparsity. In *The 38th Conference on Neural Information Processing Systems (NeurIPS)*.

Mengxia Yu, De Wang, Qi Shan, Colorado J Reed, and Alvin Wan. 2025. The super weight in large language models. *Preprint*, arXiv:2411.07191.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *Preprint*, arXiv:2401.02385.

Yang Zhao, Li Du, Xiao Ding, Kai Xiong, Ting Liu, and Bing Qin. 2024. Supervised fine-tuning achieve rapid task adaption via alternating attention head activation patterns. *Preprint*, arXiv:2409.15820.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. *Preprint*, arXiv:2305.11206.

Appendix

A Implemention Details

A.1 Observations

In our observational experiments, we investigate the relationship between the proportion of advantageous parameters and model performance by examining two types of base models when selecting strong and weak candidates. The first type comprises standard pre-trained models; we use LLaMA-7B (Touvron et al., 2023a) and Mistral-7B (Jiang et al., 2023) for comparison, as they possess highly similar architectural designs, with empirical results demonstrating that the latter significantly outperforms the former (Fourrier et al., 2024). The second type consists of efficient small models obtained by compressing standard base models. Specifically, we select Sheared-LLaMA-2.7B (Xia et al., 2024), pruned from LLaMA2 (Touvron et al., 2023b), and LLaMA3.2-3B, obtained from the larger LLaMA3 model. Our activation statistics are based on inference performed on 2048 samples with the same fixed random seed in GSM8K (Cobbe et al., 2021). For each component, we average the activations across samples and compute the standard deviation of these averaged activations across different heads in each MHA module and across different channels in each FFN module.

A.2 Instruction Tuning

In our instruction tuning experiments, we use LLaMA2-7B, LLaMA2-13B (Touvron et al., 2023b), LLaMA3.1-8B (Grattafiori et al., 2024), and Mistral-7B-v0.1 (Jiang et al., 2023) as our backbone models. We consider strong baselines under different trainable parameter settings for instruction fine-tuning, including parameter-efficient fine-tuning, half-parameter fine-tuning, and full-parameter fine-tuning for comparison.

Baselines. For PEFT methods, we select LoRA (Hu et al., 2022), which simulates full-rank model updates using a low-rank matrix. We also include three subsequent improvements to LoRA: DoRA (Liu et al., 2024), which decomposes LoRA modules into magnitude and direction components to effectively enhance LoRA's learning capacity, and ReLoRA (Lialin et al., 2024), which repeatedly merges trained LoRA parameters back into the base model and reinitializes the LoRA modules, thereby increasing the model's capacity, and PiSSA (Meng et al., 2024), which utilizes principal

singular values to initialize the LoRA modules. All of their ranks are set to 64. Additionally, we select the S^2FT (Yang et al., 2024b) method, which randomly trains either a row or a column of weights in the base model. For both S^2FT and our proposed method, APEX, we set the number of trainable parameters to match that of rank=64 in LoRA-based methods. For half-parameter fine-tuning methods, HFT (Hui et al., 2025) randomly selects half of the weight matrices along with all embedding and LM head layers for updating, resulting in 52% trainable parameters. RMT (Li et al., 2025) randomly masks 40% of the gradients during full-parameter fine-tuning, while GMT (Li et al., 2025) selects gradients based on their importance, both resulting in 60% trainable parameters. APEX follows the HFT setting, keeping the trainable parameter ratio at 52%.

Training Data. We use the TÜLU V2 dataset (Ivison et al., 2023) for training. This dataset consists of a wide range of sources to enhance the model's instruction-following capabilities. Specifically, it includes data from FLAN v2 (Chung et al., 2022), Open Assistant (Köpf et al., 2023), ShareGPT[3], GPT4-Alpaca (Peng et al., 2023), Code-Alpaca (Chaudhary, 2023), LIMA (Zhou et al., 2023), WizardLM Evol Instruct (Xu et al., 2023), Open-Orca (Mukherjee et al., 2023), Hard-coded, and Science.

Experimental Details. Under the half-parameter fine-tuning setting, we follow Hui et al. (Hui et al., 2025) and set the learning rate to 2e-5 for LLaMA2-7B/13B and 5e-6 for LLaMA3.1-8B and Mistral-7B-v0.1, consistently across all methods. Under the parameter-efficient fine-tuning setting, we apply LoRA to all linear layers for LoRA-based baselines. We select the best learning rate per method from {1e-4,2e-4} for LLaMA2-family models following (Yang et al., 2024b); for LLaMA3.1-8B and Mistral-7B-v0.1, we tune the learning rate from {2e-5,3e-5} and {8e-6,1e-5,2e-5}, respectively. All models are fine-tuned on the dataset for 2 epochs with a batch size of 128. All experiments are conducted on 8 Nvidia H100 GPUs (80GB each).

Evaluation. We follow the evaluation procedures established by Hui et al. (2025). Specifically, we report 0-shot accuracy on MMLU (Hendrycks et al., 2021) to assess the model's understanding of factual knowledge. For GSM8K (Cobbe et al., 2021) and BBH (Suzgun et al., 2023), we report 8-shot and 3-shot exact match, respectively, to evaluate reasoning abilities. We use F1 score on TyDiQA (Clark et al., 2020) to measure multilingual capabilities, and the multi-true score (MC2) on TruthfulQA (Lin et al., 2022) to evaluate the ability to generate truthful information. For code generation, we report pass@10 performance on HumanEval (Chen et al., 2021).

[3]https://sharegpt.com/

A.3 Continued Pre-Training

In our pre-training experiments, we use the TinyLLaMA-v1-1.1B model (Zhang et al., 2024) as the source model. After evaluating the advantages of APEX operators on 30M tokens, we proceed with pre-training on 10B tokens. For direct comparison, we also pre-train the original model on 10B tokens as a baseline. All training is conducted with full-parameter training.

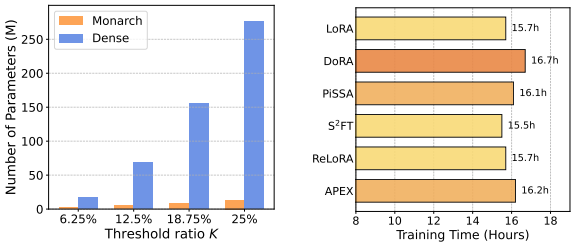
Baseline. We use continued pre-training of TinyLLaMA v1 1.1B as our primary baseline (Vanilla CPT). The original work provides a robust training framework, dynamic batch gradient pre-training strategies, and thorough evaluation protocols, resulting in convincing and competitive performance.

Training Data. We use the RedPajama dataset (Weber et al., 2024) for both APEX training and conventional training. This dataset mirrors the LLaMA1 pre-training data, covering seven domains: CommonCrawl, C4, Github, Wikipedia, Books, ArXiv, and Stack-Exchange. Consistent with LLaMA's settings, we use a sequence length of 4096. Our data splits include a 2M tokens validation set, a 4B tokens training set, and a 10B tokens continual pre-training set.

Experimental Details. Our implementation is based on the Composer package (Team, 2021). To ensure rigorous comparison, continued pre-training of TinyLLaMA v1 1.1B is carried out using identical data and experimental settings. We use a sequence length of 4096, a global train batch size of 64. For continued pre-training from the initial checkpoint (Stage 0), both APEX and the baseline adopt a learning rate of 1e-4. In training, we apply learning rate decay: 4e-5 in Stage 1, 1e-5 in Stage 2, 5e-6 in Stage 3, and 1e-6 in Stage 4. At each stage, we select advantageous and disadvantageous dimensions based on evaluation statistics from the previous stage. Each experiment is run

Method	Exact Match
Pre-trained	12.0
Mask Min 10%	12.0 (-0.0)
Mask Top 10%	0.0 (-12.0)
Mask Random 10%	11.5 (-0.5)

Table 4: Comparison of the effects of masking advantageous and disadvantageous parameters on LLaMA2-7B for GSM8K. We mask 10% of the parameters under each of the three settings.



(a) Number of parameters introduced by the expansion operator.

(b) Taining time of different methods on LLaMA2-7B.

Figure 6: Efficiency analysis of APEX.

with different random seeds, and the results are averaged.

Evaluation. We follow the procedures established by Pythia (Biderman et al., 2023) and LLaMA2 (Touvron et al., 2023b). Specifically, we employ the lm-evaluation-harness (Gao et al., 2024) to evaluate our models. Specifically, we report zero-shot accuracy on ARC-Easy (Clark et al., 2018), LAMBADA (Paperno et al., 2016), LogiQA (Liu et al., 2020), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), and WinoGrande (Sakaguchi et al., 2020). Additionally, we report the accuracy on tasks used by the OpenLLM Leaderboard (Fourrier et al., 2024), including 10-shot HellaSwag (Zellers et al., 2019), 25-shot ARC Challenge (Clark et al., 2018), and 5-shot MMLU (Hendrycks et al., 2021). To further assess the factual knowledge encoded in the model, we also report the accuracy on 32-shot BoolQ (Clark et al., 2019).

We adhere to the license terms and conditions for all codes, models, and datasets used in this work. Specifically, we ensure that our use and distribution of these artifacts comply with their respective open-source licenses, and that our usage is consistent with their intended purpose, which is limited to research contexts where specified. For any artifacts we create, we clearly specify that they are intended for research use only, and verify that this intended use is compatible with the original access conditions, particularly for any datasets derived from sources restricted to research applications.

B Extended Experiments

B.1 Extened Observations

To further investigate whether parameter advantage can be reflected by activation magnitudes, we compare the effects of masking parameters with high activations to those with low activations. As shown in Table 4, masking high-activation 10% parameters results in a significant decline in model performance, rendering the model nearly unusable. In contrast, masking low-activation 10% parameters has little to no effect on performance. These results suggest that activation magnitudes can effectively distinguish advantageous parameters from disadvantageous ones, with the former playing a critical role in overall model performance.

B.2 Experiments on Additional Models

In Table 5 and Table 6, we report the results of instruction tuning experiments using LLaMA3.1-8B, LLaMA2-13B, and Mistral-7B-v0.1 as base models. As shown, similar to the performance trends observed with fine-tuning on LLaMA2-7B in Table 1, APEX outperforms all parameter-efficient fine-tuning baselines in terms of average performance and achieves leading results on the majority of evaluation tasks. Notably, APEX attains performance comparable to full-parameter fine-tuning while only requiring around 2% of the trainable parameters. These results demonstrate that advantage expansion is effective across different base models and can consistently improve training performance.

B.3 Efficiency Analysis

Efficiency Analysis of the Expansion Operator. In Figure 6a, we compare the trend of parameters introduced by the expansion operator when implemented with Dense matrices versus Monarch matrices under varying threshold ratios (K). As K increases from 6.25% to 25%, the parameter count for the Dense implementation rises sharply, from 17.2M to 275M. In contrast, the Monarch-based expansion operator consistently maintains a much lower parameter scale, significantly below that of the Dense approach. This indicates that

Method	#Param.	MMLU 0-shot, EM	GSM8K 8-shot CoT, EM	BBH 3-shot CoT, EM	TyDiQA 1-shot, F1	TruthfulQA 0-shot, MC2	HumanEval 0-shot, Pass@10	Avg.
Full-FT	100%	62.4	65.0	66.3	71.3	44.0	73.0	63.7
LoRA	2.1%	61.0	64.0	63.4	70.9	45.8	67.8	62.2
DoRA	2.1%	61.7	62.0	65.5	68.8	45.6	71.0	62.4
PiSSA	2.1%	61.3	63.0	64.1	70.4	46.2	72.5	62.9
ReLoRA	2.1%	61.0	65.0	60.7	70.5	45.3	68.0	61.8
S^2FT	2.1%	63.1	61.0	64.4	70.9	43.8	71.3	62.4
APEX	2.1%	62.4	64.5	68.7	71.8	44.4	73.1	**64.2**

Table 5: Results for LLaMA3.1-8B. #Param. denotes trainable parameters.

Method	#Param.	MMLU 0-shot, EM	GSM8K 8-shot CoT, EM	BBH 3-shot CoT, EM	TyDiQA 1-shot, F1	TruthfulQA 0-shot, MC2	HumanEval 0-shot, Pass@10	Avg.
Full-FT	100%	53.4	41.5	47.8	59.0	44.4	41.5	47.9
LoRA	1.9%	52.3	35.5	46.4	54.7	41.3	36.5	44.5
DoRA	1.9%	51.5	40.5	44.4	56.5	43.8	35.2	45.3
PiSSA	1.9%	53.7	32.5	50.8	58.1	44.6	34.5	45.7
ReLoRA	1.9%	50.9	37.5	47.4	52.8	43.5	38.6	45.1
S^2FT	1.9%	54.2	38.5	50.3	57.2	42.8	39.1	47.0
APEX	1.9%	54.3	41.0	50.9	58.1	44.4	40.9	**48.3**

Table 6: Results for LLaMA2-13B. #Param. denotes trainable parameters.

Monarch decomposition can effectively compress the parameter overhead of the expansion operator, enabling APEX to achieve parameter-efficient expansion with minimal additional parameters. Consequently, APEX supports the expansion of more advantageous parameters without compromising the lightweight nature of PEFT.

Training Efficiency Analysis. Figure 6b presents a comparison of end-to-end training time across different methods under identical model and data settings. The results show that APEX requires approximately 16.2 hours of training, only about 0.5 hours longer than LoRA, on par with PiSSA, and faster than DoRA. Thanks to the integration of advantage parameter evaluation in the forward pass and the lightweight design of the expansion operator, the evaluation and expansion steps introduced by APEX do not noticeably slow down training throughput. Overall, APEX delivers stable performance improvements while keeping both the additional parameter and actual training time overheads at the same order of magnitude as mainstream PEFT methods, demonstrating its practical feasibility and efficiency advantage.

B.4 Extended Method Analysis

Comparison of Different Assessment Methods. We compare the performance of four assessment methods: no assessment (i.e., *Act. Rand.*), using the average activations (i.e., *Act. Avg.*), our proposed metric based on relative activation rankings (i.e., *Act. Rank*), and gradient-based ranking (i.e., *Grad. Rank*). As shown in Table 8, employing the average activations as an assessment method improved the average performance on instruction tuning tasks by 0.5 points compared to no assessment, demonstrating the effectiveness of activations in selecting advantageous parameters. Furthermore, when incorporating our proposed relative activation ranking as the assessment metric, the performance improved further. This indicates that the relative ranking of activations can effectively mitigate the fluctuations caused by data variability, leading to more accurate assessment results. Crucially, *Act. Rank* achieves performance comparable to *Grad. Rank* while avoiding the substantial overhead of gradient-based assessment, which requires backpropagation and storing gradients for all parameters. In contrast, *Act. Rank* only records forward-pass activations, providing a lightweight assessment for advantageous parameters. We note that these four methods

Method	#Param.	MMLU 0-shot, EM	GSM8K 8-shot CoT, EM	BBH 3-shot CoT, EM	TyDiQA 1-shot, F1	TruthfulQA 0-shot, MC2	HumanEval 0-shot, Pass@10	Avg.
Full-FT	100%	58.8	54.5	53.3	61.5	47.2	59.1	55.7
LoRA	2.3%	56.3	47.5	54.5	58.2	45.8	53.2	52.6
DoRA	2.3%	59.7	53.5	57.2	59.7	46.2	57.3	55.6
PiSSA	2.3%	59.2	48.0	59.2	58.8	46.6	57.3	54.9
ReLoRA	2.3%	59.2	51.0	55.6	59.7	46.2	56.7	54.7
S^2FT	2.3%	60.1	54.5	58.9	58.8	45.2	58.6	56.0
APEX	2.3%	59.8	54.0	59.0	61.9	45.9	62.7	**57.2**

Table 7: Results for Mistral-7B-v0.1. #Param. denotes trainable parameters.

Method	Average
HFT-52%	42.9
APEX-52%	
w. Act. Rand.	43.2 (+0.3)
w. Act. Avg.	43.7 (+0.8)
w. Act. Rank.	44.0 (+1.1)
w. Grad. Rank.	44.1 (+1.2)

Table 8: Analysis of assessment method in APEX.

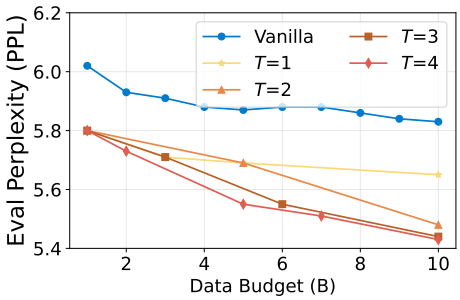


Figure 7: Perplexity curves during continued pre-training with varying stage counts T.

span a spectrum from random selection to gradient-based ranking, and our activation-based ranking already matches the most compute-intensive alternative. Other cost-efficient metrics such as weight magnitude fall between these extremes in terms of signal quality and are thus unlikely to yield further improvement.

Effect of the number of stages T for CPT settings. Unlike instruction tuning, which typically involves multi-epoch training, continued pre-training often follows a single-pass paradigm over a massive corpus. Figure 7 shows the eval PPL results of APEX with 1, 2, 3, and 4 stages under a 10B continued pre-training budget. First, APEX consistently and significantly outperforms traditional methods across all stage settings, as evidenced by a more stable and pronounced decrease in PPL. Second, increasing the number of stages can further enhance the performance of APEX. For example, $T = 3$, which has more stages than $T = 2$ under the same training data, is able to better leverage the benefits of APEX in each stage and thus achieves a lower eval PPL. Finally, the performance improvement from adding more APEX stages has its limits; as the number of stages increases, the difference between the model's advantages and disadvantages narrows, leading to diminishing returns. As a result, the gap in final evaluation performance between $T = 3$ and $T = 4$ is relatively small. Based on this trade-off between performance gains and training complexity, we adopt $T = 3$ as the default setting for all subsequent continued pre-training experiments.

Discussion of the Relationship Between Activation Distributions and Model Performance. In Section 2, we observed that strong models exhibit more uniform activation distributions (lower standard deviations). This raises a critical question: *Can we improve performance by explicitly enforcing this statistical uniformity?* To answer this, we contrast two fundamentally different paradigms.

Approach 1: Activation Regularization (Act-Regu). A natural hypothesis is that directly enforcing uniform activation distributions through regularization would improve performance. To test this, we implement Act-Regu, which penalizes the standard deviation of activations during training. However, as shown in Table 9, this approach fails catastrophically: average performance drops from 42.6 to 29.7, with TruthfulQA reaching 0% accuracy. This failure reveals a crucial insight: *the uniform distribution in strong models is an indicator of effective parameter utilization, not a direct*

Method	#Param.	MMLU 0-shot, EM	GSM8K 8-shot CoT, EM	BBH 3-shot CoT, EM	TyDiQA 1-shot, F1	TruthfulQA 0-shot, MC2	HumanEval 0-shot, Pass@10	Avg.
Pre-trained†	-	41.6	12.0	39.9	48.4	38.5	26.2	34.4
S^2FT	2.4%	47.6	25.0	45.5	52.8	43.7	33.9	41.4
Act-Regu	2.4%	23.8	30.5	41.2	51.8	0.0	31.1	29.7
APEX	2.4%	48.1	28.5	45.7	54.5	45.5	33.2	42.6

Table 9: Results of using activation regularization (Act-Regu) for LLaMA2-7B. #Param. denotes trainable parameters. All results with † are taken from HFT (Hui et al., 2025).

	MHA Top-K	MHA Min-K	FFN Top-K	FFN Min-K
Layer 5	0.75	0.60	0.78	0.63
Layer 15	0.72	0.47	0.81	0.59
Layer 25	0.69	0.54	0.87	0.66
Overall Avg.	0.72	0.54	0.82	0.63

Table 10: Jaccard similarity of Top-K and Min-K parameter sets between consecutive APEX stages on LLaMA2-7B ($T = 2$).

lever for improving it. Forcibly suppressing high activations destroys the informative representations that these parameters have learned.

Approach 2: APEX. Rather than suppressing advantageous parameters, APEX leverages them to guide the optimization of disadvantageous ones. As analyzed in App. C (Lemma 1), the expansion operator introduces a preconditioned correction term $(\mathbf{W}^P\mathbf{W}^{P\top})\hat{\mathbf{G}}$ that projects gradient updates onto the subspace spanned by advantageous parameters. This subspace, having been shaped by reliable gradient signals during prior training, provides a higher signal-to-noise ratio for optimization. Consequently, disadvantageous parameters receive more stable gradient guidance without overwriting the learned representations in advantageous ones.

Why Expansion Succeeds While Regularization Fails. The contrast between these approaches highlights a fundamental distinction: Act-Regu treats activation uniformity as a *target*, destroying useful information to achieve it. APEX instead treats advantageous parameters as a *resource*, using their learned structure to improve optimization in underutilized subspaces. The flatter activation distribution observed after APEX training (Figure 5) emerges naturally as a byproduct of broader parameter utilization, not as an imposed constraint. This validates our hypothesis that the path to enhanced performance lies in *expanding* advantageous capacity rather than *suppressing* it.

B.5 Stability of Stage-wise Parameter Re-selection

Since APEX re-selects advantageous and disadvantageous parameter sets at each training stage, a natural question is whether these sets remain stable across stages. We conduct a analysis on the LLaMA2-7B instruction tuning setting with $T = 2$ stages. We extract the Top-K and Min-K parameter sets for each stage and compute the Jaccard similarity $J = |S_t \cap S_{t+1}|/|S_t \cup S_{t+1}|$ between consecutive stages. Table 10 reports the results for representative layers.

As shown in Table 10, the Top-K (advantageous) sets maintain high consistency across consecutive stages, indicating that core advantageous parameters do not fluctuate drastically due to stage transitions and that the assessment mechanism is reliable. Meanwhile, the Jaccard similarity of the Min-K (disadvantageous) sets is systematically lower than that of Top-K, with gaps of 0.18 in MHA and 0.19 in FFN. This asymmetry suggests that some parameters originally classified as disadvantageous exit the Min-K set after expansion training, which is precisely the intended effect of APEX. We also note that FFN modules exhibit higher Jaccard similarity than MHA modules across all layers, which is consistent with Figure 1, where FFN activation distributions are sharper, implying clearer Top/Min boundaries and naturally more stable cross-stage selection.

	Full-FT	APEX
Layer 5	0.001767	0.002452
Layer 15	0.001347	0.001063
Layer 25	0.001525	0.002292
Overall Avg.	0.001546	0.001936

Table 11: Average pairwise cosine similarity between attention head output projections on LLaMA2-7B.

B.6 Representational Diversity After Expansion

To verify that the expansion mechanism preserves representational diversity across attention heads, we extract the fine-tuned LLaMA2-7B model weights and compute the average pairwise cosine similarity between output states of different attention head weight matrices within the same layer. Specifically, for each layer, we compute $\frac{1}{H(H-1)} \sum_{i \neq j} \cos(\mathbf{W}_O^{[i,:]}, \mathbf{W}_O^{[j,:]})$, where H is the number of heads.

As shown in Table 11, cosine similarity values remain at the very low 10^{-3} level for both Full-FT and APEX, indicating that the learned representations of different heads remain nearly orthogonal. The overall average similarity after APEX training is comparable to that of Full-FT, providing empirical evidence that APEX preserves representational diversity and does not induce representation collapse.

This result aligns with the design principle described in Section 3.2. Since the expansion operator (Monarch matrix) is zero-initialized, it functions as a structured preconditioner during gradient updates (as derived in Lemma 1). Rather than forcing disadvantageous parameters to become copies of advantageous ones, APEX guides them toward the advantageous subspace while retaining their capacity to learn independent, orthogonal patterns via standard gradient updates. Furthermore, the multi-stage re-selection mechanism ensures these expansion directions continually adapt across different training stages.

B.7 Robustness Across Random Seeds and Calibration Data

All experimental results reported in the main paper are averaged over three independent runs with different random seeds (0, 42, 4242). Importantly, in our implementation, the data subset $\mathcal{D}_{\text{subset}}$ used for pre-assessment in the first stage is determined by the random seed and sampled from the training set. This means that the three runs use completely different first-stage calibration data.

Method	Seed 0	Seed 42	Seed 4242	AVG
Full-FT	42.8	43.2	43.1	43.1
APEX-52%	44.3	44.0	43.6	44.0

Table 12: Per-seed average performance for instruction tuning on LLaMA2-7B.

Method	MML.	GSM.	BBH	TyD.	Tru.	Hum.	Avg.
Full-FT	25.7	3.5	23.6	24.2	38.5	7.8	20.6
APEX	25.8	3.5	25.5	24.5	38.6	9.2	21.2

Table 13: Instruction tuning results for ShearedLLaMA-1.3B.

Table 12 reports the per-seed average performance for instruction tuning on LLaMA2-7B. The per-seed results reveal that the improvement of APEX over baselines is consistent, as even the worst single run of APEX in instruction tuning (43.6) exceeds the best Full-FT run (43.2). Moreover, despite using entirely different initial calibration subsets across the three runs, the final performance remains highly stable. This robustness stems from two aspects of our design. First, the calibration subset is randomly sampled from a diverse, multi-domain training corpus, so the identified advantageous parameters reflect the model's fundamental representational strengths rather than domain-specific biases. Second, APEX's multi-stage mechanism acts as a self-correcting process: beyond the first stage, advantage assessment is based on activation statistics accumulated during training on the full data distribution, and the influence of the initial calibration gradually diminishes as training progresses.

B.8 Instruction Tuning on Small Models

To further validate that APEX helps small models make better use of their parameters, we conduct experiments on ShearedLLaMA-1.3B (Xia et al., 2024). As shown in Table 13, APEX achieves consistent improvements over Full-FT. Combined with the continued pre-training results on TinyLLaMA-1.1B (Table 2), these experiments confirm that APEX is effective on models ranging from 1.1B to 13B.

C Theoretical Perspective: Gradient Signal Alignment

We analyze why expanding *advantageous* parameters can improve optimization from a *gradient signal alignment* perspective. The key point in our actual training setup is that the disadvantageous block $\mathbf{W}_N$ is still updated, while the expansion matrix $\mathbf{M}$ is *zero-initialized*. Consequently, APEX does *not* forcibly fuse $\mathbf{W}_P$ into $\mathbf{W}_N$; instead, it provides an additional, adaptive correction term that becomes active only when the residual gradient has a non-trivial projection onto the advantageous feature span.

C.1 Setup

Consider a single linear layer with weights $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ and input $\mathbf{x} \in \mathbb{R}^{1 \times d_{\text{in}}}$. In our proposed method, the columns of $\mathbf{W}$ correspond to computational units (e.g., neurons in FFN or heads in Attention). We partition these columns into two sets based on their output activation magnitudes assessed on the dataset. Let $\mathcal{I}_P$ and $\mathcal{I}_N$ denote the index sets for advantageous and disadvantageous parameters, respectively. We conceptually partition the weight matrix as:

$$\mathbf{W} = \big[\mathbf{W}_P,\ \mathbf{W}_N\big], \tag{17}$$

where $\mathbf{W}_P \in \mathbb{R}^{d_{\text{in}} \times k}$ corresponds to the subset of parameters generating high-magnitude activations (advantageous), and $\mathbf{W}_N \in \mathbb{R}^{d_{\text{in}} \times (d_{\text{out}} - k)}$ corresponds to those with low-magnitude activations (disadvantageous). Accordingly, the layer output can be formulated as $\mathbf{y} = \mathrm{Concat}(\mathbf{y}_P, \mathbf{y}_N)$, where $\mathbf{y}_P = \mathbf{x}\mathbf{W}_P$ and $\mathbf{y}_N = \mathbf{x}\mathbf{W}_N$.

Gradient Signal Quality. To analyze the optimization dynamics, we model the stochastic gradient $\widehat{\mathbf{G}}$ (computed on a mini-batch) as the true population gradient $\mathbf{G}_\star$ perturbed by zero-mean noise $\mathbf{\Xi}$:

$$\widehat{\mathbf{G}} = \mathbf{G}_\star + \mathbf{\Xi}, \quad \text{where } \mathbb{E}[\mathbf{\Xi}] = \mathbf{0}. \tag{18}$$

We define the Signal-to-Noise Ratio (SNR) as

$$\mathrm{SNR}(\widehat{\mathbf{G}}) \ := \ \frac{\|\mathbf{G}_\star\|_F^2}{\mathbb{E}[\|\mathbf{\Xi}\|_F^2]}. \tag{19}$$

Assumption 1 (Activation-selected subspace has higher SNR). *We assume that the column space of $\mathbf{W}_P$ selected by high activations captures a larger fraction of the useful gradient signal or yields more stable gradient estimates across mini-batches.*

Formally, let $\mathbf{P}$ denote the orthogonal projector onto $\mathrm{col}(\mathbf{W}_P)$:

$$\mathbf{P} \ := \ \mathbf{W}_P(\mathbf{W}_P^\top \mathbf{W}_P)^\dagger \mathbf{W}_P^\top, \tag{20}$$

where $(\cdot)^\dagger$ is the pseudoinverse. We assume that the gradient component aligned with $\mathrm{col}(\mathbf{W}_P)$ has a higher effective SNR than the orthogonal component:

$$\mathrm{SNR}(\mathbf{P}\widehat{\mathbf{G}}) \ \geq \ \alpha \cdot \mathrm{SNR}\big((\mathbf{I} - \mathbf{P})\widehat{\mathbf{G}}\big), \quad \text{for some } \alpha > 1. \tag{21}$$

Here $\alpha > 1$ quantifies the multiplicative SNR advantage of the gradient component aligned with $\mathrm{col}(\mathbf{W}_P)$ over its orthogonal component. This assumption is intentionally coarse and does not claim that every low-activation unit must have low-SNR gradients.

Justification via Empirical Observations. Our masking experiments show that removing low-activation units ($\mathbf{W}_N$) causes negligible performance degradation, suggesting that these units are weakly coupled to the current function implemented by the layer. In such a regime, updates that rely on directions orthogonal to the active feature subspace may be inefficient on average, whereas reusing the already-active feature subspace $\mathrm{col}(\mathbf{W}_P)$ provides a structured way to express corrections. This motivates us to adopt Assumption (21) as a coarse, subspace-level abstraction: the population gradient contains a more meaningful component aligned with $\mathrm{col}(\mathbf{W}_P)$ than with its orthogonal complement. Importantly, APEX does not force fusion: $\mathbf{M}_0 = \mathbf{0}$ preserves the initial function, and the $\mathbf{M}$-path is only driven by the aligned component through $\nabla_{\mathbf{M}}\mathcal{L} = \mathbf{W}_P^\top \widehat{\mathbf{G}}$.

C.2 Baseline Gradients and the Optimization Dilemma

Consider optimizing the disadvantageous block $\mathbf{W}_N$ under standard SGD:

$$\mathbf{W}_N \leftarrow \mathbf{W}_N - \eta\,(\mathbf{G}_{N,\star} + \mathbf{\Xi}_N). \tag{22}$$

Under Assumption (21), gradient components that are *not* aligned with $\mathrm{col}(\mathbf{W}_P)$ tend to be less reliable on average, so naive updates may spend many steps in variance-dominated directions. This motivates *gradient guidance*: leveraging the activation-selected subspace $\mathrm{col}(\mathbf{W}_P)$ to supply a structured correction path while still allowing $\mathbf{W}_N$ to update directly when its gradients are informative.

C.3 APEX Reparameterization with Concurrent Updates

APEX reparameterizes the disadvantageous block as

$$\begin{aligned} \mathbf{W}'_N &:= \mathbf{W}_N + \mathbf{W}_P\mathbf{M}, \\ &\text{where } \mathbf{M} \in \mathbb{R}^{k\times(d_{\text{out}}-k)},\ \mathbf{M}_0 = \mathbf{0}. \end{aligned} \tag{23}$$

Consequently, the output of the disadvantageous units becomes:$\mathbf{y}'_N = \mathbf{x}\mathbf{W}'_N = \mathbf{x}\mathbf{W}_N + \mathbf{x}\mathbf{W}_P\mathbf{M}$. Importantly, in our training, $\mathbf{W}_N$ **is still updated** together with $\mathbf{M}$ (and typically $\mathbf{W}_P$ as well).

Let $\mathbf{G} := \nabla_{\mathbf{W}'_N}\mathcal{L} \in \mathbb{R}^{d_{\text{in}}\times(d_{\text{out}}-k)}$ denote the gradient w.r.t. the *effective* block $\mathbf{W}'_N$. By the chain rule,

$$\nabla_{\mathbf{W}_N}\mathcal{L} = \mathbf{G}, \qquad \nabla_{\mathbf{M}}\mathcal{L} = \mathbf{W}_P^\top\mathbf{G}. \tag{24}$$

Lemma 1. (Effective update equals vanilla step plus preconditioned correction.) *Consider SGD updates with learning rates $\eta_N, \eta_M > 0$:*

$$\begin{aligned} \mathbf{W}_N &\leftarrow \mathbf{W}_N - \eta_N\,\widehat{\mathbf{G}}, \\ \mathbf{M} &\leftarrow \mathbf{M} - \eta_M\,\mathbf{W}_P^\top\widehat{\mathbf{G}}. \end{aligned} \tag{25}$$

Ignoring higher-order terms due to simultaneous changes in $\mathbf{W}_P$, the induced update on $\mathbf{W}'_N$ satisfies:

$$\Delta\mathbf{W}'_N \;\approx\; -\eta_N\,\widehat{\mathbf{G}} \;-\; \eta_M\,(\mathbf{W}_P\mathbf{W}_P^\top)\widehat{\mathbf{G}}. \tag{26}$$

Proof of Lemma 1. Recall the reparameterization $\mathbf{W}'_N = \mathbf{W}_N + \mathbf{W}_P\mathbf{M}$. For one SGD step, write the increments as

$$\begin{aligned} \Delta\mathbf{W}_N &:= \mathbf{W}_N^+ - \mathbf{W}_N, \\ \Delta\mathbf{M} &:= \mathbf{M}^+ - \mathbf{M}, \\ \Delta\mathbf{W}_P &:= \mathbf{W}_P^+ - \mathbf{W}_P. \end{aligned} \tag{27}$$

Then by direct expansion,

$$\begin{aligned} \Delta\mathbf{W}'_N &= \mathbf{W}_N^+ + \mathbf{W}_P^+\mathbf{M}^+ - (\mathbf{W}_N + \mathbf{W}_P\mathbf{M}) \\ &= \Delta\mathbf{W}_N + (\mathbf{W}_P + \Delta\mathbf{W}_P)(\mathbf{M} + \Delta\mathbf{M}) \\ &\quad - \mathbf{W}_P\mathbf{M} \\ &= \Delta\mathbf{W}_N + \mathbf{W}_P\Delta\mathbf{M} + (\Delta\mathbf{W}_P)\mathbf{M} \\ &\quad + (\Delta\mathbf{W}_P)(\Delta\mathbf{M}). \end{aligned}$$

Using the chain rule $\nabla_{\mathbf{W}_N}\mathcal{L} = \mathbf{G}$ and $\nabla_{\mathbf{M}}\mathcal{L} = \mathbf{W}_P^\top\mathbf{G}$, the SGD updates

$$\mathbf{W}_N^+ = \mathbf{W}_N - \eta_N\widehat{\mathbf{G}},\ \ \mathbf{M}^+ = \mathbf{M} - \eta_M\mathbf{W}_P^\top\widehat{\mathbf{G}} \tag{28}$$

can imply:

$$\Delta\mathbf{W}_N = -\eta_N\widehat{\mathbf{G}},\ \ \Delta\mathbf{M} = -\eta_M\mathbf{W}_P^\top\widehat{\mathbf{G}}. \tag{29}$$

Substituting into (28) gives

$$\begin{aligned} \Delta\mathbf{W}'_N &= -\eta_N\widehat{\mathbf{G}} - \eta_M(\mathbf{W}_P\mathbf{W}_P^\top)\widehat{\mathbf{G}} \\ &\quad + (\Delta\mathbf{W}_P)\mathbf{M} + (\Delta\mathbf{W}_P)(\Delta\mathbf{M}). \end{aligned} \tag{30}$$

At initialization $\mathbf{M}_0 = \mathbf{0}$, the term $(\Delta\mathbf{W}_P)\mathbf{M}$ is exactly zero on the first step. More generally during training, if $\|\Delta\mathbf{W}_P\| = O(\eta_P)$ and $\|\Delta\mathbf{M}\| = O(\eta_M)$, then $(\Delta\mathbf{W}_P)(\Delta\mathbf{M}) = O(\eta_P\eta_M)$ is a higher-order term. Neglecting these higher-order contributions yields the first-order approximation

$$\Delta\mathbf{W}'_N \approx -\eta_N\widehat{\mathbf{G}} - \eta_M(\mathbf{W}_P\mathbf{W}_P^\top)\widehat{\mathbf{G}}, \tag{31}$$

Interpretation. Equation (26) shows that APEX does **not** replace the vanilla update of $\mathbf{W}_N$; it adds an extra correction term that left-multiplies the gradient by $\mathbf{W}_P\mathbf{W}_P^\top$, i.e., a *subspace-aligned preconditioner*. When $\mathbf{W}_P$ is reasonably conditioned, one may view $\mathbf{W}_P\mathbf{W}_P^\top$ as a scaled projector toward $\mathrm{col}(\mathbf{W}_P)$, steering part of the update to lie in the span of already-learned strong features.

C.4 Why Zero-Initialized $\mathbf{M}$ Does Not Force Fusion: Adaptive Activation

Since $\mathbf{M}_0 = \mathbf{0}$, the initial function is unchanged. Moreover, the update magnitude of $\mathbf{M}$ is governed by the projected gradient:

$$\|\Delta\mathbf{M}\|_F = \eta_M\|\mathbf{W}_P^\top\widehat{\mathbf{G}}\|_F. \tag{32}$$

Thus, if $\mathbf{W}_N$ is already sufficient such that the residual gradient $\widehat{\mathbf{G}}$ becomes small, or if $\widehat{\mathbf{G}}$ is (approximately) orthogonal to $\mathrm{col}(\mathbf{W}_P)$ so that $\mathbf{W}_P^\top\widehat{\mathbf{G}} \approx \mathbf{0}$, then $\mathbf{M}$ remains near zero. In this sense, APEX behaves as a *data-adaptive correction path*: it activates only when there exists a residual component that can be efficiently reduced through the advantageous feature span.

C.5 Variance Reduction and SNR Improvement of the APEX Correction

We now formalize how the additional correction term in Eq. (26) redirects part of the update from lower-SNR components to higher-SNR components, using a bias–variance decomposition. Following Eq. (18), write the stochastic gradient as $\widehat{\mathbf{G}} = \mathbf{G}_\star + \mathbf{\Xi}$. Let $\mathbf{P}$ be the orthogonal projector onto $\mathrm{col}(\mathbf{W}_P)$ as defined in Eq. (20).

Lemma 2. (Orthogonal Projection Does Not Increase Gradient Noise Energy.) *For any zero-mean noise matrix* $\mathbf{\Xi}$*,*

$$\mathbb{E}\big[\|\mathbf{P}\mathbf{\Xi}\|_F^2\big] \;\le\; \mathbb{E}\big[\|\mathbf{\Xi}\|_F^2\big]. \tag{33}$$

Define the projected signal and noise scales:

$$S_{\|} := \|\mathbf{P}\mathbf{G}_\star\|_F^2, \;\; N_{\|} := \mathbb{E}\big[\|\mathbf{P}\mathbf{\Xi}\|_F^2\big], \tag{34}$$

and the corresponding SNR:

$$\mathrm{SNR}_{\|} \;:=\; \frac{S_{\|}}{N_{\|}}. \tag{35}$$

Lemma 2 implies $N_{\|} \le \mathbb{E}\|\mathbf{\Xi}\|_F^2$, i.e., the *orthogonally projected* path which captures pure subspace alignment to $\mathrm{col}(\mathbf{W}_P)$ is no more noisy in energy than the original noise. In APEX, the actual correction uses $\mathbf{W}_P\mathbf{W}_P^\top$, which additionally applies magnitude scaling within $\mathrm{col}(\mathbf{W}_P)$; this scaling is controlled by η_M. Whether it helps depends on how much signal is retained in $\mathbf{P}\mathbf{G}_\star$, which is precisely what Eq.(32) adaptively measures during training.

Proof of Lemma 2. Let $\mathbf{P}$ be an orthogonal projector onto $\mathrm{col}(\mathbf{W}_P)$, i.e., $\mathbf{P}^\top = \mathbf{P}$ and $\mathbf{P}^2 = \mathbf{P}$, hence $0 \preceq \mathbf{P} \preceq \mathbf{I}$. For any random matrix $\mathbf{\Xi}$ with finite second moment,

$$\begin{aligned}
\mathbb{E}\big[\|\mathbf{P}\mathbf{\Xi}\|_F^2\big] &= \mathbb{E}\big[\mathrm{Tr}((\mathbf{P}\mathbf{\Xi})^\top(\mathbf{P}\mathbf{\Xi}))\big] \\
&= \mathbb{E}\big[\mathrm{Tr}(\mathbf{\Xi}^\top\mathbf{P}^\top\mathbf{P}\mathbf{\Xi})\big] \\
&= \mathbb{E}\big[\mathrm{Tr}(\mathbf{\Xi}^\top\mathbf{P}\mathbf{\Xi})\big] \\
&= \mathbb{E}\big[\mathrm{Tr}(\mathbf{\Xi}\mathbf{\Xi}^\top\mathbf{P})\big].
\end{aligned} \tag{36}$$

Since $\mathbf{\Xi}\mathbf{\Xi}^\top \succeq 0$ and $0 \preceq \mathbf{P} \preceq \mathbf{I}$, we have

$$\mathrm{Tr}(\mathbf{\Xi}\mathbf{\Xi}^\top\mathbf{P}) \le \mathrm{Tr}(\mathbf{\Xi}\mathbf{\Xi}^\top\mathbf{I}) = \mathrm{Tr}(\mathbf{\Xi}\mathbf{\Xi}^\top) = \|\mathbf{\Xi}\|_F^2. \tag{37}$$

Taking expectation on both sides and combining with Eq.(36) yields

$$\mathbb{E}\big[\|\mathbf{P}\mathbf{\Xi}\|_F^2\big] \le \mathbb{E}\big[\|\mathbf{\Xi}\|_F^2\big], \tag{38}$$

C.6 Summary of Analysis

With concurrent updates of $\mathbf{W}_N$ and zero-initialized $\mathbf{M}$, APEX is best viewed as an *auxiliary preconditioned correction path*:

- **No forced fusion:** $\mathbf{M}_0 = \mathbf{0}$ preserves the initial function, and the $\mathbf{M}$-path is activated only when the residual gradient has a non-trivial projection onto the advantageous span, i.e., when $\|\mathbf{W}_P^\top\widehat{\mathbf{G}}\|$ is non-negligible.
- **Preconditioned correction:** alongside the vanilla update on $\mathbf{W}_N$, the $\mathbf{M}$-path contributes an additional correction term proportional to $(\mathbf{W}_P\mathbf{W}_P^\top)\widehat{\mathbf{G}}$ (Lemma 1), steering part of the update toward directions spanned by $\mathbf{W}_P$.
- **Noise filtering:** the correction term maps updates into $\mathrm{col}(\mathbf{W}_P)$, i.e., it removes components in $\mathrm{col}(\mathbf{W}_P)^\perp$. Moreover, the *orthogonally projected* path $\mathbf{P}\widehat{\mathbf{G}}$ cannot increase the noise energy (Lemma 2); the additional magnitude scaling induced by $\mathbf{W}_P\mathbf{W}_P^\top$ is controlled by η_M.

Under the assumption Eq.(21), $\mathbf{W}_P$ is trained on signal-rich directions and tends to form a relatively well-conditioned subspace; APEX leverages this subspace to supply stable residual corrections while still allowing $\mathbf{W}_N$ to learn directly when it is informative.

D Broader Impact

The APEX method focuses on efficiency, demonstrating the ability to match the performance of models with more trainable parameters. This effectively reduces the computational cost required to reach target performance and mitigates the environmental impact of model training. Specifically, in instruction tuning, APEX surpassed the performance of full-parameter fine-tuning using only 52% of the trainable parameters. In continued pre-training, APEX achieved the same perplexity level as conventional training with only 33% of the pre-training data budget. Moreover, these improvements enable a wider range of research institutions to successfully train larger models within reasonable computational resources, thereby significantly reducing the cost of utilizing large base models.

E Algorithm of APEX

Here, we present the complete algorithm of the proposed method APEX in Algorithm 1.

Algorithm 1 Advantageous Parameter EXpansion Training (APEX)

Require:
Model M with trainable parameters θ, which includes $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_O, \mathbf{W}_U, \mathbf{W}_D, \mathbf{W}_G$ in each layer
Training dataset $\mathcal{D}$
Number of stages T
Proportions K
Ensure: Optimized model M with expanded advantageous parameters
Initialize $s_{\text{MHA}} \leftarrow 0$, $s_{\text{FFN}} \leftarrow 0$ ▷ Initialize advantage scores
Pre-Assessment Before Training:
for each batch in $\mathcal{D}_{\text{subset}}$ **do** ▷ Using a subset of training data
  for each head h in MHA **do** ▷ Calculate MHA scores
    $A^h_{\text{MHA}} \leftarrow$ FrobeniusNorm(activations)
    Update s^h_{MHA} with Top-K/Min-K counts
  end for
  for each channel c in FFN **do** ▷ Calculate FFN scores
    $A^c_{\text{FFN}} \leftarrow$ FrobeniusNorm(activations)
    Update s^c_{FFN} with Top-K/Min-K counts
  end for
end for
for stage $t = 0$ **to** $T - 1$ **do**
  Stage Initialization Phase:
  $d^P_{\text{MHA}} \leftarrow$ Top-$K(s_{\text{MHA}})$ ▷ Select parameter sets
  $d^N_{\text{MHA}} \leftarrow$ Min-$K(s_{\text{MHA}})$
  $d^P_{\text{FFN}} \leftarrow$ Top-$K(s_{\text{FFN}})$
  $d^N_{\text{FFN}} \leftarrow$ Min-$K(s_{\text{FFN}})$
  Initialize γ_{MHA} with Monarch matrices ($\mathbf{D}$ initialized randomly, $\mathbf{R} = 0$) ▷ Operator Initialization
  Initialize γ_{FFN} with Monarch matrices ($\mathbf{D}$ initialized randomly, $\mathbf{R} = 0$)

  Expansion Training Phase:
  for each batch in $\mathcal{D}$ **do** ▷ Using a subset of training data
    for each head h in MHA **do** ▷ Calculate MHA scores
      $A^h_{\text{MHA}} \leftarrow$ FrobeniusNorm(activations)
      Update s^h_{MHA} with Top-K/Min-K counts
    end for
    for each channel c in FFN **do** ▷ Calculate FFN scores
      $A^c_{\text{FFN}} \leftarrow$ FrobeniusNorm(activations)
      Update s^c_{FFN} with Top-K/Min-K counts
    end for
    loss $\leftarrow$ compute_loss(outputs) ▷ Backward Pass
    $\theta \leftarrow$ optimize(loss) ▷ Update trainable parameters
  end for

  Operator Fusion Phase:
  $\mathbf{W}_Q[:, d^N_{\text{MHA}}] \leftarrow \gamma_{\text{MHA}}(\mathbf{W}_Q[:, d^P_{\text{MHA}}], \mathbf{W}_Q[:, d^N_{\text{MHA}}])$
  $\mathbf{W}_K[:, d^N_{\text{MHA}}] \leftarrow \gamma_{\text{MHA}}(\mathbf{W}_K[:, d^P_{\text{MHA}}], \mathbf{W}_K[:, d^N_{\text{MHA}}])$
  $\mathbf{W}_V[:, d^N_{\text{MHA}}] \leftarrow \gamma_{\text{MHA}}(\mathbf{W}_V[:, d^P_{\text{MHA}}], \mathbf{W}_V[:, d^N_{\text{MHA}}])$
  $\mathbf{W}_O[d^N_{\text{MHA}}, :] \leftarrow \gamma_{\text{MHA}}(\mathbf{W}_O[d^P_{\text{MHA}}, :], \mathbf{W}_O[d^N_{\text{MHA}}, :])$
  $\mathbf{W}_U[:, d^N_{\text{FFN}}] \leftarrow \gamma_{\text{FFN}}(\mathbf{W}_U[:, d^P_{\text{FFN}}], \mathbf{W}_U[:, d^N_{\text{FFN}}])$
  $\mathbf{W}_G[:, d^N_{\text{FFN}}] \leftarrow \gamma_{\text{FFN}}(\mathbf{W}_G[:, d^P_{\text{FFN}}], \mathbf{W}_G[:, d^N_{\text{FFN}}])$
  $\mathbf{W}_D[d^N_{\text{FFN}}, :] \leftarrow \gamma_{\text{FFN}}(\mathbf{W}_D[d^P_{\text{FFN}}, :], \mathbf{W}_D[d^N_{\text{FFN}}, :])$
end for