DHAFT: DYNAMIC HIERARCHY-AWARE FINE-TUNING FOR EFFICIENT LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have achieved remarkable performance on complex reasoning tasks, but efficiently adapting them to diverse reasoning scenarios remains challenging. Soft prompt tuning provides a parameter-efficient adaptation strategy with a small number of trainable prompt vectors. However, existing approaches overlook the evolving nature of internal representations across model depth, making prompt effectiveness sensitive to different representation stages. To address this issue, we propose DHAFT, a hierarchy-aware soft prompt tuning framework that dynamically identifies hierarchical representation stages from internal representations and performs stage-specific prompt injection at representative layers. By aligning prompt placement with representation stages, DHAFT enables more effective prompt utilization and reduces potential cross-stage interference. Furthermore, we introduce PSMI, a prompt-stage misalignment index that quantifies the mismatch between prompt influence and representation stages. Extensive experiments across diverse LLM backbones and reasoning benchmarks demonstrate that DHAFT achieves competitive or superior performance compared with representative PEFT methods while maintaining minimal parameter overhead.

1 INTRODUCTION

In recent years, large language models (LLMs) have demonstrated remarkable capabilities in complex reasoning tasks (Brown et al., 2020; Wei et al., 2022; Kumar et al., 2025; Guo et al., 2025; Yue et al., 2025; Yang et al., 2026). However, adapting these models to diverse reasoning scenarios via full fine-tuning requires substantial computational resources and a separate set of parameters for each task. Parameter-efficient fine-tuning (PEFT) methods (Houlsby et al., 2019; Hu et al., 2022; Jung et al., 2026; Wang et al., 2026b) alleviate this adaptation cost by freezing the pretrained backbone and optimizing only a small number of additional parameters. Among them, soft prompt tuning (Lester et al., 2021; Li & Liang, 2021) has attracted increasing attention due to its minimal parameter overhead. Without modifying the backbone architecture, it provides task-relevant guidance to frozen models through a small number of learnable continuous prompt vectors.

To enhance soft prompt tuning, prior works have explored different strategies for injecting prompts into frozen language models (Liu et al., 2023; Bo et al., 2026). Existing methods mainly fall into three paradigms: input-level, layer-wise, and selective injection. Input-level methods, such as Prompt Tuning (Lester et al., 2021), P-Tuning (Liu et al., 2024b), and DePT (Shi & Lipani, 2024), prepend learnable prompt vectors to the input embeddings. Layer-wise methods, including Prefix-Tuning (Li & Liang, 2021), P-Tuning v2 (Liu et al., 2022b), IAPT (Zhu et al., 2024), CaPT (Liu et al., 2025), and PrefixMemory-Tuning (Wang et al., 2026a), introduce learnable prompts at every Transformer layer. Selective methods inject prompts at selected intermediate layers (Liu et al., 2022a) or learn injection positions through gating mechanisms (Zhu & Tan, 2023); once introduced, these prompts continue to influence subsequent layers.

Despite these advances, these paradigms generally share a common property: once injected, a prompt remains active through all subsequent layers, regardless of how internal representations evolve across depth. Recent studies have shown that LLM representations evolve throughout Transformer layers, with different layers exhibiting distinct representational characteristics (Lad et al., 2026; Skean et al., 2025; Gromov et al., 2025; Yan et al., 2026; Chuang et al., 2024; Huang et al., 2025a). As a result, a prompt injected at one depth continues to influence later layers whose representational states have

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

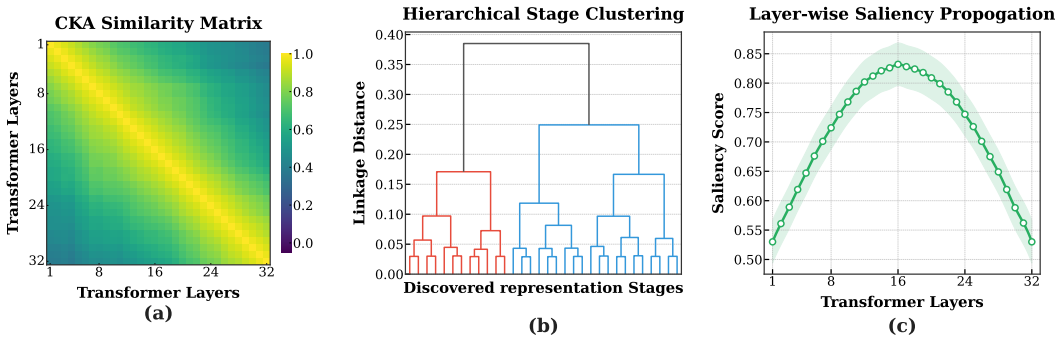


Figure 1: Representation stages on GSM8K with Llama-3-8B. (a) Cross-layer CKA similarity matrix. (b) Hierarchical clustering of layers into representation stages. (c) Layer-wise saliency under DHAFT with $G = 3$ stages.

changed, while the prompt guidance may not match their current representational state. Recently, DPC (Fan et al., 2025) identifies instances where later reasoning steps over-rely on prompts during inference and suppresses key prompt tokens accordingly. However, it regulates prompt effects after they occur, leaving open how prompt guidance should be allocated as representations evolve. This motivates a natural question: *Can prompt guidance be allocated according to the representation stages of the model, rather than fixed once at injection?*

To investigate this question, we first analyze the representation structure across Transformer layers and then examine how prompt influence varies across depth. We measure cross-layer representation similarity using Centered Kernel Alignment (CKA) (Kornblith et al., 2019), which is invariant to orthogonal transformations and isotropic scaling and thus compares representation spaces rather than raw coordinates. As shown in Figure 1(a), the similarity matrix exhibits a block structure, and hierarchical clustering in Figure 1(b) groups layers with similar representations into hierarchical representation stages. Using gradient-based prompt saliency, we then trace across depth the influence of prompts injected at the representative layers of these stages. Figure 1(c) shows that prompt influence is not evenly distributed across layers but varies across representation stages. These observations suggest that the distribution of prompt influence is tied to the stage-wise evolution of representations. To quantify the alignment between prompt influence and representation stages, we introduce the Prompt-Stage Misalignment Index (PSMI).

Based on these observations, we propose **DHAFT**, a hierarchy-aware soft prompt tuning framework for LLM reasoning. DHAFT dynamically derives hierarchical representation stages from cross-layer representation similarity and assigns stage-specific prompts to representative layers within each stage. Unlike existing strategies that determine prompt placement independently of representation stages, DHAFT injects each prompt at the representative layer of its corresponding stage, providing stage-specific guidance along the representation hierarchy. The stage partition is derived offline from unlabeled calibration samples, and inference directly reuses the derived stages and trained prompts without additional per-sample computation. Experiments across multiple LLMs and reasoning benchmarks demonstrate that DHAFT achieves competitive or superior performance compared with representative PEFT methods while maintaining minimal trainable parameters. To conclude, the main contributions of this paper are summarized as follows:

- We propose DHAFT, a hierarchy-aware soft prompt tuning framework that dynamically derives hierarchical representation stages from cross-layer similarity and injects stage-specific prompts at representative layers.
- We design a stage-specific prompt allocation mechanism that associates prompt guidance with corresponding representation stages and avoids persistent propagation across depth.
- We introduce PSMI, a stage-aware diagnostic metric that quantifies the mismatch between prompt influence and the stage-wise evolution of internal representations.
- Extensive experiments demonstrate that DHAFT achieves competitive or superior performance compared with representative PEFT methods across multiple LLMs and reasoning benchmarks while maintaining minimal trainable parameters.

2 Related Work

Prompt-based Fine-tuning for LLM Reasoning. Soft prompt tuning adapts frozen language models with a small number of trainable continuous vectors (Ding et al., 2023; Liu et al., 2022c), ranging from input-level injection (Lester et al., 2021; Shi & Lipani, 2024; Liu et al., 2025) to layer-wise (Li & Liang, 2021; Zhu et al., 2024; Wang et al., 2026a) and selective placement (Liu et al., 2022a; Zhu & Tan, 2023). Further work improves the prompt itself, through query, key, and value prompts in attention (APrompt (Wang et al., 2023b)), sparse mixtures of short prompts (SMoP (Choi et al., 2023)), decomposition (Wang et al., 2024; Lan et al., 2025), instance-adaptive selection (Yuan et al., 2024; Yang et al., 2023), hierarchical designs (Wang et al., 2022; Zeng et al., 2024; Li et al., 2026), and joint prompt–parameter optimization (Bo et al., 2026). For complex reasoning, DPC (Fan et al., 2025) suppresses prompt tokens that later reasoning steps over-rely on, at the cost of a complete reasoning pass per instance. Despite these advances, prompt placement in these methods is typically determined without explicit reference to the representation structure of the model. Our view is complementary: DHAFT derives representation stages from cross-layer similarity, injects a stage-specific prompt at the representative layer of each stage, and requires no per-instance diagnosis at inference.

Layer-wise Representation Analysis in LLMs. Beyond attribution tools such as saliency, relevance propagation, and knowledge neurons (Simonyan et al., 2013; Selvaraju et al., 2017; Chefer et al., 2021; Dai et al., 2021; Wang et al., 2023a; Voita et al., 2019), recent work characterizes depth-dependent structure directly. Lad et al. (2026) hypothesize distinct stages of inference from layer-wise interventions, Skean et al. (2025) show that intermediate layers encode richer representations than the final layer, and Gromov et al. (2025) use inter-layer similarity to identify prunable layers. Representation-level interventions (Wang et al., 2025; Huang et al., 2025a; Xu et al., 2026; Yan et al., 2026) further show that critical internal states strongly affect reasoning. Motivated by this evidence, DHAFT uses cross-layer CKA similarity (Kornblith et al., 2019) to derive representation stages and studies whether prompt influence aligns with them.

3 Method

In this section, we first formulate soft prompt tuning for LLM reasoning in Sec. 3.1. Next, Sec. 3.2 investigates how prompt-induced signals evolve across Transformer layers and observes a stage-dependent pattern of prompt effectiveness, motivating the study of prompt-stage alignment. Building on this analysis, we propose DHAFT, a hierarchy-aware prompt tuning framework that groups Transformer layers into hierarchical representation stages and performs stage-specific prompt injection at representative layers. Sec. 3.3 presents the overall framework, including CKA-based hierarchical stage partitioning and representative-layer prompt injection, as illustrated in Fig. 3 and Fig. 4. Finally, Sec. 3.4 describes the parameter-efficient training and inference procedures of DHAFT.

3.1 Preliminaries

We consider a frozen autoregressive LLM with L Transformer layers, where layer l maps hidden states $X^{(l-1)}$ to $X^{(l)} = F_l(X^{(l-1)}) \in \mathbb{R}^{n\times d}$, given input embeddings $X^{(0)} = E_{\text{in}}$ of n tokens. Standard soft prompt tuning prepends m trainable vectors $P \in \mathbb{R}^{m\times d}$ to the input embeddings E_{in}, yielding a sequence of length $n+m$, and optimizes P under the autoregressive objective on the target $y = \{y_t\}_{t=1}^{T}$ while keeping the backbone frozen; once introduced, the prompt tokens participate in the computation of every subsequent layer. Throughout this paper, a *representation stage* refers to a contiguous group of layers with similar representations along depth, as opposed to a step in the reasoning sequence. Our goal is to align prompt guidance with these representation stages.

3.2 Probing Prompt-Induced Stage Misalignment

Experimental Setup. We conduct the probing study on 500 GSM8K examples using Llama-3-8B. To isolate the effect of prompt placement, we compare four soft prompt injection strategies: shallow injection, middle injection, deep injection, and full-layer injection. For each setting, we compute layer-wise saliency scores and standardize them to make saliency trajectories comparable across different strategies. A no-prompt reference trajectory and additional details are provided in Appendix A.1.

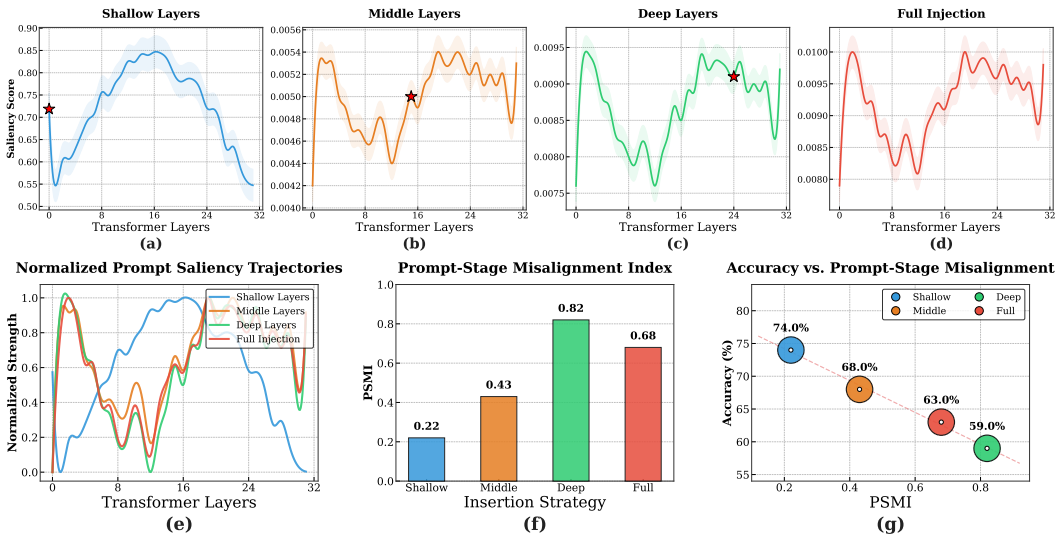


Figure 2: **Diagnostic analysis of prompt-induced cross-layer propagation. Top:** Layer-wise prompt saliency patterns under different soft prompt insertion settings: **(a)** shallow injection, **(b)** middle injection, **(c)** deep injection, and **(d)** full-layer injection. These plots illustrate how different insertion depths lead to distinct cross-layer propagation patterns. **Bottom:** Quantitative analysis of prompt-stage misalignment: **(e)** normalized prompt saliency trajectories, **(f)** Prompt-Stage Misalignment Index (PSMI), and **(g)** the relationship between PSMI and GSM8K accuracy.

Layer-wise Prompt Saliency and Information Flow. To estimate the influence of prompt-related information across layers, we compute a gradient-weighted saliency score over attention maps following (Wang et al., 2023a; Fan et al., 2025). This score measures the sensitivity of the task loss to attention patterns involving prompt tokens and serves as a proxy for the strength of prompt influence at each layer. For the h-th attention head at layer l, let $A^{h,l}$ denote the attention matrix and $\mathcal{L}(x)$ denote the task loss. The saliency score at layer l is defined as

$$I^l = \sum_h \left| A^{h,l} \odot \frac{\partial \mathcal{L}(x)}{\partial A^{h,l}} \right|, \tag{1}$$

where $\odot$ denotes element-wise multiplication. The resulting saliency maps are further averaged over token positions to obtain a scalar saliency trajectory across Transformer layers.

As illustrated in Figure 2(a)–(d), different prompt placement strategies lead to distinct saliency propagation patterns. Although all four strategies introduce soft prompts into the model, their cross-layer trajectories exhibit different degrees of fluctuation, amplification, and decay. This observation suggests that the influence of soft prompts is not uniformly propagated across layers. Instead, prompt placement affects the internal information flow, indicating that prompt influence exhibits depth-dependent behavior.

Quantifying Prompt-Stage Misalignment. To quantify how far prompt influence deviates from its corresponding representation stage, we introduce the Prompt-Stage Misalignment Index (PSMI). For a prompt assigned to stage $\mathcal{G}_i$, let $I_i^l \geq 0$ denote its layer-wise saliency; we normalize it into a distribution over layers, $q_l^{(i)} = I_i^l / (\sum_{j=1}^{L} I_i^j + \epsilon)$, so that $q_l^{(i)}$ gives the fraction of prompt influence at layer l. Let $\{\mathcal{G}_1, \ldots, \mathcal{G}_G\}$ be the stage partition from CKA-based clustering and $\mathcal{S}$ the set of prompted stages; for single-prompt injection, $\mathcal{G}_i$ is the stage containing the injection layer. PSMI is defined as

$$\text{PSMI} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left[\sum_{l \notin \mathcal{G}_i} q_l^{(i)} + \alpha \frac{1}{\max(1, |\mathcal{E}_i|)} \sum_{(l,l+1) \in \mathcal{E}_i} |q_{l+1}^{(i)} - q_l^{(i)}| \right], \tag{2}$$

where the first term measures the influence mass that leaks outside the assigned stage, and the second term measures the average change between adjacent layers within the stage, with $\mathcal{E}_i = \{(l, l+1) \mid$

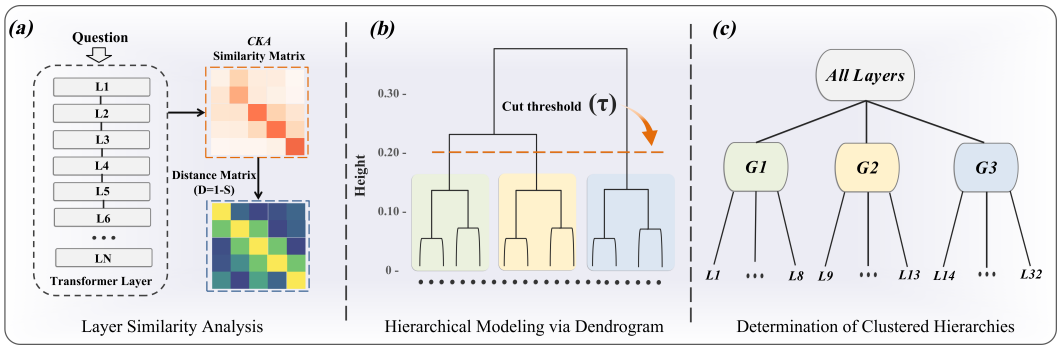


Figure 3: CKA-based hierarchical clustering in DHAFT. Cross-layer representational similarity is first computed using CKA. Hierarchical clustering then constructs a dendrogram over Transformer layers. A cut threshold produces candidate stage partitions, and the final number of stages is selected using the Silhouette coefficient with bootstrap stabilization.

$l, l+1 \in \mathcal{G}_i\}$ and the term set to zero when $|\mathcal{E}_i| = 0$. The weight α balances the two terms; we use $\alpha = 1$. A derivation and further properties are given in Appendix A.4.

Analysis Results. Figure 2(e) illustrates the normalized saliency trajectories under the four placement strategies, where each curve reflects how prompt influence evolves across Transformer layers. Different placement strategies lead to distinct propagation patterns, indicating that prompt influence varies across Transformer depth. Figure 2(f) further quantifies these differences using PSMI. Strategies with higher PSMI exhibit stronger cross-stage leakage or within-stage fluctuation, suggesting that their prompt influence is less concentrated within the assigned representation stage. As shown in Figure 2(g), higher PSMI is generally accompanied by lower GSM8K accuracy across the four strategies. These findings indicate that prompt influence under fixed injection may not fully align with the evolving representation stages across Transformer depth. We refer to this mismatch as prompt-stage misalignment.

3.3 DHAFT Framework

Based on the above analysis, we propose DHAFT, which aligns soft prompt intervention with the model's hierarchical representation stages through CKA-based stage partitioning and sparse stage-specific prompt injection. We introduce DHAFT through three tightly connected components: hierarchical stage partitioning, stage-specific soft prompt injection, and parameter-efficient training and inference.

3.3.1 Hierarchical Partitioning

Given the layer representations $\{X^{(l)}\}_{l=1}^{L}$ defined above, we construct a hierarchical partition of Transformer layers based on cross-layer representational similarity. To this end, we adopt Centered Kernel Alignment (CKA) (Kornblith et al., 2019), a normalized dependence measure for comparing neural network features. CKA is invariant to orthogonal transformations and isotropic scaling, and thus compares representation spaces rather than raw coordinates, which is essential when comparing layers whose feature bases differ.

Given two layer representations $X, Y \in \mathbb{R}^{n \times d}$, the CKA score is defined as

$$\mathrm{CKA}(K, L) = \frac{\mathrm{HSIC}(K, L)}{\sqrt{\mathrm{HSIC}(K, K)\,\mathrm{HSIC}(L, L)}} \tag{3}$$

where $K = XX^\top$, $L = YY^\top$, and HSIC denotes the Hilbert–Schmidt Independence Criterion.

After obtaining the similarity matrix, we compute pairwise CKA scores across the L layers to construct $S \in [0, 1]^{L \times L}$. We then apply agglomerative hierarchical clustering (Murtagh & Contreras, 2012) to partition the model into G stages $\{\mathcal{G}_1, \ldots, \mathcal{G}_G\}$ (see Fig. 3). Each resulting group is treated as a representation stage, consisting of layers with similar representational behavior.

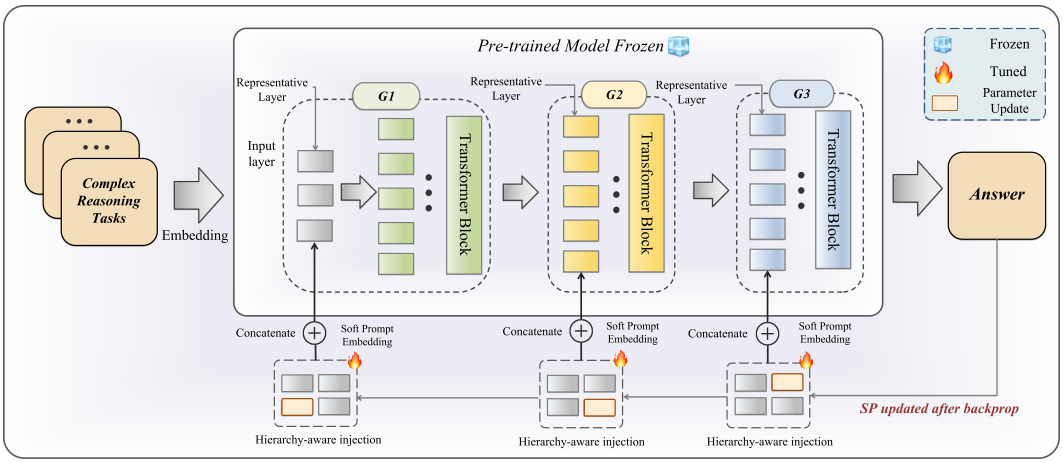


Figure 4: Stage-specific prompt injection in DHAFT. After representation stages are derived by CKA-based clustering, a trainable soft prompt is inserted at the representative layer of each stage. During training, the backbone is frozen and only stage-specific prompt parameters are optimized.

The number of stages G is determined in a data-driven manner. We construct a dendrogram $\mathcal{T}$ from the CKA-based similarity matrix, which provides multi-granularity layer partitions without requiring a pre-specified number of clusters. These partitions offer interpretable layer groupings and provide the structural basis for stage-specific soft prompt injection.

On the dendrogram $\mathcal{T}$, a cut threshold τ produces the initial number of clusters as:

$$G(\tau) = \text{NumClusters}(\mathcal{T}, \tau) \tag{4}$$

corresponding to different hierarchical partitions. To select the optimal hierarchy number, we compute the Silhouette coefficient for each partition and take the best-performing one as:

$$G^{\star} = \arg\max_{G(\tau)} \text{Silhouette}(G(\tau)) \tag{5}$$

Furthermore, to mitigate the instability caused by sample distribution randomness, we adopt bootstrap resampling and choose the value of G that appears most frequently across repetitions. In practice, G is typically constrained to the range of $[3, 5]$ to balance hierarchical granularity with additional parameter overhead. We report sensitivity analyses on G in Table 4.

3.3.2 Stage-Specific Soft Prompt Injection

After obtaining the stage partition, we incorporate stage-specific prompt injection, in which trainable SPs provide stage-aware guidance during the forward process. Unlike layer-wise methods that attach prompts to every layer, our method injects prompts only at the representative layer of each representation stage (see Fig. 4). Moreover, each stage-specific prompt is used only within its corresponding representation stage and is removed before entering the next stage, thereby avoiding prompt accumulation across depth and mitigating cross-stage prompt interference.

For each stage $\mathcal{G}_i$, we allocate a trainable soft prompt $P^{(i)} \in \mathbb{R}^{m \times d}$, where m denotes the prompt length and d the hidden dimension. The injection is implemented by concatenation along the sequence dimension, so that the prompt tokens are processed jointly with the original hidden states within the current stage.

For the first stage, the input consists of raw embeddings $E_{\text{in}} \in \mathbb{R}^{n \times d}$. We prepend the stage prompt $P^{(1)}$ to obtain:

$$\widetilde{X}^{(1)} = \text{Concat}(P^{(1)}, E_{\text{in}}) \in \mathbb{R}^{(n+m) \times d}. \tag{6}$$

At the end of this stage, the prompt tokens are explicitly discarded, and only the hidden states of the original task tokens are preserved. For each subsequent stage $\mathcal{G}_i$, we similarly prepend the new stage prompt $P^{(i)}$ to the task-token hidden states at its representative layer r_i:

$$\widetilde{X}^{(r_i)} = \text{Concat}(P^{(i)}, X_{\text{task}}^{(r_i)}) \in \mathbb{R}^{(n+m) \times d}, \tag{7}$$

where $X_{\text{task}}^{(r_i)} \in \mathbb{R}^{n \times d}$ denotes the hidden states of the original task tokens passed from the previous stage. Since these prompt tokens are removed after the corresponding stage, the sequence length remains $n + m$ during each local intervention and does not accumulate across depth. Formal details of prompt token removal, task-token extraction, and the no-accumulation property are provided in Appendix A.5.

3.4 Training and Inference

Training. DHAFT follows the standard autoregressive language modeling objective. Given the target sequence $y_{1:T}$, the model predicts each token conditioned on the ground-truth prefix under teacher forcing:

$$p_\theta(y_t \mid y_{<t}, X^{(L)}). \tag{8}$$

The training loss is defined as

$$\mathcal{L}_{\text{CE}}(\theta) = -\sum_{t=1}^{T} \log p_\theta(y_t \mid y_{<t}, X^{(L)}). \tag{9}$$

During optimization, all pretrained backbone parameters are frozen, and only the stage-specific prompts $\{P^{(i)}\}_{i=1}^{G}$ are updated. The CKA-based stage partitioning is performed offline on a small calibration set before fine-tuning and does not use label information. For clarity, we provide the detailed algorithmic workflow and pseudocode of DHAFT in Appendix A.2.

Inference. At inference time, DHAFT reuses the derived stage partition and the learned stage-specific prompts. No additional clustering or saliency computation is required. The input is propagated through the frozen LLM, and prompts are inserted only at the selected representative layers. The final hidden states are then used for autoregressive answer generation. Therefore, DHAFT introduces only lightweight prompt-related overhead while maintaining an end-to-end inference pipeline.

4 Experiment

In this section, we conduct experiments to evaluate DHAFT from multiple aspects. We first introduce the experimental settings, including datasets, baselines, and implementation details in Section 4.1. Then, we compare DHAFT with representative PEFT methods across different language model backbones and reasoning benchmarks in Section 4.2. Finally, we perform comprehensive ablation studies on stage partition strategies and hierarchy numbers to analyze the contribution of each component in Section 4.3.

4.1 Experimental Setup

Models and Datasets. We evaluate DHAFT on LLaMA-1 (7B/13B) (Touvron et al., 2023), Llama-2-7B (Touvron et al., 2023), Llama-3-8B (Dubey et al., 2024), and Qwen3-8B (Yang et al., 2025). Experiments cover 13 benchmarks from two reasoning domains. For commonsense reasoning, we use BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Easy, ARC-Challenge (Clark et al., 2018), and OBQA (Mihaylov et al., 2018). For complex reasoning, we use GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), AQuA (Ling et al., 2017), MAWPS (Koncel-Kedziorski et al., 2016), and SVAMP (Patel et al., 2021).

Baselines and Protocol. We compare DHAFT with representative PEFT baselines, including Prefix-Tuning (Li & Liang, 2021), Adapter Tuning (Houlsby et al., 2019; He et al., 2022), LoRA (Hu et al., 2022), DoRA (Liu et al., 2024a), HiRA (Huang et al., 2025b), and representation intervention methods LoReFT and DiReFT (Wu et al., 2024). Following Hu et al. (Hu et al., 2023) and Liu et al. (Liu et al., 2024a), we fine-tune models on the combined training set of eight commonsense subtasks and report results on each task-specific test set. More details of the evaluation pipeline are provided in Appendix A.7.

Metrics and Training Details. We use accuracy as the primary metric for all benchmarks. In all experiments, backbone parameters are frozen and only stage-specific soft prompts are optimized. Before fine-tuning, we perform CKA-based hierarchical clustering on 256 unlabeled calibration

Table 1: **Performance comparison on eight commonsense reasoning benchmarks.** We evaluate DHAFT against representative PEFT baselines across multiple backbones. Results marked with * are taken from prior reports. Best results are **bolded**, and second-best results are underlined.

Model	PEFT	Params (%)	Accuracy (↑)								
			BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
ChatGPT*	—	—	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA-7B	PrefT (ACL 2021)*	0.039%	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
	Adapter^S (ICML 2019)*	1.953%	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
	Adapter^P (ICLR 2022)*	3.542%	67.9	76.4	78.8	69.8	78.9	73.7	57.3	75.2	72.3
	LoRA (ICLR 2022)*	0.826%	68.9	80.7	77.4	78.1	78.8	77.8	61.3	74.8	74.7
	DoRA (ICML 2024)*	0.838%	68.5	82.9	79.6	84.8	80.8	81.4	65.8	81.0	78.1
	DiReFT (NeurIPS 2024)*	0.031%	69.5	83.0	79.0	92.5	80.5	82.2	68.0	77.5	79.0
	LoReFT (NeurIPS 2024)*	0.031%	69.3	**84.4**	80.3	**93.1**	84.2	83.2	68.2	78.9	80.2
	DHAFT (Ours)	0.018%	**72.1**	84.1	**82.0**	91.9	**84.7**	**83.7**	**69.0**	**81.3**	**81.1**
LLaMA-13B	PrefT (ACL 2021)*	0.031%	65.3	75.4	72.1	55.2	68.6	79.5	62.9	68.0	68.4
	Adapter^S (ICML 2019)*	1.586%	71.8	83.0	79.2	88.1	82.4	82.5	67.3	81.8	79.5
	Adapter^P (ICLR 2022)*	2.894%	**72.5**	84.9	79.8	92.1	84.7	84.2	71.2	82.4	81.5
	LoRA (ICLR 2022)*	0.670%	72.1	83.5	80.5	90.5	83.7	82.8	68.3	82.4	80.5
	DoRA (ICML 2024)*	0.681%	72.4	84.9	81.5	92.4	84.2	84.2	69.6	82.8	81.5
	DiReFT (NeurIPS 2024)*	0.025%	71.3	86.1	80.8	94.6	83.6	85.5	72.9	82.7	82.2
	LoReFT (NeurIPS 2024)*	0.025%	72.1	86.3	81.8	**95.1**	**87.2**	86.2	73.7	84.2	83.3
	DHAFT (Ours)	0.020%	72.4	**87.1**	**82.7**	94.9	86.8	**87.7**	**75.2**	**85.0**	**84.0**
Llama-2 7B	LoRA (ICLR 2022)*	0.826%	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
	DoRA (ICML 2024)*	0.838%	71.8	83.7	76.0	89.1	82.6	83.7	68.2	82.4	79.7
	DiReFT (NeurIPS 2024)*	0.031%	70.8	83.6	80.2	93.6	82.1	84.8	70.4	81.5	80.9
	LoReFT (NeurIPS 2024)*	0.031%	71.1	83.8	80.8	**94.3**	**84.5**	85.6	72.2	82.3	81.8
	HiRA (r=16) (ICLR 2025)*	0.413%	69.8	80.2	78.2	87.0	83.4	85.9	71.3	81.0	79.6
	HiRA (r=32) (ICLR 2025)*	0.826%	71.2	83.4	79.5	88.1	84.0	**86.7**	**73.8**	**84.6**	81.4
	DHAFT (Ours)	0.025%	**72.2**	**84.1**	**82.3**	92.9	84.4	86.1	73.1	83.3	**82.3**
Llama-3 8B	LoRA (ICLR 2022)*	0.700%	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
	DoRA (ICML 2024)*	0.710%	74.6	89.3	79.9	95.5	85.6	90.5	80.4	85.8	85.2
	DiReFT (NeurIPS 2024)*	0.026%	73.4	88.7	81.0	95.6	85.5	91.8	81.8	85.4	85.4
	LoReFT (NeurIPS 2024)*	0.026%	75.1	**90.2**	**82.0**	96.3	87.4	92.4	81.6	87.5	86.6
	HiRA (r=16) (ICLR 2025)*	0.351%	73.9	89.1	81.1	94.9	86.7	93.1	82.6	87.4	86.1
	HiRA (r=32) (ICLR 2025)*	0.700%	**75.4**	89.7	81.2	95.4	87.7	**93.3**	**82.9**	**88.3**	86.7
	DHAFT (Ours)	0.021%	75.3	89.7	81.7	**96.7**	**88.3**	93.1	82.0	88.1	**86.9**

inputs randomly sampled from the training set with a fixed seed, using only input sequences without labels or test examples. We train models with AdamW, a learning rate of 2×10^{-5}, and a linear decay scheduler. The batch size is selected from $\{4, 8\}$ according to model size and GPU memory. All experiments are conducted on 8 NVIDIA GeForce RTX 4090 GPUs.

4.2 Performance Evaluation

Table 1 and Table 2 report the main results of DHAFT against representative PEFT baselines on commonsense reasoning and complex arithmetic/logical reasoning benchmarks.

Commonsense Reasoning Benchmarks. Table 1 shows that DHAFT achieves the highest average accuracy across all evaluated backbones, obtaining 81.1%, 84.0%, 82.3%, and 86.9% on LLaMA-7B, LLaMA-13B, Llama-2-7B, and Llama-3-8B, respectively. Compared with the strongest competing baselines, DHAFT improves the average score by 0.9, 0.7, 0.5, and 0.2 points, while using only 0.018%–0.025% trainable parameters. Although the margins on commonsense tasks are moderate, DHAFT attains these results with substantially fewer trainable parameters than weight-based methods such as LoRA, DoRA, and Adapter variants, and remains competitive with ultra-light representation-based baselines such as DiReFT and LoReFT. These results suggest that stage-aligned prompt intervention provides a parameter-efficient way to adapt LLMs across diverse commonsense reasoning tasks.

Complex Reasoning Benchmarks. Table 2 further evaluates DHAFT on complex arithmetic and logical reasoning tasks. The advantage of DHAFT becomes more pronounced on these multi-step reasoning benchmarks. On LLaMA-2-7B, DHAFT achieves an average accuracy of 55.40%, surpassing the strongest baseline LoReFT by 3.34 points. On LLaMA-3-8B and Qwen3-8B, DHAFT reaches 57.90% and 57.30% average accuracy, improving over LoReFT by 1.90 and 1.58 points, respectively. The improvement is especially clear on GSM8K, where DHAFT outperforms LoReFT by 7.2 points on LLaMA-2-7B and 6.0 points on LLaMA-3-8B. DHAFT also obtains the best results on all five complex reasoning benchmarks for LLaMA-2-7B and Qwen3-8B, and on four out of five benchmarks for LLaMA-3-8B. These results indicate that DHAFT is particularly effective when reasoning requires stable cross-layer propagation of intermediate states.

Overall, DHAFT consistently achieves the best average performance across both commonsense and complex reasoning settings while maintaining only 0.018%–0.027% trainable parameters. The stronger gains on complex reasoning further align with our motivation: prompt intervention should be placed at functionally appropriate stages to reduce potential prompt-stage mismatch and provide more targeted guidance during multi-step reasoning. The following ablation studies further examine how each component of DHAFT contributes to this performance.

Table 2: **Performance comparison on complex arithmetic and logical reasoning benchmarks.** Best results are **bolded**, and second-best results are underlined.

Model	PEFT	Params(%)	Accuracy (↑)					Avg.
			GSM8K	MATH	AQuA	MAWPS	SVAMP	
LLaMA-2 7B	PrefT (ACL 2021)	0.039%	43.4	20.2	18.1	73.3	39.9	38.98
	Adapter^P (ICLR 2022)	3.542%	37.8	17.7	19.7	79.9	46.7	40.36
	LoRA (ICLR 2022)	0.826%	54.1	31.3	23.3	82.1	47.3	47.62
	DoRA (ICML 2024)	0.838%	55.4	26.7	35.6	84.7	51.9	50.86
	LoReFT (NeurIPS 2024)	0.031%	58.9	28.3	34.7	85.1	53.3	52.06
	DHAFT (Ours)	0.025%	**66.1**	**33.0**	**36.7**	**85.5**	**55.7**	**55.40**
LLaMA-3 8B	PrefT (ACL 2021)	0.045%	45.7	33.0	41.3	53.3	41.5	42.96
	Adapter^P (ICLR 2022)	3.240%	53.7	35.7	42.7	55.8	51.1	47.80
	LoRA (ICLR 2022)	0.700%	50.9	27.1	42.9	58.5	47.3	45.34
	DoRA (ICML 2024)	0.710%	65.5	34.0	41.6	66.3	53.0	52.08
	LoReFT (NeurIPS 2024)	0.026%	68.0	40.1	**44.9**	69.0	58.0	56.00
	DHAFT (Ours)	0.021%	**74.0**	**41.9**	44.7	**70.1**	**58.8**	**57.90**
Qwen3-8B	PrefT (ACL 2021)	0.040%	43.3	27.8	30.7	57.1	44.6	40.70
	Adapter^P (ICLR 2022)	3.050%	50.7	30.2	32.2	63.3	47.8	44.84
	LoRA (ICLR 2022)	0.750%	66.7	36.8	36.7	67.7	53.3	52.24
	DoRA (ICML 2024)	0.760%	65.3	42.1	40.9	70.6	51.6	54.10
	LoReFT (NeurIPS 2024)	0.028%	71.5	41.7	41.3	71.3	52.8	55.72
	DHAFT (Ours)	0.027%	**72.3**	**42.4**	**41.7**	**74.4**	**55.7**	**57.30**

4.3 Ablation Studies

Table 3: Ablation study on different stage partition strategies and similarity metrics.

Method	GSM8K	MATH	AQuA
Prompt Tuning (input-level)	65.5	33.7	38.5
Uniform partition	65.8	33.9	39.0
Cosine similarity + clustering	66.9	34.2	40.5
Euclidean distance + clustering	66.5	34.0	40.1
DHAFT (Ours)	**74.0**	**41.9**	**44.7**

Table 4: Ablation study on the number of hierarchical stages G across different models.

Model	Stages G						
	$G=2$	$G=3$	$G=4$	$G=5$	$G=6$	$G=7$	$G=8$
LLaMA-7B	41.5	**43.7**	42.8	41.2	40.5	39.8	38.2
LLaMA-13B	45.2	47.1	**48.8**	47.5	46.2	45.0	43.8
LLaMA-2 7B	63.4	**66.1**	65.2	63.8	62.5	61.2	60.0
Llama-3-8B	71.5	**74.0**	72.8	70.2	69.0	68.2	67.0
Qwen3-8B	68.5	70.1	71.2	**72.3**	70.8	69.5	68.2

Effect of Stage Partition Strategy. Table 3 reports the effect of different partition strategies on Llama-3-8B across GSM8K, MATH, and AQuA. Prompt Tuning, which uses a single input-level prompt, obtains the lowest scores. Uniform partition, which splits layers into equal-sized groups regardless of their representations, brings only marginal improvements. Clustering on cosine similarity or Euclidean distance improves further, and CKA-based clustering achieves the best results, reaching 74.0% on GSM8K, 41.9% on MATH, and 44.7% on AQuA. These results indicate that deriving stage boundaries from layer representations, rather than partitioning uniformly, is important for stage-specific prompt injection.

Effect of Stage Number. Table 4 evaluates the number of stages G under a fixed 64-token prompt budget. Across different backbones, performance generally peaks at moderate values of G, such as $G = 3$ for LLaMA-7B, LLaMA-2 7B, and Llama-3-8B, $G = 4$ for LLaMA-13B, and $G = 5$ for Qwen3-8B. Increasing G beyond the optimum consistently reduces accuracy, suggesting that over-segmentation fragments prompt guidance. These results indicate that DHAFT benefits from balanced stage granularity rather than simply increasing the number of prompt insertion points.

5 Conclusion

In this paper, we present DHAFT, a hierarchy-aware soft prompt tuning framework for efficient LLM reasoning. DHAFT dynamically identifies hierarchical representation stages from cross-layer representation similarity and assigns stage-specific prompts to representative layers, reducing potential cross-stage prompt interference. To characterize this behavior, we introduce PSMI, a prompt-stage misalignment index that quantifies the mismatch between prompt influence and the stage-wise evolution of internal representations. Extensive experiments across multiple LLM backbones and reasoning benchmarks demonstrate that DHAFT achieves competitive or superior performance compared with representative PEFT methods while maintaining minimal parameter overhead.

AI USE STATEMENT

The authors used LLM assistance only for language polishing, including grammar correction, spelling refinement, word choice, and clarity improvement. All scientific claims, experimental results, citations, methodological designs, tables, formulas, code, scripts, figures, LaTeX source, and the final manuscript were produced, verified, edited, and approved by the authors. LLM assistance was not used as an autonomous source of scientific claims, experimental findings, citations, methodological decisions, or final research conclusions.

REPRODUCIBILITY STATEMENT

We emphasize the reproducibility of our work and provide detailed descriptions across the main paper, appendix, and supplementary materials. Section 3 presents the proposed DHAFT framework in detail, including cross-layer representation similarity computation, hierarchical stage partitioning, and stage-specific soft prompt injection. To facilitate implementation, Appendix A.2 provides pseudocode for the core procedures, including stage partitioning, prompt assignment, and training, together with a step-by-step description of the overall workflow. Section 4 describes the experimental settings, including evaluated models, reasoning benchmarks, training configurations, parameter settings, and evaluation protocols. Additional analyses in Section 4 and Appendix A.1 report ablation studies on stage partition strategies and the number of representation stages, as well as analyses of prompt influence across model depth. Variance over three random seeds is reported in Appendix A.8. All datasets used in this work are publicly available, and their sources are clearly cited in the paper. Upon acceptance, we will release the source code and training scripts to facilitate reproduction and further research.

REFERENCES

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

Xiaohe Bo, Rui Li, Zexu Sun, Quanyu Dai, Zeyu Zhang, Zihang Tian, Xu Chen, and Zhenhua Dong. Prompt and parameter co-optimization for large language models. In *International Conference on Learning Representations*, volume 2026, pp. 116697–116723, 2026.

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

Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 782–791, 2021.

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

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54158–54183, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/edc36117f795ca52a0cbf6a7b3882859-Paper-Conference.pdf.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pp. 2924–2936, 2019.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature machine intelligence*, 5(3):220–235, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

Sinan Fan, Liang Xie, Chen Shen, Ge Teng, Xiaosong Yuan, Xiaofeng Zhang, Chenxi Huang, Wenxiao Wang, Xiaofei He, and Jieping Ye. Improving complex reasoning with dynamic prompt corruption: A soft prompt optimization approach. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=h7Qz1ulnvF.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. In *International Conference on Learning Representations*, volume 2025, pp. 81906–81920, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0RDcd5Axok.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, et al. Data interpreter: An llm agent for data science. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19796–19821, 2025.

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

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 5254–5276, 2023.

Chenxi Huang, Shaotian Yan, Liang Xie, Binbin Lin, Sinan Fan, Yue Xin, Deng Cai, Chen Shen, and Jieping Ye. Enhancing chain-of-thought reasoning with critical representation fine-tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23173–23195, 2025a.

Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. Hira: Parameter-efficient hadamard high-rank adaptation for large language models. In *13th International Conference on Learning Representations (ICLR 2025)*, pp. 83935–83954. International Conference on Learning Representations, ICLR, 2025b.

Yeonjoon Jung, Daehyun Ahn, Hyungjun Kim, Taesu Kim, and Eunhyeok Park. Gralora: Granular low-rank adaptation for parameter-efficient fine-tuning. *Advances in Neural Information Processing Systems*, 38:95822–95845, 2026.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. Mawps: A math word problem repository. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*, pp. 1152–1157, 2016.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 54523–54549, 2025.

Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. Remarkable robustness of llms: Stages of inference? *Advances in Neural Information Processing Systems*, 38:130050–130083, 2026.

Pengxiang Lan, Haoyu Xu, Enneng Yang, Yuliang Liang, Guibing Guo, Jianzhe Zhao, and Xingwei Wang. Efficient and effective prompt tuning via prompt decomposition and compressed outer product. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4406–4421, 2025.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

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

Yuqi Li, Xi Xiao, Yunbei Zhang, Lin Zhao, Yu Li, Aiden Zhao, Tianyang Wang, Hao Xu, and Yingli Tian. Rethinking layer-wise information allocation for vision foundation model adaptation. *arXiv preprint arXiv:2607.21973*, 2026.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35, 2023.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121. PMLR, 21–27 Jul 2024a. URL https://proceedings.mlr.press/v235/liu24bn.html.

Xiangyang Liu, Tianxiang Sun, Xuanjing Huang, and Xipeng Qiu. Late prompt tuning: A late prompt could be better than many prompts. *arXiv preprint arXiv:2210.11292*, 2022a.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, Dublin, Ireland, May 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.8. URL https://aclanthology.org/2022.acl-short.8/.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, 2022c.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *AI open*, 5:208–215, 2024b.

Yiyang Liu, James C Liang, Heng Fan, Wenhao Yang, Yiming Cui, Xiaotian Han, Lifu Huang, Dongfang Liu, Qifan Wang, and Cheng Han. All you need is one: Capsule prompt tuning with a single vector. *arXiv preprint arXiv:2510.16670*, 2025.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 2(1):86–97, 2012.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Zhengxiang Shi and Aldo Lipani. Dept: Decomposed prompt tuning for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, volume 2024, pp. 51862–51882, 2024.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=WGXb7UdvTX.

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

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

Haonan Wang, Brian Chen, Siquan Li, Liang Xinhe, Hwee Lee, Kenji Kawaguchi, and Tianyang Hu. Prefixmemory-tuning: Modernizing prefix-tuning by decoupling the prefix from attention. In *International Conference on Learning Representations*, volume 2026, pp. 130687–130708, 2026a.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. Label words are anchors: An information flow perspective for understanding in-context learning. *arXiv preprint arXiv:2305.14160*, 2023a.

Qifan Wang, Yuning Mao, Jingang Wang, Hanchao Yu, Shaoliang Nie, Sinong Wang, Fuli Feng, Lifu Huang, Xiaojun Quan, Zenglin Xu, et al. Aprompt: Attention prompt tuning for efficient adaptation of pre-trained language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 9147–9160, 2023b.

Shangshang Wang, Julian Asilis, Ömer Faruk Akgül, Enes Bilgin, Ollie Liu, and Willie Neiswanger. Tina: Tiny reasoning models via lora. In *International Conference on Learning Representations*, volume 2026, pp. 90106–90140, 2026b.

Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: low-rank adaptation with gradient approximation. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pp. 54905–54931, 2024.

Weixuan Wang, Jingyuan Yang, and Wei Peng. Semantics-adaptive activation intervention for llms via dynamic steering vectors. In *International Conference on Learning Representations*, volume 2025, pp. 79334–79351, 2025.

Zihan Wang, Peiyi Wang, Tianyu Liu, Binghuai Lin, Yunbo Cao, Zhifang Sui, and Houfeng Wang. Hpt: Hierarchy-aware prompt tuning for hierarchical text classification. *arXiv preprint arXiv:2204.13413*, 2022.

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

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962, 2024.

Xiaoyu Xu, Yulan Pan, Xiaosong Yuan, Zhihong Shen, Minghao Su, Yuanhao Su, and Xiaofeng Zhang. Reasoning fails where step flow breaks. *arXiv preprint arXiv:2604.06695*, 2026.

Hanqi Yan, Hainiu Xu, Siya Qi, Shu Yang, and Yulan He. When thinking backfires: Mechanistic insights into reason-induced misalignment. In *The Fourteenth International Conference on Learning Representations*. International Conference on Learning Representations, ICLR, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. Dynamic early exit in reasoning models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=NpU7ZXafRi.

Xianjun Yang, Wei Cheng, Xujiang Zhao, Wenchao Yu, Linda Petzold, and Haifeng Chen. Dynamic prompting: A unified framework for prompt tuning. *arXiv preprint arXiv:2303.02909*, 2023.

Xiaosong Yuan, Chen Shen, Shaotian Yan, Xiaofeng Zhang, Liang Xie, Wenxiao Wang, Renchu Guan, Ying Wang, and Jieping Ye. Instance-adaptive zero-shot chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 37:125469–125486, 2024.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 57654–57689. Curran Associates, Inc., 2025. doi: 10.52202/085713-1933. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/537d5aa768c2d534016a4d06f87bc8fb-Paper-Conference.pdf.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Lei Zeng, Ruifang He, Haowen Sun, Jing Xu, Chang Liu, and Bo Wang. Global and local hierarchical prompt tuning framework for multi-level implicit discourse relation recognition. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 7760–7773, 2024.

Wei Zhu and Ming Tan. Spt: Learning to selectively insert prompts for better prompt tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 11862–11878, 2023.

Wei Zhu, Aaron Tian, Congrui Yin, Yuan Ni, Xiaoling Wang, and Guotong Xie. Iapt: instance-aware prompt tuning for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14285–14304, 2024.

A APPENDIX

CONTENTS

1 Introduction **1**

2 Related Work **3**

3 Method **3**

3.1 Preliminaries . . . 3

3.2 Probing Prompt-Induced Stage Misalignment . . . 3

3.3 DHAFT Framework . . . 5

3.3.1 Hierarchical Partitioning . . . 5

3.3.2 Stage-Specific Soft Prompt Injection . . . 6

3.4 Training and Inference . . . 7

4 Experiment **7**

4.1 Experimental Setup . . . 7

4.2 Performance Evaluation . . . 8

4.3 Ablation Studies . . . 9

5 Conclusion **9**

A Appendix **16**

A.1 Analysis of Stage Misalignment in Soft Prompts . . . 16

A.2 Pseudocode . . . 19

A.2.1 Pseudocode Explanation . . . 19

A.3 Centered Kernel Alignment and Its Invariance . . . 21

A.4 Derivation and Properties of the Prompt-Stage Misalignment Index . . . 21

A.5 Formal Details of Prompt Injection and Removal . . . 23

A.6 Notation . . . 23

A.7 More Experimental Setup . . . 23

A.8 More Experimental Results . . . 25

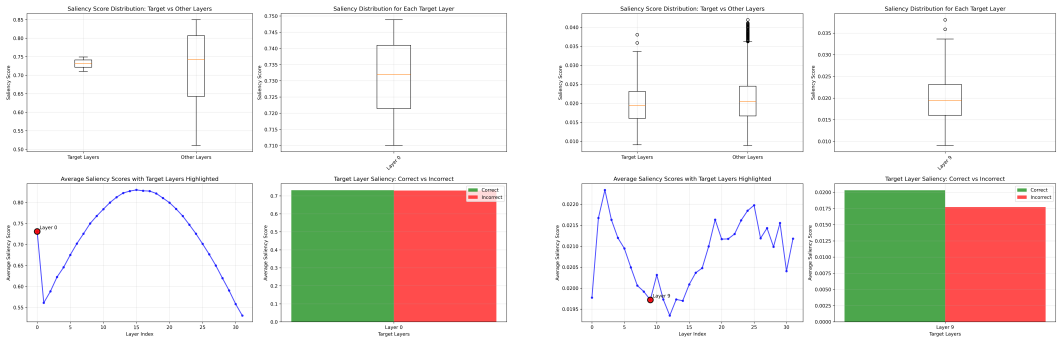
A.9 Limitations and Future Work. . . . 25

A.10 Broader impacts. . . . 26

A.11 Use of AI Assistance. . . . 26

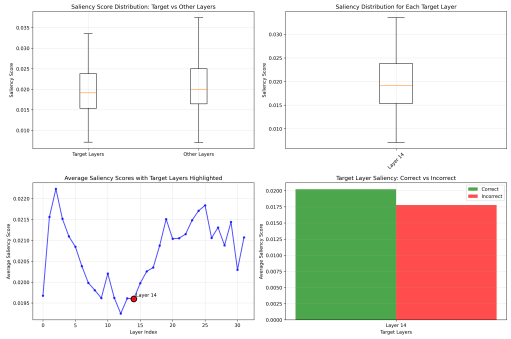
A.1 ANALYSIS OF STAGE MISALIGNMENT IN SOFT PROMPTS

This section provides the layer-wise saliency trajectories underlying the probing study in Section 3.2. All trajectories are computed on the same 500 GSM8K examples with Llama-3-8B, using the saliency score in Eq. equation 1 and the same normalization across settings, so that differences between panels reflect the insertion layer rather than the input data. Figure 5a serves as a reference: it shows the

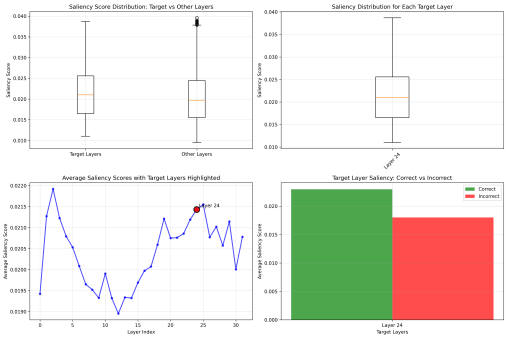


(a) No soft prompt

(b) Insertion at Layer 1



(c) Insertion at Layer 9



(d) Insertion at Layer 24

Figure 5: Layer-wise saliency trajectories on GSM8K with Llama-3-8B. (a) Reference trajectory of the frozen backbone without soft prompts. (b)–(d) Trajectories when a single soft prompt is inserted at Layer 1, Layer 9, and Layer 24, respectively. Each panel reports the saliency distribution across layers and the average trajectory.

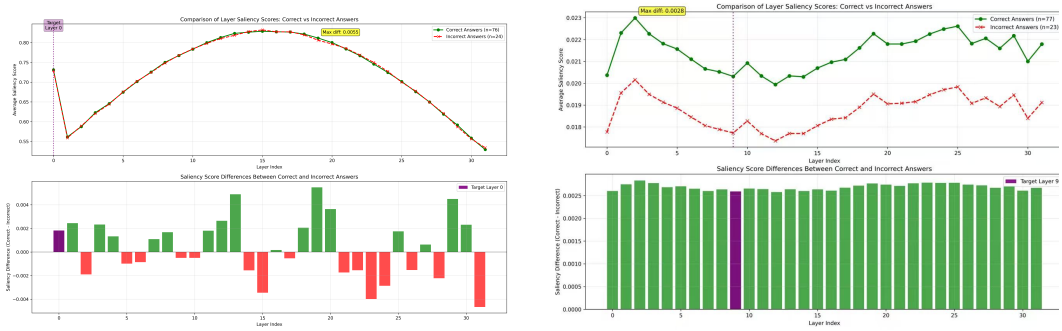
layer-wise saliency of the frozen backbone without any soft prompt, which characterizes the intrinsic depth profile of the model before prompt tuning.

Relative to this reference, inserting a soft prompt changes the shape of the trajectory, and the change depends on where the prompt enters the model. With insertion at Layer 1 (Figure 5b), the trajectory follows a single rise-and-fall profile, with saliency increasing through the shallow layers and decreasing toward the deep layers. With insertion at Layer 9 (Figure 5c), the trajectory exhibits several local peaks after the insertion layer, and its variation across adjacent layers becomes larger. With insertion at Layer 24 (Figure 5d), prompt-related saliency is concentrated in the final layers, where the trajectory changes more sharply within a narrow depth range.

Taken together, these trajectories indicate that the depth profile of prompt influence is shaped by the insertion layer: a single prompt, once inserted, affects all subsequent layers, and the resulting distribution of its influence differs across insertion positions. Because these layers span multiple representation stages derived in Section 3.3.1, the same prompt interacts with layers whose representations differ. PSMI (Eq. equation 2) summarizes this behavior by measuring how much of the prompt influence falls outside the stage containing the insertion layer and how much it varies within that stage; the corresponding values are reported in Figure 2(f).

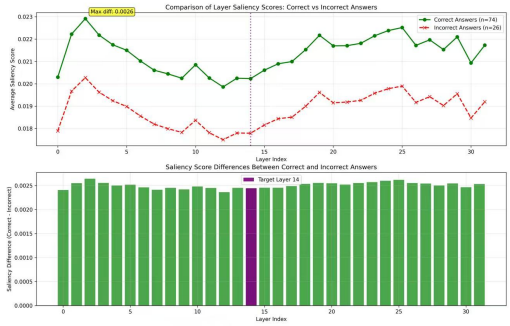
To examine how the insertion layer relates to reasoning outcomes, we split the examples in each setting by whether the final answer is correct and compare the average saliency trajectories of the two groups. All settings use the same examples, saliency score, and normalization as in Figure 5, so that differences across panels reflect the insertion layer.

With insertion at Layer 1 (Figure 6a), the trajectories of correct and incorrect examples largely overlap, and their difference is confined to a small number of shallow layers. With insertion at Layer 9 and Layer 14 (Figures 6b and 6c), the two trajectories diverge after the insertion layer, and the

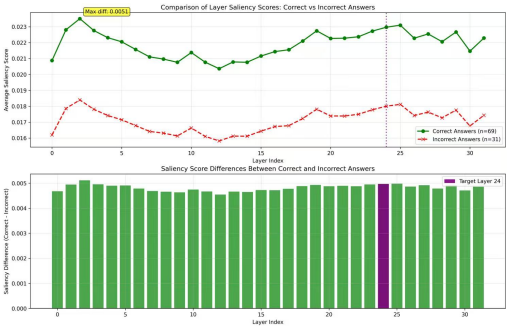


(a) Insertion at Layer 1

(b) Insertion at Layer 9



(c) Insertion at Layer 14



(d) Insertion at Layer 24

Figure 6: Average layer-wise saliency trajectories of correctly and incorrectly answered examples on GSM8K with Llama-3-8B, under a single soft prompt inserted at (a) Layer 1, (b) Layer 9, (c) Layer 14, and (d) Layer 24. Each panel shows the two average trajectories and their layer-wise difference.

difference extends over a wider range of depth. With insertion at Layer 24 (Figure 6d), the difference is concentrated in the final layers, where both trajectories change most sharply.

These comparisons indicate that the depth at which correct and incorrect examples differ in prompt influence shifts with the insertion layer, and tends to follow the layers after the prompt enters. Since these layers belong to different representation stages, the gap between the two groups is distributed across stages in a way that depends on where the prompt is inserted. This observation is consistent with the stage-level analysis in Section 3.2 and complements the PSMI results in Figure 2(f)–(g).

Figure 7 summarizes saliency statistics aggregated over all examples, complementing the per-setting trajectories in Figures 5 and 6. The four panels report, respectively, the average trajectories of correctly and incorrectly answered examples, the overall average trajectory, the trajectories of individual examples, and the distribution of saliency values.

The average trajectories of correct and incorrect examples (top-left) follow similar trends across depth, with differences smaller than the variation of either trajectory across layers. At this aggregate level, the two groups share a common depth profile, and the differences between them are more visible when examples are grouped by insertion layer, as in Figure 6.

The overall average trajectory (top-right) is unimodal, rising through the shallow layers, peaking around Layer 15, and decreasing toward the final layers. The sample-wise heatmap (bottom-left) shows the same profile at the level of individual examples: for most examples, saliency increases from the shallow layers toward the middle layers, and the position of the peak varies only moderately across examples. The peak region can be compared with the representation stages in Figure 1(b), which indicates which stage receives the largest share of prompt influence under this setting.

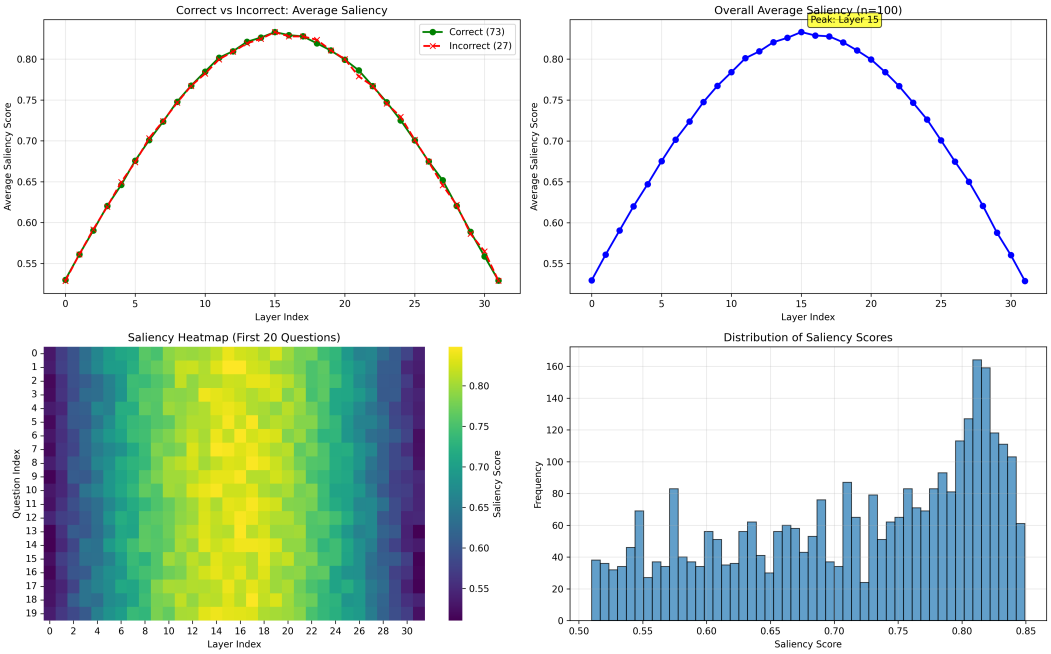


Figure 7: Comprehensive saliency statistics across layers. (Top-left) Average saliency trajectories for correct vs. incorrect predictions. (Top-right) Overall average saliency with peak at Layer 15. (Bottom-left) Heatmap of saliency evolution for the first 20 samples. (Bottom-right) Histogram of saliency score distribution. Statistics in this figure are computed on 100 of the 500 probing examples.

The histogram (bottom-right) shows that most saliency values lie between 0.70 and 0.82, so that the layer-wise differences in the trajectories above correspond to changes within a relatively narrow range of values. Together with the per-setting results, these statistics describe the depth profile against which the stage-level comparisons in Section 3.2 are made.

A.2 Pseudocode

This section provides the pseudocode of DHAFT, including hierarchical stage partitioning (Algorithm 1), the stage-wise forward pass with stage-specific prompt injection and removal (Algorithm 2), and prompt optimization (Algorithm 3).

A.2.1 Pseudocode Explanation

Hierarchical Stage Partitioning (Algorithm 1). DHAFT first collects hidden states of the frozen LLM on a small unlabeled calibration set and computes pairwise CKA scores between layers, yielding the similarity matrix S. To select the number of stages, the calibration set is resampled with replacement B times; for each resample, agglomerative hierarchical clustering builds a dendrogram, candidate thresholds τ produce partitions of different granularity, and the partition with the highest Silhouette coefficient determines G_b. The most frequent value among $\{G_b\}$ is taken as G, and the dendrogram built from the full calibration set is cut into G representation stages. The first layer of each stage is taken as its representative layer, which serves as the insertion point for the corresponding stage-specific prompt.

Forward Pass (Algorithm 2). DHAFT processes the layers stage by stage. Within stage $\mathcal{G}_i$, the trainable prompt $P^{(i)}$ is prepended to the task-token hidden states at the representative layer r_i along the sequence dimension and is attended to by layers $r_i, \ldots, e_i$, so the sequence length is $n + m$ within the stage while the hidden dimension d is unchanged. At the end of the stage, the prompt positions are discarded by Π_{task} and only the n task-token hidden states are passed to the next stage; hence every layer processes a sequence of length at most $n + m$, and prompts do not accumulate

Algorithm 1 Hierarchical Stage Partitioning

Require: Frozen LLM $\mathcal{M}$ with L layers; unlabeled calibration set $\mathcal{C}$; candidate thresholds $\{\tau\}$; bootstrap rounds B
Ensure: Representation stages $\{\mathcal{G}_i\}_{i=1}^{G}$ and representative layers $\{r_i\}_{i=1}^{G}$

Step 1: Collect layer representations
Run $\mathcal{M}$ on $\mathcal{C}$ and cache the hidden states of every layer
Stack the hidden states of layer l over all calibration tokens into $X^{(l)} \in \mathbb{R}^{N\times d}$, for $l = 1, \ldots, L$
Step 2: Compute the CKA similarity matrix
for $l = 1$ to L **do**
  for $l' = 1$ to L **do**
    $S_{l,l'} \leftarrow \mathrm{CKA}\big(X^{(l)}, X^{(l')}\big)$
  end for
end for
Step 3: Select the number of stages
for $b = 1$ to B **do**
  Resample $\mathcal{C}_b$ from $\mathcal{C}$ with replacement and compute $S^{(b)}$ as in Steps 1–2
  Build dendrogram $\mathcal{T}^{(b)}$ from $S^{(b)}$ by agglomerative hierarchical clustering
  for each threshold τ **do**
    $G_b(\tau) \leftarrow \mathrm{NumClusters}\big(\mathcal{T}^{(b)}, \tau\big)$ and compute $\mathrm{Silhouette}\big(G_b(\tau)\big)$
  end for
  $G_b \leftarrow \arg\max_{G_b(\tau)} \mathrm{Silhouette}\big(G_b(\tau)\big)$
end for
$G \leftarrow$ the most frequent value in $\{G_b\}_{b=1}^{B}$
Step 4: Derive stages and representative layers
Build dendrogram $\mathcal{T}$ from S and cut it into G clusters $\{\mathcal{G}_i\}_{i=1}^{G}$, indexed in order of their first layer
for $i = 1$ to G **do**
  $r_i \leftarrow \min \mathcal{G}_i$ ▷ first layer of $\mathcal{G}_i$
end for
return $\{\mathcal{G}_i\}_{i=1}^{G}, \{r_i\}_{i=1}^{G}$

Algorithm 2 DHAFT Forward Pass with Stage-Specific Prompt Injection and Removal

Require: Input embeddings $E_{\text{in}} \in \mathbb{R}^{n\times d}$; stages $\{\mathcal{G}_i = \{b_i, \ldots, e_i\}\}_{i=1}^{G}$ with representative layers $r_i \in \mathcal{G}_i$; stage-specific prompts $\{P^{(i)}\}_{i=1}^{G}$; frozen layers $\{F_l\}_{l=1}^{L}$
Ensure: Final hidden states of the task tokens $X^{(L)} \in \mathbb{R}^{n\times d}$

$Z \leftarrow E_{\text{in}}$ ▷ task-token hidden states, length n
for $i = 1$ to G **do**
  $H \leftarrow Z$
  for $l = b_i$ to e_i **do**
    if $l = r_i$ **then**
      $H \leftarrow \mathrm{Concat}\big(P^{(i)}, H\big)$ ▷ inject the stage prompt; length $n + m$
    end if
    $H \leftarrow F_l(H)$
  end for
  $Z \leftarrow \Pi_{\text{task}}(H) = H_{m+1:m+n}$ ▷ remove $P^{(i)}$; length back to n
end for
$X^{(L)} \leftarrow Z$
return $X^{(L)}$

across stages (Appendix A.5). Unlike input-level prompt tuning, which relies on a single prompt at the input, and layer-wise methods, which attach prompts to every layer, DHAFT introduces a new prompt only where a new representation stage begins and removes it where that stage ends. The task-token hidden states after the last stage are used for generation.

Training (Algorithm 3). All parameters of the pretrained backbone, including the LM head, are frozen, and only the stage-specific prompts $\{P^{(i)}\}$ are optimized. The objective is the standard autoregressive cross-entropy loss under teacher forcing. Gradients are backpropagated through the frozen backbone to the prompts, which are updated with AdamW. Stage partitioning is performed once before training, and the derived stages and representative layers are fixed during training and inference.

Algorithm 3 Training DHAFT with Teacher Forcing

Require: Training set $\mathcal{D}$; frozen LLM $\mathcal{M}$; stages $\{\mathcal{G}_i\}_{i=1}^{G}$ and representative layers $\{r_i\}_{i=1}^{G}$; trainable prompts $\{P^{(i)}\}_{i=1}^{G}$; optimizer $\mathcal{O}$
Ensure: Optimized prompts $\{P^{(i)}\}_{i=1}^{G}$
1: Freeze all parameters of $\mathcal{M}$, including the LM head
2: **for** each minibatch $(x, y) \in \mathcal{D}$ **do**
3: $E_{\text{in}} \leftarrow \text{Embed}(x)$
4: $X^{(L)} \leftarrow \text{DHAFT-FORWARD}\big(E_{\text{in}}, \{\mathcal{G}_i\}, \{r_i\}, \{P^{(i)}\}, \mathcal{M}\big)$ ▷ Algorithm 2
5: $\mathcal{L}_{\text{CE}} \leftarrow -\sum_{t=1}^{T} \log p_\theta\big(y_t \mid y_{<t}, X^{(L)}\big)$
6: Update $\{P^{(i)}\}$ with $\mathcal{O}$ using $\nabla_{\{P^{(i)}\}} \mathcal{L}_{\text{CE}}$
7: **end for**
8: **return** $\{P^{(i)}\}_{i=1}^{G}$

A.3 Centered Kernel Alignment and Its Invariance

Definition. Let $X, Y \in \mathbb{R}^{N\times d}$ be the representations of two layers on N calibration tokens, and let $H = I_N - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$ be the centering matrix. With linear kernels $K_X = XX^\top$ and $K_Y = YY^\top$, the Hilbert–Schmidt Independence Criterion is $\text{HSIC}(K_X, K_Y) = \frac{1}{(N-1)^2}\,\text{tr}(K_X H K_Y H)$, and

$$\text{CKA}(X, Y) = \frac{\text{HSIC}(K_X, K_Y)}{\sqrt{\text{HSIC}(K_X, K_X)\,\text{HSIC}(K_Y, K_Y)}}. \tag{10}$$

Writing $\bar{X} = HX$ and $\bar{Y} = HY$ for the column-centered representations, this equals

$$\text{CKA}(X, Y) = \frac{\|\bar{Y}^\top \bar{X}\|_F^2}{\|\bar{X}^\top \bar{X}\|_F\,\|\bar{Y}^\top \bar{Y}\|_F} \in [0, 1]. \tag{11}$$

Proposition 1 (Invariance). For any orthogonal $Q \in \mathbb{R}^{d\times d}$ and any scalar $c \neq 0$, $\text{CKA}(cXQ, Y) = \text{CKA}(X, Y)$.

Proof. Since $H(cXQ) = c\bar{X}Q$, we have $K_{cXQ} = c^2 XQQ^\top X^\top = c^2 K_X$. The numerator of the definition scales as c^2, and the denominator as $\sqrt{c^4} = c^2$, so the ratio is unchanged. □

Remark. Cosine similarity between flattened representations, $\langle \text{vec}(X), \text{vec}(Y)\rangle / (\|X\|_F \|Y\|_F)$, is invariant to scaling but not to orthogonal transformations of either layer, and Euclidean distance $\|X - Y\|_F$ is invariant to neither. Since the feature basis of hidden states can rotate and rescale across depth, CKA compares the geometry of two representation spaces rather than their coordinates, which is consistent with the gap between CKA-based and cosine- or Euclidean-based partitioning in Table 3.

A.4 Derivation and Properties of the Prompt-Stage Misalignment Index

This section derives the cross-stage leakage term of PSMI, defines the within-stage fluctuation term, and summarizes the properties of the resulting index.

Setup and Notation. For a prompt assigned to representation stage $\mathcal{G}_i$, let $I_i^l \geq 0$ denote the saliency associated with the prompt at layer $l \in \{1, \ldots, L\}$, with $\sum_{j=1}^{L} I_i^j > 0$. We normalize it into a distribution over layers,

$$q_l^{(i)} = \frac{I_i^l}{\sum_{j=1}^{L} I_i^j}, \tag{12}$$

so that $q^{(i)} \in \Delta^{L-1}$, the probability simplex over L layers; in implementation, a small constant (10^{-8}) is added to the denominator for numerical stability. For single-prompt injection, $\mathcal{G}_i$ is the stage containing the injection layer. Let $\{\mathcal{G}_1, \ldots, \mathcal{G}_G\}$ denote the stage partition obtained by CKA-based hierarchical clustering. A distribution fully aligned with stage $\mathcal{G}_i$ places no mass outside it; we denote the set of such distributions by

$$\mathcal{A}_i = \left\{p \in \Delta^{L-1} \mid p_l = 0,\ \forall l \notin \mathcal{G}_i\right\}. \tag{13}$$

Cross-Stage Leakage. We measure how far $q^{(i)}$ is from $\mathcal{A}_i$ in L_1 distance,

$$d(q^{(i)}, \mathcal{A}_i) = \min_{p \in \mathcal{A}_i} \|q^{(i)} - p\|_1. \tag{14}$$

Let $Q_i^{\text{out}} = \sum_{l \notin \mathcal{G}_i} q_l^{(i)}$ and $Q_i^{\text{in}} = \sum_{l \in \mathcal{G}_i} q_l^{(i)} = 1 - Q_i^{\text{out}}$. For any $p \in \mathcal{A}_i$,

$$\|q^{(i)} - p\|_1 = Q_i^{\text{out}} + \sum_{l \in \mathcal{G}_i} \left| q_l^{(i)} - p_l \right| \;\geq\; Q_i^{\text{out}} + \left| \sum_{l \in \mathcal{G}_i} \left(p_l - q_l^{(i)} \right) \right| = Q_i^{\text{out}} + \left(1 - Q_i^{\text{in}}\right) = 2Q_i^{\text{out}}, \tag{15}$$

where the inequality follows from the triangle inequality and the last step uses $\sum_{l \in \mathcal{G}_i} p_l = 1$. Equality holds whenever $p_l \geq q_l^{(i)}$ for all $l \in \mathcal{G}_i$; for $Q_i^{\text{in}} > 0$, one such minimizer is the renormalized in-stage distribution $p_l^\star = q_l^{(i)} / Q_i^{\text{in}}$ for $l \in \mathcal{G}_i$ and $p_l^\star = 0$ otherwise, and for $Q_i^{\text{in}} = 0$ every $p \in \mathcal{A}_i$ attains the bound. Hence $d(q^{(i)}, \mathcal{A}_i) = 2Q_i^{\text{out}}$. Dropping the constant factor, we define the cross-stage leakage as the out-of-stage mass,

$$\text{CSL}(\mathcal{G}_i) = \sum_{l \notin \mathcal{G}_i} q_l^{(i)} = \tfrac{1}{2}\, d(q^{(i)}, \mathcal{A}_i) \in [0, 1], \tag{16}$$

which equals the minimum total-variation distance between $q^{(i)}$ and the set of stage-aligned distributions.

Within-Stage Fluctuation. Leakage does not capture how prompt influence varies inside the assigned stage. Let $\mathcal{E}_i = \{(l, l+1) \mid l, l+1 \in \mathcal{G}_i\}$ denote the adjacent layer pairs within $\mathcal{G}_i$. We define the within-stage fluctuation as the average absolute change between adjacent layers in the stage,

$$\text{WSF}(\mathcal{G}_i) = \frac{1}{\max(1, |\mathcal{E}_i|)} \sum_{(l, l+1) \in \mathcal{E}_i} \left| q_{l+1}^{(i)} - q_l^{(i)} \right|, \tag{17}$$

which is zero when $|\mathcal{E}_i| = 0$.

Final Form. The stage-wise score is $\text{PSMI}(\mathcal{G}_i) = \text{CSL}(\mathcal{G}_i) + \alpha\, \text{WSF}(\mathcal{G}_i)$, and PSMI averages it over the set $\mathcal{S}$ of prompted stages:

$$\text{PSMI} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left[\sum_{l \notin \mathcal{G}_i} q_l^{(i)} + \alpha \frac{1}{\max(1, |\mathcal{E}_i|)} \sum_{(l, l+1) \in \mathcal{E}_i} \left| q_{l+1}^{(i)} - q_l^{(i)} \right| \right], \tag{18}$$

which is Eq. equation 2 in the main text. We use $\alpha = 1$ in all experiments.

Properties. *(i) Zero.* $\text{PSMI}(\mathcal{G}_i) = 0$ if and only if all prompt influence lies within $\mathcal{G}_i$ and $q_l^{(i)} = q_{l+1}^{(i)}$ for every adjacent pair $(l, l+1) \in \mathcal{E}_i$; for a single-layer stage, the second condition holds trivially. *(ii) Range of each term.* The leakage term satisfies $\text{CSL}(\mathcal{G}_i) \in [0, 1]$. For nonnegative $q_l^{(i)}$, $\sum_{(l, l+1) \in \mathcal{E}_i} |q_{l+1}^{(i)} - q_l^{(i)}| \leq 2Q_i^{\text{in}}$, so $\text{WSF}(\mathcal{G}_i) \leq 2Q_i^{\text{in}} / |\mathcal{E}_i|$ for $|\mathcal{E}_i| \geq 1$. *(iii) Relative scale.* By (ii), the within-stage term is typically much smaller than the leakage term; with $\alpha = 1$, PSMI is dominated by cross-stage leakage, and the within-stage term distinguishes placements with similar leakage. *(iv) Coupling with in-stage mass.* Because $q^{(i)}$ is normalized over all layers, the within-stage term scales with Q_i^{in}. *(v) Dependence on stage size.* For a uniform distribution $q_l^{(i)} = 1/L$, $\text{CSL}(\mathcal{G}_i) = 1 - |\mathcal{G}_i|/L$; PSMI is therefore used to compare prompt placements under a fixed stage partition, as in Figure 2(f)–(g), rather than to compare partitions of different granularity. *(vi) Non-contiguous stages.* Since $\mathcal{E}_i$ contains only adjacent pairs that both belong to $\mathcal{G}_i$, the definition applies unchanged when a stage is not contiguous along depth.

Relation to Total Variation. The total variation of the full trajectory, $\text{TV}(q^{(i)}) = \sum_{l=1}^{L-1} |q_{l+1}^{(i)} - q_l^{(i)}|$, measures fluctuation across all layers without reference to stages. When the partition consists of a single contiguous stage containing all layers ($G = 1$), the leakage term vanishes and PSMI coincides with $\alpha\, \text{TV}(q^{(i)})/(L-1)$. For $G > 1$, PSMI separately accounts for the influence outside the assigned stage and the variation inside it.

Remark. PSMI summarizes how prompt influence is distributed relative to a given stage partition: a higher value indicates more influence outside the assigned stage or larger variation within it, and a lower value indicates influence that is more concentrated and more uniform within the stage.

A.5 Formal Details of Prompt Injection and Removal

This section describes how stage-specific prompts are inserted into and removed from the forward pass of DHAFT, and characterizes the resulting sequence length.

Setup and Notation. Let the task sequence contain n tokens and let d be the hidden dimension. We describe the procedure for stages that are contiguous along depth, writing $\mathcal{G}_i = \{b_i, \dots, e_i\}$ with $b_1 = 1$, $e_G = L$, and $b_{i+1} = e_i + 1$. Each stage has a representative layer $r_i \in \mathcal{G}_i$ and a stage-specific prompt $P^{(i)} \in \mathbb{R}^{m \times d}$. Let F_l denote the l-th frozen Transformer layer applied to a sequence of any length, and let $F_{a:c} = F_c \circ \cdots \circ F_a$ for $a \le c$, with $F_{a:c}$ the identity when $a > c$.

Forward Pass. Let $Z_i \in \mathbb{R}^{n \times d}$ denote the task-token hidden states entering stage $\mathcal{G}_i$, with $Z_1 = E_{\text{in}}$. Within stage $\mathcal{G}_i$, the layers before the representative layer process the task tokens alone:

$$X_{\text{task}}^{(r_i)} = F_{b_i:r_i-1}\big(Z_i\big) \in \mathbb{R}^{n \times d}. \tag{19}$$

At the representative layer, the stage prompt is prepended along the sequence dimension,

$$\widetilde{X}^{(r_i)} = \text{Concat}\big(P^{(i)}, X_{\text{task}}^{(r_i)}\big) \in \mathbb{R}^{(n+m) \times d}, \tag{20}$$

and the augmented sequence is processed by the remaining layers of the stage:

$$\widetilde{H}^{(i)} = F_{r_i:e_i}\big(\widetilde{X}^{(r_i)}\big) \in \mathbb{R}^{(n+m) \times d}. \tag{21}$$

When $r_i = b_i$, Eq. equation 19 is the identity and the prompt is active throughout the stage.

Prompt Removal. The first m positions of $\widetilde{H}^{(i)}$ correspond to prompt tokens and the remaining n to task tokens. We define the task-token extraction operator

$$\Pi_{\text{task}}\big(\widetilde{H}^{(i)}\big) = \widetilde{H}^{(i)}_{m+1:m+n} \in \mathbb{R}^{n \times d}, \tag{22}$$

and pass only the task-token hidden states to the next stage:

$$Z_{i+1} = \Pi_{\text{task}}\big(\widetilde{H}^{(i)}\big). \tag{23}$$

The final hidden states used for generation are $X^{(L)} = Z_{G+1}$.

Sequence Length. Within stage $\mathcal{G}_i$, layers $b_i, \dots, r_i - 1$ process sequences of length n and layers $r_i, \dots, e_i$ process sequences of length $n + m$. Since Z_{i+1} has length n regardless of i, every layer of the model processes a sequence of length at most $n + m$, independent of the number of stages G. If previously injected prompts were instead retained, the sequence length in stage $\mathcal{G}_i$ would be $n + im$, reaching $n + Gm$ in the last stage.

Scope of Each Prompt. The tokens of $P^{(i)}$ participate in attention only at layers $r_i, \dots, e_i$; layers in subsequent stages do not attend to them. The effect of $P^{(i)}$ on later stages is therefore carried only through the task-token hidden states Z_{i+1}, which later stages process together with their own prompts. This differs from input-level and selective injection, where a prompt, once injected, remains directly attended to by all subsequent layers.

A.6 Notation

A.7 More Experimental Setup

Evaluation Protocol. To ensure a fair and reproducible evaluation, we adopt a strict separation between model optimization and final evaluation. For all benchmarks, training or validation data are used for prompt optimization, hyperparameter selection, and stage discovery, while the test split is held out and accessed only after the final configuration is fixed. No test labels are used for CKA-based stage partitioning, prompt updates, hyperparameter tuning, or model selection. All models and datasets used in this work are publicly available for research purposes, and we follow their official licenses and usage terms.

Table 5: Summary of notation.

Symbol	Meaning
Backbone and prompts	
L	Number of Transformer layers
d	Hidden dimension
n	Number of task tokens
m	Number of prompt tokens per stage
F_l	The l-th frozen Transformer layer
$E_{\text{in}} \in \mathbb{R}^{n\times d}$	Input token embeddings, $X^{(0)} = E_{\text{in}}$
$X^{(l)} \in \mathbb{R}^{n\times d}$	Hidden states at layer l
$P^{(i)} \in \mathbb{R}^{m\times d}$	Trainable soft prompt of stage $\mathcal{G}_i$
Stage partitioning	
$\mathcal{C}$	Unlabeled calibration set
N	Number of calibration tokens used for CKA
$S \in [0,1]^{L\times L}$	Cross-layer CKA similarity matrix
$\mathcal{T}$	Dendrogram from hierarchical clustering on S
τ	Cut threshold on $\mathcal{T}$
$G(\tau)$	Number of stages obtained at threshold τ
$G^\star$	Number of stages selected by the Silhouette coefficient
B	Number of bootstrap rounds
$\{\mathcal{G}_i\}_{i=1}^{G}$	Partition of layers into G representation stages
$b_i,\ e_i$	First and last layers of stage $\mathcal{G}_i$
r_i	Representative layer of stage $\mathcal{G}_i$
Stage-specific injection	
$Z_i \in \mathbb{R}^{n\times d}$	Task-token hidden states entering stage $\mathcal{G}_i$
$X_{\text{task}}^{(r_i)} \in \mathbb{R}^{n\times d}$	Task-token hidden states at layer r_i
$\widetilde{X}^{(r_i)} \in \mathbb{R}^{(n+m)\times d}$	Prompt-augmented sequence at layer r_i
Π_{task}	Operator that removes prompt tokens and keeps task tokens
Saliency and PSMI	
$A^{h,l}$	Attention matrix of head h at layer l
I_i^l	Layer-l saliency associated with the prompt of stage $\mathcal{G}_i$
$q_l^{(i)}$	Normalized saliency at layer l, $\sum_l q_l^{(i)} = 1$
$\mathcal{S}$	Set of prompted stages
$\mathcal{E}_i$	Adjacent layer pairs within stage $\mathcal{G}_i$
α	Weight of the within-stage term in PSMI
Training	
$\mathcal{D} = \{(x, y)\}$	Training set of inputs x and target sequences y
T	Length of the target sequence
$p_\theta(y_t \mid y_{<t}, X^{(L)})$	Predicted probability of token y_t
$\mathcal{L}_{\text{CE}}$	Autoregressive cross-entropy loss

Commonsense Reasoning. For commonsense reasoning, we follow the experimental protocol of LLM-Adapters (Hu et al., 2023) and DoRA (Liu et al., 2024a). The eight subtasks, including BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, and OBQA, each provide predefined training and test sets. Following prior work, the training sets of all eight subtasks are combined into a single fine-tuning mixture, and evaluation is conducted separately on the predefined test set of each subtask. This protocol allows us to consistently evaluate model responses across all eight commonsense reasoning tasks and ensures direct comparability with prior PEFT studies.

Complex Reasoning. For complex reasoning benchmarks, we use the official training and test splits whenever available. For GSM8K, AQuA, MAWPS, and SVAMP, prompt optimization is performed only on the training or development side, and the corresponding test split is used solely for final evaluation. For MATH, we follow the commonly used Level-5 evaluation setting in prior work (Hong et al., 2025), where evaluation is conducted on 617 Level-5 problems from four representative categories. Optimization follows the same separation protocol and uses only labeled data from the training side, ensuring that no test labels provide feedback during training or configuration selection.

Table 6: **Key hyperparameter settings of compared methods.** We report the main adaptation-related hyperparameters used for each method. * indicates results taken from prior reports under the corresponding published settings.

Method	Rank r	Alpha α	Prompt Length m	Layers / Positions
LoRA / DoRA*	$8/16$	$16/32$	–	All target modules (W_q, W_v)
Prefix-Tuning*	–	–	100	All layers
Prompt Tuning*	–	–	100	Input layer only
LoReFT / DiReFT*	4	–	–	Last 4 layers, prefix/suffix positions $(p = 2, s = 2)$
HiRA*	$16/32$	–	–	Attention + FFN modules
DHAFT (Ours)	–	–	$64/100$	Sparse representative layers discovered by CKA

Fairness and Reproducibility. To ensure a fair comparison, we use the same train/test split for all methods under each benchmark and strictly separate model optimization from final evaluation. Training and validation data are used for prompt optimization, stage discovery, and hyperparameter selection, while test labels are never used during CKA-based partitioning, prompt updates, or model selection. For baseline methods marked with *, we follow the hyperparameter settings and evaluation protocols reported in prior studies, and we explicitly summarize their key adaptation-related configurations in Table 6, including LoRA/DoRA rank and scaling factor, prompt length for prompt-based methods, and intervention positions for ReFT-based methods. For our DHAFT, all hyperparameters are fixed before final evaluation. In particular, the total prompt budget is kept comparable to prompt-based baselines and is distributed across the discovered representative stages rather than assigned independently to every stage. Thus, DHAFT does not gain additional advantage from using more prompt tokens or accessing test-set information. These settings ensure that the comparison focuses on the effectiveness of stage-aligned prompt intervention rather than differences in data usage, test-set tuning, or parameter budget.

A.8 More Experimental Results

Table 7: **Variance analysis of DHAFT on complex reasoning benchmarks.** We report mean accuracy and standard deviation over three independent runs with different random seeds.

Model	GSM8K	MATH	AQuA	MAWPS	SVAMP	Avg.
LLaMA-2 7B	66.1 ± 0.28	33.0 ± 0.21	36.7 ± 0.32	85.5 ± 0.24	55.7 ± 0.30	55.40 ± 0.22
LLaMA-3 8B	74.0 ± 0.24	41.9 ± 0.18	44.7 ± 0.27	70.1 ± 0.22	58.8 ± 0.25	57.90 ± 0.19
Qwen3-8B	72.3 ± 0.31	42.4 ± 0.20	41.7 ± 0.29	74.4 ± 0.26	55.7 ± 0.33	57.30 ± 0.23

To assess the stability of DHAFT, we repeat the complex reasoning experiments with three different random seeds, i.e., 42, 123, and 2024, under the same data split, backbone, and hyperparameter configuration. These seeds affect prompt initialization, calibration sampling for CKA-based stage discovery, and mini-batch ordering during optimization. The results reported in Table 2 are the means over these three runs. As shown in Table 7, the standard deviation of each benchmark remains below 0.35 points on all evaluated backbones. This suggests that the performance of DHAFT is not sensitive to a particular prompt initialization, calibration subset, or mini-batch ordering.

A.9 Limitations and Future Work.

While our ablation results suggest that moderate stage numbers ($3 \leq G \leq 5$) strike a good balance between semantic granularity and parameter efficiency, further exploration is warranted. One promising direction is to make the stage partitioning process *adaptive*, allowing G to vary dynamically across tasks, datasets, or even input instances. Another avenue is to jointly optimize the partitioning and prompt parameters in an end-to-end manner, rather than precomputing the hierarchy. Such adaptive and task-aware extensions could further enhance the generality and robustness of DHAFT.

A.10 BROADER IMPACTS.

DHAFT aims to improve the parameter efficiency and reasoning performance of LLM adaptation. Its positive impact lies in reducing the cost of adapting large models to reasoning tasks, which may make LLM-based reasoning systems more accessible to researchers and practitioners with limited computational resources. However, more efficient adaptation may also lower the barrier to deploying LLMs in sensitive applications, where incorrect reasoning outputs could lead to unreliable decisions. Therefore, DHAFT should be used with appropriate evaluation, human oversight, and task-specific safety considerations when applied to high-stakes domains.

A.11 USE OF AI ASSISTANCE.

The authors used LLM assistance only for language polishing, including grammar correction, spelling refinement, word choice, and clarity improvement. All scientific claims, experimental results, citations, methodological designs, tables, formulas, code, scripts, figures, LaTeX source, and the final manuscript were produced, verified, edited, and approved by the authors. LLM assistance was not used as an autonomous source of scientific claims, experimental findings, citations, methodological decisions, or final research conclusions.