Freeze Deep, Train Shallow: Interpretable Layer Allocation for Continued Pre-Training

Anonymous ACL submission

Abstract

Selective layer-wise updates are essential for low-cost continued pre-training of Large Language Models (LLMs), yet determining which layers to freeze or train remains an empirical black-box problem due to the lack of interpretable guidance. To address this issue, we propose LayerTracer, an architecture-agnostic diagnostic framework that reveals the evolution patterns of layer-wise representations and stability by locating task execution positions and quantifying layer sensitivity. Analysis results reveal that deep layers act as critical regions for task execution and maintain high stability against disruptive updates. Guided by this finding, we conduct three controlled continued pre-training trials to compare diverse freeze-train strategies, demonstrating that training shallow layers while freezing deep layers consistently outperforms full-parameter fine-tuning and the opposite allocation on both C-Eval and CMMLU benchmarks. We further present a hybrid model case study, which validates that placing high-quality pre-trained modules in deep layers effectively preserves inherent knowledge of the model. This work delivers a low-cost and interpretable solution for resource-constrained teams, offering actionable guidance for layer-wise parameter allocation in continued pre-training and hybrid model construction. Our code is publicly available at `https://anonymous.4open.science/r/LayerTracer-E41E`.

1 Introduction

With the rapid development of Large Language Models (LLMs), leading general-purpose models such as GPT (OpenAI, 2023), LLaMA (Touvron et al., 2023b,a; Dubey et al., 2024), and Qwen (Team, 2024b,a; Yang et al., 2025a) have achieved significant breakthroughs in semantic understanding, logical reasoning, and generation,

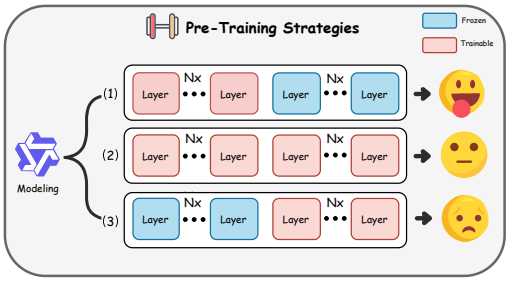


Figure 1: Comparison of three layer-wise pre-training strategies on the CEval dataset (Huang et al., 2023) using the Qwen3-0.6B architecture. (1) (Freeze-Deep/Train-Shallow) achieves the best performance, followed by (2) (Full Training), while (3) (Freeze-Shallow/Train-Deep) yields the worst results.

thereby accelerating industrial adoption and engineering deployment (Zhao et al., 2023; Liu et al., 2023; Abo El-Enen et al., 2025; Wu et al., 2025). However, the superior performance of these models relies heavily on massive high-quality pretraining corpora and large-scale computing clusters, creating substantial technical barriers (Bai et al., 2023; Touvron et al., 2023b). For small and medium-sized teams with limited computational power and data reserves, pretraining general-purpose models from scratch is largely infeasible. Consequently, a mainstream paradigm for vertical domain modeling has emerged. This approach leverages the stacked decoder layers of open-source pretrained models by freezing a portion of the decoder layers, while only training the remaining native layers as well as newly added custom layers. Combined with high-quality domain-low-budget continued pre-training data for domain adaptation, this strategy enables the rapid deployment of commercially viable vertical domain models at minimal computational and data costs (Bao et al., 2023; Chen et al., 2023b; Roziere et al., 2023; Chen et al., 2023a; Labrak et al., 2024).

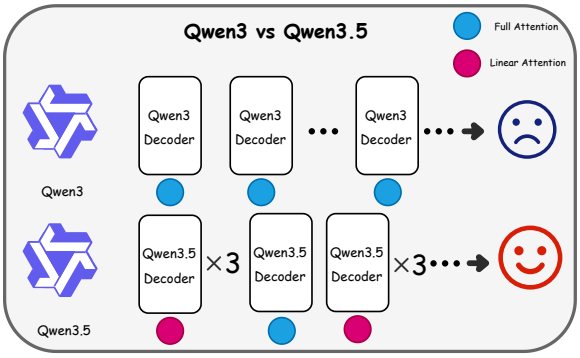


Figure 2: The architectures of Qwen3 model and Qwen3.5 model. Qwen3 adopts a single architecture, while Qwen3.5 is a hybrid architecture with a 3:1 ratio of Linear Attention to Full Attention.

Although this approach has established a standardized pipeline, it suffers from a critical limitation: the lack of interpretable guidance for layer-wise freeze–train allocation during continued pre-training. Thus, small and medium-sized teams are forced to rely on empirical heuristics when determining which layers to freeze or train, leaving the allocation process largely opaque. This reliance on trial-and-error often leads to undesirable overwriting of pre-trained knowledge and performance fluctuations, while substantially increasing computational costs. As illustrated in Figure 1, the specific arrangement of frozen and trainable layers critically dictates convergence stability and final performance. Therefore, uncovering functional differentiation patterns across layers and deriving actionable freeze-train rules emerges as a critical imperative. Specifically, we aim to address the following two research questions:

- *RQ1: What are the functional differentiation patterns across layers in pretrained models?*
- *RQ2: How can these patterns inform practical layer-wise freeze/train allocation strategies for continued pre-training?*

Nevertheless, existing research offers limited guidance for layer-wise parameter allocation in continued pre-training. Although prior layer probing studies (Belinkov, 2022b; Gurnee et al., 2023; Ju et al., 2024; Xiao et al., 2025; Eisenstadt et al., 2025; Zhang et al., 2026) and parameter-efficient tuning methods (Bao et al., 2023; Liu et al., 2023) have yielded valuable insights into internal model dynamics and fine-tuning strategies, they mainly focus on interpreting model behaviors or task-specific fine-tuning, rather than producing actionable rules for layer freeze-train decisions. More importantly, few existing efforts develop quantitative paradigms to characterize intrinsic layer-wise properties, resulting in a lack of interpretable principles to address the core research questions *RQ1* and *RQ2*.

To address these issues, we propose LayerTracer, a diagnostic framework for layer-wise freeze/train allocation in continued pre-training. It quantifies two core metrics: *Task Particle (TP)* identifies the layers where task probability undergoes meaningful relative shifts, marking where task evidence actively consolidates. *Layer-wise Sensitivity (LS)* measures the relative change in Jensen-Shannon divergence (Lin, 1991) across consecutive layers under controlled perturbation, capturing zones Sensitive to disruptive updates. Across the Qwen3 series, we observe a consistent finding: shallow layers exhibit higher sensitivity to perturbations, while deep layers consolidate task evidence and stabilize execution. Based on this finding, we derive a practical allocation rule: freeze deep pre-trained layers and train shallow ones. We validate this rule through three controlled continued pre-training experiments, complemented by a hybrid architecture case study to simulate its practical value in resource-constrained industrial scenarios. Results show that Our train-shallow/freeze-deep strategy achieves a notable relative improvement over the reverse allocation, with an average gain of 15.72% across C-Eval (Huang et al., 2023) and CMMLU (Li et al., 2024).

2 Related Work

2.1 Evolution of LLM Architectures

The Transformer (Vaswani et al., 2017) remains the dominant backbone across modern foundation models (OpenAI, 2023; Touvron et al., 2023a,b; Dubey et al., 2024; Yang et al., 2025a). To overcome the scalability bottleneck of self-attention, efficient alternatives such as State Space Models (Gu and Dao, 2024; Dao and Gu, 2024), Linear Attention (Ahn et al., 2024), and GatedDeltaNet (Yang et al., 2025b) have been proposed, giving rise to hybrid systems like Jamba (Lieber et al., 2024) and Qwen3.5 (Team, 2025) that interleave full-attention and efficient layers as illustrated in Figure 2. Such designs enable low-cost domain adaptation by freezing pre-trained foundation layers while tuning lightweight modules (Bao et al., 2023; Chen et al., 2023b; Roziere et al., 2023; Chen et al.,

2023a; Labrak et al., 2024). Nevertheless, layer-wise freeze–train decisions remain purely heuristic. Without quantitative and interpretable guidance, improper allocation easily breaks knowledge coherence and degrades robustness, especially in fragile hybrid architectures.

2.2 Layer-Wise Representation Analysis

Layer-wise interpretability has evolved from early linguistic probing (Pimentel et al., 2020; Belinkov, 2022a; Youssef et al., 2023) to advanced mechanistic analysis that locates task-critical components and knowledge boundaries (Gurnee et al., 2023; Meng et al., 2022; Geva et al., 2021; Ju et al., 2024; Xiao et al., 2025). Closely related are logit/tuned lens and activation patching and causal tracing techniques (Meng et al., 2023; Hernandez et al., 2024; Liu et al., 2024), which project intermediate states through the LM head or intervene on activations to reveal internal dynamics. However, these works focus on post-hoc interpretation rather than actionable guidance. They lack quantitative metrics for layer robustness and cannot be directly generalized to hybrid models. No prior framework unifies task localization and sensitivity measurement for continued pre-training.

2.3 Parameter-Efficient Adaptation and Allocation

Parameter-efficient tuning methods such as LoRA (Hu et al., 2021), AdaLoRA (Zhang et al., 2023), and DCFT (Zhang et al., 2025) optimize layer-wise allocation for fine-tuning, with extensions to instruction tuning and alignment via RLHF (Schulman et al., 2017; Rafailov et al., 2023; Guo et al., 2025). Yet these approaches rely on task-specific gradients or labeled data, making them unsuitable for unsupervised continued pre-training on raw text. Existing strategies also ignore intrinsic layer stability and fail to generalize to hybrid architectures. This creates a critical need for a gradient-free, interpretable diagnostic to guide principled layer allocation.

3 Method

To answer *RQ1* and *RQ2* raised in this paper, this section proposes a hierarchical layer analysis framework named LayerTracer. The overview of the LayerTracer framework is illustrated in Figure 3.

3.1 Preliminary

We first define unified notations for layer-wise analysis. The input follows a composite task structure $t = s_1 \oplus s_2$, where s_1 denotes context examples and s_2 denotes the query input. Given such structured input, the model outputs a series of hidden states $\{h_1, h_2, \ldots, h_N\}$, where h_l represents the hidden state at the l-th layer. All layer-wise hidden states are projected to the vocabulary space via the shared final-layer LM head, ensuring consistent distribution calibration and controlled variables across layers.

The final output probability distribution P is obtained by projecting the last-layer hidden state h_N through the shared LM head. We define the target token t^* as the token with the maximum probability in P. This token is determined by the model's own final prediction rather than external labels, and is used to trace how the model's preferred answer emerges across layers. Let $P_t(l)$ denote the probability of t^* derived from the hidden state h_l projected by the shared LM head. For brevity, we refer to $P_t(l)$ as task probability in subsequent analysis, and the collection of layers with meaningful $P_t(l)$ shifts as task evidence consolidation zones. For a masking perturbation applied at layer l, we mask only the context s_1 while keeping the query s_2 unchanged and use $Q(l)$ to denote the final output distribution after perturbation. The Jensen-Shannon divergence between two distributions P and Q is denoted as $\mathrm{JS}(P \parallel Q)$.

3.2 Task Particle

The *Task Particle (TP)* is designed to locate the critical layer where the model initiates task execution. Following the notation in section 3.1, for each layer l, we compute the probability of the target token t^* and measure its relative increase ratio between consecutive layers:

$$\mathrm{Ratio}(l) = \left| \frac{P_t(l) - P_t(l-1)}{P_t(l) + \epsilon} \right|, \qquad (1)$$

where $\epsilon = 10^{-6}$ ensures numerical stability, and $P_t(l-1)$ denotes the task probability from the preceding layer. The Task Particle is formally defined as the set of layers where $\mathrm{Ratio}(l) > 0$, forming a continuous task-execution interval. Unlike single-point localization methods, this interval-based view captures all depths where task evidence undergoes meaningful relative shifts, encompassing both task probability promotion and suppression. This continuous zone marks the region where the model

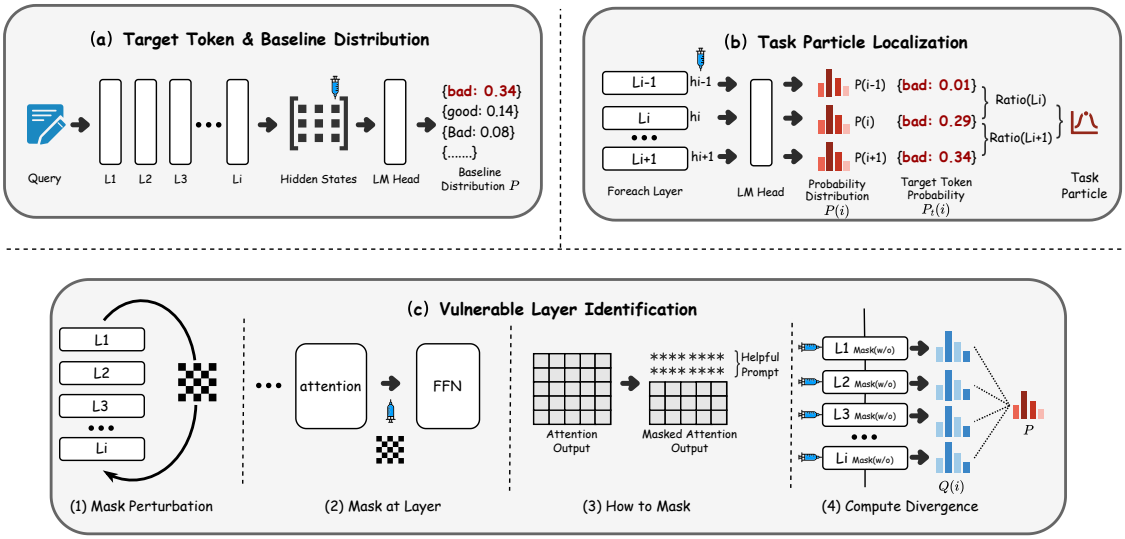


Figure 3: Overview of the LayerTracer framework. (a) Baseline projection: hidden states at each layer are projected via the shared LM head to obtain the target token probability $P_t(l)$, with t^* selected from the final distribution P. (b) Task Particle: we compute the relative probability shift $\mathrm{Ratio}(l)$ across consecutive layers. Layers satisfying $\mathrm{Ratio}(l) > 0$ form a execution interval, marking depths where task evidence actively consolidates. (c) Layer-wise Sensitivity: we apply context-targeted masking perturbations at layer l and compute the relative fluctuation $\Delta\mathrm{JS}(l)$ of JS divergence across adjacent layers, identifying zones highly sensitive to parameter updates and information flow disruptions.

actively consolidates and executes task-relevant representations.

3.3 Layer-wise Sensitivity

To evaluate layer-wise stability under perturbations, we propose the Layer-wise Sensitivity (LS) metric. Different from layer-level evaluation that directly adopts raw JS divergence, LS focuses on the relative variation trend across consecutive layers, thereby capturing abrupt changes in the stability of context information flow.

Specifically, let $\mathcal{I}_c$ and $\mathcal{I}_q$ denote the token indices of the context examples s_1 and the query s_2, respectively. For a perturbation applied at layer l, we intervene on the residual hidden states after the l-th Transformer block by masking only the context-token positions:

$$\tilde{h}_{l,i} = \begin{cases} \mathbf{0}, & i \in \mathcal{I}_c, \\ h_{l,i}, & i \in \mathcal{I}_q. \end{cases} \tag{2}$$

The perturbed hidden states $\tilde{H}_l$ are then fed into the remaining layers $l+1, \ldots, N$ to obtain the final output distribution $Q(l)$.

Given the original intact output distribution P and the perturbed distribution $Q(l)$ at layer l, we define the JS divergence at layer l as:

$$\mathrm{JS}(l) = \mathrm{JS}\big(P \parallel Q(l)\big). \tag{3}$$

The standard Jensen-Shannon divergence is formulated as:

$$\begin{aligned} \mathrm{JS}\big(P \parallel Q(l)\big) = \frac{1}{2}\mathrm{KL}\big(P \parallel M(l)\big) \\ + \frac{1}{2}\mathrm{KL}\big(Q(l) \parallel M(l)\big), \end{aligned} \tag{4}$$

where $M(l) = \frac{1}{2}\big(P + Q(l)\big)$ represents the average mixed distribution, and $\mathrm{KL}(\cdot \parallel \cdot)$ denotes the Kullback-Leibler divergence:

$$\mathrm{KL}(X \parallel Y) = \sum_i X(i) \log \frac{X(i)}{Y(i)}. \tag{5}$$

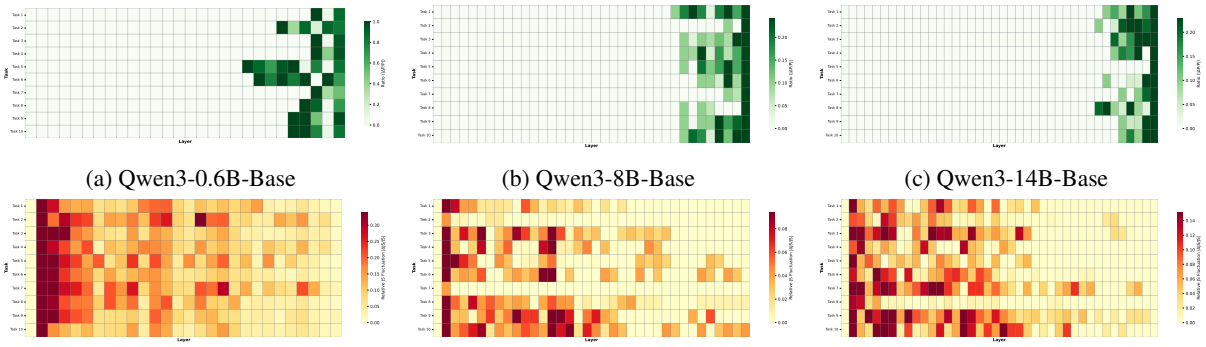
Based on the layer-level JS divergence sequence, we further compute the relative fluctuation between adjacent layers:

$$\Delta\mathrm{JS}(l) = \left| \frac{\mathrm{JS}(l) - \mathrm{JS}(l-1)}{\mathrm{JS}(l-1) + \epsilon} \right|, \tag{6}$$

A large $\Delta\mathrm{JS}(l)$ value indicates an abrupt shift in layer sensitivity, meaning the stability of information flow changes significantly at this layer. This metric effectively identifies critical layers that are sensitive to parameter updates.

4 Experiment

To empirically validate the diagnostic utility of LayerTracer and derive actionable freeze-train rules,



(a) Qwen3-0.6B-Base (b) Qwen3-8B-Base (c) Qwen3-14B-Base

(d) Qwen3-0.6B-Base (e) Qwen3-8B-Base (f) Qwen3-14B-Base

Figure 4: Distribution details in the AntSynNET dataset, where 1000 samples are evenly divided into 10 groups of 100 samples, denoted as Task 1–Task 10 for visualization. (a)–(c) present the variation of Ratio across different layers for Qwen3-0.6B-Base to Qwen3-14B-Base. A deeper green color indicates a larger Ratio, meaning that the target token probability changes more significantly at this layer, marking a key position for task execution. (d)–(f) present the variation of $\Delta\mathrm{JS}$ across different layers, which measures the relative fluctuation of JS divergence between adjacent layers. A deeper color indicates a larger abrupt change in layer sensitivity, reflecting higher vulnerability and lower robustness.

we conduct comprehensive layer-wise analysis across Qwen3 variants. Figure 4 illustrates the distribution of Task Particle and Layer-wise Sensitivity.

4.1 Experimental Settings

Dataset. We use a composite dataset constructed upon AntSynNET (Nguyen et al., 2017), following four standard categories for layer-wise reasoning analysis: Knowledge–Algorithmic, Extractive–Knowledge, Extractive–Algorithmic, and Knowledge–Translation. We reorganize each sample into a structured form $t = s_1 \oplus s_2$. This structural separation enables s_1-targeted masking perturbation, which helps distinguish the roles of shallow and deep layers by isolating context influence during layer-wise sensitivity evaluation. For clarity, dataset details are presented in Appendix B.

Experimental Details. All experiments are conducted on a server equipped with 8 NVIDIA H200 141G GPUs.This study uses three Qwen3 variants of 0.6B, 8B, and 14B parameters, as the primary subjects for layer-wise analysis. The experiment aims to characterize the distribution trends of Task Particles and Layer-wise Sensitivity across varying model capacities. For each layer, the hidden state is projected via the language modeling head to obtain the full vocabulary distribution, from which we retain only the top-10 candidate tokens with the highest probabilities. We verify in Appendix C.1 that results remain consistent when extending to top-50, confirming the robustness of our layer-wise patterns.

4.2 Experimental Results

4.2.1 Deep Layers as Execution Zones

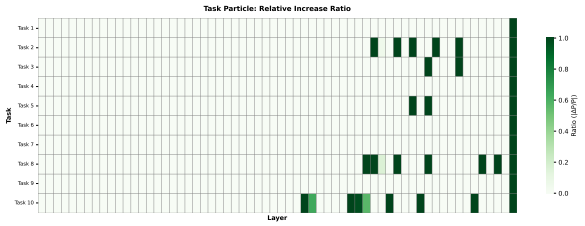
💡 Finding1 ››› *RQ1 :* Deep layers serve as stable execution zones where task evidence is actively consolidated.

Figure 4a–4c visualize the layer-wise distribution of $\mathrm{Ratio}(l)$, the relative change of task probability defined in Equation 1. Following our definition in Section 3.2, layers with $\mathrm{Ratio}(l) > 0$ constitute the Task Particle interval, marking where task evidence undergoes meaningful shifts. We observe that high Ratio values (darker green) concentrate in deep layers across all model scales, forming a stable task-execution zone. This indicates that deep layers serve as the primary region where the model executes task-specific reasoning, while shallow layers show minimal task-related activity.

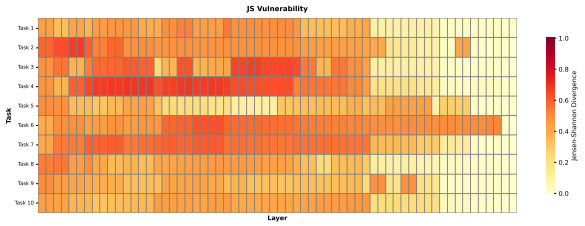
4.2.2 Shallow Layers as Sensitive Zones

💡 Finding2 ››› *RQ1 :* Shallow layers serve as sensitive zones where representations are vulnerable to perturbation.

Figure 4d–4f illustrate $\Delta\mathrm{JS}(l)$, which quantifies the relative change in layer-wise sensitivity under masking perturbations. Based on Equation 6, higher $\Delta\mathrm{JS}$ values (darker red) indicate layers where sensitivity to context perturbations changes abruptly. The results reveal that shallow layers exhibit significantly larger $\Delta\mathrm{JS}$ fluctuations compared to deep layers, which maintain stable sensitivity profiles. This indicates that shallow layers remain highly responsive to parameter updates, mak-



(a) Nemotron-12B-Base



(b) Nemotron-12B-Base

Figure 5: Layer-wise analysis results on Nemotron-12B-Base, a Mamba-based non-Transformer architecture. Consistent with observations on the Qwen3 series, the Task Particle concentrates in deep layers while shallow layers exhibit higher instability, verifying the universality of our findings across diverse model architectures.

ing them natural candidates for adaptation, while deep layers have consolidated stable knowledge representations.

4.2.3 Core Principle: Matching Zones with Strategies

> 💡 Finding3 ››› RQ2 : Training shallow sensitive zones while freezing deep execution zones offers an interpretable freeze-train allocation strategy for continued pre-training.

The above results reveal a consistent shallow-sensitive and deep-stable hierarchy, suggesting a simple layer-wise allocation principle for continued pre-training. We instantiate this principle with a model-midpoint split: the shallow half is treated as sensitive zones, where high ΔJS indicates greater sensitivity to perturbations and thus higher plasticity for absorbing domain-specific changes. In contrast, the deep half is treated as execution zones, where high Ratio values indicate active task-evidence consolidation and stabilized task execution. Since these deep layers carry task-execution functions and inherit high-quality parameters from strong pre-trained models, they should be preserved rather than frequently updated. This midpoint split provides a symmetric parameter partition and a clean controlled setting for validation, instead of exhaustively searching for the optimal boundary.

As illustrated in Figure 6, we therefore match different strategies to different layer zones: shallow sensitive zones are trained to absorb domain-specific patterns, while deep execution zones are frozen to preserve strong pre-trained parameters and avoid disrupting consolidated task evidence. Details as elaborated in C.2.

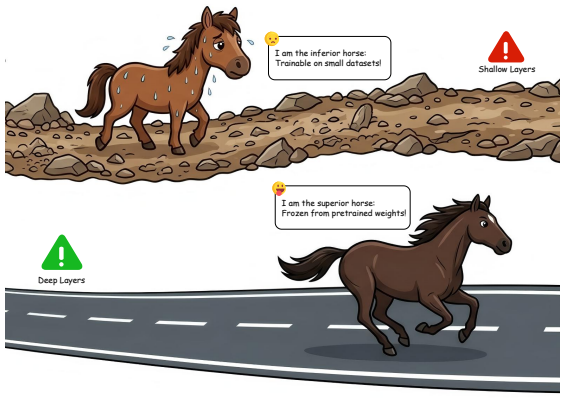


Figure 6: Illustration of the proposed zone-strategy matching principle. Shallow layers are treated as sensitive zones, where active training supports domain-specific updates, while deep layers are treated as execution zones, where freezing preserves pre-trained task evidence.

5 Why is LayerTracer Effective?

Guided by the hierarchical patterns revealed by LayerTracer in Section 4, we further evaluate whether the derived strategy leads to measurable gains in practical continued pre-training settings. While Section 4 focuses on representation dynamics and intrinsic layer properties, this section examines whether the LayerTracer-guided allocation strategy improves downstream task performance.

> 🔬 Results ››› Finding : These results provide empirical evidence for the practical effectiveness of LayerTracer-guided layer allocation.

5.1 Pre-Training and Evaluation Setup

We conduct continued pre-training on Qwen3-0.6B-Base using the high-quality Chinese corpus CCI3.0-HQ (Wang et al., 2024) with 262B tokens, where all training samples are in the form of natural text streams rather than question–answer pairs, and compare three strategies: full-parameter pre-training as the baseline, freezing the shallow module while training the deep module, and training the shallow module while freezing the deep module. Evaluation is performed on two standard Chinese benchmarks, C-Eval (Huang et al., 2023)

Models	CEval				
	STEM	Social Science	Humanities	Other	Average
Qwen3-0.6B-Base (Trainable)	26.60 / 26.55	27.19 / 27.21	26.18 / 25.83	26.53 / 25.67	26.62 / 26.32
Qwen3-0.6B-Base (Frozen/Trainable)	24.86 / 24.97	25.33 / 25.60	25.51 / 25.71	25.85 / 25.92	25.39 / 25.55
Qwen3-0.6B-Base (Trainable/Frozen)	**27.76 / 27.65**	**32.07 / 32.17**	**28.94 / 29.39**	**28.82 / 28.99**	**29.40 / 29.55**

Table 1: Performance evaluation on the C-Eval benchmark (Huang et al., 2023) across three training strategies for Qwen3-0.6B-Base. Each cell reports Generate / Logit scores. The first row denotes the baseline with full-parameter pre-tuning. The second row represents the strategy of freezing shallow layers and training deep layers. The third row illustrates our strategy of training shallow layers and freezing deep layers. Red values indicates performance better than the full-parameter baseline, while green values indicates performance worse than the baseline. **Bold values** denote the best performance across all settings. Standard deviation details over five repeated runs for all experiments are provided in Appendix C.3.

Models	CMMLU				
	STEM	Social Science	Humanities	Other	Average
Qwen3-0.6B-Base (Trainable)	25.22 / 24.99	24.10 / 25.00	25.26 / 25.53	25.85 / 25.54	25.11 / 25.27
Qwen3-0.6B-Base (Frozen/Trainable)	24.28 / 24.14	24.98 / 26.01	25.14 / 25.50	26.30 / 26.80	25.17 / 25.61
Qwen3-0.6B-Base (Trainable/Frozen)	**26.69 / 26.72**	**30.62 / 30.65**	**27.93 / 28.00**	**27.76 / 28.32**	**28.25 / 28.42**

Table 2: Performance evaluation on the CMMLU benchmark (Li et al., 2024) across the same three pre-training strategies. Settings and notations follow Table 1.

Dataset	Tokens
Qwen Private Corpus	36T
CCI3.0-HQ	262B

Table 3: Statistics of pre-training corpora. The Qwen private corpus is a large-scale high-quality proprietary dataset built by the Qwen team, while CCI3.0-HQ (Wang et al., 2024) is a public high-quality Chinese corpus with a smaller scale.

and CMMLU (Li et al., 2024), using two mainstream protocols widely adopted for pre-trained model assessment: logit-based evaluation infers answers directly from model logits to reflect inherent knowledge fitting, while generation-based evaluation judges whether the correct answer is included within a 16-token autoregressive output with `do_sample=False` to characterize generation quality, and consistency between these two metrics indicates reliable model behavior. Detailed settings are summarized in Appendix A.

5.2 Results and Analysis

As shown in Table 1 and Table 2, the train-shallow/freeze-deep strategy consistently achieves the best performance across both benchmarks and both evaluation protocols. On C-Eval, it improves the average score from 26.62% / 26.32% under full-parameter continued pre-training to 29.40% / 29.55%, yielding absolute gains of 2.78% in Generate evaluation and 3.23% in Logit evaluation. In contrast, the reverse strategy, which freezes shallow layers and trains deep layers, drops to 25.39% / 25.55%, falling below the full-parameter baseline.

A similar trend is observed on CMMLU. The train-shallow/freeze-deep strategy reaches 28.25% / 28.42% on average, outperforming the full-parameter baseline by 3.14% and 3.15% under Generate and Logit evaluation, respectively. By comparison, the freeze-shallow/train-deep strategy obtains only 25.17% / 25.61%, again showing clear degradation. These results indicate that updating deep layers can disrupt pre-trained task evidence, while preserving deep execution layers and adapting shallow layers leads to more reliable continued pre-training. More case studies are provied in Appendix D.

6 A Case Study: Industrial Implications

💡 Value >>> Industry: This paper provides an industrial example of building hybrid models through low-cost continued pre-training under resource constraints.

This section examines the industrial value of

	STEM	Social Science	Humanities	Other	Average
Qwen3-Nemotron-Base	17.39 / 24.78	17.95 / 24.65	14.65 / 25.60	15.79 / 26.94	16.45 / 25.49
Nemotron-Qwen3-Base	**20.40 / 29.40**	**19.77 / 34.30**	**20.37 / 26.78**	**20.19 / 28.48**	**20.19 / 29.73**

Table 4: Performance evaluation on the C-Eval benchmark for two hybrid architectures. Nemotron-Qwen3-Base places pre-trained Qwen3 decoder layers in the deep half, while Qwen3-Nemotron-Base places them in the shallow half. Bold values denote the best performance.

Models	CMMLU				
	STEM	Social Science	Humanities	Other	Average
Qwen3-Nemotron-Base	17.25 / 24.59	18.73 / 24.62	15.75 / 24.66	15.83 / 24.67	16.89 / 24.64
Nemotron-Qwen3-Base	**20.79 / 26.87**	**21.76 / 29.94**	**22.21 / 27.95**	**19.43 / 28.43**	**20.80 / 28.30**

Table 5: Performance evaluation on the CMMLU benchmark for the same two hybrid strategies. Settings and notations follow Table 4.

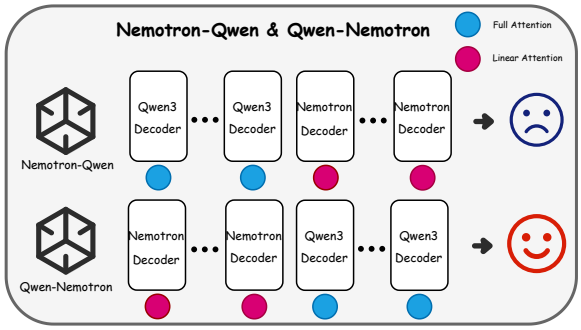


Figure 7: The architectures of Nemotron-Qwen model and Qwen-Nemotron model. They are hybrid architectures with a 1:1 ratio of Full Attention to Linear Attention.

LayerTracer for resource-constrained LLM development. As hybrid architectures become more common, a key challenge is how to place heterogeneous modules while preserving the generalization ability of foundation models. Guided by the LayerTracer principle of preserving deep execution zones while adapting shallow sensitive zones, we provide a practical case study for hybrid model construction. Figure 7 illustrates the hybrid architecture used in this study.

To illustrate its industrial value, we construct two hybrid models, Nemotron-Qwen3-Base and Qwen3-Nemotron-Base, both with 550M parameters. In Nemotron-Qwen3-Base, Mamba2-based Nemotron layers are placed in the first half of the network and actively trained, while Qwen3 pre-trained decoder layers are placed in the second half and frozen to preserve high-quality pre-trained parameters. In Qwen3-Nemotron-Base, the allocation is reversed, placing Qwen3 layers in the first half and Nemotron layers in the second half. We adopt the same pre-training data, benchmarks, and evaluation metrics as Section 5.

Experimental results are reported in Table 4 and Table 5. On C-Eval, placing Qwen3 pre-trained layers in the deep half improves the average score from 16.45% / 25.49% to 20.19% / 29.73%, yielding absolute gains of 3.74 and 4.24 percentage points under Generate and Logit evaluation, respectively. On CMMLU, the same strategy improves the average score from 16.89% / 24.64% to 20.80% / 28.30%, with gains of 3.91 and 3.66 percentage points. The improvements are consistent across STEM, Social Science, Humanities, and Other categories.

7 Conclusion

This paper proposes LayerTracer, an interpretable diagnostic framework for layer-wise freeze-train allocation in continued pre-training. By combining Task Particle and Layer-wise Sensitivity, LayerTracer reveals a consistent hierarchy: shallow layers are more sensitive and suitable for adaptation, while deep layers form stable execution zones where task evidence is consolidated. Based on this finding, we derive a train-shallow/freeze-deep strategy. Experiments show that this strategy outperforms both full-parameter continued pre-training and the reverse allocation, while the hybrid architecture case study further demonstrates the value of preserving high-quality pre-trained parameters in deep layers. Overall, LayerTracer provides practical and low-cost guidance for efficient model adaptation and hybrid model construction.

Limitations

LayerTracer is designed as an interpretable diagnostic framework for continued pre-training, and its main limitation is computational scale rather than methodological scope. Due to limited resources, we validate it on mainstream model sizes instead of extending the analysis to much larger parameter scales. Even so, our experiments already cover comparatively large continued pre-training settings, and the results are fully reproducible, stable, and consistent across benchmarks. We believe the framework provides practical and effective guidance for layer-wise allocation, while larger-scale validation remains an important direction for future work.

Ethical Statement

This work focuses on fundamental interpretability and efficient adaptation techniques for large language models, with no involvement of unethical data collection, harmful content generation, or biased application scenarios. All experiments comply with standard AI research ethics and model usage norms. The proposed LayerTracer framework aims to improve training efficiency and stability while reducing computational costs, which supports green and responsible development of large models.

References

Mohamed Abo El-Enen, Sally Saad, and Taymoor Nazmy. 2025. A survey on retrieval-augmentation generation (rag) models for healthcare applications. *Neural Computing and Applications*, 37(33):28191–28267.

K. Ahn, X. Cheng, M. Song, C. Yun, A. Jadbabaie, and S. Sra. 2024. Linear attention is (maybe) all you need (to understand transformer optimization). In *International Conference on Learning Representations (ICLR)*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Zhijie Bao, Wei Chen, Shengze Xiao, Kuang Ren, Jiaao Wu, Cheng Zhong, Jiajie Peng, Xuanjing Huang, and Zhongyu Wei. 2023. Disc-medllm: Bridging general large language models and real-world medical consultation. *arXiv preprint arXiv:2308.14346*.

Y. Belinkov. 2022a. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Yonatan Belinkov. 2022b. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Junying Chen, Xidong Wang, Ke Ji, Anningzhe Gao, Feng Jiang, Shunian Chen, Hongbo Zhang, Dingjie Song, Wenya Xie, Chuyi Kong, et al. 2023a. Huatuogpt-ii, one-stage training for medical adaption of llms. *arXiv preprint arXiv:2311.09774*.

Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, et al. 2023b. Meditron-70b: Scaling medical pretraining for large language models. *arXiv preprint arXiv:2311.16079*.

T. Dao and A. Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*.

A. Dubey, A. Jauhri, A. Pandey, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Roy Eisenstadt, Itamar Zimerman, and Lior Wolf. 2025. Overclocking llm reasoning: Monitoring and controlling thinking path lengths in llms. *Preprint*, arXiv:2506.07240.

M. Geva, R. Schuster, J. Berant, and O. Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5484–5495.

A. Gu and T. Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *International Conference on Learning Representations (ICLR)*.

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

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. In *Advances in Neural Information Processing Systems*, volume 36.

E. Hernandez, A. S. Sharma, T. Haklay, K. Meng, M. Wattenberg, J. Andreas, Y. Belinkov, and D. Bau. 2024. Linearity of relation decoding in transformer language models. In *International Conference on Learning Representations (ICLR)*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu,

Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*.

Tianjie Ju, Weiwei Sun, Wei Du, Xinwei Yuan, Zhaochun Ren, and Gongshen Liu. 2024. How large language models encode context knowledge? a layer-wise probing study.

Yanis Labrak, Adrien Bazoge, Emmanuel Morin, Pierre-Antoine Gourraud, Mickael Rouvier, and Richard Dufour. 2024. Biomistral: A collection of open-source pretrained large language models for medical domains. In *Findings of the association for computational linguistics: acl 2024*.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2024. CMMLU: Measuring massive multitask language understanding in Chinese. In *Findings of the Association for Computational Linguistics*.

A. Lieber, O. Sharir, B. Lenz, and Y. Shoham. 2024. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.

J. Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.

F. Liu, P. Shi, X. Liu, Y. Zhang, and G. Neubig. 2024. What do probes actually probe? on the role of surface statistics in linguistic probing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 12345–12359.

T. Liu, Y. Li, Q. Xie, X. Wang, and H. Li. 2023. A survey on the robustness of large language models. In *Findings of the Conference on Empirical Methods in Natural Language Processing (EMNLP Findings)*, pages 14521–14538.

K. Meng, D. Bau, A. Andonian, and Y. Belinkov. 2022. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 17359–17372.

K. Meng, D. Bau, A. Andonian, and Y. Belinkov. 2023. Mass editing memory in a transformer. In *International Conference on Learning Representations (ICLR)*.

K. A. Nguyen, S. Schulte im Walde, and N. T. Vu. 2017. Distinguishing antonyms and synonyms in a pattern-based neural network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622.

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

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Qwen Team. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Qwen Team. 2024b. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Qwen Team. 2025. Qwen3.5 official repository.

H. Touvron, T. Lavril, G. Izacard, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

H. Touvron, L. Martin, K. Stone, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

A. Vaswani, N. Shazeer, N. Parmar, et al. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008.

Liangdong Wang, Bo-Wen Zhang, Chengwei Wu, Hanyu Zhao, Xiaofeng Shi, Shuhao Gu, Jijie Li, Quanyue Ma, TengFei Pan, and Guang Liu. 2024. Cci3.0-hq: a large-scale chinese dataset of high quality designed for pre-training large language models. *Preprint*, arXiv:2410.18505.

Yu-Hang Wu, Yu-Jie Xiong, Hao Zhang, Jia-Chen Zhang, and Zheng Zhou. 2025. Sugar-coated poison: Benign generation unlocks llm jailbreaking. *arXiv preprint arXiv:2504.05652*.

Chenghao Xiao, Hou Pong Chan, Hao Zhang, Mahani Aljunied, Lidong Bing, Noura Al Moubayed, and Yu Rong. 2025. Analyzing llms' knowledge boundary cognition across languages through the lens of internal representation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*.

A. Yang, A. Li, B. Yang, et al. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

S. Yang, J. Kautz, and A. Hatamizadeh. 2025b. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations (ICLR)*.

Paul Youssef, Osman Alperen Koras, Meijie Li, Jörg Schlötterer, and Christin Seifert. 2023. Give me the facts! a survey on factual knowledge probing in pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15588–15605.

Hengyuan Zhang, Zhihao Zhang, and Mingyang Wang. 2026. Locate, steer, and improve: A practical survey of actionable mechanistic interpretability in large language models. *Preprint*, arXiv:2601.14004.

Jia-Chen Zhang, Yu-Jie Xiong, Chun-Ming Xia, Dong-Hai Zhu, and Xi-He Qiu. 2025. Parameter-efficient fine-tuning of large language models via deconvolution in subspace. In *Proceedings of the 31st International Conference on Computational Linguistics*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*.

W. X. Zhao, K. Zhou, J. Li, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Hyperparameter	Value
Sequence Length	4096
Precision	BF16
Optimizer (AdamW)	$\beta_1 = 0.9$, $\beta_2 = 0.95$
Learning Rate	3×10^{-5}
Warmup Ratio	0.1
Weight Decay	0.01
Per-GPU Micro Batch Size	8
Gradient Accumulation Steps	1
Global Batch Size	64
Training Epochs	1
Training Tokens	~262B

Table 6: Unified hyperparameter configuration for all continued pre-training experiments.

A Training and Evaluation Details

To ensure rigorous and reproducible comparisons, all continued pre-training experiments were conducted under a unified configuration. This appendix details the hardware setup, hyperparameter settings, and training efficiency metrics across all evaluated layer-wise allocation strategies.

A.1 Hardware and Implementation.

All experiments were implemented using the native Accelerate training framework and executed on a single server equipped with 8 $\times$ NVIDIA H200 141G. To strictly isolate the impact of layer-wise parameter allocation, we maintained identical data pipelines, hardware environments, and training schedules across all trials. Wall-clock training time was measured from the first optimizer step to the final checkpoint save, explicitly excluding data loading and model initialization overhead.

A.2 Hyperparameter Configuration

We adopted a consistent hyperparameter configuration for all continued pre-training runs, as summarized in Table 6. Models were trained on the CCI3.0-HQ corpus (~262B tokens) for a single epoch with a sequence length of 4096 and BF16 precision. Optimization was performed using AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$) with a learning rate of 3×10^{-5}, a linear warmup ratio of 0.1, and a weight decay of 0.01. The effective global batch size of 64 was achieved using a per-GPU micro-batch size of 8 with a single gradient accumulation step.

A.3 Training Efficiency and Strategy Mapping.

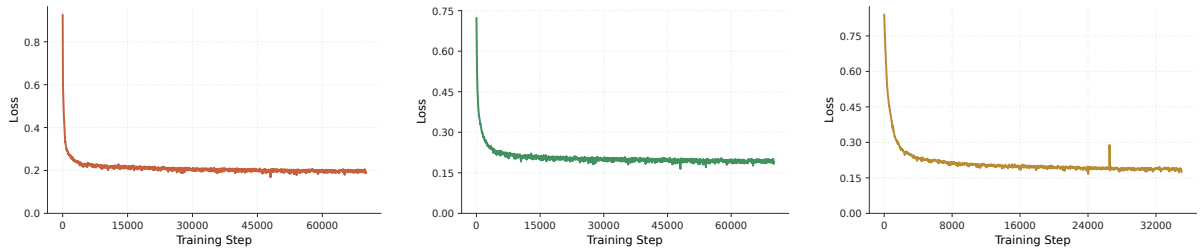
Table 7 reports the computational efficiency across all evaluated strategies. Strategies (1)–(3) correspond to the vanilla Qwen3-0.6B-Base variants reported in Table 1 of the main text: (1) full-parameter training, (2) Train-Shallow/Freeze-Deep, and (3) Freeze-Shallow/Train-Deep. Strategies (4) and (5) correspond to the hybrid architectures evaluated in Table 4: (4) Nemotron-Qwen3-Base (training shallow NemotronDecoderLayer while freezing deep QwenDecoderLayer) and (5) Qwen3-Nemotron-Base (training shallow QwenDecoderLayer while freezing deep NemotronDecoderLayer).

As shown in Table 7, partial-parameter strategies substantially reduce the number of trainable parameters, from approximately 0.6B in full-parameter training to about 0.3B in the vanilla partial-training setting and 0.23B in the hybrid setting. However, the wall-clock training time does not decrease monotonically with the number of trainable parameters. For the vanilla Qwen3-0.6B-Base variants, both partial-training strategies reduce training time compared with full-parameter training, from approximately 141h to 112h and 107h, respectively. In contrast, the hybrid architectures introduce additional optimization difficulty due to heterogeneous layer composition. In particular, when trainable modules must adapt their representations to frozen modules from a different architecture family, the optimization process involves not only parameter updating but also cross-architecture representation alignment. This explains why the hybrid strategies can require comparable or even longer wall-clock time despite involving fewer trainable parameters.

Figure 8 illustrates the training loss curves of the three models investigated in Section 4. Overall, the three curves show broadly similar decreasing trends under the same corpus and training schedule, indicating stable continued pre-training across different allocation strategies.

A.4 Evaluation Details

To ensure fair comparison with existing baselines and official leaderboards, all evaluations are conducted in a zero-shot setting without any task-specific demonstrations or fine-tuning. The evaluation prompts strictly follow the official templates provided by C-Eval (Huang et al., 2023) and CMMLU (Li et al., 2024), preserving the original question formatting, option ordering, and answer extraction rules. For generation-based evaluation, we adopt the official decoding configuration with temperature set to 0.95, top-p sampling disabled, and a maximum output length of 16 tokens to align



(a) Train-Shallow/Freeze-Deep (b) Freeze-Shallow/Train-Deep (c) Full-parameter Training

Figure 8: Training loss curves across three layer-wise allocation strategies on the CCI3.0-HQ corpus. (a) Train-Shallow/Freeze-Deep: shallow layers are updated while deep layers are frozen, showing stable convergence and the lowest final loss. (b) Freeze-Shallow/Train-Deep: deep layers are updated while shallow layers are frozen, exhibiting slower convergence and higher final loss due to disruption of task-critical representations. (c) Full-parameter Training: all layers are updated, serving as the baseline with intermediate convergence behavior. The consistent pattern validates that preserving deep execution zones while adapting shallow sensitive zones yields more efficient and stable continued pre-training.

with the benchmark protocols. For logit-based evaluation, we directly extract the probabilities of candidate answer tokens from the model's vocabulary logits without any decoding,. All evaluations are performed on the validation sets of both benchmarks, and results are reported as accuracy percentages across four domain categories (STEM, Social Science, Humanities, and Other) as well as the overall average.

B Experimental Dataset

B.1 Dataset Construction

The original AntSynNET dataset (Nguyen et al., 2017) provides word pairs for lexical semantic relation tasks, where each instance is represented as a tuple $\langle w_1, w_2 \rangle$ indicating an antonym or synonym relation, such as `(good, bad)` or `(hot, cold)`. To support our analysis in Section 4, we reformulate each instance into a structured prompt $t = s_1 \oplus s_2$, where s_1 contains contextual demonstration pairs and s_2 contains the query word to be predicted. The structured prompt follows the template:

```
Example: [pair_1], [pair_2];
Query: [target_word]->
```

Here, s_1 spans the contextual examples before the query, while s_2 corresponds to the query segment. Specifically, we perturb token positions belonging to s_1 while keeping the query segment s_2 unchanged. In this way, the task query itself remains intact, and the resulting distributional change mainly reflects how each layer responds to contextual information disruption. Thus, s_1-targeted masking serves as a unified and controlled disturbance for measuring layer-wise sensitivity.

B.2 Why Controlled Lexical Prompts

LayerTracer tracks how the probability of a target token evolves across layers by projecting intermediate hidden states through the LM head. Therefore, the diagnostic setting requires the first predicted token to be meaningful and directly related to the answer. Complex QA benchmarks, such as math or medical QA, are less suitable for this purpose because the model may need to generate a reasoning process or follow a specific answer format. In contrast, the structured AntSynNET prompts provide short and controlled one-step lexical prediction tasks, making it easier to isolate how contextual demonstrations affect target-token emergence across layers.

Our analysis uses 1000 structured samples in total. For visualization, these samples are evenly divided into 10 groups, with 100 samples per group, denoted as Task 1–Task 10. Layer-wise metrics are computed for each sample and then averaged within each group to produce the heatmap rows. Table 8 provides 10 representative prompts to illustrate the dataset format.

C Additional Experiments

C.1 Robustness to Top-K Probability Support

To examine whether the Layer-wise Sensitivity results are affected by the choice of probability support, we further repeat the analysis using a larger Top-K probability support with $K = 50$, as shown in Figure 9. For each layer, the JS divergence is computed over the aligned Top-50 probability distribution rather than a smaller truncated support. This Top-50 setting refers to the probability support

Strategy	Trainable Parameters	Training Time
(1) Full-Parameter	~0.6B	~141h
(2) Train-Shallow / Freeze-Deep	~0.3B	~107h
(3) Train-Deep / Freeze-Shallow	~0.3B	~112h
(4) Hybrid Architectures (Train-Shallow)	~0.23B	~115h
(5) Hybrid Architectures (Train-Deep)	~0.23B	~146h

Table 7: Training efficiency comparison across layer-wise allocation strategies. Strategies (1)-(3) correspond to vanilla Qwen3-0.6B-Base variants, while (4)-(5) correspond to the hybrid architectures (Nemotron-Qwen3-550M-Base and Qwen3-Nemotron-550M-Base).

ID	Prompt
1	`Example:good->Bad, no-Yes; Query:bad->`
2	`Example:hot->Cold, big-Small; Query:cold->`
3	`Example:fast->Slow, light-Heavy; Query:slow->`
4	`Example:love->Hate, start-End; Query:hate->`
5	`Example:day->Night, up-Down; Query:night->`
6	`Example:rich->Poor, win-Lose; Query:poor->`
7	`Example:early->Late, high-Low; Query:late->`
8	`Example:strong->Weak, loud-Quiet; Query:weak->`
9	`Example:hard->Soft, dark-Bright; Query:soft->`
10	`Example:full->Empty, open-Close; Query:empty->`

Table 8: Representative structured prompts from the AntSynNET-based dataset. These examples illustrate the prompt format rather than the full set of visualization groups.

used for JS-based sensitivity computation, which is distinct from the top-10 candidate-token selection used for target-token analysis and visualization.

The results show that enlarging the probability support to Top-50 preserves the same qualitative trend observed in the main analysis: shallow layers still exhibit stronger sensitivity fluctuations, while deep layers remain comparatively stable and act as execution-dominant zones. Consequently, it further justifies our practical allocation strategy in Section 4.2.3, where we adopt a symmetric midpoint split to train the shallow sensitive zones and freeze the deep execution zones.

This consistency indicates that the LayerTracer diagnosis is robust to the choice of Top-K probability support, further supporting the conclusion that shallow layers are more suitable for adaptation whereas deep layers should be preserved during continued pre-training.

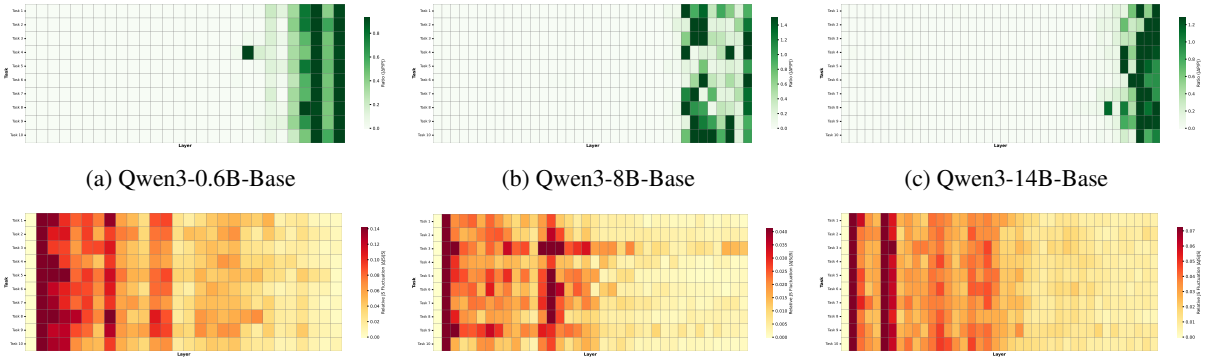
C.2 Boundary Selection

An exhaustive training sweep over all possible split boundaries can identify a model-specific optimum, but this is not the goal of our study. Our objective is to derive a general allocation rule from layer-wise diagnostics to address *RQ2*. Figure 10 shows the normalized TP and LS profiles of Qwen3-0.6B-Base. LS is concentrated in shallow and middle layers, indicating that these layers are sensitive to perturbation and suitable for adaptation. In contrast, TP becomes more prominent in later layers, suggesting that deep layers mainly serve as task-evidence consolidation and execution zones.

Based on this observation, we use the 50% midpoint as a simple and model-agnostic split in the main continued pre-training experiments. This setting keeps a substantial sensitive shallow layer trainable while preserving the deep execution layer from disruptive updates. It also provides a symmetric comparison with matched trainable-parameter budgets. Thus, the midpoint is not treated as an optimal boundary, but as a controlled test of the Train-Shallow/Freeze-Deep principle.

To examine whether this allocation rule depends on the midpoint, we further conduct a split-boundary diagnostic using LayerTracer. For a model with N layers and a split ratio r, we set the split layer to the nearest feasible layer $b = \mathrm{round}(rN)$. Layers 1 to b form the shallow layer, and layers $b+1$ to N form the deep layers. A desirable Train-Shallow/Freeze-Deep split should place high-LS layers in the trainable layer and high-TP layers in the frozen layer. We therefore define the following boundary alignment score:

$$\mathcal{S}(b) = \overline{\widehat{\mathrm{LS}}}_{1:b} + \overline{\widehat{\mathrm{TP}}}_{b+1:N} - \overline{\widehat{\mathrm{TP}}}_{1:b} - \overline{\widehat{\mathrm{LS}}}_{b+1:N}, \tag{7}$$



(a) Qwen3-0.6B-Base (b) Qwen3-8B-Base (c) Qwen3-14B-Base

(d) Qwen3-0.6B-Base (e) Qwen3-8B-Base (f) Qwen3-14B-Base

Figure 9: Robustness analysis under Top-50 probability support on the AntSynNET dataset, where 1000 samples are evenly divided into 10 groups of 100 samples for visualization. Here, Top-50 refers to the probability support used for JS-based layer-wise sensitivity computation, which is distinct from the top-10 candidate-token display in decoding. The single target-token probability used in Ratio remains unchanged. (a)–(c) show the variation of Ratio across layers for Qwen3-0.6B-Base to Qwen3-14B-Base. A deeper green color indicates a larger Ratio, meaning that the target token probability changes more significantly at this layer. (d)–(f) show the variation of ΔJS across layers under the same Top-50 setting. A deeper color indicates a larger abrupt change in layer sensitivity, reflecting higher vulnerability and lower robustness.

Trainable Shallow Ratio	Split Layer	$\mathcal{S}(b)$
33%	9	0.004
50%	14	0.138
66%	19	0.392

Table 9: Split-boundary diagnostic based on normalized TP and LS. Positive scores indicate that sensitive layers are assigned to the trainable layer and task-execution layers are assigned to the frozen layer, supporting the Train-Shallow/Freeze-Deep rule across multiple split boundaries.

Benchmark	Generate Std.	Logit Std.
C-Eval	0.00	0.00
CMMLU	0.00	0.00

Table 10: Standard deviation over five repeated evaluations for all models used in Section 5 and Section 6.

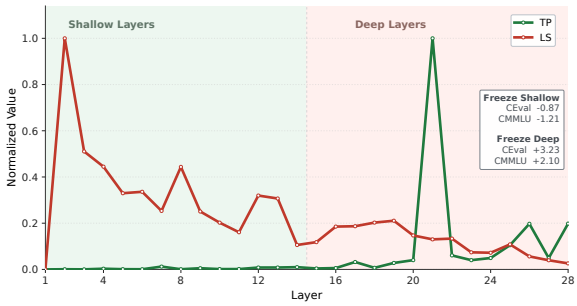


Figure 10: Layer-wise TP and LS profiles of Qwen3-0.6B-Base. The light green region denotes the shallow layer used in the midpoint split, where LS is relatively stronger and layers are more sensitive to perturbation. The light red region denotes the deep layers, where TP becomes more prominent and task evidence is mainly consolidated. The dashed line marks the 50% split used in the main experiments.

where $\widehat{\mathrm{TP}}$ and $\widehat{\mathrm{LS}}$ denote normalized TP and LS values. A positive score indicates that the split is better aligned with the Train-Shallow/Freeze-Deep rule than with the reverse allocation.

As shown in Table 9, $\mathcal{S}(b)$ remains positive under 33%, 50%, and 66% shallow-layer splits. This result indicates that the allocation direction is not tied to the exact midpoint. The performance deltas in Figure 10 further support this interpretation: freezing the shallow sensitive layer and training only deep layers leads to performance drops, whereas training shallow layers while freezing deep layers yields consistent gains over full-parameter training.

C.3 Standard Deviation under Deterministic Evaluation

As shown in Table 10, all models used in Section 5 and Section 6 achieve zero standard deviation over five repeated evaluations on both C-Eval and CMMLU under the two evaluation protocols. For the generation-based evaluation, we use greedy decoding with `do_sample=False`, so the generated answer is deterministic given the same model and input. For the logit-based evaluation, the score is computed directly from the fixed probabilities of the candidate answer tokens, without any decoding

ID	Prompt	Response
1	According to traffic regulations, a motor vehicle running a red light will	In 2019, the National Bureau of Statistics of China released the *China Statistical Yearbook 2018*. The book states that . . . ranked second, but per capita GDP was only around USD 35,000. It also notes that China's total economic output had reached first place in the world . . .
2	Beijing is the capital of China, while Shanghai is	In 2019, many major changes occurred in China and around the world. These changes include: 1. China and the world . . . the economies of China and the world are undergoing major changes. These changes include: 1. the economies of China and the world are undergoing major . . .
3	Washington, D.C. is the capital of the United States, while New York is	The capital of the United States. In 1940, New York City was the capital of the United States, and also the capital of the United States. New York City . . . capital. New York City is the capital of the United States, and also the capital of the United States . . .

Table 11: Bad cases from Qwen3-0.6B-Base (Frozen/Trainable). The responses exhibit severe semantic drift: instead of completing simple factual prompts, the model generates unrelated news-style or encyclopedic text.

ID	Prompt	Response
1	According to traffic regulations, a motor vehicle running a red light will	Be fined 20 yuan. If the driver does not stop in time when running the red light, an additional fine of no more than 15 yuan will be imposed. The car I was driving hit . . .
2	Beijing is the capital of China, while Shanghai is	The capital of China after 1949. Therefore, from a historical perspective, there have been two capitals in Chinese history. The first was during the Qin period, when Xianyang was established as the capital in order to strengthen centralization. . . .
3	Washington, D.C. is the capital of the United States, while New York is	The capital of the United States. In the 19th century, New York City's population had already exceeded two million, making it one of the most populous cities in the world at that time. . . .

Table 12: Good cases from Qwen3-0.6B-Base (Trainable/Frozen). Compared with the Frozen/Trainable model, the responses are more logically coherent and more likely to start with the correct completion, although some continuations still show mild topic drift.

randomness. Therefore, the repeated evaluations produce identical results. This confirms the reproducibility of our reported scores under the fixed evaluation setting.

D Case Study

To intuitively compare different layer-wise training allocation strategies, we present representative generation examples from two Qwen3-0.6B-Base variants: Qwen3-0.6B-Base (Frozen/Trainable), where shallow layers are frozen and deep layers are trainable, and Qwen3-0.6B-Base (Trainable/Frozen), where shallow layers are trainable and deep layers are frozen.

As shown in Table 11, the Frozen/Trainable model often fails to follow simple factual prompts and quickly drifts into unrelated news-style or encyclopedic text. In contrast, Table 12 shows that the Trainable/Frozen model produces more coherent continuations and is more likely to begin with the correct answer, although some outputs still contain later topic drift. These qualitative cases further support our conclusion that updating deep execution layers can disrupt inherited knowledge, while freezing deep layers better preserves factual consistency and generation stability.