

Path-Lock Expert: Separating Reasoning Mode in Hybrid Thinking via Architecture-Level Separation

Shouren Wang^{1,*}, Wang Yang^{1,*}, Chuang Ma^{2,3}, Debargha Ganguly¹, Vikash Singh¹,
Chaoda Song¹, Xinpeng Li¹, Xianxuan Long⁴, Vipin Chaudhary^{1,†}, Xiaotian Han^{1,†}

¹Case Western Reserve University ²Kyoto University ³NII LLMC

⁴Michigan State University

{sxw992, wxy320, dxg512, vxs465, cxs965, xxl1337, vipin, xhan}@case.edu
ma.chuang.52h@st.kyoto-u.ac.jp longxia2@msu.edu

*Equal contribution [†]Corresponding authors

Abstract

Hybrid-thinking language models expose explicit /think and /no_think modes, but current designs do not separate them cleanly. Even in /no_think mode, models often emit long and self-reflective responses, causing *reasoning leakage*. Existing work reduces this issue through better data curation and multi-stage training, yet leakage remains because both modes are still encoded in the same feed-forward parameters. We propose **Path-Lock Expert (PLE)**, an architecture-level solution that replaces the single MLP in each decoder layer with two semantically locked experts, one for /think and one for /no_think, while keeping attention, embeddings, normalization, and the language-model head shared. A deterministic control-token router selects exactly one expert path for the entire sequence, so inference preserves the dense model’s per-token computation pattern and each expert receives mode-pure updates during supervised fine-tuning. Across math and science reasoning benchmarks, PLE maintains strong /think performance while producing a substantially stronger mode separation—a /no_think mode with higher accuracy and far less reasoning leakage. On Qwen3-4B, for example, compared to the SFT-only baseline on AIME24, PLE generates $17\times$ fewer reflective tokens (6.01 vs. 0.35 per answer) and $2\times$ shorter outputs (8665 vs. 4101 tokens), and improves /no_think accuracy from 35.33% to 44.67%, while maintaining /think-mode performance (61.33% vs. 60.00%). These results suggest that controllable hybrid thinking is fundamentally an architectural problem, and separating mode-specific feed-forward pathways is a simple and effective solution. The code is available at: <https://github.com/SR-A-W/path-lock-expert>

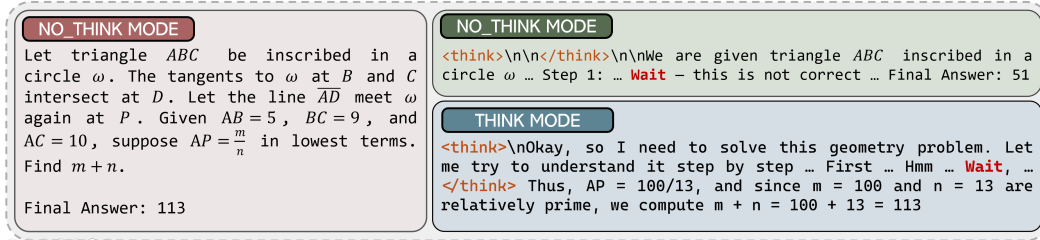


Figure 1: Motivating example of reasoning leakage. On an AIME24 problem, Qwen3-8B in /no_think mode still emits self-reflective tokens (e.g., **Wait**) outside the empty <think> block and arrives at an incorrect answer, while /think mode solves it correctly. This shows that current hybrid-thinking models cannot fully separate the different reasoning modes.

Table 1: Reasoning leakage in hybrid-thinking models. We compare Qwen2.5-7B-Instruct (Instruct, a pure instruction-tuned model) with Qwen3-4B (Hybrid, a hybrid-thinking model). We report accuracy (Acc.), average output length (Len.), and average reflective tokens per response (#Refl./Ans.). Δ denotes the difference between each model’s /no_think performance and the Instruct baseline. The hybrid model’s /no_think mode stays markedly longer and more reflective than a genuine direct-answer mode.

Model	Acc. (%)			Len.			#Refl./Ans.		
	Think	No-think	Δ	Think	No-think	Δ	Think	No-think	Δ
MATH500									
Instruct	–	59.94		–	703		–	0	
Hybrid	92.82	82.60	+22.66	4384	1583	+880	16.74	0.04	+0.04
AIME24									
Instruct	–	6.67		–	1729		–	0	
Hybrid	68.00	23.33	+16.66	16466	12506	+10777	98.35	0.17	+0.17
GPQA									
Instruct	–	30.15		–	775		–	0	
Hybrid	51.01	46.97	+16.82	11371	2679	+1904	188.59	0.05	+0.05

1 Introduction

Hybrid-thinking language models are only useful if their control tokens actually control behavior. In principle, /think should invoke deliberate reasoning when a problem requires it, while /no_think should return a direct answer when extended reasoning is unnecessary (Yang et al., 2025; Comanici et al., 2025; Agarwal et al., 2025). This distinction matters because long reasoning traces come with an associated cost: they increase latency, consume token budget, and occupy context-window space (Sui et al., 2025; Chen et al., 2024; Zhang et al., 2025a). A practical /no_think mode therefore must not be a weakened version of /think; it must operate as a genuine direct-answer mode.

In current hybrid-thinking systems, this separation is weak. Even under explicit /no_think instructions, models often continue to emit signs of deliberation: responses become unnecessarily long, reflective markers such as wait or hmm appear, and outputs drift toward partial chain-of-thought generation. Prior literature defined this failure mode as **reasoning leakage**. Under this failure mode, /no_think becomes an unstable halfway mode between direct answering and explicit reasoning, thereby the interface no longer delivers the efficiency, predictability, and behavioral control that motivate hybrid thinking in the first place.

Most existing attempts to fix leakage operate through training alone. Prior work shows that changing data scale, /think vs. /no_think ratios, and multi-stage training schedules can reduce leakage substantially (Wang et al., 2025). However, lower leakage alone does not guarantee a strong /no_think mode: answers may remain verbose, partially reflective, or weaker in accuracy. This suggests that the limitation is not only in data or optimization. A single dense decoder is being asked to realize two competing output behaviors with the same feed-forward parameters; one mode should externalize reasoning, while the other should suppress it. We therefore hypothesize that weak mode separation is partly an architectural interference problem.

We address this with **Path-Lock Expert (PLE)**, a minimal architecture that makes the mode choice part of the architecture itself. PLE replaces each decoder MLP with two semantically locked experts, one dedicated to /think and one dedicated to /no_think, and uses deterministic control-token routing to activate exactly one expert path for the entire sequence. Attention, embeddings, normalization, and the language-model head remain shared. This design separates the parameters most directly tied to generation behavior while preserving a shared representational backbone, and it does so without a learned router, auxiliary balancing losses, or a complex RL-based training pipeline.

Across mathematical and scientific reasoning benchmarks, PLE yields a substantially stronger `/no_think` mode than dense baselines and prior training-only hybrid approaches: no-think outputs are shorter, exhibit far less reasoning leakage, and achieve better no-think accuracy in our primary evaluations. At the same time, PLE maintains strong `/think`-mode performance. Our claim is intentionally specific: we do not argue that PLE solves hybrid thinking in general or establishes a new overall state of the art; rather, we show that architecture-level mode separation can materially improve the usefulness of the `/no_think` mode while preserving the value of `/think`. Our contributions are as follows:

- We motivate an architectural view of reasoning leakage in hybrid-thinking models, arguing that training heuristics alone may be insufficient because shared feed-forward parameters interfere with clean mode separation.
- We introduce PLE, a deterministic dual-expert decoder architecture that separates `/think` and `/no_think` behavior at the MLP level while preserving a shared Transformer backbone and dense-model-like per-token inference computation.
- We show that, across our evaluated settings, PLE strengthens the `/no_think` mode through lower leakage, shorter outputs, and improved no-think accuracy in the primary evaluations, while maintaining strong `/think` performance.

2 Motivation

We define **hybrid thinking** as an interface in which a single model exposes two response behaviors—an extended chain-of-thought reasoning mode (`/think`) and a concise direct-answer mode (`/no_think`), termed the *thinking* and *non-thinking* modes in the Qwen3 Technical Report (Yang et al., 2025)—selected by a user-supplied control token appended to the query, following that report and Wang et al. (2025). This explicit, user-directed setting is distinct from *implicit-routing* variants, where the model decides internally. Our problem statement follows the explicit setting: *can a hybrid-thinking model truly switch between `/think` and `/no_think` modes—in particular, given that the user has selected `/no_think`, can the model reliably yield a short and direct answer?*

We use a simple empirical comparison: we compare a pure instruction-tuned model, an off-the-shelf hybrid-thinking model, and the strongest prior training-level mitigation, and ask whether `/no_think` behaves like a genuine direct-answer mode. First, `/no_think` should suppress explicit reasoning artifacts such as long deliberative outputs and reflective phrases. Second, it should preserve the behavioral character of direct answering rather than merely producing a weaker version of `/think`.

Off-the-shelf hybrid thinking leaks by default. Table 1 shows that released hybrid-thinking models do learn to respond differently under `/think` and `/no_think` prompts, but the separation is incomplete. Relative to the pure Instruct baseline, the hybrid model’s `/no_think` outputs remain longer and contain nonzero reflective-token counts across benchmarks. This means the issue is not that hybrid models fail to react to the control token at all; rather, they react only partially. `/no_think` behaves like a softened form of thinking, not like a truly distinct direct-answer mode.

Training-only fixes help, but they stop short of true separation. Wang et al. (2025) systematically explored the most natural mitigation strategies: scaling training data, changing the think/no-think ratio, and applying multi-stage training. Their best recipe significantly reduces leakage, confirming that data and optimization matter. However, it still does not match the behavioral target defined above. Reflective tokens remain nonzero, and the resulting `/no_think` behavior does not consistently recover the direct-answer profile of the pure Instruct model across benchmarks. This is the key empirical observation: even after substantial recipe engineering, the model remains only partially controllable.

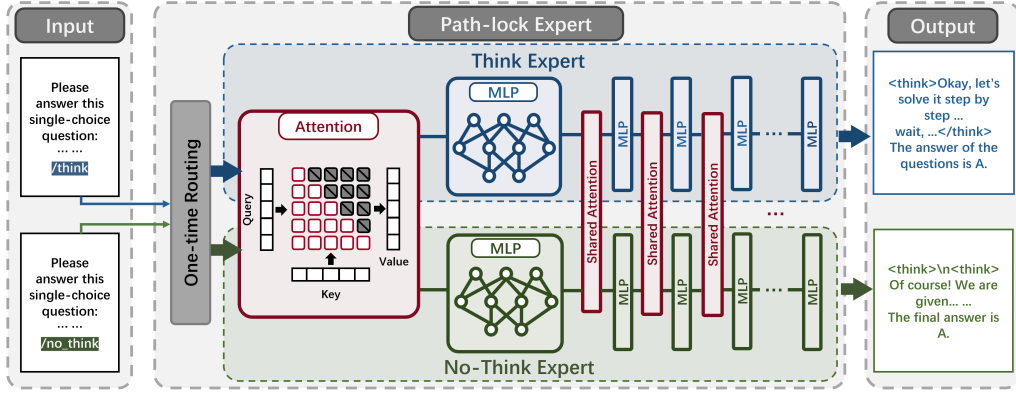


Figure 2: Path-Lock Expert replaces the single MLP in each decoder layer with two mode-specific experts while keeping the attention backbone shared. A single routing decision is made from the control tokens and reused across all layers and all decoding steps.

3 Method: Path-Lock Expert

Path-Lock Expert (PLE) aims to achieve three primary objectives in hybrid-thinking language models: 1) better reasoning mode separation than conventional hybrid-thinking architectures; 2) performance preservation, ensuring the model’s core reasoning capabilities are not degraded; and 3) parameter efficiency, avoiding the prohibitive cost of training or deploying two independent models. PLE achieves these with a minimal modification to a standard decoder-only Transformer, described below.

Path-locked dual-expert decoder. In every layer, the single MLP is replaced by two structurally identical copies—one for `/think` and one for `/no_think`. All other components (self-attention, positional encoding, normalization, embeddings, LM head) remain shared. Because exactly one expert is active per response, per-token computation is unchanged. By replicating only the MLP rather than duplicating the full model, PLE saves approximately 33% of total parameters relative to deploying two independent copies of the backbone.

Deterministic control-token routing. The active expert is selected by the last `/think` or `/no_think` token appearing in the input; the choice is resolved once and locked for all layers and all decoding steps. No learned router or balancing loss is involved. This one-time, all-layer lock is what motivates the name Path-Lock Expert: swapping the control token flips a single routing index and activates the other expert, leaving attention, embeddings, norms, and the LM head unchanged. We formalize the detection procedure, its string-level fallback, and edge-case handling, and also reported detailed validation tests in Appendix A.6.

Routing-conditioned fine-tuning. PLE is fine-tuned with standard causal LM loss on chat examples tagged with either mode. The inactive expert receives zero gradient; the shared backbone is updated by both modes. This removes direct parameter coupling between the two MLP pathways while preserving a common representational substrate. We provide a rigorous definition in Appendix B.

Relation to Mixture-of-Experts. While previous work such as *Metis-HOME* (Lan et al., 2025) has explored using a two-expert MoE for mode separation, their approach still relies on the **conventional MoE framework** with learned, token-level routers. In contrast, PLE shifts from token-level competition to **deterministic, sequence-level routing**. By tying expert identity directly to the `/think` or `/no_think` interface via control tokens, PLE eliminates the need for router parameters, balancing losses, or complex dispatch decisions. This ensures a “locked” path that is structurally stable and specifically designed for behavioral control.

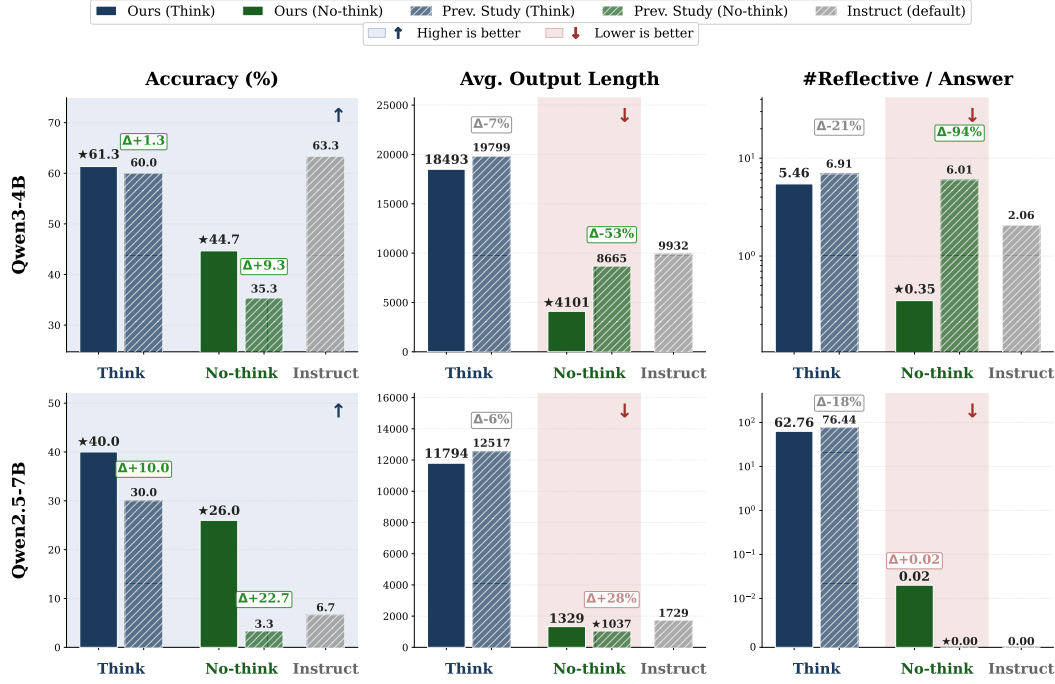


Figure 3: Accuracy, average output length, and per-answer reflective token count on AIME24 for two PLE instantiations (Qwen2.5-7B, top; Qwen3-4B, bottom) compared with baselines. “Ours” denotes PLE models built on each backbone; “SFT-only” is an SFT-only baseline in previous studies (Wang et al., 2025); “Instruct” is the off-the-shelf instruction-tuned backbone, shown under its single default-mode response. On Qwen2.5-7B, PLE matches the near-zero reflective token count of the SFT-only baseline despite much higher /no_think accuracy. On Qwen3-4B, PLE achieves substantially stronger mode separation than SFT-only: reasoning leakage drops by 17 $\times$ and /no_think accuracy improves by +9.3 points, while /think-mode performance is maintained.

4 Experiments

In this section, we evaluate the effectiveness of PLE in achieving architectural mode separation. Our experiments are designed to verify whether physical parameter isolation can improve mode separation by addressing three key questions: 1) **Leakage Reduction**: Does PLE significantly reduce reasoning leakage (reflective tokens) in /no_think mode compared to dense baselines? 2) **Mode Performance**: Does PLE improve the performance of the /no_think mode by reducing gradient interference? 3) **Reasoning Preservation**: Does the architectural separation maintain the original /think performance without degradation?

4.1 Experimental setup

Dataset Construction We build our training corpus from the *Superior-Reasoning* (Yan et al., 2026) problem set, pairing its reasoning traces with /no_think targets synthesized by Qwen3-235B in /no_think mode and retained only if they match the ground-truth answer, respect domain-specific length limits, and contain no reflective tokens. The corpus holds 54k examples: 27k /think traces and 27k direct answers, a 1:1 ratio. Because the /no_think targets are distilled from a stronger teacher, one might ask whether PLE’s gains reflect teacher-style imitation rather than architecture; PLE and its dense baselines train on identical data, so any such bias acts on both equally, and Appendix A.11 rules it out with a controlled data-construction ablation.

Backbone weights. We evaluate PLE across two representative model families to demonstrate its generalizability. For both families we initialize from the corresponding *Instruct*

checkpoint, keeping the two families structurally symmetric: *Qwen2.5-7B-Instruct* for the Qwen2.5 family, and *Qwen3-4B-Instruct* for the Qwen3 family. For each backbone, we initialize the PLE architecture by cloning their MLP weights, and fine-tune on our training corpus constructed from the *Superior-Reasoning* dataset (Yan et al., 2026).

Baselines. We compare PLE (*Ours*) against two baselines per family: (1) *Instruct*, the off-the-shelf backbone; and (2) *SFT-only*, the SFT-only recipe (Wang et al., 2025). All three models start from the same Instruct checkpoint. Results with additional backbone initializations (*Qwen3-4B* and *Qwen3-4B-Base*) are presented in Appendix A.2 and Appendix A.7.1.

Benchmarks and Metrics. We evaluate all models in both `/think` and `/no.think` modes across four reasoning benchmarks: MATH500 (Lightman et al., 2023), AIME24 (Mathematical Association of America, 2024), MMLU-STEM (Hendrycks et al., 2021), and GPQA-Diamond (Rein et al., 2024). To measure performance and mode separation, we report: (1) *Accuracy (%)* for task proficiency; (2) *Average output length* to monitor verbosity; and (3) *Reflective tokens per answer (#Refl./Ans.)*—the mean count of self-reflective markers (e.g., wait, hmm, alternatively). As our two metric for reasoning leakage, average output length and *#Refl./Ans.* are complementary rather than redundant: average output length measures verbosity, while *#Refl./Ans.* measures whether reflective-reasoning tokens are specifically present. We validate these metrics track genuine semantic leakage rather than incidental token counts using a model-based judge (GPT-OSS-20B) applied to off-the-shelf hybrid-thinking baselines across three model sizes and three benchmarks (Appendix A.4): judge verdicts agree with *#Refl./Ans.* in 92–100% of cases in both directions.

4.2 Main Results

We present a comprehensive evaluation of PLE across two model families to verify its effectiveness in mitigating reasoning leakage while maintaining task performance. Overall, our results demonstrate that PLE provides a superior Pareto frontier for hybrid thinking: it delivers a `/no.think` mode that is cleaner and more proficient than prior training-level recipes, while simultaneously safeguarding (or even enhancing) the model’s capacity for deep, multi-step reasoning. Our primary analysis focuses on the challenging AIME24 benchmark (Figure 3), which serves as a stress test for hybrid thinking due to its requirement for deep deliberation; results for MATH500 are consistent with these findings and are provided in Figure 6 (with additional details in Appendix A.1).

Effective Mode Separation with Minimal Leakage. Table 2 reports `/no.think`-mode behavior for every model; we begin with mode separation. On Qwen3-4B, PLE cuts reasoning leakage $17\times$ below the SFT-only baseline (6.01 to 0.35 reflective tokens per answer) and raises `/no.think` accuracy by +9.3 points (35.33% to 44.67%), while `/think`-mode performance is maintained (61.33% vs. 60.00%). SFT-only is only partially effective: it halves output length between modes (19,799 to 8,665 tokens) but leaves reflective content untouched (6.91 to 6.01), whereas PLE halves length to 4,101 tokens and removes the reflective content (Figure 3).

Significant Enhancement of `/no.think` Proficiency. PLE’s `/no.think` mode is not merely cleaner but more capable, consistently across both families. On Qwen3-4B it answers at 44.67%—+9.3 points over the SFT-only recipe—in 4,101 tokens at 0.35 reflective tokens, and the margin is larger on Qwen2.5-7B (26.00% vs. 3.33%, +22.7 points) at 0.02 reflective tokens (Table 2). These gains are not won on easy ground: despite its name, *Qwen3-4B-Instruct* is a non-thinking release of a hybrid-thinking base (Yang et al., 2025; Team, 2025b), reasoning at length with no direct-answer path (Appendix A.8). From that the SFT-only recipe supplies a path but leaks through it (8,665 tokens at 6.01); PLE makes it short and accurate.

Preservation and Synergy of Reasoning Capabilities. A vital requirement for PLE is that architectural isolation must not degrade the model’s core reasoning power. Our results confirm that the `/think` mode’s performance remains robust across both model families. Interestingly, we observe a synergistic effect in the Qwen2.5-7B-Instruct family: the `/think` mode of PLE actually **outperforms** the original model on AIME24 accuracy (Figure 3).

Table 2: **Reasoning leakage and accuracy in the /no_think (direct-answer) mode.** #Refl./Ans.: mean reflective token count per response (e.g., wait, hmm). **Bold marks PLE (Ours).** *Hybrid* and *Instruct* report each off-the-shelf model’s own default /no_think behavior; coming from different post-training lineages, their accuracies are not directly comparable across lineages—the controlled comparison is between *SFT-only* and *PLE (Ours)*, both initialized from the same *Instruct* checkpoint. *SFT-only*: the two-phase SFT recipe of Wang et al. (2025) on the same backbone. /think-mode performance is preserved (Figures 3, 6, 8 and 9).

Model	Type	AIME24			MATH500		
		Acc.	Len.	#Refl./Ans.	Acc.	Len.	#Refl./Ans.
Qwen3-4B	Hybrid	23.33	12506	0.17	82.60	1583	0.04
	Instruct	63.33	9932	2.06	94.52	1812.55	0.36
	SFT-only	35.33	8665	6.01	83.28	1040.37	0.15
	PLE (Ours)	44.67	4101	0.35	82.36	702.23	0.05
Qwen2.5-7B	Instruct	6.67	1729	0	59.94	703.11	0
	SFT-only	3.33	1037	0.00	63.56	593.09	0.31
	PLE (Ours)	26.00	1329	0.02	83.20	614.46	0.01

Model	Type	GPQA-Diamond			MMLU-STEM		
		Acc.	Len.	#Refl./Ans.	Acc.	Len.	#Refl./Ans.
Qwen3-4B	Hybrid	46.97	2679	0.05	85.95	483.79	0.01
	Instruct	42.32	445.08	0.14	87.54	295.95	0.05
	SFT-only	45.96	2513.78	5.34	86.71	263.92	0.06
	PLE (Ours)	45.86	736.34	0.23	86.61	180.19	0.02
Qwen2.5-7B	Instruct	30.15	775	0	62.00	211	0
	SFT-only	30.00	2910	3.22	59.44	774	0.83
	PLE (Ours)	30.00	3857	0.00	92.22	139	0.00

This indicates that sharing the attention backbone preserves the underlying logic while the dual-expert MLP structure lets the /think mode focus on deliberation.

Generalization Beyond Qwen and Math/STEM. To test whether PLE’s mode separation is specific to the Qwen family or to mathematical reasoning, we extend our evaluation to a structurally different model family, *Llama-3.1-8B*, and to a broad general-knowledge benchmark, full *MMLU*. Mode separation holds throughout—on *Llama-3.1-8B*, PLE’s /no_think mode is the most accurate no-think model on all four benchmarks tested, most strikingly on *MMLU-STEM* (76.7% vs. 53.0–53.4%)—so PLE spans three model families and five benchmark types; see Appendix A.10 for full results.

5 Ablation Studies

Beyond the main evaluations in Section 4, we conduct a series of ablation studies to dissect the impact of various design choices on PLE’s performance and mode separation behavior. Our analysis reveals that two primary factors significantly dictate the model’s efficacy: the selection of **backbone weights** for initialization and the composition of the **training dataset**. In the following subsections, we explore how these factors influence the delicate balance between reasoning power and leakage mitigation.

5.1 Effect of backbone initialization

The choice of which pretrained model to use as the backbone for PLE is a critical design decision. We compare four backbones: 1) *Qwen2.5-7B-Instruct*, 2) *Qwen3-4B-Instruct*, the backbone used for our main-text Qwen3 results (Section 4), 3) *Qwen3-4B*, and 4) *Qwen3-4B-Base* (the pretrained model without any post-training).

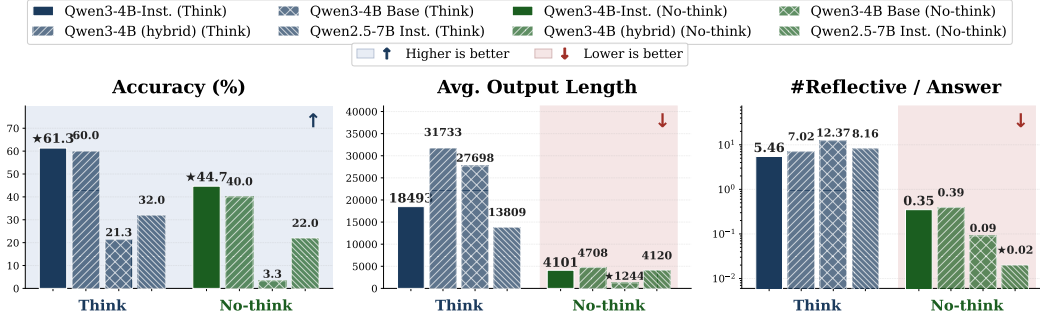


Figure 4: Ablation on backbone choice (AIME24). PLE is initialized from “Qwen2.5-7B Instruct” (pure instruction-tuned weights), “Qwen3-4B Instruct”, “Qwen3-4B Base” (raw pretrained weights), or “Qwen3-4B” (hybrid-thinking weights). Residual /no_think leakage orders monotonically with each initialization’s own reasoning behavior on both dimensions (reflective tokens/answer $0.02 < 0.35 < 0.39$, with /no_think output length in the same order), at some cost in accuracy. The pretrained base collapses on this high-difficulty task, confirming that PLE requires a backbone with existing instruction-following capabilities.

All variants are trained with identical hyperparameters to isolate the effect of backbone weights. Figure 4 provides the results on the AIME24 task; more comprehensive experimental results are provided in Appendix A.7.1.

Comparison between Three Backbones: Effect of Retained Latent Reasoning Behavior. Comparing *Qwen3-4B* (a hybrid-thinking model) with *Qwen2.5-7B-Instruct* (a pure instruction-tuned model) reveals a clear divergence in behavior. We find that using a pure *Instruct* model as the base generally yields superior mode separation, characterized by a “cleaner” /no_think mode with virtually zero leakage. In contrast, using the hybrid *Qwen3-4B* leads to higher absolute performance (accuracy) but carries a persistent residue of reasoning artifacts.

We hypothesize that hybrid-thinking models carry reasoning-heavy patterns from their intensive post-training. This residual reasoning behavior is deeply embedded in the shared attention representations, making it difficult to fully extinguish reflective behaviors in the /no_think mode, even with architectural isolation. However, because the hybrid base already possesses a stronger foundation for the /think mode, it ultimately achieves higher task scores, suggesting that the initial “reasoning literacy” of the weights sets a higher ceiling for performance. Not every instruction-tuned checkpoint is equally free of these patterns: *Qwen3-4B-Instruct* is derived from a hybrid-thinking base and released as a non-thinking update within the same family (Yang et al., 2025; Team, 2025b). Accordingly, residual /no_think leakage after PLE fine-tuning orders with each backbone’s own reasoning behavior on both dimensions: reflective tokens per answer run $0.02 < 0.35 < 0.39$, and AIME24 /no_think length follows the same order, $1,329 < 4,101 < 5,597$ tokens.

The Necessity of Post-training: Qwen3-4B vs. Qwen3-Base. To isolate the impact of post-training, we evaluated *Qwen3-4B-Base* (the raw pretrained weights). While this base shows reasonable performance on simpler benchmarks like MATH500, its performance collapses on the high-difficulty AIME24 tasks. In many cases, the base-initialized PLE fails to converge on a coherent reasoning strategy, often exceeding token limits without reaching a solution. This result underscores that PLE requires a backbone that has already acquired basic instruction-following and logical structuring capabilities. Relying solely on the PLE to “learn” reasoning from scratch is insufficient for complex domains.

5.2 Effect of training dataset

We investigate the impact of dataset quality and composition on PLE training. Specifically, we compare the *OpenR1* dataset used in Wang et al. (2025), constructed from OpenR1-Math (Hugging Face, 2025) and *Superior-reasoning*: our improved dataset featuring harder



Figure 5: Ablation on training dataset (AIME24) for two backbones: *Qwen3-4B-Instruct* (top) and *Qwen2.5-7B-Instruct* (bottom). “Superior”: PLE trained on our high-difficulty dataset with longer CoT traces; “OpenR1”: on simpler data. On *Qwen3-4B-Instruct*, Superior wins in both modes (61.33% vs. 48.00% /think; 44.67% vs. 14.00% /no_think) and reasons more efficiently (5.46 vs. 34.46 reflective tokens in /think). On *Qwen2.5-7B*, OpenR1 remains competitive—dataset difficulty must match backbone capacity.

reasoning questions and higher-quality responses. Our findings are summarized in Figure 5 and Table 24 (Appendix A.12), and point to a trade-off between proficiency and mode purity.

We investigate the impact of dataset quality and difficulty on PLE training by comparing two distinct sources: the **OpenR1** dataset (a relatively simpler collection) and our **Superior-reasoning** dataset (higher difficulty and longer reasoning chains). Our analysis, supported by Table 24 and Figure 5, reveals how data composition interacts with architectural experts.

Difficulty-Capacity Alignment. A key observation from our ablation is that the effectiveness of a dataset is closely tied to the inherent capability of the backbone. As shown in Table 24, the simpler *OpenR1* dataset yields better relative gains when paired with the *Qwen2.5-7B-Instruct* backbone. Conversely, the more challenging *Superior-reasoning* dataset excels when paired with the more capable *Qwen3-4B* model.

For the *Qwen3-4B* backbone, the advantage of the superior dataset is particularly pronounced in the /no_think mode, where it significantly outperforms the OpenR1 baseline in accuracy. This suggests that as backbones become more proficient, they require higher-quality, “harder” data to fully specialize their dual experts. We conclude that dataset “difficulty” must be aligned with the backbone’s capacity to achieve optimal results.

The Leakage-Performance Trade-off. While the *Superior-reasoning* dataset provides clear performance benefits, it also induces more noticeable reasoning leakage compared to OpenR1. We attribute this to the nature of the data: the superior dataset contains significantly longer and more complex chain-of-thought (CoT) traces. These high-intensity reasoning patterns appear to be more “contagious,” exerting greater pressure on the architectural separation and making the /no_think expert more prone to emitting reflective artifacts.

This represents a fundamental trade-off in hybrid thinking: achieving the highest levels of reasoning proficiency often requires sacrificing a degree of mode-control purity. In most practical scenarios, we argue that this trade-off is acceptable, as the gains in /no_think accuracy and /think depth outweigh the presence of minor leakage artifacts.

5.3 Isolating the architectural contribution from added parameters

Do PLE’s gains come from deterministic routing, or merely from extra parameters? From an identical *Qwen3-4B-Base* checkpoint we train a *think-only* and a *no_think-only* specialist on matched data, and compare both against *Base-PLE* trained on the combined data and against a naive weight-merge of the two specialists. Across MATH500, AIME24, and GPQA-Diamond, *Base-PLE* matches or exceeds both dedicated specialists within one model, while the weight-merge collapses accuracy on AIME24 and MATH500 and still leaks reasoning in */no_think* mode (Table 13; full tables in Appendix A.9).

An **initialization-equivalence test** confirms the source: before fine-tuning, PLE is token-for-token identical to the backbone on 12/12 prompt-mode pairs and all 108/108 tensors match numerically, so the extra expert contributes nothing until training decouples the two objectives. Relative to deploying two independent models, **PLE saves up to 33%** of a single model’s parameters ($\approx 1.67 \times$ one model in total), at single-dense-model inference cost (Table 11). Appendix A.11 separately rules out teacher-style bias from the distilled */no_think* targets.

6 Related Work

LLM Reasoning. Reasoning in LLMs has been advanced through RL (Guo et al., 2025; Jaech et al., 2024; Team et al., 2025; Shao et al., 2024; Xie et al., 2025; Zeng et al., 2025) and SFT (Li et al., 2025a; Muennighoff et al., 2025; Ye et al., 2025; Ji et al., 2025). Our work is orthogonal: rather than improving reasoning quality, we focus on *controlling when reasoning occurs* through architectural design.

Hybrid Thinking and Efficient Reasoning. Hybrid thinking (Sui et al., 2025) lets models switch between chain-of-thought and direct answering via control tokens, and has been adopted by several recent systems (Comanici et al., 2025; Yang et al., 2025; Agarwal et al., 2025; DeepSeek-AI, 2025). However, reasoning behaviors still leak into */no_think* mode. Wang et al. (2025) showed that training-level interventions—data scale, ratio tuning, multi-stage schedules—reduce but cannot eliminate this leakage. Other efficiency-oriented approaches include CoT compression (Xia et al., 2025; Zhang et al., 2025a; Xu et al., 2025; Aytes et al., 2025), early stopping (Fu et al., 2024), and preference optimization (Team, 2025a). Unlike all of these, we address leakage at the *architectural level*.

Mixture-of-Experts. MoE architectures (Shazeer et al., 2017; Fedus et al., 2022; Lepikhin et al., 2020; Jiang et al., 2024; Dai et al., 2024) use learned token-level routers for conditional computation, but face challenges including load imbalance, training instability, and the need for auxiliary balancing losses. Most relevant to our setting, Metis-HOME (Lan et al., 2025) applies a two-expert MoE with a learned router to hybrid thinking, differing from PLE in routing (a trainable gate vs. our parameter-free control-token lookup) and in modality (vision-language vs. text-only). Substituting a learned router into our setting would change the research question from “can deterministic routing achieve clean mode separation?” to “which router design wins?”—a different paper. We compare the design space in Appendix A.5 (Table 8).

7 Conclusion

In this paper, we argued that reasoning leakage in hybrid-thinking models is partly an architectural problem. We introduced Path-Lock Expert (PLE), a minimal modification that replaces each decoder MLP with two semantically locked experts routed deterministically by control tokens. Across math and science benchmarks on two model families, PLE substantially strengthens the */no_think* mode—reducing reflective tokens, shortening outputs, and improving accuracy—while preserving */think* performance. Our ablations further show that PLE’s effectiveness depends on backbone initialization and dataset-capacity alignment. These results demonstrate that separating mode-specific feed-forward pathways is a straightforward and effective complement to training-level interventions for achieving more controllable hybrid thinking models.

Acknowledgments

This work was supported in part by NSF awards 2117439 and 2112606.

The first author would like to express his sincere gratitude to Yanbo Zhao, whose guidance and influence inspired him to pursue computer science and AI research, and whose care and support have accompanied him throughout this journey. He would also like to thank Kaijun Tan for his continued support and for generously sharing his practical advice and valuable insights on LLM engineering.

References

- Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.
- Simon A Aytes, Jinheon Baek, and Sung Ju Hwang. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *arXiv preprint arXiv:2503.05179*, 2025.
- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for $2+3=?$ on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.
- Zhikai Chen, Jialiang Gu, Junyu Yin, Xianxuan Long, Shenglai Zeng, Xiaozhe Liu, Kai Guo, Keren Zhou, and Jiliang Tang. Exploring cross-scenario generality of agentic memory systems: Diagnostics and a strong baseline. *arXiv preprint arXiv:2606.04315*, 2026.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.
- Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, 2022.
- Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.
- DeepSeek-AI. Deepseek-v3.1 (model card). <https://huggingface.co/deepseek-ai/DeepSeek-V3.1>, 2025. Accessed 2026-08-08.
- William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.
- Elias Frantar, Saleh Ashkboos, Torsten Hoeftler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023.
- Yao Fu, Xianxuan Long, Runchao Li, Haotian Yu, Mu Sheng, Xiaotian Han, Yu Yin, and Pan Li. Quantized but deceptive? a multi-dimensional truthfulness evaluation of quantized LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.
- Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Aurick Qiao, and Hao Zhang. Efficiently serving llm reasoning programs with certindex. *arXiv preprint arXiv:2412.20993*, 2024.

- Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.
- Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 30–45, 2022.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Mohsen Hariri, Weicong Chen, Nahal Shahini, Vikash Singh, Kai Ye, Amirhossein Samandar, Debargha Ganguly, S. Sankar, Yanyan Zhang, Shouren Wang, Jerry Peng, Biyao Zhang, Michael Hinczewski, and Vipin Chaudhary. Test-time scaling in reasoning LLMs: Inference regimes, evaluation, and reproducibility. *arXiv preprint arXiv:2608.04001*, 2026a.
- Mohsen Hariri, Alan Luo, Weicong Chen, Tianyi Zhang, Qifan Wang, Xiaotian Han, and Vipin Chaudhary. Quantize what counts: More for keys, less for values. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026b.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.
- Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL <https://github.com/huggingface/open-r1>.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- Ke Ji, Jiahao Xu, Tian Liang, Qiuzhi Liu, Zhiwei He, Xingyu Chen, Xiaoyuan Liu, Zhijie Wang, Junying Chen, Benyou Wang, et al. The first few tokens are all you need: An efficient and effective unsupervised prefix fine-tuning method for reasoning models. *arXiv preprint arXiv:2503.02875*, 2025.
- Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.
- Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.
- Xiaohan Lan, Fanfan Liu, Haibo Qiu, Siqi Yang, Delian Ruan, Peng Shi, and Lin Ma. Metis-home: Hybrid optimized mixture-of-experts for multimodal reasoning. *arXiv preprint arXiv:2510.20519*, 2025.
- Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.
- Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hakhmaneshi, Shishir G. Patil, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Llm can easily learn to reason from demonstrations structure, not content, is what matters!, 2025a. URL <https://arxiv.org/abs/2502.07374>.
- Runchao Li, Yao Fu, Mu Sheng, Xianxuan Long, Haotian Yu, and Pan Li. FAEDKV: Infinite-window fourier transform for unbiased KV cache compression. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025b.

- Runchao Li, Yao Fu, Mu Sheng, Haotian Yu, Xianxuan Long, and Kenneth A. Loparo. JW-SVD: Bridging the cross-modal mismatch in post-training MLLM compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, 2024.
- Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in neural information processing systems*, 34: 18878–18890, 2021.
- Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024a.
- Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *International Conference on Machine Learning*, 2024b.
- Xianxuan Long, Zhikai Chen, Shenglai Zeng, Shouren Wang, Kai Guo, and Jiliang Tang. Memtrace: Probing what final accuracy misses in long-term memory. *arXiv preprint arXiv:2606.17328*, 2026.
- Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1930–1939, 2018.
- Mathematical Association of America. American invitational mathematics examination (aime). <https://www.maa.org/math-competitions/aime>, 2024. Math reasoning benchmark derived from 2024 competition problems.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL <https://arxiv.org/abs/2501.19393>.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.
- Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarsz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.
- Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Hanjie Chen, Xia Hu, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.
- NovaSky Team. Think less, achieve more: Cut reasoning costs by 50% without sacrificing accuracy. <https://novasky-ai.github.io/posts/reduce-overthinking>, 2025a. Accessed: 2025-01-23.
- Qwen Team. Qwen3-4b-instruct-2507 (model card). <https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507>, 2025b. Accessed 2026-08-07.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.
- Shouren Wang. Agent team work zone: An automated, persistent workspace for long-lived claude code agent teams. *arXiv preprint arXiv:2607.22917*, 2026.
- Shouren Wang, Wang Yang, Xianxuan Long, Qifan Wang, Vipin Chaudhary, and Xiaotian Han. Demystifying hybrid thinking: Can llms truly switch between think and no-think? *arXiv preprint arXiv:2510.12680*, 2025.
- Shouren Wang, Debargha Ganguly, Vinooth Rao Kulkarni, Wang Yang, Zhuoran Qiao, Daniel Blankenberg, Vipin Chaudhary, and Xiaotian Han. A survey on efficient protein language models. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL <https://openreview.net/forum?id=PTReuOwsXz>. Survey Certification.
- Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms, 2025. URL <https://arxiv.org/abs/2502.12067>.
- Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning, 2025. URL <https://arxiv.org/abs/2502.14768>.
- Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.
- Shaotian Yan, Kaiyuan Liu, Chen Shen, Bing Wang, Sinan Fan, Jun Zhang, Yue Wu, Zheng Wang, and Jieping Ye. Distribution-aligned sequence distillation for superior long-cot reasoning. *arXiv preprint arXiv:2601.09088*, 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.
- Wang Yang, Chaoda Song, Xinpeng Li, Debargha Ganguly, Chuang Ma, Shouren Wang, Zhihao Dou, Yuli Zhou, Vipin Chaudhary, and Xiaotian Han. Agentce-bench: Agent configurable evaluation with scalable horizons and controllable difficulty under lightweight environments. *arXiv preprint arXiv:2604.06111*, 2026a.
- Wang Yang, Chaoda Song, Xinpeng Li, Shouren Wang, Nengbo Wang, Yanyan Zhang, Chuang Ma, Debargha Ganguly, Vikash Singh, Shuai Xu, Jing Ma, Yin Yu, Vipin Chaudhary, and Xiaotian Han. A survey on agent skills for LLMs: A lifecycle perspective from construction to ecosystems. SSRN preprint 6746498, 2026b.
- Wang Yang, Shouren Wang, Chaoda Song, Chuang Ma, Xinpeng Li, Nengbo Wang, Kaixiong Zhou, Vipin Chaudhary, and Xiaotian Han. When domains interact: Asymmetric and order-sensitive cross-domain effects in reinforcement learning for reasoning. *arXiv preprint arXiv:2602.01365*, 2026c.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

- Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning, 2025. URL <https://arxiv.org/abs/2502.03387>.
- Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020.
- Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplertl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild, 2025. URL <https://arxiv.org/abs/2503.18892>.
- Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*, 2025a.
- Tianyi Zhang, Mohsen Hariri, Shaochen Zhong, Vipin Chaudhary, Yang Sui, Xia Hu, and Anshumali Shrivastava. 70% size, 100% accuracy: Lossless LLM compression for efficient GPU inference via dynamic-length float. In *Advances in Neural Information Processing Systems*, 2025b.

A Appendix

LLM Usage Statement

The authors used LLM-based assistants to help with LaTeX editing, polishing writing, proofreading, and code implementation; all content was reviewed and verified by the authors.

A.1 Supplemental Results for Main Experiment Results

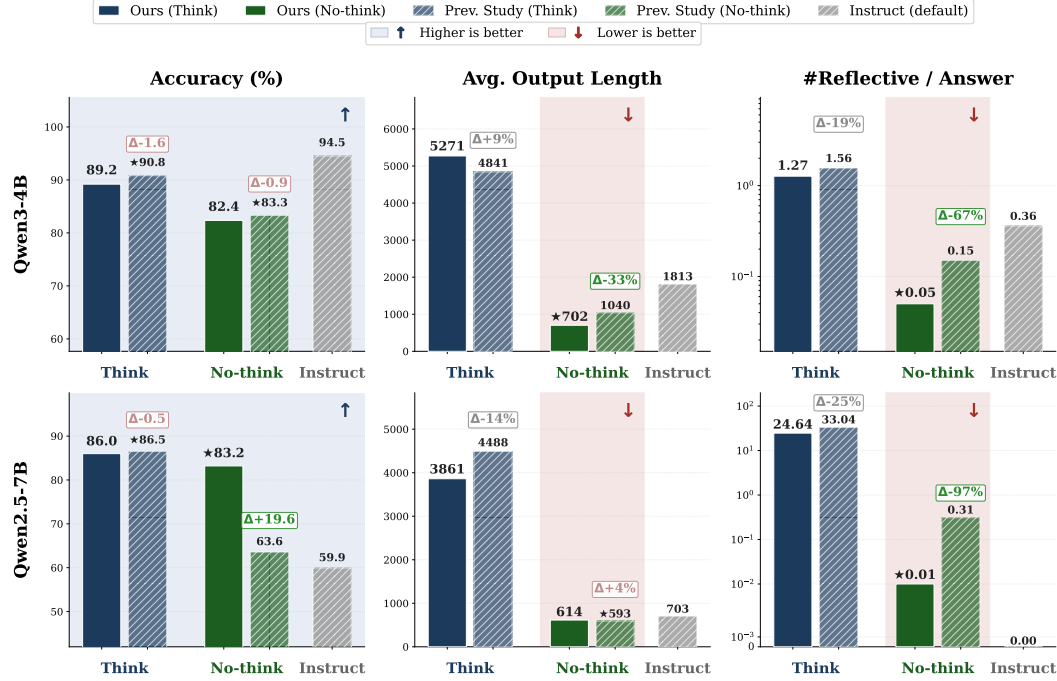


Figure 6: Accuracy, average output length, and per-answer reflective token count on MATH500 for two PLE instantiations (Qwen2.5-7B, top; Qwen3-4B, bottom) compared with baselines, in the same format as Figure 3. “Instruct” is shown under its single default-mode response (no control-token suffix appended).

Table 3: Reasoning leakage comparison on **MMLU-STEM** and **MMLU-full**. Same setup and notation as Table 2.

Model	Type	Mode	MMLU-STEM			MMLU-full		
			Acc.	Len.	#Refl./Ans.	Acc.	Len.	#Refl./Ans.
Qwen2.5-7B	Instruct	Think	N/A	N/A	N/A	N/A	N/A	N/A
		No-think	62.00	211	0	73.82	275.01	0.0000
	SFT-only	Think	85.82	3100	24.74	—	—	—
		No-think	59.44	774	0.83	—	—	—
	Ours	Think	88.89	429	0	—	—	—
		No-think	92.22	139	0.00	76.28	357.27	0.0000
Qwen3-4B	Instruct	Think	N/A	N/A	N/A	N/A	N/A	N/A
		No-think	87.54	295.95	0.05	79.79	264.88	0.02
	SFT-only	Think	90.96	1491.51	0.22	78.43	2777.88	2.69
		No-think	86.71	263.92	0.06	75.58	188.76	0.16
	Ours	Think	91.06	1606.65	0.49	78.17	3175.23	2.85
		No-think	86.61	180.19	0.02	75.60	117.18	0.00

A.2 Results with the Native Hybrid Qwen3-4B Backbone

These tables report PLE initialized from the native hybrid-thinking *Qwen3-4B* checkpoint, rather than from the *Qwen3-4B-Instruct* checkpoint used for the Qwen3 family in the main text (Section 4.1). Initializing both families from an Instruct checkpoint keeps the main-text comparison structurally symmetric; the hybrid-initialized variant is reported here for reference.

Table 4: Results with the native hybrid-thinking *Qwen3-4B* backbone on MATH500 and AIME24. “Hybrid” is the off-the-shelf checkpoint; “Ours” is PLE initialized from it. No SFT-only baseline was run on this backbone.

Type	Mode	MATH500			AIME24		
		Acc.	Len.	#Refl./Ans.	Acc.	Len.	#Refl./Ans.
Hybrid	Think	92.02	4679.54	19.60	61.67	11595.51	45.85
	No-think	82.22	1004.13	0.12	20.67	4636.37	2.54
Ours	Think	94.8	6050.45	3.56	60.00	31733.18	7.02
	No-think	80.80	676.78	0.13	40.0	5597.0	0.39

Table 5: Results with the native hybrid-thinking *Qwen3-4B* backbone on MMLU-STEM and GPQA-Diamond. Same setup as Table 4.

Type	Mode	MMLU-STEM			GPQA-Diamond		
		Acc.	Len.	#Refl./Ans.	Acc.	Len.	#Refl./Ans.
Hybrid	Think	90.26	2338	16.82	53.13	7615	67.01
	No-think	84.49	612	0.01	42.83	1504	0.08
Ours	Think	92.22	1041	0	48.00	25741	5.14
	No-think	83.70	253	0.01	40.00	402	0.00

A.3 Additional Main-Results Figures

Composite accuracy/length/reflective-token figures for the remaining three benchmarks not already shown in Figure 3 (AIME24, main text) or Figure 6 (MATH500, above), in the same format. “Instruct” rows in all three figures show each model’s single mode response (no control-token suffix appended).

A.4 Model-Based Judge Validation of the #Refl./Ans. Metric

To confirm that #Refl./Ans.—a keyword-count metric—captures genuine implicit reasoning rather than surface token matches, we apply an external LLM judge (GPT-OSS-20B) to off-the-shelf hybrid-thinking baselines (*not* PLE, to avoid circularity) across three model sizes (Qwen3-4B, Qwen3-8B, Qwen3-14B) and three benchmarks. The judge classifies each response three-way (no / implicit / explicit reasoning). Table 6 reports agreement between the judge’s verdict and #Refl./Ans.: /no.think outputs flagged as leaking by #Refl./Ans. are independently judged as containing implicit reasoning in 92–100% of cases, and /think outputs with high #Refl./Ans. are judged as explicit reasoning in 94–100% of cases. Table 7 further reports a true-negative control: a genuinely non-hybrid model (Qwen2.5-7B-Instruct) shows #Refl. = 0 on every benchmark, and the judge independently and consistently returns “no reasoning,” confirming the metric does not fire on outputs that are already direct. Because this validation uses independent off-the-shelf baselines rather than PLE itself, it establishes that #Refl./Ans. is a valid leakage signal in general, which then applies to our own results without circularity.

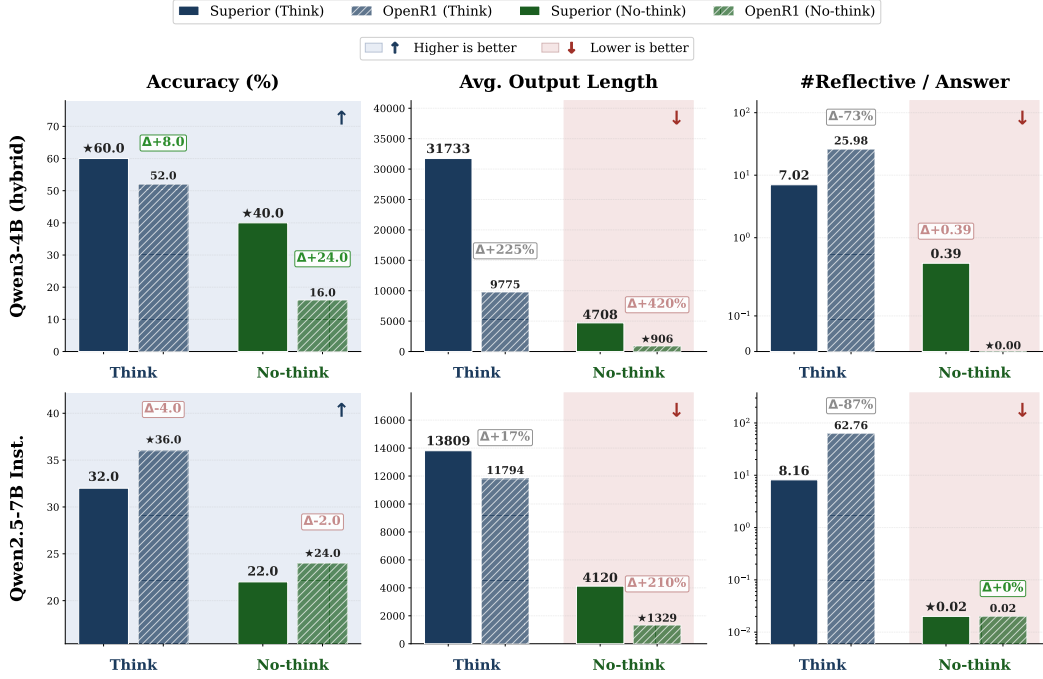


Figure 7: Ablation on training dataset (AIME24) with the native hybrid-thinking *Qwen3-4B* backbone. “Superior”: PLE trained on our high-difficulty dataset with longer CoT traces; “OpenR1”: trained on simpler data. Results for *Qwen3-4B* (top) and *Qwen2.5-7B* (bottom). On the stronger *Qwen3-4B* backbone, Superior yields higher accuracy and fewer /think reflective tokens (7.02 vs. 26), suggesting higher-quality data enables more efficient reasoning. On the weaker *Qwen2.5-7B*, OpenR1 performs comparably, indicating dataset difficulty must align with backbone capacity. Both maintain near-zero /no_think leakage, though Superior produces slightly more reflective artifacts (0.39 vs. 0 on *Qwen3-4B*), illustrating a leakage–performance trade-off.

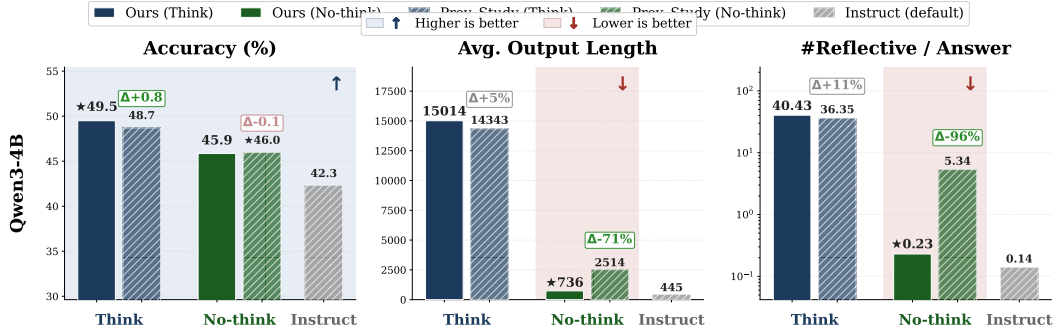


Figure 8: Accuracy, average output length, and per-answer reflective token count on GPQA-Diamond for two PLE instantiations (*Qwen2.5-7B*, top; *Qwen3-4B*, bottom) compared with baselines, in the same format as Figure 3. “Instruct” is shown under its single default-mode response (no control-token suffix appended).

A.5 Extended Related-Work Discussion: Positioning Against Learned-Router and Parameter-Adaptation Methods

PLE is one instance of a broader family of *mode-specific parameter separation* designs, which also includes adapter/LoRA-based approaches and learned or soft-routing Mixture-of-Experts (MoE) methods. We expand here on how PLE differs from each.

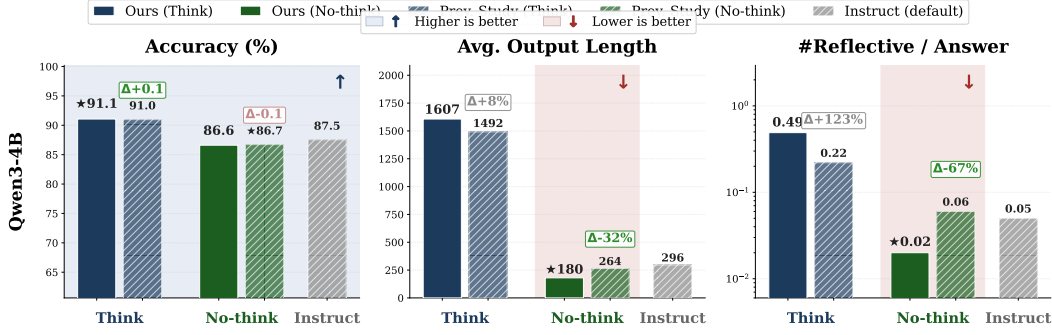


Figure 9: Accuracy, average output length, and per-answer reflective token count on MMLU-STEM for two PLE instantiations (Qwen2.5-7B, top; Qwen3-4B, bottom) compared with baselines, in the same format as Figure 3. “Instruct” is shown under its single default-mode response (no control-token suffix appended).

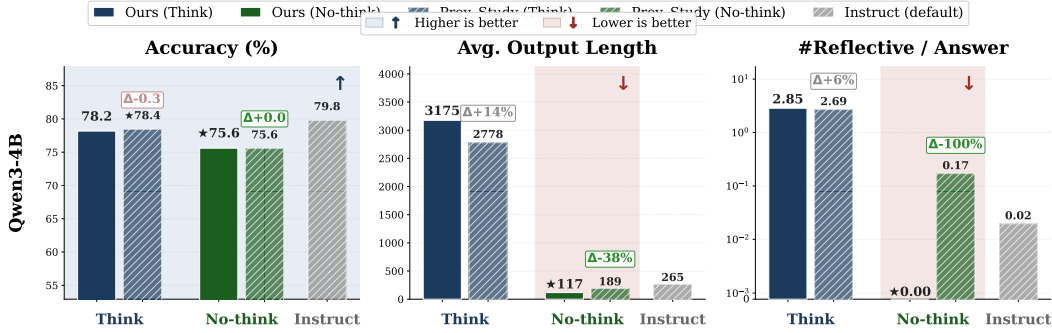


Figure 10: Accuracy, average output length, and per-answer reflective token count on full MMLU for the Qwen3-4B PLE instantiation compared with baselines, in the same format as Figure 3. “Instruct” is shown under its single default-mode response (no control-token suffix appended).

Learned and soft-routing MoE. Token-level MoE architectures (Shazeer et al., 2017; Fedus et al., 2022; Lepikhin et al., 2020; Jiang et al., 2024; Dai et al., 2024) learn a gating network that selects experts per token, typically requiring auxiliary load-balancing losses and incurring routing overhead at every layer. Metis-HOME (Lan et al., 2025) is the closest such system to our setting: it applies a two-expert MoE with a *learned* router to hybrid thinking, trained via a multi-stage RL+SFT pipeline. PLE differs on two axes simultaneously: (1) routing mechanism—PLE uses a deterministic, parameter-free control-token lookup resolved once per sequence, with no learnable gate, no routing supervision, and no balancing loss, whereas Metis-HOME’s router is trained; and (2) modality and scope—Metis-HOME operates on cross-modal vision-language inputs, while PLE is text-only. These differences make a direct head-to-head comparison uninformative: substituting a learned router into our architecture would answer a different question (*which router design is best?*) than the one this paper asks (*is deterministic, parameter-free routing sufficient to achieve clean mode separation?*). We therefore position learned-router MoE methods, including Metis-HOME, as related work rather than as baselines.

Adapters and LoRA-style methods. An alternative approach to mode-specific behavior is to keep a single shared backbone and attach lightweight, mode-conditioned adapter or LoRA modules that are switched in or out per mode. Such methods trade parameter cost for a different failure mode: because the adapted capacity is small relative to the backbone, they may be less able to fully decouple two competing generation objectives (extended deliberation vs. concise direct answering) from the shared feed-forward pathway, which is precisely the interference PLE targets by giving each mode a full, independent MLP. We

Table 6: Model-based judge validation of #Refl./Ans. as a genuine (not surface-level) leakage signal, on off-the-shelf Qwen3 hybrid baselines. Judge: GPT-OSS-20B, three-way no/implicit/explicit verdict.

Benchmark	Model	Mode	Output Len.	#Refl.	Judge verdict (agreement)
MATH500	Qwen3-4B	No-think	969	210	Implicit reasoning (100%)
	Qwen3-4B	Think	4750	163160	Explicit reasoning (98%)
	Qwen3-8B	No-think	1044	113	Implicit reasoning (92%)
	Qwen3-8B	Think	4419	154670	Explicit reasoning (98%)
	Qwen3-14B	No-think	885	131	Implicit reasoning (98%)
	Qwen3-14B	Think	4468	127342	Explicit reasoning (98%)
AIME24	Qwen3-4B	No-think	5022	672	Implicit reasoning (94%)
	Qwen3-4B	Think	11410	21481	Explicit reasoning (94%)
	Qwen3-8B	No-think	4677	24	Implicit reasoning (98%)
	Qwen3-8B	Think	11578	23222	Explicit reasoning (96%)
	Qwen3-14B	No-think	4013	74	Implicit reasoning (98%)
	Qwen3-14B	Think	11341	18814	Explicit reasoning (98%)
GPQA-Diamond	Qwen3-4B	No-think	1504	150	Implicit reasoning (100%)
	Qwen3-4B	Think	7615	132680	Explicit reasoning (100%)
	Qwen3-8B	No-think	1245	196	Implicit reasoning (96%)
	Qwen3-8B	Think	7481	131701	Explicit reasoning (98%)
	Qwen3-14B	No-think	1294	181	Implicit reasoning (98%)
	Qwen3-14B	Think	7246	105493	Explicit reasoning (96%)

Table 7: True-negative control: a non-hybrid Instruct model shows zero reflective tokens and is consistently judged “no reasoning,” confirming the metric does not fire on genuinely direct outputs. Compare to the hybrid baseline (Qwen3-8B), which shows non-zero #Refl. and is judged implicit reasoning on all three benchmarks.

Benchmark	Model	Output Len.	#Refl.	Judge verdict
MATH500	Qwen2.5-7B-Instruct	703	0	No reasoning
MATH500	Qwen3-8B (hybrid)	1044	113	Implicit reasoning
AIME24	Qwen2.5-7B-Instruct	1729	0	No reasoning
AIME24	Qwen3-8B (hybrid)	4677	24	Implicit reasoning
GPQA-Diamond	Qwen2.5-7B-Instruct	775	0	No reasoning
GPQA-Diamond	Qwen3-8B (hybrid)	1245	196	Implicit reasoning

view adapter-based mode separation as a complementary, more parameter-efficient point in the same design space, and a natural direction for follow-up work rather than a claim this paper makes.

General multi-task parameter separation. Beyond hybrid thinking specifically, multi-task learning more broadly has explored task-specific sub-networks, mixture-of-adapters, and conditional computation to reduce cross-task interference. PLE’s contribution within this space is narrow and specific: a minimal, two-expert instantiation targeted at the think/no-think mode boundary, using the control token already present in the hybrid-thinking interface as a free, deterministic routing signal—requiring no additional supervision beyond what standard hybrid-thinking SFT already provides.

Table 8 summarizes these design-space differences along structural properties only; it makes no performance claims and reports no numbers, since we did not run these methods as baselines (see main text above for why).

A.6 Routing Mechanism: Formal Description, Validation, and Worked Example

Each input sequence is assigned a single routing index $r \in \{0, 1\}$ ($r=0$: /no_think $\rightarrow$ expert 0; $r=1$: /think $\rightarrow$ expert 1). This index is shared across all tokens and all layers for that sequence; there is no learnable gate or router parameter anywhere in the model.

Table 8: Structural comparison of mode-specific parameter-separation designs. Properties only—no performance numbers or claims of superiority, since these methods were not run as baselines in this paper (see Section 6).

<i>(a) Routing</i>			
Method	Routing signal	Learnable routing params	Granularity
PLE (ours)	User-specified control token	None	Sequence-level (one decision per sequence)
Learned / soft-routing MoE	Learned gate (trained)	Yes	Token-level (per-token top- k)
Adapters / LoRA-style	Mode flag selects adapter	None for selection; adapter weights are trained	Sequence- or module-level
General multi-task separation	Task/mode identifier	Varies by method	Typically task-level
Metis-HOME (Lan et al., 2025)	Learned query/token-level gate	Yes	Token/query-level
<i>(b) Architecture and cost</i>			
Method	Aux. balancing loss	What is duplicated	
PLE (ours)	None	Full MLP per mode	
Learned / soft-routing MoE	Required	Full MLP per expert	
Adapters / LoRA-style	None	Low-rank adapter, not full MLP	
General multi-task separation	Varies by method	Task-specific sub-network (size varies)	
Metis-HOME (Lan et al., 2025)	Required (MoE-style)	Full expert MLP (2 experts: think/non-think)	

Primary routing: registered control tokens. In the primary setup, `/think` and `/no_think` are registered as dedicated special tokens, and routing is detected by an anchor-bounded, last-occurrence-wins scan over the prompt region. The detector locates the last `<|im_start|>assistant` boundary token pair and searches only before it, structurally excluding the assistant’s own response from influencing routing. Within the prompt, the last control token found is taken as the sequence’s routing intent, which naturally handles multi-turn inputs containing multiple control words. If no control token is found, a safe-mode default applies: strict mode (training and strict evaluation) raises an error, while non-strict mode falls back to `config.default_routing_index`. The index is computed once at the start of `model.forward` and propagated to every layer; during generation, it is detected at prefill and cached for all subsequent decode steps, keeping prefill and decode routing consistent.

Fallback routing: unregistered control tokens. For backbones where the control words cannot be registered as atomic tokens, the same detector operates at the decoded-string level with guarded matching, so a single mechanism covers both regimes. Some tokenizers cannot register `/think` and `/no_think` as atomic tokens; they split into subwords that can collide with ordinary text (e.g., the substring `_th` inside `/think` colliding with a word like `R_th`). The fallback detector operates on the tokenizer .decode string with a guarded regular expression: a negative lookbehind excludes the `</think>` closing tag, a negative lookahead blocks longer substrings such as `/think_hard`, and the alternation order places `/no_think` before `/think` to avoid a prefix-overlap match. Because `tokenizer.decode` maps both the registered-token and the multi-subword case to the same surface string, one detector implementation covers both regimes with no code branching.

Edge cases. The detector explicitly handles: (i) no control token present in the prompt (safe-mode default, see above); (ii) multiple control tokens in a multi-turn prompt (last

occurrence wins); (iii) control-token-like substrings appearing inside the model’s own generated response (structurally excluded, since the scan region is bounded to end at the prompt/response anchor); and (iv) tokenizers that cannot register the control words as atomic tokens (string-level fallback with guarded regex, above).

Table 9: Routing detection robustness, validated on the full training set and a unit-test suite.

Property	Result
Anchor (prompt/response boundary) found — full training set	37,500 / 37,500 (100%)
Detection accuracy (decoded-string + guarded regex)	37,500 / 37,500 (100%)
Adversarial injection false-positives	0 / 37,500
Routing & dispatch unit tests (detection / expert-selection correctness / backward-compat / batching)	42 / 42 pass

Worked example. Table 10 traces the routing decision end-to-end for a representative prompt ending in /think (“...leap year? /think”). The detector locates the last `<|im_start|>assistant` anchor, scans only the prompt region to its left, finds /think as the last control-token match, and sets `routing_index=1`, activating expert 1 (/think) for every layer and decoding step. Swapping only the final control token to /no_think changes exactly the last three rows of the trace—the matched token, the routing index, and the activated expert—while every other step of the pipeline (attention, embeddings, normalization layers, the LM head) is identical.

Table 10: Typical-case token trace for a representative /think-routed prompt.

Stage	Content
Prompt tail	... leap year? /think <code>< im_end ></code>
Anchor (boundary)	<code>< im_start >assistant</code>
Scan region	Everything left of the anchor (the prompt)
Decode → regex	Last match = /think
Decision	<code>routing_index=1</code> → expert 1 (think), all layers, all steps
Output shape	Non-empty <code><think>...</think></code> + full answer

Activated vs. total parameters. Table 11 summarizes the parameter accounting underlying the efficiency claim in Section 4. The routing index selects one expert, and that expert’s MLP is called directly for the whole sequence; there is no per-token gather/scatter and no dispatch machinery. One MLP is evaluated per sequence, so PLE’s per-sequence activated parameters and MLP FLOPs equal a single dense model’s by construction; the only overhead relative to a dense model is the storage of one extra expert MLP.

Table 11: Total vs. activated parameters. PLE shares attention, embeddings, norms, and the LM head across both experts; only one additional expert MLP is stored, and sequence-level routing activates a single expert at inference.

	Total params	Activated params / sequence
Dense backbone	P	P
Two independent models	$2P$	P (per model)
PLE	$P + \Delta_{\text{MLP}} (\approx 1.67P)$	P

A.7 Supplemental Results for Ablation Studies

A.7.1 Supplemental Results for Backbone Weight Ablation

Tables 23 and 25 extend the ablations beyond the mathematical benchmarks and clarify the scope of our claims.

MMLU-STEM favors concise, nearly artifact-free direct answering. Across several initializations, `/no_think` accuracy on MMLU-STEM stays close to `/think` accuracy (typically within a few points, and exceeding it for some initializations) while remaining short and consistently near-artifact-free—reflective-token counts stay near zero in `/no_think` mode across every model we evaluate, including the off-the-shelf baseline. This may simply reflect that many MMLU-STEM questions can be answered correctly without extended reasoning. These results reinforce the practical motivation for hybrid thinking: a well-behaved `/no_think` mode should not be interpreted as a fallback setting.

GPQA-Diamond is more backbone-dependent. The GPQA results are more mixed. `/no_think` leakage remains low, but accuracy depends strongly on the initialization and training dataset. We view this as an important boundary of our claim. PLE addresses *mode interference*; it does not by itself supply missing scientific knowledge or guarantee uniformly strong direct answering on every benchmark. The most stable conclusion on GPQA is therefore about control rather than universal accuracy gains.

Why these supplementary results matter. Taken together with the main math results, the appendix suggests a simple task-dependent picture. Architectural separation is most valuable when a model must clearly choose between “reason deeply” and “answer directly.” On benchmarks that mainly test retrieval or short reasoning, the main benefit of PLE is a clean and efficient `/no_think` mode; on long-horizon mathematical benchmarks, it can also substantially improve `/no_think` accuracy.

A.8 Behavioral Probe of the Instruct Backbone

Qwen3-4B-Instruct exposes a single response mode, with no control token to select between reasoning and direct answering. To characterise how that mode actually behaves, we evaluate the off-the-shelf checkpoint twice: once on the bare prompt, and once with `/no_think` appended.

Table 12: Behavioral probe of the off-the-shelf *Qwen3-4B-Instruct* backbone. “Plain” is the bare prompt; `/no_think` appends the control token used by hybrid-thinking models.

Benchmark	Prompt	Acc. (%)	Avg. Len.	#Refl./Ans.
AIME24	Plain	63.33	9932	2.06
	<code>/no_think</code>	56.67	7950	1.81
MATH500	Plain	94.52	1813	0.36
	<code>/no_think</code>	94.36	1675	0.30
GPQA-Diamond	Plain	42.32	445	0.14
	<code>/no_think</code>	42.02	433	0.10
MMLU-STEM	Plain	87.54	296	0.05
	<code>/no_think</code>	87.58	283	0.04

Two observations follow. First, the backbone reasons at length by default: on AIME24 it emits over 9,000 tokens carrying two reflective tokens per answer, so its single mode is a reasoning mode rather than a direct-answer mode. Second, appending `/no_think` does not give it a direct-answer mode, however slightly but consistently reduces both its performance and reasoning leakage: it emits slightly less tokens and reflective tokens per answer than default mode on all 4 benchmarks. While on other three benchmarks its accuracy moves by at most 0.3 points, on AIME24, the accuracy drops by 6.66%. Both observations are consistent

with its lineage as a non-thinking release derived from a hybrid-thinking base (Yang et al., 2025; Team, 2025b), and they motivate supplying the direct-answer path architecturally rather than through prompting.

A.9 Two-Separate-Models Controlled Ablation

All models below are trained from an identical Qwen3-4B-Base checkpoint on matched data, isolating the effect of deterministic routing from any difference in base weights or data. “Specialist” models are trained on only one mode’s data; “weight-merge” naively averages the two specialists’ weights with no routing mechanism.

Table 13: *Base-PLE* (PLE initialized from Qwen3-4B-Base) vs. single-mode specialists and a naive weight-merge, all trained from an identical Qwen3-4B-Base checkpoint on matched data. Accuracy (%) shown; per-benchmark tables with output length and #Refl./Ans. follow below.

Model	MATH500	AIME24	GPQA-Diamond
Think-only specialist (think mode)	80.50	22.00	30.30
No_think-only specialist (no_think mode)	64.30	4.70	31.00
Weight-merge of the two specialists	66.50	9.30	32.90
<i>Base-PLE, think</i>	82.40	21.33	34.00
<i>Base-PLE, no_think</i>	64.50	3.33	34.00

Table 14: AIME24 — PLE vs. single-mode specialists and weight-merge.

Model	Acc. (%)	Avg. Len.	#Refl./Ans.
Qwen3-4B-Base + think-only data	22.00	28895	13.53
Qwen3-4B-Base + no_think-only data	4.70	941	0.07
Weight-merge of the two specialists	9.30	4062	3.76
<i>Base-PLE, think</i>	21.33	27698	12.37
<i>Base-PLE, no_think</i>	3.33	1244	0.09

Table 15: MATH500 — PLE vs. single-mode specialists and weight-merge.

Model	Acc. (%)	Avg. Len.	#Refl./Ans.
Qwen3-4B-Base + think-only data	80.50	10390	4.73
Qwen3-4B-Base + no_think-only data	64.30	358	0.02
Weight-merge of the two specialists	66.50	1044	2.12
<i>Base-PLE, think</i>	82.40	8514	4.64
<i>Base-PLE, no_think</i>	64.50	368	0.02

Table 16: GPQA-Diamond — PLE vs. single-mode specialists and weight-merge.

Model	Acc. (%)	Avg. Len.	#Refl./Ans.
Qwen3-4B-Base + think-only data	30.30	19742	24.86
Qwen3-4B-Base + no_think-only data	31.00	457	0.21
Weight-merge of the two specialists	32.90	1152	2.95
<i>Base-PLE, think</i>	34.00	18433	37.33
<i>Base-PLE, no_think</i>	34.00	397	0.02

The initialization-equivalence test (12/12 prompt-mode pairs byte-identical; 108/108 parameter tensors numerically equal at init) and the parameter accounting ($P + \Delta_{\text{MLP}} \approx 1.67P$)

for PLE vs. 2P for two independent models, a $\approx 0.33P$ saving) are reported in Appendix A.6 (Table 11); we do not repeat the derivation here to avoid duplication.

A.10 Generalization: Llama-3.1-8B and Full MMLU

To test generalization beyond the Qwen family and beyond math/STEM tasks, we apply PLE to *Llama-3.1-8B* (trained on OpenR1 data at the 20k scale, matching the data scale of the SFT-only comparison recipe) and evaluate on full MMLU (57 subjects, 14,042 questions) with the Qwen3-4B backbone used throughout the main paper. In Tables 17 to 20, the “Instruct” row is the off-the-shelf Llama-3.1-8B-Instruct model (no-think only, no hybrid capability); “+SFT only” is a dense two-phase SFT hybrid recipe following Wang et al. (2025) at the matched 20k data scale; “PLE” is our architecture applied to the same backbone and data.

Table 17: Cross-family generalization (Llama-3.1-8B): MATH500.

Model	Mode	Acc. (%)
Llama-3.1-8B-Instruct	No-think	49.3
Llama-3.1-8B-Instruct + SFT only	No-think	47.2
Llama-3.1-8B-Instruct + SFT only	Think	68.6
Llama-3.1-8B-PLE	No-think	56.0
Llama-3.1-8B-PLE	Think	66.4

Table 18: Cross-family generalization (Llama-3.1-8B): AIME24.

Model	Mode	Acc. (%)
Llama-3.1-8B-Instruct	No-think	6.0
Llama-3.1-8B-Instruct + SFT only	No-think	4.3
Llama-3.1-8B-Instruct + SFT only	Think	7.0
Llama-3.1-8B-PLE	No-think	12.0
Llama-3.1-8B-PLE	Think	18.0

Table 19: Cross-family generalization (Llama-3.1-8B): GPQA-Diamond.

Model	Mode	Acc. (%)
Llama-3.1-8B-Instruct	No-think	30.5
Llama-3.1-8B-Instruct + SFT only	No-think	26.7
Llama-3.1-8B-Instruct + SFT only	Think	36.3
Llama-3.1-8B-PLE	No-think	32.0
Llama-3.1-8B-PLE	Think	38.0

Table 20: Cross-family generalization (Llama-3.1-8B): MMLU-STEM.

Model	Mode	Acc. (%)
Llama-3.1-8B-Instruct	No-think	53.0
Llama-3.1-8B-Instruct + SFT only	No-think	53.4
Llama-3.1-8B-Instruct + SFT only	Think	77.3
Llama-3.1-8B-PLE	No-think	76.7
Llama-3.1-8B-PLE	Think	78.9

A.11 Ruling Out Data-Construction Bias

Because PLE and its dense SFT baseline (the “SFT-only” row in Table 2) are trained on the identical Qwen3-235B-distilled dataset—same teacher, same filtering pipeline, same

Table 21: Full MMLU (57 subjects, 14,042 questions), Qwen3-4B backbone; the same results are shown graphically in Figure 10. †: PLE think accuracy is over the full question set; average length and #Refl./response for PLE think are over a 90.6% subset (56/57 subjects; professional_law partially truncated).

Model	Mode	Acc. (%)	Avg. Len.	#Refl./response
No_think-only SFT (single mode)	No-think	46.2	268	0.001
Think-only SFT (single mode)	Think	75.2	3723	36.804
Qwen3-4B (off-the-shelf hybrid)	No-think	75.1	583	0.003
Qwen3-4B (off-the-shelf hybrid)	Think	79.8	1916	13.500
SFT-recipe (Wang et al., 2025)	No-think	57.1	1548	0.646
SFT-recipe (Wang et al., 2025)	Think	76.9	2823	19.548
PLE (ours)	No-think	73.4	164	0.003
PLE (ours)	Think	77.3†	2349†	4.803†

examples—any teacher-style bias in the /no_think targets would inflate both equally, and cannot selectively favor PLE. As a direct test, we further retrain PLE on a newer, more aggressively *condensed* /no_think dataset: a stronger model rewrites each full chain-of-thought into a concise, reasoning-free answer, an even more aggressive direct-answer construction than the original filtering pipeline. If teacher-style distillation were inflating /no_think performance, this more aggressively condensed data should help further; instead, Table 22 shows it makes /no_think accuracy substantially *worse* (AIME24: 40.0% $\rightarrow$ 3.3%; MATH500: 80.8% $\rightarrow$ 64.3%), with /think accuracy also somewhat lower. This is the opposite of what a teacher-bias explanation predicts, and is consistent with PLE’s benefit being architectural rather than a data-construction artifact. We note that identifying the single best /no_think-construction recipe is outside this paper’s scope; the architecture is a separable, independently improvable component.

Table 22: Data-construction-bias ablation: PLE retrained on the original vs. a more aggressively condensed /no_think dataset (Qwen3-4B-Base). Condensation worsens /no_think accuracy, ruling out teacher-style bias inflation as an explanation for PLE’s gains.

Benchmark	Training data	Mode	Acc. (%)
AIME24	Original data (as in the paper)	Think	60.0
	Original data (as in the paper)	No-think	40.0
	Condensed /no_think data (new)	Think	53.3
	Condensed /no_think data (new)	No-think	3.3
MATH500	Original data (as in the paper)	Think	94.8
	Original data (as in the paper)	No-think	80.8
	Condensed /no_think data (new)	Think	91.0
	Condensed /no_think data (new)	No-think	64.3

A.12 Supplemental Results for Dataset Ablation

A.13 Broader Impact and Future Applications

Agents. Agent trajectories interleave routine steps—parsing a tool result, filling a template, deciding which file to open—with a much smaller number of genuinely deliberative ones (Yao et al., 2023). Because agent benchmarks score long horizons in which each step conditions the next, residual leakage compounds rather than staying local, so a /no_think mode that can be trusted lets a system spend tokens only where deliberation is actually needed (Liu et al., 2024a; Yang et al., 2026a). The same property helps downstream: records written back into long-term memory stay compact when the underlying responses are direct rather than padded with abandoned reasoning (Shinn et al., 2023; Long et al., 2026; Chen et al., 2026). It also composes with how agent capabilities are increasingly packaged—a

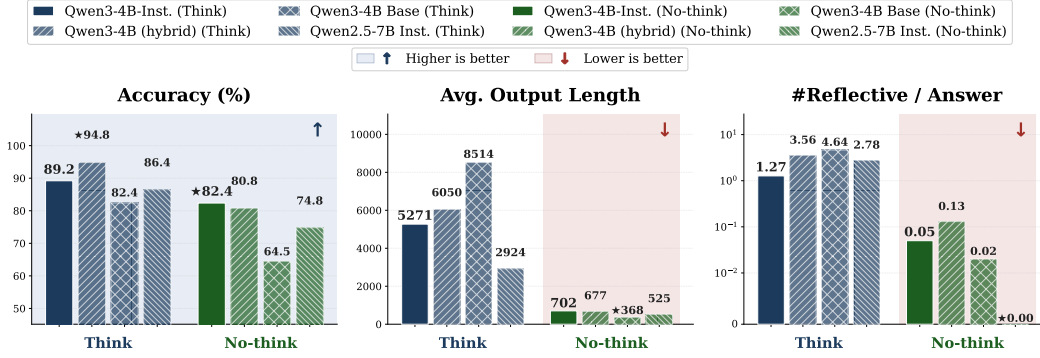


Figure 11: Ablation on backbone choice (MATH500); same models as Figure 4. PLE is initialized from “Qwen2.5-7B-Instruct” (pure instruction-tuned weights), “Qwen/Qwen3-4B-Instruct-2507”, “Qwen3-4B Base” (raw pretrained weights), or “Qwen3-4B (hybrid)” (hybrid-thinking weights). Residual /no_think leakage again orders with each backbone’s own reasoning behavior (0.00 for Qwen2.5-7B-Instruct < 0.05 for Qwen/Qwen3-4B-Instruct-2507 < 0.13 for the hybrid variant). Unlike AIME24, the pretrained base stays usable on this easier benchmark, though at the lowest /no_think accuracy.

Table 23: Backbone-weight ablation under **superior-reasoning** training data on MMLU-STEM and GPQA-Diamond. We report accuracy (Acc., %), output length (Len.), and reflective tokens per answer (#Refl./Ans.).

Model	Mode	MMLU-STEM			GPQA-Diamond		
		Acc.	Len.	#Refl./Ans.	Acc.	Len.	#Refl./Ans.
Qwen3-4B Base	Think	94.44	-	-	30.00	-	-
	No-think	88.89	-	-	44.00	-	-
Qwen3-4B	Think	91.11	794	6.36	42.00	8268	3.26
	No-think	93.33	117	0.06	26.00	298	0.00
Qwen3-4B Instruct	Think	91.06	1606.65	0.49	49.49	15013.74	40.43
	No-think	86.61	180.19	0.02	45.86	736.34	0.23
Qwen2.5-7B Instruct	Think	88.89	429	8.16	34.00	7044	4.44
	No-think	92.22	139	0.02	30.00	3857	0.00

skill library can declare the reasoning mode a skill expects at call time (Wang et al., 2024; Yang et al., 2026b), and persistent multi-agent workspaces, where many calls are routine bookkeeping, benefit from the same effect (Wang, 2026).

Conventional inference serving. In interactive deployment most everyday queries want a short, direct answer, and a /no_think mode that reliably delivers one lowers both the cost of a request and the time until the user has a complete response, while /think remains available for the queries that need it. This acts on a different axis from most inference-time work: it reduces how many tokens are generated in the first place rather than the cost of producing each one. IO-aware exact attention and lossless weight compression leave the token stream untouched (Dao et al., 2022; Zhang et al., 2025b), as does post-training weight quantization or low-rank compression (Frantar et al., 2023; Lin et al., 2024; Wang et al., 2026; Li et al., 2026), though quantization can also shift model behaviour rather than merely compress it (Fu et al., 2025). Cache-side methods instead target the memory that accumulates as generation proceeds (Liu et al., 2024b; Hariri et al., 2026b; Li et al., 2025b). Because these axes are largely independent, an architectural mode boundary can be stacked with any of them.

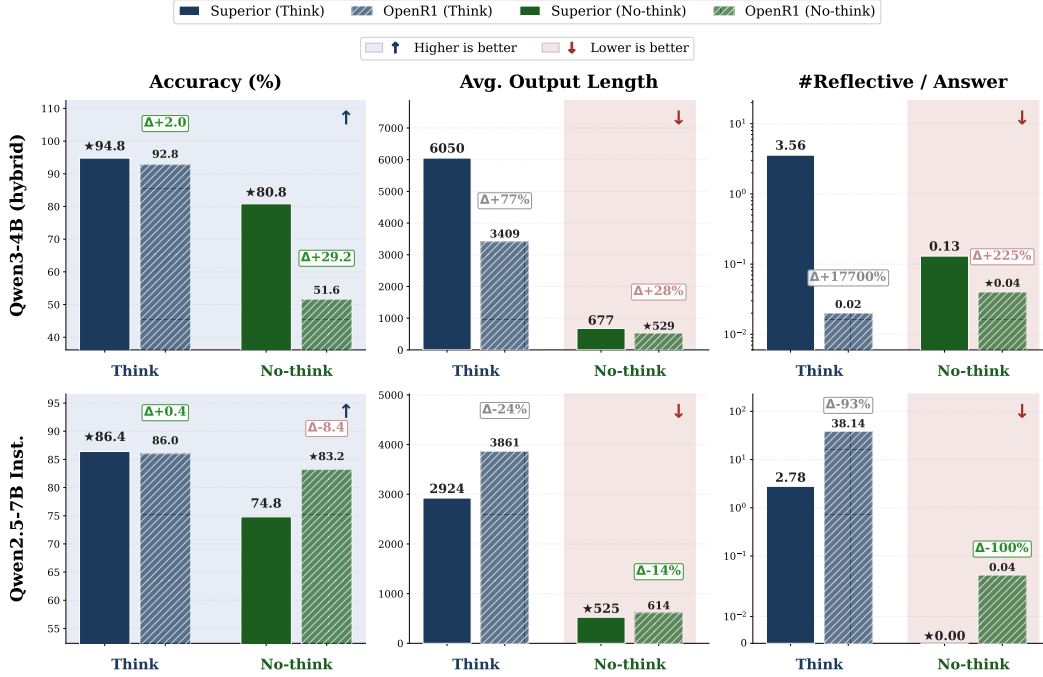


Figure 12: Ablation on training dataset (MATH500) with the native hybrid-thinking *Qwen3-4B* backbone (top) and *Qwen2.5-7B-Instruct* (bottom). “Superior”: PLE trained on our high-difficulty dataset with longer CoT traces; “OpenR1”: trained on simpler data. On the stronger *Qwen3-4B* backbone Superior wins in both modes, while on *Qwen2.5-7B* OpenR1 remains competitive—the same capacity-dependent pattern seen on AIME24 (Figure 7).

Table 24: Dataset ablation across different backbone weights. Leftmost column groups results by dataset (**OpenR1** vs. **superior-reasoning 54k**); within each group, we report think/no-think performance for each backbone on MATH500 and AIME24. The *Qwen3-4B* rows use the native hybrid-thinking checkpoint.

Dataset	Backbone	Mode	MATH500			AIME24		
			Acc.	Len.	#Refl./Ans.	Acc.	Len.	#Refl./Ans.
OpenR1	Qwen3-4B Base	Think	80.00	16384	28.80	50.00	16384	19.40
		No-think	56.00	16262	0.00	10.00	16384	0.00
	Qwen3-4B (hybrid)	Think	92.80	3409	0.02	52.00	9775	25.98
		No-think	51.60	529	0.04	16.00	906	0.00
Superior-reasoning 54k	Qwen2.5-7B Instruct	Think	86.00	3861	38.14	36.00	11794	62.76
		No-think	83.20	614	0.04	24.00	1329	0.02
	Qwen3-4B Base	Think	90.00	15769	7.00	20.00	16384	5.00
		No-think	74.00	16224	0.00	30.00	16384	0.00
	Qwen3-4B (hybrid)	Think	94.80	6050	3.56	60.00	31733	7.02
		No-think	80.80	677	0.13	40.00	4708	0.39
	Qwen2.5-7B Instruct	Think	86.40	2924	2.78	32.00	13809	8.16
		No-think	74.80	525	0.00	22.00	4120	0.02

Model architecture. Conditional computation is the established route to spending capacity selectively, activating a subset of parameters per token through learned sparse experts (Fedus et al., 2022; Jiang et al., 2024). PLE occupies a deliberately minimal point in that space: two experts, no router parameters, and a routing decision fixed once per sequence by a token the interface already exposes. Methods that instead shorten reasoning after the de-

Table 25: Dataset ablation across different backbone weights on MMLU-STEM and GPQA-Diamond. Leftmost column groups results by dataset (**OpenR1** vs. **superior-reasoning**); within each group, we report think/no-think performance for each backbone. The *Qwen3-4B* rows use the native hybrid-thinking checkpoint.

Dataset	Backbone	Mode	MMLU-STEM			GPQA-Diamond		
			Acc.	Len.	#Refl./Ans.	Acc.	Len.	#Refl./Ans.
OpenR1	Qwen3-4B	Think	33.33	16384	19.40	8.00	16384	14.10
	Base	No-think	83.33	16384	0.00	24.00	16384	0.00
	Qwen3-4B	Think	88.89	999	25.98	66.00	5372	16.88
	(hybrid)	No-think	83.33	160	0.00	30.00	603	0.00
	Qwen2.5-7B	Think	77.78	848	62.76	6.00	1821	41.00
	Instruct	No-think	88.89	352	0.02	40.00	871	0.00
Superior-reasoning	Qwen3-4B	Think	94.44	-	-	30.00	-	-
	Base	No-think	88.89	-	-	44.00	-	-
	Qwen3-4B	Think	91.11	794	6.36	42.00	8268	3.26
	(hybrid)	No-think	93.33	117	0.06	26.00	298	0.00
	Qwen2.5-7B	Think	88.89	429	8.16	34.00	7044	4.44
	Instruct	No-think	92.22	139	0.02	30.00	3857	0.00

cision to reason has been made (Sui et al., 2025) are complementary, since they act on the length of a /think trace rather than on the boundary between modes. Both sit inside the broader question of how inference-time compute should be allocated and compared across regimes (Hariri et al., 2026a).

Post-training. The /no.think behaviour PLE produces is a property of post-training, not of the architecture alone. Post-training effects are known to interact across domains in order-sensitive ways, and training-level mitigation of leakage is itself sensitive to data scale and mixture (Yang et al., 2026c; Wang et al., 2025). Mode separation should therefore be re-measured whenever the training mixture changes, rather than assumed to transfer from one recipe to another.

B Formal Architecture Specification

Path-locked dual-expert decoder Consider a standard pre-norm decoder layer:

$$\mathbf{h}_{\text{attn}}^{(l)} = \mathbf{h}^{(l)} + \text{Attn}^{(l)}\left(\text{LN}_1^{(l)}(\mathbf{h}^{(l)})\right); \mathbf{h}^{(l+1)} = \mathbf{h}_{\text{attn}}^{(l)} + \text{MLP}^{(l)}\left(\text{LN}_2^{(l)}(\mathbf{h}_{\text{attn}}^{(l)})\right). \quad (1)$$

PLE changes only the second part. Each decoder MLP is replaced by a pair of experts, and the selected route $r \in \{0, 1\}$ determines which expert is active:

$$\mathbf{h}^{(l+1)} = \mathbf{h}_{\text{attn}}^{(l)} + \text{MLP}_r^{(l)}\left(\text{LN}_2^{(l)}(\mathbf{h}_{\text{attn}}^{(l)})\right), \quad (2)$$

where $r = 0$ denotes the /no_think expert and $r = 1$ denotes the /think expert. For the Qwen backbones used in our main experiments, each expert preserves the source model’s original SwiGLU-style MLP parameterization:

$$\text{MLP}_r^{(l)}(\mathbf{x}) = W_{\text{down}}^{(l,r)}\left(\phi\left(W_{\text{gate}}^{(l,r)}\mathbf{x}\right)\right) \odot W_{\text{up}}^{(l,r)}\mathbf{x}, \quad (3)$$

where $\phi(\cdot)$ is the activation function and $\odot$ is elementwise multiplication. The two experts are structurally identical; they differ only in parameters and in which control mode activates them. All non-MLP components remain shared across modes, including self-attention, positional encoding, normalization, token embeddings, and the LM head. We duplicate only the MLP because the two modes should share the same contextualization and stored knowledge, but need not share the final feed-forward transformation that turns those representations into long-form reasoning or concise direct answers. This makes PLE a targeted intervention rather than a full model duplication scheme. **Deterministic control-token routing** PLE does not use a learned router. Instead, routing is determined by control tokens in the input. We register two tokenizer-level special tokens, /no_think and /think, each represented by a single token ID. Given an input sequence $\mathbf{x}$, the routing function is

$$r(\mathbf{x}) = \begin{cases} 1, & \text{if the final control token in } \mathbf{x} \text{ is /think,} \\ 0, & \text{if the final control token in } \mathbf{x} \text{ is /no_think,} \\ r_{\text{default}}, & \text{if no control token is present,} \end{cases} \quad (4)$$

where $r_{\text{default}} = 0$ routes to the /no_think expert. The *last control token wins* rule serves two purposes. First, it makes routing fully deterministic and interpretable from the prompt alone. Second, it allows a later template or system token to override an earlier user-supplied token without introducing any learned gating behavior. During autoregressive generation, the route is determined from the accumulated prompt at the first decoding step and then cached and reused for all later steps. This ensures that the active expert remains fixed throughout decoding. Our current implementation uses one scalar routing index per forward pass. Accordingly, we do not mix /think and /no_think examples within the same batch; each forward pass uses a single mode. This keeps the routing logic simple while preserving the intended turn-level control semantics.

Routing-conditioned supervised fine-tuning We fine-tune PLE with standard causal language modeling loss on chat-formatted examples tagged with either /think or /no_think. Let $\mathcal{B}_r$ denote a minibatch routed to mode $r \in \{0, 1\}$. The training objective is

$$\mathcal{L}(\theta; \mathcal{B}_r) = - \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{B}_r} \sum_{t=1}^{|\mathbf{y}|} \log p_{\theta}(y_t \mid \mathbf{x}, \mathbf{y}_{<t}; r). \quad (5)$$

Let θ_{sh} denote the shared parameters and θ_0, θ_1 denote the two expert parameter sets. Because only the routed expert is executed, for any batch using route r ,

$$\frac{\partial \mathcal{L}}{\partial \theta_{1-r}} = 0, \quad \frac{\partial \mathcal{L}}{\partial \theta_r} \neq 0, \quad \frac{\partial \mathcal{L}}{\partial \theta_{\text{sh}}} \neq 0. \quad (6)$$

The inactive expert therefore receives no gradient, while the shared backbone is updated by both modes. This is the central mechanism of PLE: it removes direct parameter sharing between the two MLP pathways while still allowing both modes to benefit from a common representational substrate. At the same time, because attention and output layers remain shared, PLE should be understood as *targeted architectural separation*, not as full model duplication.

C Theoretical account: Path-Lock as partial block-diagonalization

This section formalizes PLE as a partially decoupled two-objective optimization problem. The derivation is guided by three observations from prior work: (i) multi-task learning is inherently multi-objective and can suffer from conflicting gradients (Sener & Koltun, 2018; Yu et al., 2020; Liu et al., 2021); (ii) expert routing is useful when different objectives prefer different submodels (Jordan & Jacobs, 1994; Ma et al., 2018); and (iii) Transformer feed-forward layers are a privileged locus for token-level prediction shaping (Geva et al., 2021; 2022; Dai et al., 2022).

Setup. We model hybrid thinking as conditional language modeling with an observed mode variable $r \in \{0, 1\}$, where $r = 0$ denotes /no_think and $r = 1$ denotes /think. Let

$$\begin{aligned}\alpha &= \text{shared parameters (embeddings, attention, norms, LM head),} \\ \beta_0 &= \text{no-think expert parameters,} \\ \beta_1 &= \text{think expert parameters.}\end{aligned}$$

Thus the full parameter vector is

$$\theta = (\alpha, \beta_0, \beta_1).$$

For an example (x, y, r) , the PLE model defines

$$p_\theta(y \mid x, r) = \prod_{t=1}^{|y|} p_\theta(y_t \mid x, y_{<t}; \alpha, \beta_r).$$

Observed-route formulation. Introduce an expert variable $z \in \{0, 1\}$. Because routing is deterministic from the control token c ,

$$p(z \mid c) = \mathbf{1}\{z = r(c)\},$$

where $r(c)$ is the route selected by the control-token rule. Therefore

$$\begin{aligned}p_\theta(y \mid x, c) &= \sum_{z \in \{0, 1\}} p(z \mid c) p_\theta(y \mid x, z) \\ &= \sum_{z \in \{0, 1\}} \mathbf{1}\{z = r(c)\} p_\theta(y \mid x, z) \\ &= p_\theta(y \mid x, r(c)).\end{aligned}$$

Hence the route variable is *observed*, not latent. In this observed-route setting, a learned router is not statistically necessary for identifying the routed conditional model; outside this setting (e.g., if the route is unobserved or must be inferred at test time), a learned router may still be useful or necessary.

Modeling assumption. The reduction from the control token c to the observed route $r(c)$ assumes that, within this theoretical abstraction, c affects the conditional distribution only through route selection:

$$p_\theta(y \mid x, c, z) = p_\theta(y \mid x, z).$$

Equivalently, once the route z is fixed, the control token carries no additional route-independent signal for prediction. If, in the implemented model, c is also processed by the shared backbone through a distinct embedding or any other route-specific component, then the identities below should be read as an abstraction that isolates routing from those additional effects.

Dataset decomposition. Let

$$\mathcal{D} = \mathcal{D}_0 \sqcup \mathcal{D}_1,$$

where $\mathcal{D}_0$ contains no-think examples and $\mathcal{D}_1$ contains think examples. Define mode frequencies

$$\pi_r = \frac{|\mathcal{D}_r|}{|\mathcal{D}|}, \quad \pi_0 + \pi_1 = 1.$$

For each mode, define the expected autoregressive loss

$$\mathcal{L}_r(\alpha, \beta_r) = \mathbb{E}_{(x,y) \sim \mathcal{D}_r} \left[- \sum_{t=1}^{|y|} \log p_\theta(y_t \mid x, y_{<t}; \alpha, \beta_r) \right]. \quad (7)$$

Then the full PLE objective is

$$\mathcal{L}_{\text{PLE}}(\alpha, \beta_0, \beta_1) = \mathbb{E}_{(x,y,r) \sim \mathcal{D}} \left[- \sum_{t=1}^{|y|} \log p_\theta(y_t \mid x, y_{<t}; \alpha, \beta_r) \right] \quad (8)$$

$$= \sum_{r \in \{0,1\}} \pi_r \mathcal{L}_r(\alpha, \beta_r) \quad (9)$$

$$= \pi_0 \mathcal{L}_0(\alpha, \beta_0) + \pi_1 \mathcal{L}_1(\alpha, \beta_1). \quad (10)$$

Unless stated otherwise, all exact statements below refer to the example-weighted objective in (10), with $\pi_r = |\mathcal{D}_r|/|\mathcal{D}|$. The later length-asymmetry discussion explicitly switches to a token-summed training heuristic. If training instead uses token-weighted normalization, or adds auxiliary router losses, cross-expert regularizers, or other route-coupling terms, then the gradient and Hessian identities below acquire additional additive terms.

Proposition C.1 (Exact expert-gradient decoupling). *Under (10),*

$$\begin{aligned} \nabla_{\beta_0} \mathcal{L}_{\text{PLE}} &= \pi_0 \nabla_{\beta_0} \mathcal{L}_0, \\ \nabla_{\beta_1} \mathcal{L}_{\text{PLE}} &= \pi_1 \nabla_{\beta_1} \mathcal{L}_1, \\ \nabla_\alpha \mathcal{L}_{\text{PLE}} &= \pi_0 \nabla_\alpha \mathcal{L}_0 + \pi_1 \nabla_\alpha \mathcal{L}_1. \end{aligned}$$

In particular,

$$\nabla_{\beta_0} \mathcal{L}_1 = 0, \quad \nabla_{\beta_1} \mathcal{L}_0 = 0.$$

Proof. Differentiate (10) with respect to β_0 :

$$\begin{aligned} \nabla_{\beta_0} \mathcal{L}_{\text{PLE}} &= \nabla_{\beta_0} \left(\pi_0 \mathcal{L}_0(\alpha, \beta_0) + \pi_1 \mathcal{L}_1(\alpha, \beta_1) \right) \\ &= \pi_0 \nabla_{\beta_0} \mathcal{L}_0 + \pi_1 \underbrace{\nabla_{\beta_0} \mathcal{L}_1}_{=0} \\ &= \pi_0 \nabla_{\beta_0} \mathcal{L}_0. \end{aligned}$$

The zero term appears because $\mathcal{L}_1$ depends on β_1 but not on β_0 .

Similarly,

$$\begin{aligned} \nabla_{\beta_1} \mathcal{L}_{\text{PLE}} &= \nabla_{\beta_1} \left(\pi_0 \mathcal{L}_0(\alpha, \beta_0) + \pi_1 \mathcal{L}_1(\alpha, \beta_1) \right) \\ &= \pi_0 \underbrace{\nabla_{\beta_1} \mathcal{L}_0}_{=0} + \pi_1 \nabla_{\beta_1} \mathcal{L}_1 \\ &= \pi_1 \nabla_{\beta_1} \mathcal{L}_1. \end{aligned}$$

For the shared parameters α , both mode losses depend on α , so

$$\nabla_\alpha \mathcal{L}_{\text{PLE}} = \pi_0 \nabla_\alpha \mathcal{L}_0 + \pi_1 \nabla_\alpha \mathcal{L}_1.$$

□

Pointwise consequence. For a single example (x_i, y_i, r_i) , define

$$\ell_i(\alpha, \beta_0, \beta_1) = - \sum_{t=1}^{|y_i|} \log p_\theta(y_{i,t} \mid x_i, y_{i,<t}; \alpha, \beta_{r_i}).$$

Then for any $k \in \{0, 1\}$,

$$\frac{\partial \ell_i}{\partial \beta_k} = \begin{cases} - \sum_{t=1}^{|y_i|} \frac{\partial}{\partial \beta_{r_i}} \log p_\theta(y_{i,t} \mid x_i, y_{i,<t}; \alpha, \beta_{r_i}), & k = r_i, \\ 0, & k \neq r_i. \end{cases} \quad (11)$$

So the inactive expert receives *exactly zero* gradient on that example.

All curvature-based statements below assume the relevant local second derivatives exist and that differentiation may be interchanged with expectation in the usual way.

Partial block-diagonalization of the curvature. Let

$$H_{ab}^{(r)} := \nabla_{a,b}^2 \mathcal{L}_r, \quad a, b \in \{\alpha, \beta_0, \beta_1\}.$$

Because $\mathcal{L}_0$ does not depend on β_1 and $\mathcal{L}_1$ does not depend on β_0 ,

$$\nabla_{\beta_0\beta_1}^2 \mathcal{L}_{\text{PLE}} = 0, \quad \nabla_{\beta_1\beta_0}^2 \mathcal{L}_{\text{PLE}} = 0.$$

Hence the Hessian takes the form

$$\nabla^2 \mathcal{L}_{\text{PLE}} = \begin{bmatrix} \pi_0 H_{\alpha\alpha}^{(0)} + \pi_1 H_{\alpha\alpha}^{(1)} & \pi_0 H_{\alpha\beta_0}^{(0)} & \pi_1 H_{\alpha\beta_1}^{(1)} \\ \pi_0 H_{\beta_0\alpha}^{(0)} & \pi_0 H_{\beta_0\beta_0}^{(0)} & 0 \\ \pi_1 H_{\beta_1\alpha}^{(1)} & 0 & \pi_1 H_{\beta_1\beta_1}^{(1)} \end{bmatrix}. \quad (12)$$

Equation (12) is the precise sense in which PLE *partially block-diagonalizes* the hybrid-thinking optimization problem: the expert blocks are exactly decoupled from each other, while both remain coupled to the shared backbone.

This block structure is exact only for the direct expert–expert interactions in the stated objective. It does not imply full disentanglement over training, because both modes remain coupled through the shared parameters α : an update from one mode can change α , which in turn changes future gradients for the other mode.

Dense hybrid training as a conflicting shared-objective problem. Now consider a dense baseline with the same shared backbone α but a *single* shared MLP parameter vector β . Its objective is

$$\mathcal{L}_{\text{dense}}(\alpha, \beta) = \pi_0 \mathcal{L}_0(\alpha, \beta) + \pi_1 \mathcal{L}_1(\alpha, \beta).$$

Let

$$g_0 := \nabla_\beta \mathcal{L}_0(\alpha, \beta), \quad g_1 := \nabla_\beta \mathcal{L}_1(\alpha, \beta).$$

Then the dense expert gradient is

$$\nabla_\beta \mathcal{L}_{\text{dense}} = \pi_0 g_0 + \pi_1 g_1.$$

Proposition C.2 (One-step interference criterion). *Consider one SGD step with learning rate $\eta > 0$:*

$$\Delta \beta_{\text{dense}} = -\eta(\pi_0 g_0 + \pi_1 g_1).$$

Then the first-order change in mode-0 loss is

$$\mathcal{L}_0(\beta + \Delta \beta_{\text{dense}}) = \mathcal{L}_0(\beta) - \eta \left(\pi_0 \|g_0\|^2 + \pi_1 g_0^\top g_1 \right) + O(\eta^2). \quad (13)$$

Therefore the dense update increases the mode-0 loss at first order whenever

$$g_0^\top g_1 < -\frac{\pi_0}{\pi_1} \|g_0\|^2. \quad (14)$$

The same statement holds symmetrically for mode 1.

Proof. Take a first-order Taylor expansion of $\mathcal{L}_0$ around β :

$$\mathcal{L}_0(\beta + \Delta\beta_{\text{dense}}) = \mathcal{L}_0(\beta) + g_0^\top \Delta\beta_{\text{dense}} + O(\|\Delta\beta_{\text{dense}}\|^2).$$

Substitute the dense update:

$$\begin{aligned} g_0^\top \Delta\beta_{\text{dense}} &= g_0^\top [-\eta(\pi_0 g_0 + \pi_1 g_1)] \\ &= -\eta\pi_0 g_0^\top g_0 - \eta\pi_1 g_0^\top g_1 \\ &= -\eta\pi_0 \|g_0\|^2 - \eta\pi_1 g_0^\top g_1. \end{aligned}$$

Substituting back yields (13). The dense update increases $\mathcal{L}_0$ at first order iff

$$-\eta(\pi_0 \|g_0\|^2 + \pi_1 g_0^\top g_1) > 0,$$

which is equivalent to (14). $\square$

By contrast, under the same full-objective weighting, the expected PLE update for the no-think expert is

$$\Delta\beta_0 = -\eta\pi_0 g_0,$$

so

$$\mathcal{L}_0(\beta_0 + \Delta\beta_0) = \mathcal{L}_0(\beta_0) - \eta\pi_0 \|g_0\|^2 + O(\eta^2),$$

with no cross-term involving g_1 . If one instead conditions on a mode-0 minibatch, the same calculation gives $\Delta\beta_0 = -\eta g_0$. Thus PLE removes *direct* cross-mode interference exactly in the expert parameters, while indirect coupling through the shared backbone remains.

A local quadratic theory of the dense compromise. To isolate the mechanism, fix the shared parameters at some local value $\bar{\alpha}$ and write

$$L_r(\beta) := \mathcal{L}_r(\bar{\alpha}, \beta).$$

Proposition C.3 (Exact fixed-backbone dominance of separate experts). *Fix any shared parameter value $\bar{\alpha}$, and define $L_r(\beta) := \mathcal{L}_r(\bar{\alpha}, \beta)$. Then*

$$\min_{\beta_0, \beta_1} \pi_0 L_0(\beta_0) + \pi_1 L_1(\beta_1) \leq \min_{\beta} \pi_0 L_0(\beta) + \pi_1 L_1(\beta).$$

Proof. The dense model is the constrained subset of the separate-expert model obtained by imposing $\beta_0 = \beta_1 = \beta$. Minimizing over the larger feasible set (β_0, β_1) therefore cannot yield a larger objective value than minimizing over the restricted set $\beta_0 = \beta_1$. $\square$

Let β_r^* denote a local minimizer of L_r , and let $H_r \succeq 0$ be the local Hessian at that point. A second-order Taylor expansion gives

$$L_r(\beta) \approx L_r(\beta_r^*) + \frac{1}{2}(\beta - \beta_r^*)^\top H_r(\beta - \beta_r^*). \quad (15)$$

For the remainder of this subsection, it is convenient to promote (15) to the local surrogate

$$\tilde{L}_r(\beta) := L_r(\beta_r^*) + \frac{1}{2}(\beta - \beta_r^*)^\top H_r(\beta - \beta_r^*),$$

and to interpret all equalities below in this subsection as statements about the surrogate $\tilde{L}_r$, not the exact nonquadratic loss L_r .

The dense expert objective becomes

$$L_{\text{dense}}(\beta) = \pi_0 L_0(\beta) + \pi_1 L_1(\beta) \quad (16)$$

$$\approx \pi_0 L_0(\beta_0^*) + \pi_1 L_1(\beta_1^*) + \frac{\pi_0}{2}(\beta - \beta_0^*)^\top H_0(\beta - \beta_0^*) + \frac{\pi_1}{2}(\beta - \beta_1^*)^\top H_1(\beta - \beta_1^*). \quad (17)$$

Expand the first quadratic term:

$$(\beta - \beta_0^*)^\top H_0(\beta - \beta_0^*) = \beta^\top H_0 \beta - 2\beta^\top H_0 \beta_0^* + (\beta_0^*)^\top H_0 \beta_0^*.$$

Expand the second:

$$(\beta - \beta_1^*)^\top H_1(\beta - \beta_1^*) = \beta^\top H_1 \beta - 2\beta^\top H_1 \beta_1^* + (\beta_1^*)^\top H_1 \beta_1^*.$$

Substitute both expansions into (17):

$$L_{\text{dense}}(\beta) \approx C + \frac{1}{2} \beta^\top (\pi_0 H_0 + \pi_1 H_1) \beta - \beta^\top (\pi_0 H_0 \beta_0^* + \pi_1 H_1 \beta_1^*), \quad (18)$$

where

$$C = \pi_0 L_0(\beta_0^*) + \pi_1 L_1(\beta_1^*) + \frac{\pi_0}{2} (\beta_0^*)^\top H_0 \beta_0^* + \frac{\pi_1}{2} (\beta_1^*)^\top H_1 \beta_1^*.$$

Differentiate (18) with respect to β :

$$\nabla_\beta L_{\text{dense}}(\beta) = (\pi_0 H_0 + \pi_1 H_1) \beta - (\pi_0 H_0 \beta_0^* + \pi_1 H_1 \beta_1^*).$$

Setting the gradient to zero gives

$$(\pi_0 H_0 + \pi_1 H_1) \beta_{\text{dense}}^* = \pi_0 H_0 \beta_0^* + \pi_1 H_1 \beta_1^*,$$

hence

$$\beta_{\text{dense}}^* = (\pi_0 H_0 + \pi_1 H_1)^{-1} (\pi_0 H_0 \beta_0^* + \pi_1 H_1 \beta_1^*), \quad (19)$$

assuming $\pi_0 H_0 + \pi_1 H_1$ is invertible.

Equation (19) shows that the dense MLP is forced to sit at a curvature-weighted compromise between the two mode-specific optima.

Theorem C.4 (Mode-conflict gap for the local quadratic surrogate). *Under the local quadratic model (15), the best PLE solution in the expert subspace is*

$$(\beta_0, \beta_1) = (\beta_0^*, \beta_1^*),$$

with value

$$L_{\text{PLE}}^* = \pi_0 L_0(\beta_0^*) + \pi_1 L_1(\beta_1^*).$$

The dense model incurs excess loss

$$\Delta_{\text{conflict}} := L_{\text{dense}}(\beta_{\text{dense}}^*) - L_{\text{PLE}}^* = \frac{1}{2} \sum_{r \in \{0,1\}} \pi_r (\beta_{\text{dense}}^* - \beta_r^*)^\top H_r (\beta_{\text{dense}}^* - \beta_r^*) \geq 0. \quad (20)$$

Proof. For PLE, the expert-subspace objective is

$$L_{\text{PLE}}(\beta_0, \beta_1) = \pi_0 L_0(\beta_0) + \pi_1 L_1(\beta_1).$$

Because β_0 and β_1 are separated, minimization reduces to two independent problems, so the optimum is attained at (β_0^*, β_1^*) , with value

$$L_{\text{PLE}}^* = \pi_0 L_0(\beta_0^*) + \pi_1 L_1(\beta_1^*).$$

For the dense model, evaluate each mode loss at β_{dense}^* :

$$L_r(\beta_{\text{dense}}^*) = L_r(\beta_r^*) + \frac{1}{2} (\beta_{\text{dense}}^* - \beta_r^*)^\top H_r (\beta_{\text{dense}}^* - \beta_r^*).$$

Multiply by π_r and sum over $r \in \{0,1\}$:

$$\begin{aligned} L_{\text{dense}}(\beta_{\text{dense}}^*) &= \sum_{r \in \{0,1\}} \pi_r L_r(\beta_{\text{dense}}^*) \\ &= \sum_{r \in \{0,1\}} \pi_r L_r(\beta_r^*) + \frac{1}{2} \sum_{r \in \{0,1\}} \pi_r (\beta_{\text{dense}}^* - \beta_r^*)^\top H_r (\beta_{\text{dense}}^* - \beta_r^*). \end{aligned}$$

Subtract L_{PLE}^* to obtain (20). Since each $H_r \succeq 0$, every quadratic term is nonnegative, so $\Delta_{\text{conflict}} \geq 0$. $\square$

Equation (20) is the formal version of the core intuition for the local quadratic surrogate: if think and no-think prefer different MLP parameters in directions with nonzero local curvature, then a dense hybrid MLP pays a nonzero compromise penalty. PLE removes that surrogate penalty in the expert subspace by allowing each mode to occupy its own local optimum.

Corollary C.5 (Equal-curvature closed form for the local quadratic surrogate). *If $H_0 = H_1 = H \succ 0$, then*

$$\beta_{\text{dense}}^* = \pi_0 \beta_0^* + \pi_1 \beta_1^*,$$

and the compromise penalty simplifies to

$$\Delta_{\text{conflict}} = \frac{1}{2} \pi_0 \pi_1 (\beta_0^* - \beta_1^*)^\top H (\beta_0^* - \beta_1^*). \quad (21)$$

Proof. Substitute $H_0 = H_1 = H$ into (19):

$$\begin{aligned} \beta_{\text{dense}}^* &= (\pi_0 H + \pi_1 H)^{-1} (\pi_0 H \beta_0^* + \pi_1 H \beta_1^*) \\ &= ((\pi_0 + \pi_1) H)^{-1} H (\pi_0 \beta_0^* + \pi_1 \beta_1^*) \\ &= \pi_0 \beta_0^* + \pi_1 \beta_1^*, \end{aligned}$$

since $\pi_0 + \pi_1 = 1$.

Now define $\Delta\beta := \beta_0^* - \beta_1^*$. Then

$$\begin{aligned} \beta_{\text{dense}}^* - \beta_0^* &= (\pi_0 \beta_0^* + \pi_1 \beta_1^*) - \beta_0^* = -\pi_1 (\beta_0^* - \beta_1^*) = -\pi_1 \Delta\beta, \\ \beta_{\text{dense}}^* - \beta_1^* &= (\pi_0 \beta_0^* + \pi_1 \beta_1^*) - \beta_1^* = \pi_0 (\beta_0^* - \beta_1^*) = \pi_0 \Delta\beta. \end{aligned}$$

Plug these into (20):

$$\begin{aligned} \Delta_{\text{conflict}} &= \frac{1}{2} \left[\pi_0 (-\pi_1 \Delta\beta)^\top H (-\pi_1 \Delta\beta) + \pi_1 (\pi_0 \Delta\beta)^\top H (\pi_0 \Delta\beta) \right] \\ &= \frac{1}{2} \left[\pi_0 \pi_1^2 \Delta\beta^\top H \Delta\beta + \pi_1 \pi_0^2 \Delta\beta^\top H \Delta\beta \right] \\ &= \frac{1}{2} \pi_0 \pi_1 (\pi_0 + \pi_1) \Delta\beta^\top H \Delta\beta \\ &= \frac{1}{2} \pi_0 \pi_1 \Delta\beta^\top H \Delta\beta. \end{aligned}$$

□

Equation (21) is especially useful for intuition: the dense compromise penalty grows with (i) the frequency of both modes, through $\pi_0 \pi_1$, (ii) the geometric separation between their preferred optima, and (iii) the curvature in the directions where they disagree.

Why sequence-level routing matters. For sequence-level routing, each example has a single route r_i , so

$$\ell_i = \sum_{t=1}^{|y_i|} \ell_{i,t}(\alpha, \beta_{r_i}).$$

Thus

$$\nabla_{\beta_k} \ell_i = \mathbf{1}\{k = r_i\} \sum_{t=1}^{|y_i|} \nabla_{\beta_k} \ell_{i,t}. \quad (22)$$

A single response updates at most one expert.

If routing were token-level, with $r_{i,t} \in \{0, 1\}$, then

$$\ell_i^{\text{token}} = \sum_{t=1}^{|y_i|} \ell_{i,t}(\alpha, \beta_{r_{i,t}}),$$

and hence

$$\nabla_{\beta_k} \ell_i^{\text{token}} = \sum_{t=1}^{|y_i|} \mathbf{1}\{k = r_{i,t}\} \nabla_{\beta_k} \ell_{i,t}.$$

Now the same response can update both experts, weakening the clean decoupling in (11) and (22). This is why sequence-level routing is the natural match to a turn-level /think//no_think interface.

Why identical initialization matters. PLE initializes both experts from the same dense source MLP:

$$\beta_0^{(0)} = \beta_1^{(0)} = \beta_{\text{src}}.$$

If the two routes differ only through the duplicated expert weights at initialization, then before fine-tuning

$$p_\theta(y \mid x, r = 0) = p_\theta(y \mid x, r = 1) = p_{\text{src}}(y \mid x).$$

When newly introduced control-token embeddings or other route-specific components are present, this equality need not hold exactly. In that case, the intended claim is narrower: the two experts start from identical weights, so specialization is not driven by random asymmetry in the expert parameters.

The update identities below are exact for plain SGD with matched learning rates and no optimizer state. With momentum, Adam, gradient clipping, or asynchronous minibatch schedules, the same conclusion is qualitative rather than algebraically exact.

After one SGD step on a no-think batch and one SGD step on a think batch,

$$\begin{aligned} \beta_0^{(1)} &= \beta_{\text{src}} - \eta g_0^{(0)}, \\ \beta_1^{(1)} &= \beta_{\text{src}} - \eta g_1^{(0)}. \end{aligned}$$

Subtracting yields

$$\beta_1^{(1)} - \beta_0^{(1)} = -\eta (g_1^{(0)} - g_0^{(0)}).$$

After T paired updates,

$$\beta_1^{(T)} - \beta_0^{(T)} = -\eta \sum_{s=0}^{T-1} (g_1^{(s)} - g_0^{(s)}).$$

Thus expert divergence is driven purely by the cumulative difference between mode-specific gradients.

Why duplicating only the MLP can be sufficient. Let u_t denote the shared residual-stream representation entering a routed MLP block, and let $f_{\beta_r}(u_t)$ denote the routed expert output. Abstract the remainder of the network (all later layers plus the LM head) as a differentiable map G . Then the route- r logits can be written locally as

$$o_t^{(r)} = G(u_t + f_{\beta_r}(u_t)).$$

A first-order expansion around u_t gives

$$o_t^{(r)} \approx G(u_t) + J_G(u_t) f_{\beta_r}(u_t), \quad (23)$$

where $J_G(u_t)$ is the Jacobian of G at u_t . This linearization should be interpreted as a local sensitivity argument rather than a global proof of sufficiency. It is accurate when the downstream map G is locally smooth around u_t and the routed residual update $f_{\beta_r}(u_t)$ remains within a neighborhood where higher-order terms are small; in affine or shared linear-region regimes, the relation is exact. Hence the think/no-think logit difference is approximately

$$o_t^{(1)} - o_t^{(0)} \approx J_G(u_t) (f_{\beta_1}(u_t) - f_{\beta_0}(u_t)). \quad (24)$$

Equations (23)–(24) make the architectural intuition explicit: the shared backbone can keep constructing a common contextual representation u_t , while the mode-specific MLPs directly control how that representation is pushed toward different token distributions. In this sense, PLE separates *expression policy* more than *core competence*.

A length-induced asymmetry heuristic. Let mode r have N_r examples with average target length m_r , and let $g_{i,t}^{(r)}$ denote the per-token gradient contribution in the expert subspace. A token-summed dense objective produces an epoch-level update

$$\Delta\beta_{\text{dense}} \propto - \sum_{r \in \{0,1\}} \sum_{i=1}^{N_r} \sum_{t=1}^{m_i^{(r)}} g_{i,t}^{(r)}.$$

If think targets are much longer than no-think targets, then even with similar example counts, the dense expert receives more cumulative update mass from think-mode tokens. PLE changes this to

$$\Delta\beta_0 \propto - \sum_{i=1}^{N_0} \sum_{t=1}^{m_i^{(0)}} g_{i,t}^{(0)}, \quad \Delta\beta_1 \propto - \sum_{i=1}^{N_1} \sum_{t=1}^{m_i^{(1)}} g_{i,t}^{(1)},$$

so long think traces can no longer wash over the no-think expert. This gives a conditional explanation for why PLE may especially strengthen the cleanliness and stability of /no_think mode when optimization effectively scales with token count and think targets are substantially longer.