

AdaMix: Adaptive Mixing for Short and Long Reasoning Adapters

Hao Luo¹, Xiao Yan², Xinyan Li¹, Qiming Zeng¹, Yuhao Lin¹,
Shanshan Feng¹, Hao Wang¹, Jiawei Jiang^{1,3,*}

¹School of Computer Science, Wuhan University, ²Institute for Math & AI, Wuhan University,
³Wuhu Overseas Students Pioneer Park, Wuhu, China

{lohozz, yanxiaosunny, xinyan_li, kirin_z, yuhao_lin}@whu.edu.cn,
{victor_fengss, wanghao.cs, jiawei.jiang}@whu.edu.cn

Abstract

Large Reasoning Models (LRMs) have achieved remarkable success on complex tasks by generating detailed Chain-of-Thought (CoT) reasoning. However, they tend to apply a uniform, computation-intensive deep reasoning strategy to all problems, leading to unnecessary overhead on simple tasks. This significantly hinders their efficiency in real-world applications. While existing methods have improved reasoning efficiency to some extent, they still face critical challenges such as conflicting objectives, limited adaptability. To address these limitations, we propose AdaMix, an adaptive reasoning framework via decoupled optimization. To mitigate optimization conflicts, AdaMix first constructs two specialized adapters: an efficiency-oriented short adapter and an accuracy-oriented long adapter. It then incorporates a difficulty-aware routing model that assesses problem complexity to predict a reasoning intensity coefficient. This coefficient is used to dynamically interpolate a mixed adapter from the two base adapters, enabling fine-grained reasoning control. Our experiment demonstrates that our AdaMix reduces the average response length of DeepSeek-R1-Distill-Qwen-7B by 54.9% while improving accuracy by up to 4.8% on five mathematical datasets, thus indicating a favorable accuracy-efficiency trade-off.

1 Introduction

In recent years, the field of artificial intelligence has been undergoing a paradigmatic transformation driven by Large Reasoning Models (LRMs) (Jaech et al., 2024; Guo et al., 2025; Yang et al., 2025a; Zhang et al., 2025a). These models simulate humans’ in-depth critical thinking processes by generating detailed step-by-step Chain-of-Thought (CoT) (Wei et al., 2022), thereby

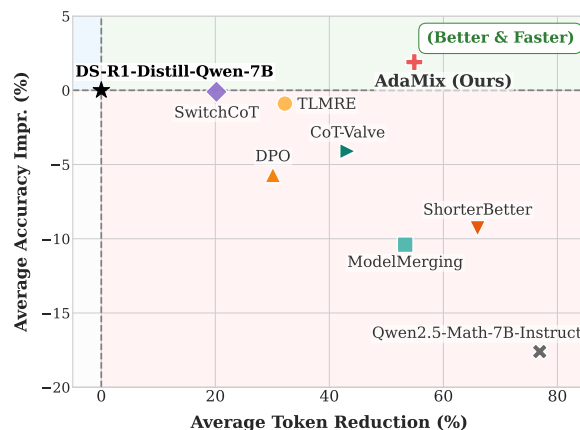


Figure 1: Trade-off between accuracy improvement and token reduction rate (relative to the DS-R1-Distill-Qwen-7B baseline) for AdaMix and baseline methods, averaged across five mathematical datasets.

achieving groundbreaking progress in complex reasoning tasks (Chang et al., 2024) such as mathematics (Cobbe et al., 2021) and code generation (Chen et al., 2021). However, beneath this powerful capability lies a core efficiency paradox: regardless of the inherent complexity of various problems, models tend to adopt reasoning patterns that are equally lengthy and computationally intensive. While this uniformly applied deep thinking strategy is relatively applicable to challenging problems, it leads to the phenomenon of *overthinking* when confronted with simple problems that do not require complex reasoning (Chen et al., 2025; Sui et al., 2025; Yang et al., 2025b). This results in significant waste of computational resources and delays.

To address the efficiency challenges posed by *overthinking*, the academic community has explored diverse optimization approaches from multiple dimensions. A major research orientation employs reinforcement learning to directly restrict response length (Team et al., 2025; Hou et al., 2025). By devising reward functions that penalize the generation of redundant tokens, it guides

*Corresponding author.

models to yield more concise outputs (Luo et al., 2025; Aggarwal and Welleck, 2025). Other approaches adopt non-training strategies, such as utilizing model merging techniques to fuse a slow-thinking model proficient in in-depth reasoning with a fast-thinking model capable of quick responses using fixed parameter weights (Wu et al., 2025; Qiu et al., 2025), with the aim of creating a single model that combines the strengths of both. Furthermore, researchers propose the model routing mechanism (Ong et al., 2025), which introduces an external decision-making module during inference to assign tasks to different expert models according to input features (Zhang et al., 2025b; Liang et al., 2025; Wang et al., 2025).

Despite improving reasoning efficiency, existing methods still face the following challenges:

❶ **Optimization Objective Conflicts.** Many reinforcement learning-based methods (Team et al., 2025; Yi et al., 2025; Arora and Zanette, 2025) use a single model to simultaneously pursue accuracy and efficiency. Because these goals are often inherently competing, this approach can lead to conflicting optimization signals and training instability (Dai et al., 2025; Lu et al., 2025), leading to unsatisfactory model performance.

❷ **Limited Adaptability.** While model merging can effectively combine different capabilities (Wu et al., 2025), the resulting static weights may limit model flexibility in handling problems with varying difficulties. Similarly, model routing often relies on coarse, binary decisions (e.g., selecting a short or long reasoning path) (Zhang et al., 2025b; Liang et al., 2025), which can restrict their capacity for fine-grained real-world problem complexities.

To address these issues, this paper focuses on the following core research question:

How can we resolve the optimization conflict between reasoning depth and efficiency, and achieve fine-grained adaptation to problem complexity?

To this end, we propose AdaMix, an adaptive reasoning framework based on a decoupled strategy.

❶ To mitigate optimization conflicts, we employ a decoupled optimization approach guided by specialized dynamic reward functions. Specifically, we independently train two adapter modules with distinct behavioral preferences: a short adapter optimized for generating efficient and concise reasoning paths, and a long adapter specialized in solving complex problems with in-depth reasoning. ❷ To overcome the limitations of adaptability, we introduce a difficulty-aware routing model. This

model learns to perceive the intrinsic complexity of the input to predict a complexity coefficient, which drives the linear interpolation between the two adapters, enabling AdaMix to activate a mixed adapter matching the problem’s complexity.

Our experiments demonstrate that AdaMix reduces inference costs while maintaining or even enhancing reasoning performance. For instance, on the GSM8K and AIME25 datasets, AdaMix reduces response length by over 70% and 42%, respectively, while simultaneously improving accuracy by 0.7% and 4.8%. Notably, AdaMix outperforms strong baselines, such as ShorterBetter and TLMRE, by at least 30% in the accuracy-efficiency score. As illustrated in Figure 1, AdaMix achieves a superior accuracy-efficiency trade-off compared to baselines on average across five datasets.

Our contributions can be summarized as follows:

- **Dual-Adapter Decoupled Optimization.** We propose a decoupled optimization strategy with specialized dynamic rewards to train two distinct adapters, optimizing them separately for conciseness and deep reasoning capabilities.
- **Fine-Grained Adaptive Reasoning.** We introduce a difficulty-aware routing model that predicts a complexity coefficient to drive the linear interpolation of adapter weights, achieving fine-grained adaptive reasoning.
- **Favorable Accuracy-Efficiency Trade-off.** Experiments on five mathematical datasets demonstrate that AdaMix reduces the average response length by 54.9% while simultaneously improving accuracy by up to 4.8%.

2 Preliminaries

2.1 Controllable Reasoning

The core objective of controllable reasoning is to steer the behavior of a large reasoning model to align with specific efficiency and accuracy requirements. A primary mechanism for achieving this control is through the design of the reward function that guides policy optimization. A general formulation for such a reward function, $R(y|x)$, for a response y given an input x , can be expressed as:

$$R(y|x) = R_{\text{acc}}(y|x) + w_{\text{len}} \cdot R_{\text{len}}(y|x) \quad (1)$$

Here, $R_{\text{acc}}(y|x)$ is a binary accuracy reward, yielding 1 if y is correct and 0 otherwise. $R_{\text{len}}(y|x)$ is a length-based reward component, typically designed to penalize responses that are excessively long.

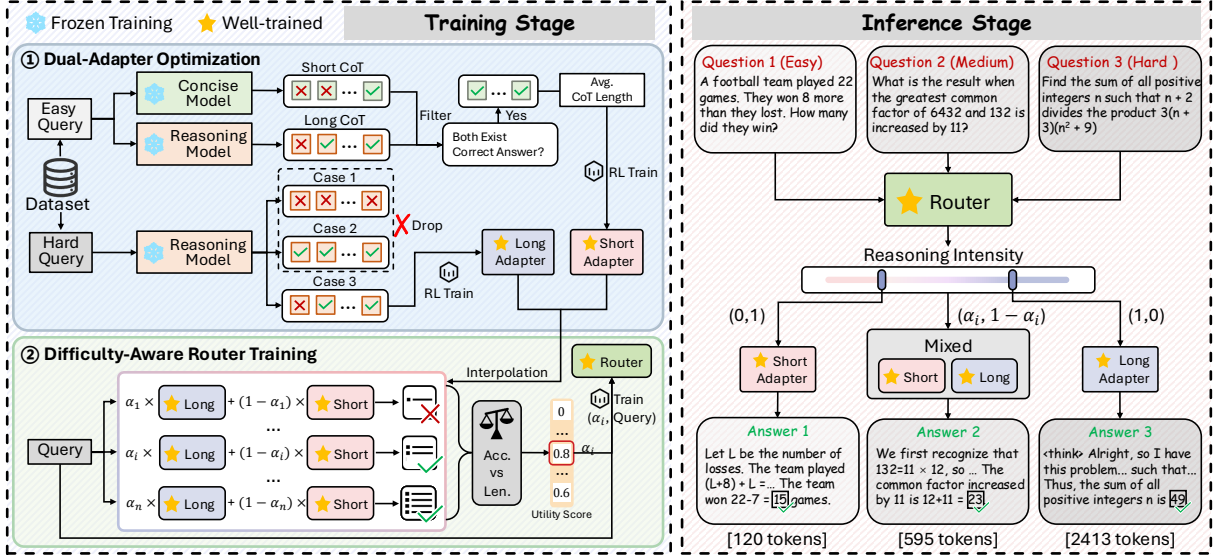


Figure 2: Overview of the AdaMix framework.

The key to achieving adaptability lies in the term w_{len} , which acts as a control knob for the model’s behavior. By adjusting this weighting factor, it is possible to modulate the model’s reasoning style: a high w_{len} value encourages concise and efficient responses by penalizing length, while a low w_{len} value permits more elaborate reasoning trajectories to enhance accuracy.

2.2 Parameter-Efficient Adapter Merging

Parameter-Efficient Fine-Tuning (PEFT) methods like Low-Rank Adaptation (LoRA) (Hu et al., 2022) allow adapter parameters to be manipulated algebraically. This enables model merging, a technique that combines specialist adapters into a single module by operating directly on their weights, often without requiring additional training (Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024).

A straightforward yet effective merging technique is Task Arithmetic. In this framework, the learned parameters of an adapter are treated as a task vector (θ) representing a specific direction in the parameter space. Given two task vectors, adapter A (θ_A) and adapter B (θ_B), each specialized for a different behavior, a new mixed adapter (θ_{mixed}) can be created via their weighted sum:

$$\theta_{\text{mixed}} = \alpha \cdot \theta_A + (1 - \alpha) \cdot \theta_B, \quad \text{where } \alpha \in [0, 1] \quad (2)$$

The mixing weight α determines the magnitude of each task vector’s contribution, allowing for a smooth transition between the behaviors of the two source adapters.

3 AdaMix Framework

Figure 2 presents an overview of the proposed AdaMix framework. We first employ a decoupled optimization strategy (▷ Section 3.1) to independently train an efficiency-oriented short adapter (▷ Section 3.2) and an accuracy-oriented long adapter (▷ Section 3.3). Subsequently, we introduce a continuous interpolation mechanism and train a difficulty-aware router (▷ Section 3.4). Finally, this router learns to map each problem to a complexity coefficient, thereby activating an adapter instance that aligns with the required reasoning intensity for adaptive inference (▷ Section 3.5).

3.1 Decoupled Optimization Strategy

Simultaneously optimizing a single model for both reasoning depth (accuracy) and conciseness (efficiency) is often challenging (Dai et al., 2025), as the two objectives can interfere with each other during the reinforcement learning (RL) process. To mitigate this optimization conflict, we propose a decoupled optimization strategy that separates the acquisition of distinct reasoning capabilities.

To operationalize this strategy, we first establish capability-aligned training environments via difficulty-based data stratification. Adopting the LLM-based automatic scoring approach from (Gao et al., 2025; He et al., 2025), we quantify problem complexity to partition the training data into an easy subset, $\mathcal{D}_{\text{easy}}$, and a hard subset, $\mathcal{D}_{\text{hard}}$ (details in Appendix A.2.1). Building on this setup, we denote the target long-reasoning model as π_L and employ Low-Rank Adaptation (LoRA) (Hu et al.,

2022) based on Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to independently train the two adapters on their respective datasets.

3.2 Efficiency-Oriented Short Adapter

Data Selection. To compress reasoning trajectories without compromising correctness, we propose an intrinsic complexity matching strategy for data selection. We hypothesize that every problem possesses an inherent reasoning complexity, which can be effectively approximated by a more concise reference model π_S acting as an efficiency baseline for our target reasoning model π_L . For instance, when π_L is a reasoning model like DeepSeek-R1-Distill-Qwen (Guo et al., 2025), π_S corresponds to its non-reasoning counterpart, Qwen2.5-Math-Instruct (Yang et al., 2024).

We focus the optimization on the intersection of capabilities between the two models, defined as $\mathcal{D}_{short} = \{x \in \mathcal{D}_{easy} \mid \mathcal{Y}_L^c(x) \neq \emptyset \wedge \mathcal{Y}_S^c(x) \neq \emptyset\}$, where $\mathcal{Y}_L^c(x)$ and $\mathcal{Y}_S^c(x)$ denote the sets of correct solutions generated by π_L and π_S , respectively. This ensures the short adapter focuses only on problems having concise reasoning feasibility.

Dynamic Relative Constraint. Instead of a static length penalty, we implement a dynamic length budget based on the problem’s intrinsic complexity. We define a problem-specific reference length $L_{ref}(x)$ derived from the average correct response length $l(\cdot)$ of π_S , adjusted by a tolerance margin δ :

$$L_{ref}(x) = \frac{\sum_{y \in \mathcal{Y}_S^c(x)} l(y)}{|\mathcal{Y}_S^c(x)|} + \delta, \quad \forall x \in \mathcal{D}_{short} \quad (3)$$

Here, the tolerance δ adapts dynamically to the group accuracy (acc_G), formulated as $\delta \propto (1 - acc_G)$, allowing larger length when model accuracy is low. We define the length reward function as:

$$R_{len}^S(y|x) = \mathbb{I}_c(y) \cdot (1 - \lambda \cdot \max(0, \ell(y) - L_{ref}(x))) \quad (4)$$

where $\mathbb{I}_c(\cdot)$ is the correctness indicator and $\lambda > 0$ is a coefficient that controls the strength of the length penalty. As illustrated in Figure 3 (top), this reward function only penalizes redundant reasoning trajectories that exceed the difficulty-aware threshold, while directly assigning a reward of 0 to incorrect responses. Such a mechanism guides the adapter to converge towards the minimal sufficient reasoning trajectory dictated by the problem’s complexity.

3.3 Accuracy-Oriented Long Adapter

Data Selection. To extend the reasoning boundaries on complex tasks, we propose a capability-

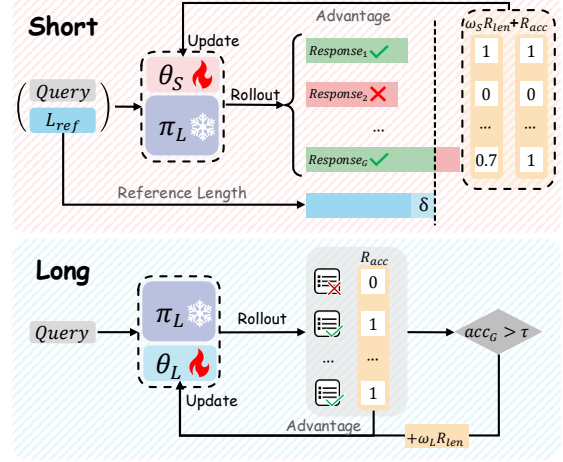


Figure 3: The short adapter utilizes a dynamic reference length to exclusively penalize redundant tokens, while the long adapter employs a capability-gated penalty to ensure sufficient reasoning depth for complex problems.

prioritized optimization strategy. We direct the training focus towards the model’s reasoning frontier, targeting instances where the policy π_L demonstrates potential but lacks consistency. Thus, we define the target subset \mathcal{D}_{long} as the collection of problems where *the model achieves partial success*. Formally, the optimization domain is defined as $\mathcal{D}_{long} = \{x \in \mathcal{D}_{hard} \mid 0 < |\mathcal{Y}_L^c(x)| < |\mathcal{Y}_L(x)|\}$, where $\mathcal{Y}_L(x)$ and $\mathcal{Y}_L^c(x)$ denote the sets of generated solutions and correct solutions, respectively. This formulation effectively filters out both fully mastered and currently intractable instances.

Capability-Gated Optimization. Within this domain, while accuracy is paramount, unregulated exploration often leads to inefficient over-thinking or unbounded generation. To mitigate this, we define the length reward function as:

$$R_{len}^L(y|x) = \mathbb{I}(acc_G \geq \tau) \cdot \mathbb{I}_c(y) \cdot \sigma\left(-\kappa \cdot \frac{\ell(y) - \mu_L}{\sigma_L}\right) \quad (5)$$

where μ_L and σ_L are the mean and standard deviation of correct response lengths in the group. $\kappa > 0$ is a hyperparameter controlling the penalty steepness, and $\sigma(\cdot)$ is the sigmoid function. As displayed in the bottom of Figure 3, this design introduces a capability-gated preference mechanism. By gating the length penalty with the group accuracy acc_G relative to a threshold τ , this mechanism ensures the optimizer prioritizes consolidating correct reasoning before refining its conciseness.

3.4 Interpolation and Router Training

Parametric Adapter Interpolation. With the adapters θ_S and θ_L specialized for efficiency and

capability, respectively, we utilize task arithmetic to establish a continuous interpolation mechanism between them (We compare different merging methods in Appendix B.6). We treat the optimized adapters as basis vectors in the parameter domain. Any intermediate reasoning strategy θ_α can be synthesized via linear interpolation:

$$\theta_\alpha = \alpha \cdot \theta_L + (1 - \alpha) \cdot \theta_S, \quad \text{where } \alpha \in [0, 1] \quad (6)$$

Here, the interpolation weight α acts as a complexity coefficient that governs the model’s reasoning intensity. As α increases from 0 to 1, the model’s parameters smoothly transition from the efficiency-oriented configuration to the deep reasoning, enabling a continuous scaling of reasoning intensity.

Difficulty-Aware Router Training. To enable autonomous difficulty perception and reasoning intensity adjustment, we employ a BERT-based routing model M_{route} . This model maps each problem instance x to a suitable complexity coefficient $\alpha^*(x)$, aiming to balance solution accuracy and computational cost. We formalize this objective using a scalar utility function $U(x, \alpha)$:

$$U(x, \alpha) = \underbrace{\mathbb{I}_c(y_\alpha)}_{\text{Solution Accuracy}} - \beta \cdot \underbrace{\frac{\bar{\ell}(y_\alpha)}{L_{\max}}}_{\text{Efficiency Penalty}} \quad (7)$$

where y_α is the response generated by the mixed adapter θ_α , $\bar{\ell}(y_\alpha)$ is the average response length, L_{\max} is a maximum length for normalization, and $\beta \geq 0$ is a preference coefficient for efficiency.

During the training phase, we utilize this utility function to construct ground-truth labels. For practical implementation, we discretize the continuous range of α into a candidate set \mathcal{A} (e.g., $\{0, 0.1, \dots, 1.0\}$) and treat routing as a classification task. The model is optimized to minimize the cross-entropy loss between its predicted probability distribution $P(\cdot|x)$ and the target label derived from the coefficient $\alpha^*(x) = \arg \max_{\alpha \in \mathcal{A}} U(x, \alpha)$:

$$\mathcal{L}_{\text{route}} = \mathbb{E}_{x \sim \mathcal{D}} \left[- \sum_{\alpha_i \in \mathcal{A}} \mathbb{I}(\alpha_i = \alpha^*(x)) \log P(\alpha_i|x) \right] \quad (8)$$

3.5 Adaptive Inference

During the inference phase, the trained router operates in concert with the parameterized adapters to achieve instance-level adaptive reasoning.

Dynamic Adapter Composition. For a given problem x , the router selects the target complexity coefficient $\hat{\alpha}(x) = \arg \max_{\alpha_i \in \mathcal{A}} P(\alpha_i|x)$. By dynamically loading the corresponding adapter $\theta_{\hat{\alpha}}$, the

base model is transformed into an instance-specific active model $\pi_{\text{active}} = \pi_L + \theta_{\hat{\alpha}}$. This mechanism enables the model to apply efficiency-oriented policies ($\hat{\alpha} \rightarrow 0$) for simple problems and deep reasoning strategies ($\hat{\alpha} \rightarrow 1$) for complex ones.

Efficient Serving via Grouping. To mitigate the overhead of switching adapters per sample, we implement a request grouping strategy similar to S-LoRA (Sheng et al., 2024). In a batch processing scenario, the router first predicts $\hat{\alpha}$ for all pending requests. Requests sharing the same target weight are then grouped into sub-batches and processed in parallel using the corresponding shared adapter parameters. This approach is compatible with inference frameworks supporting multi-LoRA (e.g., vLLM), allowing for the efficient handling of heterogeneous request streams with negligible latency overhead (detailed latency and throughput analyses are provided in Appendix B.7).

4 Experimental Evaluation

4.1 Experimental Setup

We largely follow the settings of recent work like Yi et al. (2025) and Arora and Zanette (2025) to ensure a fair and rigorous comparison.

Models. We select DeepSeek-R1-Distill-Qwen-1.5B / 7B (Guo et al., 2025) (hereafter R1-Distill-Qwen) and their base models Qwen2.5-Math-Instruct (Yang et al., 2024) for evaluation. Additionally, we apply AdaMix to DeepSeek-R1-Distill-Llama-8B and Qwen3-4B (Yang et al., 2025a) to verify its generality across different backbones.

Training Data. Using difficulty annotations from DeepMath-103K (He et al., 2025), we construct an *easy* subset $\mathcal{D}_{\text{short}}$ and a *hard* subset $\mathcal{D}_{\text{long}}$ to train the short and long adapters, respectively. A separate uniform subset is sampled to train the router.

Implementation Details. Our experiments are implemented using the veRL (Sheng et al., 2025) framework on 8 NVIDIA A6000 GPUs. Detailed training configurations and other hyperparameters are provided in Appendix A.6.

Baselines. We compare our method against several representative baselines: ❶ DPO (Rafailov et al., 2023) aligns the model to prefer shorter correct answers via preference pairs. ❷ Shorter-Better (Yi et al., 2025) uses RL with the shortest sampled correct response as a dynamic reward. ❸ TLMRE (Arora and Zanette, 2025) employs RL to directly penalize token length. ❹ ModelMerg-

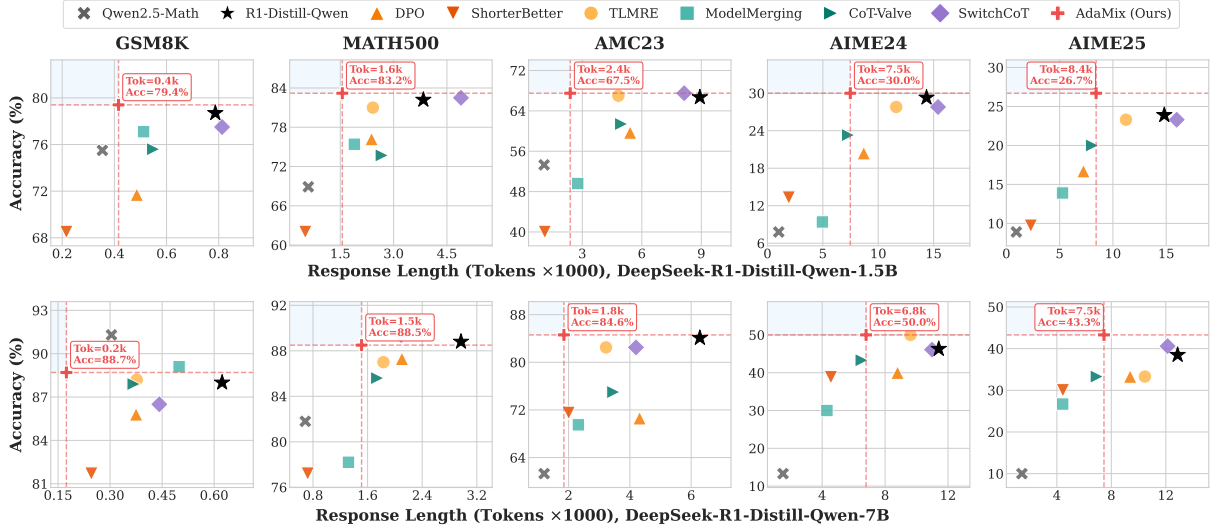


Figure 4: Accuracy-efficiency trade-off comparison on five mathematical benchmarks using 1.5B (top) and 7B (bottom) backbones. We compare AdaMix (red cross) against the reasoning baseline DeepSeek-R1-Distill-Qwen (black star) and various reasoning optimization methods. The results demonstrate that AdaMix consistently achieves comparable or superior accuracy with significantly reduced response lengths across varying problem difficulties.

ing (Wu et al., 2025) statically fuses a reasoning model with a concise model without training. ⑤ CoT-Valve (Ma et al., 2025) modulates CoT length by scaling a specific LoRA update vector via task arithmetic. ⑥ SwitchCoT (Zhang et al., 2025b) uses an external classifier to select between long and short reasoning strategies. Further details on these baselines are provided in Appendix A.5.

Evaluation. We evaluate all methods on five mathematical benchmarks spanning from primary school to Olympic competition levels: GSM8K (Cobbe et al., 2021) test set, MATH500 (Hendrycks et al., 2021b), AMC 2023, AIME 2024, and AIME 2025. We report accuracy (pass@1), response length, and the accuracy-efficiency score (AES) (Luo et al., 2025), a metric quantifying the efficiency-accuracy trade-off. Detailed sampling configurations and metric definitions are provided in Appendix A.3 and Appendix A.4, respectively.

4.2 Main Results

As shown in Figure 4, AdaMix outperforms most baselines in terms of both accuracy and length reduction across varying problem difficulties. While methods like CoT-Valve effectively compress inference length, they typically fail to yield accuracy gains. In contrast, our long adapter is specifically designed to tackle problems that are intrinsically difficult for the model, thereby elevating the accuracy upper bound. This ensures that AdaMix maintains robust performance even on computationally

Method	Average		
	Δ Acc.	Δ #Tok.	AES \uparrow
<i>Qwen2.5-1.5B-Math-Ins</i>			
<i>R1-Distill-Qwen-1.5B</i>	-	-	-
+ DPO	-7.2%	-41.2%	0.15
+ ShorterBetter	-17.5%	-83.6%	<u>0.26</u>
+ TLMRE	-0.3%	-28.2%	<u>0.26</u>
+ ModelMerging	-11.1%	<u>-56.9%</u>	0.13
+ CoT-Valve	-5.4%	-40.0%	-0.10
+ SwitchCoT	-0.4%	+7.4%	0.22
+ AdaMix	+1.2%	-54.2%	0.58
<i>Qwen2.5-7B-Math-Ins</i>			
<i>R1-Distill-Qwen-7B</i>	-	-	-
+ DPO	-5.7%	-30.1%	0.16
+ ShorterBetter	-9.3%	-66.0%	<u>0.44</u>
+ TLMRE	-0.9%	-32.2%	0.29
+ ModelMerging	-10.4%	-53.3%	0.25
+ CoT-Valve	-4.1%	-43.2%	0.33
+ SwitchCoT	<u>-0.1%</u>	-20.2%	0.20
+ AdaMix	+1.9%	-54.9%	0.59

Table 1: Comparison of accuracy improvement (Δ Acc.), token reduction (Δ #Tok.), and AES score averaged across five math benchmarks. Full detailed results per dataset are provided in Appendix Table 8.

intensive benchmarks (e.g., AIME).

Results in Table 1 further corroborate these observations. Compared to the original 1.5B and 7B baselines, AdaMix reduces the average response length by 54.2% and 54.9%, while increasing

Backbone	MATH500		AIME24		AIME25	
	Acc.	#Tok.	Acc.	#Tok.	Acc.	#Tok.
R1-Llama-8B	86.7	3675	46.7	12717	30.0	11595
+ AdaMix	88.2	2164	53.3	7344	33.3	7492
Qwen3-4B	93.4	4696	64.3	11523	55.6	12476
+ AdaMix	92.5	2577	62.0	8029	58.3	7863

Table 2: Generalization performance across different backbones on MATH500, AIME24, and AIME25.

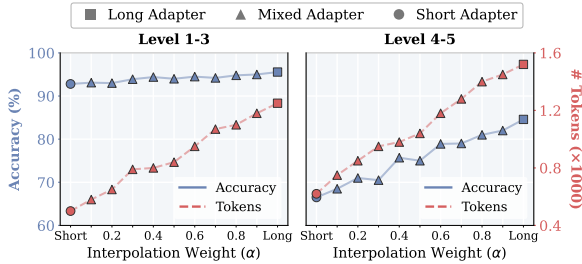


Figure 5: The weight α modulates the model’s behavior between efficiency and accuracy on MATH500.

the average accuracy by 1.2% and 1.9%, respectively. Furthermore, AdaMix achieves the highest Accuracy-Efficiency Scores of 0.58 and 0.59 across the two scales. This result significantly outperforms baselines such as ShorterBetter.

4.3 Generality and Insights

Across Different Model Backbones. We further applied AdaMix to DeepSeek-R1-Distill-Llama-8B and Qwen3-4B to verify its universality. As shown in Table 2, AdaMix consistently achieves substantial length reduction across different backbones on both datasets while maintaining comparable or better accuracy. This demonstrates the good robustness of our framework across different model architectures, with results on more datasets provided in Appendix B.4.

Generalization to Other Domains. We evaluated AdaMix on *Code*, *Science*, and *General Knowledge* tasks (dataset details in Appendix A.2.2). The sustained length reduction (Table 3) indicates that the distinct reasoning patterns captured by the two adapters are transferable across domains. While direct transfer caused a slight accuracy drop due to domain-specific difficulty shifts, fine-tuning *only* the lightweight router on a small mixed-domain dataset effectively resolved this. The updated router recalibrates difficulty criteria across domains, recovering accuracy while preserving efficiency. This demonstrates that AdaMix enables efficient generalization by simply updating the router, without

Method	LiveCodeBench		GPQA-Diamond		MMLU	
	Pass@1	#Tok.	Acc.	#Tok.	Acc.	#Tok.
R1-Qwen-7B	36.7	8870	45.3	4485	62.4	1472
AdaMix (Zero-Shot)	34.0	5123	43.2	2274	61.3	865
		(-42%)		(-49%)		(-41%)
AdaMix (+ Router-FT)	35.6	5364	47.8	2485	64.7	972
		(-40%)		(-45%)		(-34%)

Table 3: Performance on other domain tasks. We compare the base model with AdaMix in Zero-Shot (math-trained only) and Router-FT (router fine-tuned).

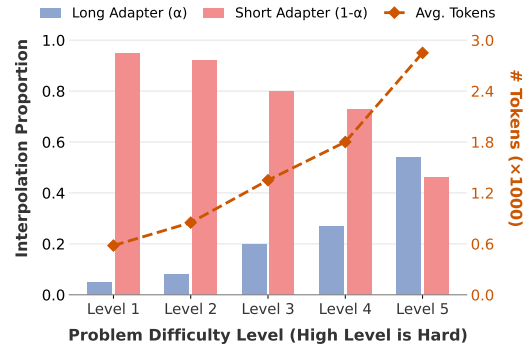


Figure 6: Interpolation proportions of short and long adapters across difficulty levels on MATH500.

retraining the adapters.

Can the Weight α Modulate Behavior? As shown in Figure 5, adjusting the interpolation weight modulates the model’s reasoning behavior between the short and long adapters. For lower-difficulty problems (Levels 1–3), incorporating a significant portion of the short adapter can reduce response length with minimal impact on accuracy. In contrast, for high-difficulty problems (Levels 4–5), while the long adapter’s contribution remains essential, introducing a small proportion of the short adapter can still shorten response length, albeit with an associated accuracy trade-off. This highlights the interpolation weight’s role in balancing efficiency and accuracy when addressing problems of varying difficulty.

Can the Router Judge Problem Difficulty? Figure 6 shows the interpolation proportions selected by the router across five difficulty levels. For the simplest problems (Level 1), the router assigns a dominant weight to the short adapter. As difficulty increases, the proportion of the long adapter rises correspondingly. By Level 5, the long adapter becomes the primary contributor. This indicates that the router effectively maps difficulty to reasoning strategy, prioritizing efficiency for simple tasks and depth for complex ones.

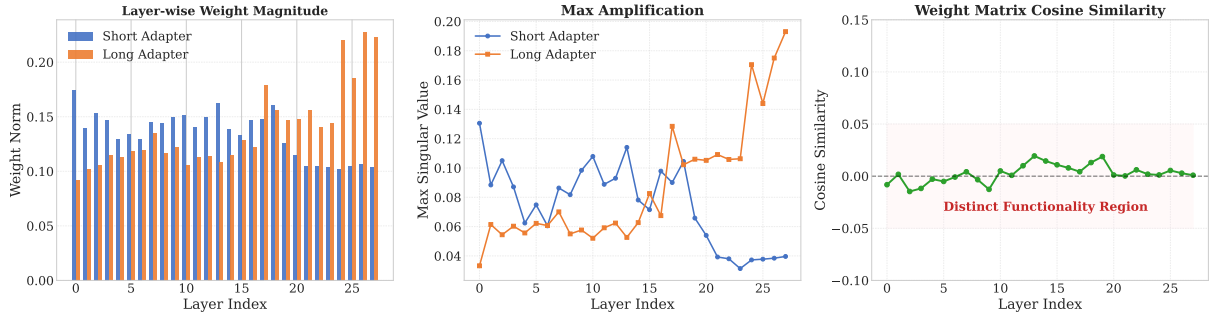


Figure 7: Layer-wise visualization of adapter weight updates. The panels display the magnitude of weight updates (**left**), the maximum singular value (**center**), and the cosine similarity between the two adapters (**right**).

Internal Representation Visualization. We visualized the layer-wise weight updates of the two adapters to examine their internal representations (Figure 7). Observations show that the short adapter’s updates are concentrated in shallow-to-middle layers, whereas the long adapter exhibits dominant magnitudes in deeper layers. Crucially, the cosine similarity between their weight matrices remains near zero across all layers, indicating that their parameter update directions are nearly orthogonal. This confirms that the decoupled training effectively induces distinct internal features for the two adapters.

4.4 More Analysis

Sensitivity Analysis. We conducted a comprehensive sensitivity analysis on the key hyperparameters of our reward functions, including the penalty coefficient λ , steepness factor κ , accuracy threshold τ , and length reward weight w_{len} . The results, detailed in Appendix B.1, demonstrate that AdaMix maintains robust performance across a reasonable range of parameter values.

Qualitative Analysis. We provide qualitative examples in Appendix D to show how AdaMix adaptively adjusts reasoning trajectories across varying problem complexities.

Routing Capability Comparison. To further evaluate the effectiveness of our routing mechanism, we compared it against a strong frontier model (GPT-5 via API) acting as a router. Although GPT-5 achieves a more aggressive token reduction, it leads to a significant degradation in accuracy, resulting in a lower Accuracy-Efficiency Score (AES) than our BERT-based router (Table 4).

5 Related Work

Large Reasoning Models. Large Language Models that excel at complex logic and multi-

Router	Δ Acc.	Δ \#Tok.	AES
Our BERT	+1.2%	-54.2%	0.58
GPT-5	-5.7%	-60.4%	0.50

Table 4: Comparison with a strong frontier model router.

step reasoning are often referred to as Large Reasoning Models (LRMs). They generate detailed Chains of Thought (CoT) (Wei et al., 2022) by explicitly outlining intermediate reasoning steps, mimicking human “System 2” cognition (Zhang et al., 2025a). Prominent examples include OpenAI’s GPT-o1 (Jaech et al., 2024), DeepSeek-AI’s DeepSeek-R1 (Guo et al., 2025), and the Qwen team’s QwQ (Yang et al., 2025a), which have demonstrated strong performance on challenging benchmarks. However, while beneficial for complex problems, this detailed reasoning approach can lead to overthinking (Chen et al., 2025)—producing excessively long responses for simpler tasks, hindering their practical efficiency in resource-constrained or latency-sensitive scenarios.

Efficient Reasoning for LRMs. To address the issue of overthinking, various approaches have been proposed to reduce response length while preserving accuracy. One line of research applies reinforcement learning (RL) with length constraints. For instance, O1-Pruner (Luo et al., 2025) introduces a length-harmonizing reward, ShorterBetter (Yi et al., 2025) uses the shortest correct sample for guidance, and LCPO (Aggarwal and Welleck, 2025) aligns outputs with pre-defined token budgets. ThinkPrune (Hou et al., 2025) dynamically restricts token usage during training via multi-round fine-tuning. Model merging offers an alternative, training-free solution by statically combining a slow-thinking model with a fast-thinking one (Wu et al., 2025). Techniques such as task vector merg-

ing (Ilharco et al., 2023) and singular value decomposition (SVD)-based merging (Stoica et al., 2025; Qiu et al., 2025) integrate the parameters of multiple models into a single one (Marczak et al., 2025; Lee et al., 2025). Another strategy is model routing, which employs a dedicated module to decide whether an input should be handled by a long-reasoning expert or a short-reasoning one (Ong et al., 2025; Liang et al., 2025).

6 Conclusion

We introduce AdaMix, a framework that addresses LRM overthinking via parametric adapter interpolation. By utilizing a difficulty-aware router to blend efficiency and accuracy adapters, AdaMix achieves fine-grained control over reasoning strategies. Experimental results confirm an effective accuracy-efficiency trade-off, reducing response length without compromising performance.

Limitations

We discuss several limitations of our work in this section: (1) Domain-Specific Calibration. Although the distinct reasoning patterns captured by the Short and Long adapters demonstrate transferability across tasks, the router’s perception of problem difficulty is inherently domain-dependent. Consequently, extending AdaMix to significantly different domains typically necessitates lightweight fine-tuning of the routing module to recalibrate the difficulty mapping. (2) Discrete Reasoning Granularity. Our framework currently utilizes a discrete set of fusion weights, which theoretically limits the continuous modulation of reasoning behavior compared to a continuous regression approach. Nevertheless, empirical observations indicate that the current discrete resolution is sufficient to effectively distinguish between the necessary reasoning depths for standard benchmarks.

Ethical Considerations

All models and datasets utilized in this work are publicly available and used in strict adherence to their respective licenses. While our framework focuses on optimizing the efficiency of Large Reasoning Models, we acknowledge that the underlying base models may still exhibit inherent biases or hallucinations, though we do not foresee significant ethical risks arising directly from this study.

Acknowledgments

This work was sponsored by National Key Research and Development Program of China (2023YFB2703604), National Natural Science Foundation of China (62441229), State Key Laboratory of Cognitive Intelligence of China (Open Project iED2023-004), and Sichuan Clinical Research Center for Imaging Medicine (YXYX2402).

References

- Pranjal Aggarwal and Sean Welleck. 2025. [L1: Controlling how long a reasoning model thinks with reinforcement learning](#). In *Second Conference on Language Modeling*.
- Daman Arora and Andrea Zanette. 2025. [Training language models to reason efficiently](#). In *The Thirtieth Annual Conference on Neural Information Processing Systems*.
- Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. [A survey on evaluation of large language models](#). *ACM transactions on intelligent systems and technology*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. [Evaluating large language models trained on code](#). *arXiv preprint arXiv:2107.03374*.
- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. [Do not think that much for 2+3=? on the overthinking of long reasoning models](#). In *Forty-second International Conference on Machine Learning*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- Muzhi Dai, Shixuan Liu, and Qingyi Si. 2025. [Stable reinforcement learning for efficient reasoning](#). *arXiv preprint arXiv:2505.18086*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*.

- Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2025. [Omni-math: A universal olympiad level mathematic benchmark for large language models](#). In *The Thirteenth International Conference on Learning Representations*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. [Deepseek-r1 incentivizes reasoning in llms through reinforcement learning](#). *Nature*.
- Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, and 1 others. 2025. [Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning](#). *arXiv preprint arXiv:2504.11456*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. [Measuring mathematical problem solving with the math dataset](#). In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. [Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning](#). *arXiv preprint arXiv:2504.01296*.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [Lora: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. [Editing models with task arithmetic](#). In *The Eleventh International Conference on Learning Representations*.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. [Openai o1 system card](#). *arXiv preprint arXiv:2412.16720*.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. [Livecodebench: Holistic and contamination free evaluation of large language models for code](#). In *The Thirteenth International Conference on Learning Representations*.
- Chanhyuk Lee, Jiho Choi, Chanryeol Lee, Donggyun Kim, and Seunghoon Hong. 2025. [Adarank: Adaptive rank pruning for enhanced model merging](#). *arXiv preprint arXiv:2503.22178*.
- Guosheng Liang, Longguang Zhong, Ziyi Yang, and Xiaojun Quan. 2025. [Thinkswitcher: When to think hard, when to think fast](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *The Twelfth International Conference on Learning Representations*.
- Yining Lu, Zilong Wang, Shiyang Li, Xin Liu, Changlong Yu, Qingyu Yin, Zhan Shi, Zixuan Zhang, and Meng Jiang. 2025. [Learning to optimize multi-objective alignment through dynamic reward weighting](#). *arXiv preprint arXiv:2509.11452*.
- Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. [O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning](#). *arXiv preprint arXiv:2501.12570*.
- Xinyin Ma, Guangnian Wan, Rungeng Yu, Gongfan Fang, and Xinchao Wang. 2025. [Cot-valve: Length-compressible chain-of-thought tuning](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*.
- Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D. Bagdanov, and Joost van de Weijer. 2025. [No task left behind: Isotropic model merging with common and task-specific subspaces](#). In *Forty-second International Conference on Machine Learning*.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. [Routellm: Learning to route llms from preference data](#). In *The Thirteenth International Conference on Learning Representations*.
- Haiquan Qiu, You Wu, Dong Li, Jianmin Guo, and Quanming Yao. 2025. [Superpose task-specific features for model merging](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. [Direct preference optimization: Your language model is secretly a reward model](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. [Gpqa: A graduate-level google-proof q&a benchmark](#). In *First Conference on Language Modeling*.

- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *arXiv preprint arXiv:2402.03300*.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. [Hybridflow: A flexible and efficient rlhf framework](#). In *Proceedings of the Twentieth European Conference on Computer Systems*.
- Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. 2024. [Slora: Scalable serving of thousands of lora adapters](#). In *Proceedings of Machine Learning and Systems*.
- George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. 2025. [Model merging with svd to tie the knots](#). In *The Thirteenth International Conference on Learning Representations*.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. [Stop overthinking: A survey on efficient reasoning for large language models](#). *Transactions on Machine Learning Research*.
- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. [Kimi k1. 5: Scaling reinforcement learning with llms](#). *arXiv preprint arXiv:2501.12599*.
- Chen Wang, Xunzhuo Liu, Yuhan Liu, Yue Zhu, Xiangxi Mo, Junchen Jiang, and Huamin Chen. 2025. [When to reason: Semantic router for vllm](#). *arXiv preprint arXiv:2510.08731*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. [Chain of thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*.
- Han Wu, Yuxuan Yao, Shuqi Liu, Zehua Liu, Xiaojin Fu, Xiongwei Han, Xing Li, Hui-Ling Zhen, Tao Zhong, and Mingxuan Yuan. 2025. [Unlocking efficient long-to-short llm reasoning with model merging](#). *arXiv preprint arXiv:2503.20641*.
- Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. [Ties-merging: Resolving interference when merging models](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. [Qwen3 technical report](#). *arXiv preprint arXiv:2505.09388*.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. [Qwen2.5-math technical report: Toward mathematical expert model via self-improvement](#). *arXiv preprint arXiv:2409.12122*.
- Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. 2025b. [Towards thinking-optimal scaling of test-time compute for LLM reasoning](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Jingyang Yi, Justin Wang, and Sida Li. 2025. [Shorter-better: Guiding reasoning models to find optimal inference length for efficient reasoning](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. [Language models are super mario: Absorbing abilities from homologous models as a free lunch](#). In *Forty-first International Conference on Machine Learning*.
- Duzhen Zhang, Zhong-Zhi Li, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhijiang Guo, Le Song, and Cheng-Lin Liu. 2025a. [From system 1 to system 2: A survey of reasoning large language models](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Ruiqi Zhang, Changyi Xiao, and Yixin Cao. 2025b. [Long or short cot? investigating instance-level switch of large reasoning models](#). *arXiv preprint arXiv:2506.04182*.

APPENDIX

A	Additional Experimental Details	12
A.1	Environment	12
A.2	Dataset Details	12
	A.2.1 Training Dataset.	12
	A.2.2 Evaluation Benchmarks.	12
A.3	Sampling Settings	13
A.4	Evaluation Metrics Details.	13
A.5	Baselines Details	14
A.6	Adapter Training	14
A.7	Routing Model Training	15
A.8	Reward Function Pseudocode	15
B	Additional Experiments	15
B.1	Sensitivity Analysis	15
B.2	Internal Representation Visualization	16
B.3	Detailed Experimental Results	17
B.4	Generalization across Diverse Backbones	17
B.5	Ablation Study on Data Filtering	17
B.6	Impact of Different Merging Methods	18
B.7	Efficiency Analysis	18
B.8	Training Efficiency.	19
C	Future Works	19
D	Qualitative Examples	19

A Additional Experimental Details

A.1 Environment

All experiments were conducted on a server equipped with 8 NVIDIA A6000 GPUs, Our implementation is built upon the veRL (Sheng et al. 2025) framework.

A.2 Dataset Details

This section provides detailed specifications of the datasets used for training and evaluating our proposed framework.

A.2.1 Training Dataset.

We utilize DeepMath-103K as our primary training corpus. This is a large-scale mathematical instruction tuning dataset curated to advance reasoning capabilities in Large Reasoning Models. It comprises approximately 103,000 mathematical problems spanning diverse topics. Each problem in the dataset is annotated with a difficulty score

```
# CONTEXT#
I am a teacher, and I have some high-level olympiad
↪ math problems.
I want to evaluate the difficulty of these math
↪ problems. There are some references available
↪ regarding the difficulty of the problems:
↪ <difficulty reference>
## Examples for difficulty levels
For reference, here are problems from each of the
↪ difficulty levels 1-10:
## Some known difficulty ratings of the competitions.
(Too long to show...)
</difficulty reference>
# OBJECTIVE#
A. Summarize the math problem in a brief sentence,
↪ describing the concepts involved in the math
↪ problem.
B. Based on the source of the given problem, as well
↪ as the difficulty of the problems referenced in
↪ these materials and the solution to the current
↪ problem, please provide an overall difficulty
↪ score for the current problem. The score should be
↪ a number between 1 and 10, with increments of 0.5,
↪ and should align perfectly with the materials.
# STYLE#
Data report.
# TONE#
Professional, scientific.
# AUDIENCE#
Students. Enable them to better understand the
↪ difficulty of the math problems.
# RESPONSE: MARKDOWN REPORT#
## Summarization
[Summarize the math problem in a brief paragraph.]
## Difficulty
[Rate the difficulty of the math problem and give the
↪ reason.]
# ATTENTION#
- Add "=== report over===" at the end of the report.

<math problem>
[Question]:
{{Question Here}}
[Solution]:
{{Solution Here}}
[Source]:
{{Source Here}}
</math problem>
```

Figure 8: The detailed difficulty classification prompt.

$d(x) \in [1.0, 10.0]$, enabling the precise stratification required by our decoupled training strategy. Specifically, we partition the dataset into an Easy Subset (problems with $d(x) \leq 5.5$) for the short adapter and a Hard Subset (problems with $d(x) > 5.5$) for the long adapter.

For datasets lacking difficulty annotations, we automate the scoring process using LLM (e.g., gpt4o). We adopt the methodology from recent work on LLM-based data processing (Gao et al., 2025; He et al., 2025), which uses a detailed prompt to instruct a model to assign a difficulty score from 1.0 to 10.0 for each problem. The prompt used for this task is shown in Figure 8.

A.2.2 Evaluation Benchmarks.

We employ a comprehensive set of benchmarks to evaluate the model’s performance across varying difficulty levels and domains.

❶ **GSM8K**. A dataset (Cobbe et al., 2021) consist-

Dataset	Domain	Size	License
DeepMath-103K	Math	103,111	MIT
GSM8K	Math	1,319	MIT
MATH500	Math	500	MIT
AMC23	Math	40	©MAA
AIME24	Math	30	©MAA
AIME25	Math	30	©MAA
LiveCodeBench	Code	1055	MIT
GPQA-Diamond	Science	198	MIT
MMLU	General	5,700	MIT

Model	License
DeepSeek-R1-Distill-Qwen-1.5B / 7B	MIT
Qwen2.5-Math-Instruct-1.5B / 7B	Apache 2.0
DeepSeek-R1-Distill-Llama-8B	MIT
Qwen3-4B	MIT

Table 5: Datasets and models used in this work.

ing of 8.5k high-quality grade school math word problems. We evaluate on the standard test set of 1,319 examples, a widely used benchmark for multi-step reasoning.

② **MATH500**. A curated subset (Lightman et al., 2024) of 500 problems from the MATH dataset, covering seven subjects including Algebra, Geometry, and Number Theory. It provides a representative evaluation of advanced mathematical reasoning across varying difficulties.

AMC23. This dataset consists of 40 problems selected from the 2023 AMC 12A and 12B contests. Targeting students in grade 12 and below, these exams feature challenges in algebra, geometry, number theory, and combinatorics, serving as a standard benchmark for high-school competition difficulty.

AIME24 / AIME25. These datasets are specialized benchmark collections derived from the 2024 and 2025 American Invitational Mathematics Examinations, respectively. Covering core secondary-school mathematics topics such as arithmetic, combinatorics, algebra, geometry, number theory, and probability, these collections place rigorous demands on both solution accuracy and conceptual depth. They are frequently employed to distinguish the upper limits of frontier reasoning models.

④ **LiveCodeBench**. A holistic benchmark (Jain et al., 2025) for code generation that collects problems from recent programming contests to prevent data contamination. We use this dataset to evaluate the model’s ability to transfer its reasoning patterns to code generation tasks.

⑤ **GPQA**. A challenging dataset (Rein et al., 2024) containing biology, physics, and chemistry questions written by domain experts. It is designed to be resistant to simple retrieval and requires high-level

scientific reasoning.

⑥ **MMLU**. A massive multitask benchmark (Hendrycks et al., 2021a) covering 57 subjects across STEM, the humanities, and social sciences. We use it to verify the model’s general knowledge understanding and robustness.

A.3 Sampling Settings

For all evaluations, we employ a consistent sampling configuration with a temperature of 0.6 and a top-p of 0.95. To minimize variance and ensure statistical stability, the reported metrics are averaged over multiple independent sampling runs for each question: 6 runs for large-scale datasets and 12 runs for smaller competition datasets (AMC 2023, AIME 2024, AIME 2025).

A.4 Evaluation Metrics Details.

Pass@K. To reliably estimate the functional correctness of the models, we employ the Pass@K metric. Given a problem, we generate n solutions ($n \geq k$) and count the number of correct solutions c . The unbiased estimator for Pass@K (Chen et al., 2021) is calculated as:

$$\text{Pass}@k := \mathbb{E}_{\text{Problems}} \left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right] \quad (9)$$

In our main experiments, we report the Pass@1 metric ($k = 1$) to reflect the expected accuracy.

AES. We adopt the Accuracy-Efficiency Score (AES) (Luo et al., 2025) to evaluate the effectiveness of reasoning compression. This metric quantifies whether the model can reduce response length without sacrificing accuracy.

The AES is derived from the percentage changes in output length (ΔL) and accuracy (ΔA) relative to the baseline model:

$$\Delta L = \frac{L_{\text{base}} - L_{\text{model}}}{L_{\text{base}}}, \quad \Delta A = \frac{A_{\text{model}} - A_{\text{base}}}{A_{\text{base}}} \quad (10)$$

where L and A denote the average token count and accuracy, respectively.

Based on these variations, the AES is defined as:

$$\text{AES} = \begin{cases} \varphi \cdot \Delta L + \eta \cdot |\Delta A| & \text{if } \Delta A \geq 0 \\ \varphi \cdot \Delta L - \theta \cdot |\Delta A| & \text{if } \Delta A < 0 \end{cases} \quad (11)$$

In our experiments, we set the parameters to $\varphi = 1$, $\eta = 1$, and $\theta = 1.5$. The condition $\theta > \eta$ imposes a stricter penalty for accuracy loss compared to the reward for accuracy gain, ensuring efficiency gains do not compromise reasoning reliability.

A.5 Baselines Details

This section provides a more detailed description of the baseline methods compared in our experiments.

❶ **DPO**: DPO (Rafailov et al., 2023) is a preference alignment method that eliminates reward modeling. For reasoning efficiency, we adapt it by creating preference pairs from responses to the same problem: shorter correct answers are preferred over longer ones. The model is fine-tuned to favor concise responses.

❷ **ShorterBetter**: This is a reinforcement learning (RL) method (Yi et al., 2025) designed to teach models to find their own efficient reasoning lengths. Its core contribution is the concept of Sample Optimal Length (SOL). For each prompt, multiple responses are generated, and the SOL is defined as the length of the *shortest correct* response in that sample set. This SOL then acts as a dynamic target in a reward function that positively rewards correctness while penalizing the deviation of a response’s length from the SOL. The model is optimized with on-policy RL like GRPO to produce correct, near-optimal-length outputs.

❸ **TLMRE**: This approach (Arora and Zanette, 2025) also uses reinforcement learning to directly incentivize shorter outputs. Its reward function combines a standard binary reward for accuracy with a direct penalty for the length of the response. A key feature is that the length penalty is only applied to correct answers, ensuring that accuracy is not unduly sacrificed. The length of each response is normalized on a per-prompt basis to account for varying problem difficulty. Controlled by a single hyperparameter adjusting length penalty strength, the method generates model families along the accuracy-efficiency trade-off.

❹ **ModelMerging**: This is a training-free approach (Wu et al., 2025) that statically improves efficiency by operating directly in the parameter space. It merges two models with distinct characteristics: a slow-thinking model proficient in detailed, long-form reasoning and a fast-thinking model capable of generating concise answers. Merging usually uses linear interpolation of model weights (or task vectors), creating a static model that combines parent strengths to shorten responses without further training or dynamic inference decisions.

❺ **CoT-Valve**: This method (Ma et al., 2025) aims to decouple the reasoning length from the model’s core capabilities by finding a length-controlling direction $\Delta\theta$ in the parameter space. It utilizes a

Parameter	Value
<i>RL Training Parameters</i>	
Length Reward Weight (w_{len})	3
Penalty Coefficient (λ)	0.5
Learning Rate	1×10^{-5}
Batch Size	128
Epochs	1
Cutoff Length	4096
Optimizer	constant
<i>LoRA Parameters</i>	
LoRA Rank (r)	64
LoRA Alpha (α)	128
LoRA Dropout	0.05
Target Modules	q_proj, v_proj

Table 6: Training hyperparameters for the short adapter.

valve mechanism based on Low-Rank Adaptation (LoRA). During tuning, it leverages a constructed dataset (MixChain) containing reasoning paths of varying lengths. During inference, the length of the reasoning chain can be dynamically adjusted by scaling the adapter weights using a coefficient α (via task arithmetic), effectively compressing or extending the CoT without retraining.

❻ **SwitchCoT**: This method (Zhang et al., 2025b) employs a two-stage, modular framework. In the first stage, a lightweight, independent classifier is trained to analyze an input prompt and make a binary decision: whether to use a long CoT or a short CoT strategy. In the second stage, the main reasoning model executes the inference based on the classifier’s choice. The decision dictates which prompt template is used to guide the main model, either encouraging a detailed thought process or a direct, concise answer. This strategy from reasoning but limits choices to hard binary selections.

A.6 Adapter Training

During the data generation stage for both adapters (to construct \mathcal{D}_{short} and \mathcal{D}_{long}), we set the number of candidate reasoning trajectories generated by each model (π_L and π_S) to $N=8$ for each problem. The specific hyperparameters for the RL training and LoRA configuration for each adapter are detailed in Tables 6 and 7. In terms of computational cost, the training required approximately 40 hours for the long adapter and 20 hours for the short adapter.

Parameter	Value
<i>RL Training Parameters</i>	
Length Reward Weight (w_{len})	0.2
Accuracy Threshold (τ)	0.3
Penalty Steepness (κ)	0.5
Learning Rate	1×10^{-5}
Batch Size	64
Epochs	1
Cutoff Length	24576
Optimizer	constant
<i>LoRA Parameters</i>	
LoRA Rank (r)	64
LoRA Alpha (α)	128
LoRA Dropout	0.05
Target Modules	q_proj, v_proj

Table 7: Training hyperparameters for the long adapter.

A.7 Routing Model Training

The training of the routing model involves two key stages: supervision data generation and model fine-tuning. We define the set of candidate mixing weights as $\mathcal{A} = \{0, 0.1, \dots, 0.9, 1\}$. The entire training process is conducted on a dataset of 6,000 samples and takes approximately half an hour.

❶ **Supervision Data Generation.** In this offline process, we generate the ground-truth labels for training the router. To calculate the utility score $U(x, \alpha_i)$ for each problem-weight pair, we generate $N=6$ samples to obtain a stable accuracy and average length. The preference coefficient β in the utility function was set to 0.2.

❷ **Model Fine-tuning.** The routing model, built on bert-base-uncased (Devlin et al., 2019), is fine-tuned for 100 epochs with a learning rate of $2e-5$ and a batch size of 64.

A.8 Reward Function Pseudocode

Algorithm 1 details the reward calculation logic.

B Additional Experiments

B.1 Sensitivity Analysis

To evaluate the robustness of AdaMix and verify the rationality of our hyperparameter selection, we conducted a sensitivity analysis on key parameters governing the reward mechanism: the penalty coefficient (λ) for the Short adapter, and the steepness factor (κ) and accuracy threshold (τ) for the Long

Algorithm 1 Reward Calculation Mechanism

Require: Response y , Group Accuracy acc_G

Require: Short Params: \mathcal{Y}_S (Reference samples), λ, γ

Require: Long Params: \mathcal{L}_{batch} (Batch correct lengths), τ, κ

Ensure: Reward R

```

1:  $r_{acc} \leftarrow \mathbb{I}(y \text{ is correct})$ 
2: if Optimizing Short Adapter then
3:   // Filter correct reference samples
4:    $\mathcal{Y}_S^c \leftarrow \{y' \in \mathcal{Y}_S \mid y' \text{ is correct}\}$ 
5:   // Compute dynamic length threshold
6:    $l_{avg} \leftarrow \frac{1}{|\mathcal{Y}_S^c|} \sum_{y' \in \mathcal{Y}_S^c} l(y')$ 
7:    $\delta \leftarrow \gamma \cdot (1 - acc_G)$ 
8:    $L_{ref} \leftarrow l_{avg} + \delta$ 
9:   if  $y$  is correct then
10:     $r_{len} \leftarrow 1 - \lambda \cdot \max(0, l(y) - L_{ref})$ 
11:   else
12:     $r_{len} \leftarrow 0$ 
13:   end if
14:    $R \leftarrow r_{acc} + w_{len}^S \cdot r_{len}$ 
15: else if Optimizing Long Adapter then
16:    $r_{len} \leftarrow 0$ 
17:   if  $acc_G \geq \tau$  and  $y$  is correct then
18:     $\mu_L, \sigma_L \leftarrow \text{Mean}(\mathcal{L}_{batch}), \text{Std}(\mathcal{L}_{batch})$ 
19:     $r_{len} \leftarrow \sigma \left( -\kappa \cdot \frac{l(y) - \mu_L}{\sigma_L} \right)$ 
20:   end if
21:    $R \leftarrow r_{acc} + w_{len}^L \cdot r_{len}$ 
22: end if
23: return  $R$ 

```

adapter. We also provide a detailed ablation on the length reward weight (w_{len}) across different levels.

- **Short Adapter (λ).** As λ increases from 0.1 to 0.5, the trajectory moves horizontally to the left, indicating significant token reduction (2.2k \rightarrow 1.4k) with minimal accuracy loss. However, beyond $\lambda = 0.5$, the curve dips sharply downwards, signaling a detrimental trade-off. This identifies $\lambda = 0.5$ as the inflection point.
- **Long Adapter (κ).** The trajectory shows a gradual shift towards higher efficiency. Increasing κ up to 0.5 yields a reasonable reduction in tokens while maintaining accuracy above 85.8%. Larger κ values (> 0.5) push the model into a region of diminishing returns where accuracy begins to degrade.
- **Long Adapter (τ).** The trajectory moves diagonally upwards. Lower τ values (0.1–0.2) suppress token usage but cap accuracy. Increasing

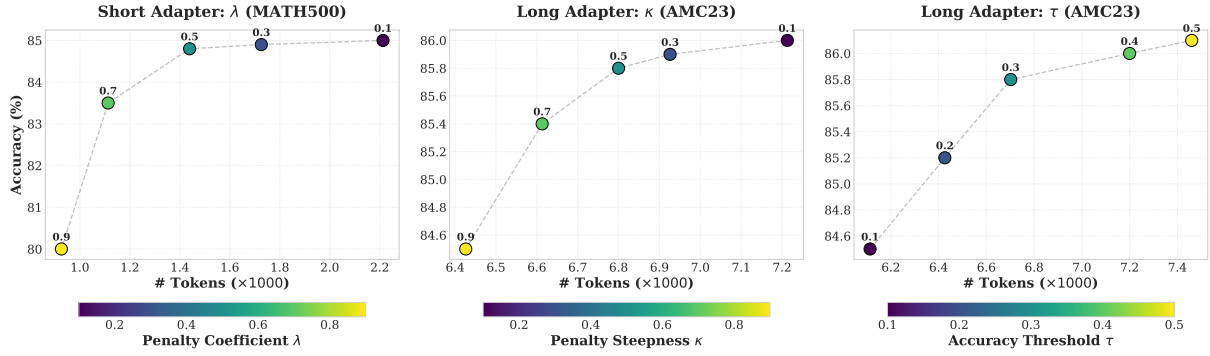


Figure 9: Hyperparameter sensitivity analysis visualized as trajectories in the Accuracy-Efficiency plane. The plots show the impact of varying λ for the Short adapter on MATH500 (Left), and varying κ (Center) and τ (Right) for the Long adapter on AMC23. Color intensity indicates the parameter value.

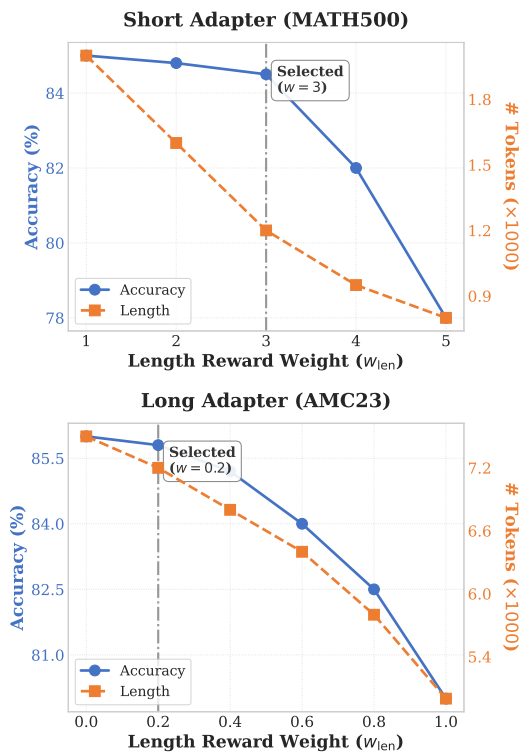


Figure 10: Impact of length reward weight (w_{len}) on accuracy and response length. **Top**: Short adapter evaluated on MATH500. **Bottom**: Long adapter evaluated on AMC23.

τ to 0.3 allows the model to reach a higher accuracy plateau ($\sim 85.8\%$) with a moderate increase in length, marking a balanced operating point before the token cost becomes excessive.

- **Length Reward Weight.** We conducted a sensitivity analysis on representative datasets of varying difficulty to investigate the impact of w_{len} . For the Short adapter evaluated on the relatively simple MATH500 benchmark, Figure 10 (Top) shows that increasing w_{len} significantly reduces

length, yet accuracy begins to drop noticeably beyond 3. For the Long adapter tested on the more challenging AMC23 benchmark, Figure 10 (Bottom) reveals that a mild penalty ($w_{len} = 0.2$) trims redundancy without hindering the deep reasoning required for complex problems, whereas larger weights impair performance. Based on these observations across different difficulty levels, we adopted $w_{len} = 3$ and $w_{len} = 0.2$ as our default settings.

B.2 Internal Representation Visualization

To investigate how the decoupled training strategy impacts the underlying model parameters, we visualized the layer-wise weight updates of the learned short and long adapters. As illustrated in Figure 7, we observe distinct patterns in their internal representations:

- **Layer-wise Distribution:** The short adapter exhibits weight updates that are primarily concentrated in the shallow and middle layers of the network. In contrast, the long adapter shows a marked increase in both weight magnitude and maximum singular value in the deeper layers (specifically after Layer 20), suggesting that its parameter modifications are predominantly localized in the final stages of the transformer.
- **Parameter Orthogonality:** A key observation is that the cosine similarity between the weight update matrices of the two adapters remains consistently near zero across all layers. This indicates that the update directions of the short and long adapters are nearly orthogonal in the parameter space. Such separation implies that the two adapters have acquired distinct internal features without significant interference, validating the

Method	GSM8K		MATH500		AMC23		AIME24		AIME25		Average		
	Acc.	#Tok.	Acc.	#Tok.	Acc.	#Tok.	Acc.	#Tok.	Acc.	#Tok.	Δ Acc.	Δ #Tok.	AES \uparrow
<i>Qwen2.5-1.5B-Math-Ins</i>	75.5	354	68.9	601	53.3	1077	7.8	1020	8.9	941	-	-	-
<i>R1-Distill-Qwen-1.5B</i>	78.7	787	82.2	3837	66.7	8928	29.3	14358	23.9	14820	0.0%	0.0%	0.00
+ DPO	71.7	486	76.2	2380	59.7	5412	20.4	8694	16.7	7234	-7.2%	-41.2%	0.15
+ ShorterBetter	68.5	216	62.0	515	40.0	1109	13.3	1928	9.7	2298	-17.5%	-83.6%	<u>0.26</u>
+ TLMRE	80.4	672	81.0	2417	<u>67.0</u>	4821	<u>27.8</u>	11623	<u>23.3</u>	11222	<u>-0.3%</u>	-28.2%	<u>0.26</u>
+ ModelMerging	77.1	512	75.4	1896	49.6	2768	9.4	<u>4961</u>	13.9	<u>5273</u>	-11.1%	<u>-56.9%</u>	0.13
+ CoT-Valve	75.6	549	73.7	2668	61.4	4965	23.3	7285	20.0	8047	-5.4%	-40.0%	-0.10
+ SwitchCoT	77.5	813	<u>82.5</u>	4895	67.5	8138	<u>27.8</u>	15397	<u>23.3</u>	15984	-0.4%	+7.4%	0.22
+ AdaMix	<u>79.4</u>	<u>416</u>	83.2	<u>1557</u>	67.5	<u>2392</u>	30.0	7476	26.7	8426	+1.2%	-54.2%	0.58
<i>Qwen2.5-7B-Math-Ins</i>	91.3	304	81.8	687	61.3	1202	13.3	1511	10.0	1432	-	-	-
<i>R1-Distill-Qwen-7B</i>	88.0	623	88.8	2968	84.1	6284	46.3	11412	38.5	12869	0.0%	0.0%	0.00
+ DPO	85.8	375	87.3	2103	70.6	4316	40.0	8793	33.3	9381	-5.7%	-30.1%	0.16
+ ShorterBetter	81.7	<u>246</u>	77.2	723	71.5	<u>2003</u>	38.8	<u>4556</u>	30.0	<u>4442</u>	-9.3%	-66.0%	<u>0.44</u>
+ TLMRE	88.2	377	87.0	1832	<u>82.5</u>	3220	50.0	9614	33.3	10465	-0.9%	-32.2%	0.29
+ ModelMerging	89.1	499	78.2	<u>1320</u>	69.5	2320	30.0	4302	26.7	4428	-10.4%	-53.3%	0.25
+ CoT-Valve	87.9	368	85.6	<u>1742</u>	75.0	3462	43.3	6515	33.3	6937	-4.1%	-43.2%	0.33
+ SwitchCoT	86.5	442	89.4	2091	<u>82.5</u>	4200	<u>46.1</u>	10988	<u>40.6</u>	12133	<u>-0.1%</u>	-20.2%	0.20
+ AdaMix	<u>88.7</u>	174	<u>88.5</u>	1512	84.6	1847	50.0	6792	43.3	7460	+1.9%	<u>-54.9%</u>	0.59

Table 8: Comparison on five math benchmarks, covering accuracy (Acc.), average response length (#Tok.), and accuracy-efficiency score (AES). The best and second-best results are marked in **bold** and underlined, respectively.

Backbone	GSM8K		MATH500		AMC23		AIME24		AIME25		Average	
	Acc.	#Tok.	Acc.	#Tok.	Acc.	#Tok.	Acc.	#Tok.	Acc.	#Tok.	Δ Acc.	Δ #Tok.
<i>R1-Distill-Llama-8B</i>												
Base	90.7	929	86.7	3675	81.2	6790	46.7	12717	30.0	11595	-	-
+AdaMix	89.4	213	88.2	2164	79.6	2045	53.3	7344	33.3	7492	+1.7%	-53.1%
<i>Qwen3-4B</i>												
Base	95.0	1428	93.4	4696	91.2	7034	64.3	11523	55.6	12476	-	-
+AdaMix	96.2	356	92.5	2577	92.4	2308	62.0	8029	58.3	7863	+0.4%	-50.9%

Table 9: Generalization performance across different model backbones. We report accuracy and token length across five mathematical datasets and their average improvements.

effectiveness of the decoupled training approach in learning specialized reasoning behaviors.

B.3 Detailed Experimental Results

Table 8 presents the comprehensive performance breakdown across all five benchmarks for both 1.5B and 7B models. Consistent with the averaged results, AdaMix demonstrates robust adaptability across varying difficulty levels.

B.4 Generalization across Diverse Backbones

This section provides the per-dataset performance breakdown for different model backbones, including DeepSeek-R1-Distill-Llama-8B and Qwen3-4B. As detailed in Table 9, AdaMix achieves significant token reduction across all five mathematical benchmarks while maintaining or even improving

the baseline accuracy. These results further validate that our decoupled reasoning framework is architecture-agnostic.

B.5 Ablation Study on Data Filtering

To investigate the impact of our data filtering strategies (i.e., filtering based on intrinsic complexity and capability boundaries), we conducted an ablation study. Table 10 presents the average results across the five mathematical benchmarks. We compared the full AdaMix framework against variants without short adapter data filtering and without long adapter data filtering. Additionally, we included strong baselines (ShorterBetter and CoT-Valve) for reference.

Setting	Δ Acc.	Δ #Tok.	AES
AdaMix	+1.9%	-54.9%	0.59
w/o short filtering	+0.8%	-50.3%	0.52
w/o long filtering	+0.6%	-48.7%	0.49
ShorterBetter	-9.3%	-66.0%	0.44
CoT-Valve	-4.1%	-43.2%	0.33

Table 10: Ablation study on the impact of data filtering strategies. Results are averaged across the five evaluation benchmarks.

B.6 Impact of Different Merging Methods

In the AdaMix framework, we employ Task Arithmetic to enable continuous modulation of reasoning behaviors. To investigate the impact of model merging techniques on experimental results, we compared the following mainstream merging methods:

- **Task Arithmetic:** The weighted sum of task vectors. By treating the weights of fine-tuned models as vectors in the parameter space, this approach (Ilharco et al., 2023) allows for arithmetic operations like addition and subtraction to combine or negate specific model capabilities effectively.
- **Ties-Merging:** A method (Yadav et al., 2023) designed to resolve parameter interference through three steps: trimming redundant parameters based on magnitude, resolving sign conflicts to maintain direction consistency, and disjoint merging.
- **DARE-Linear:** A method (Yu et al., 2024) based on "Drop And REscale". It randomly sets a fraction of delta parameters to zero to reduce redundancy and rescales the remaining parameters to approximate the original embeddings, followed by linear merging.
- **SVD:** A merging approach (Stoica et al., 2025) based on Singular Value Decomposition, which typically operates in the eigenspace of the weight matrices. By decomposing the weight matrices into singular vectors, this method can align the internal representations of different models more effectively than direct parameter averaging.
- **Ties-SVD:** A technique that combines the interference resolution mechanism of Ties-Merging with the decomposition approach of SVD.

We integrated these different merging algorithms into the AdaMix framework and evaluated their per-

Merging Method	MATH500		AIME25	
	Acc.	#Tok.	Acc.	#Tok.
Task Arithmetic	88.5	1512	43.3	7460
Ties-Merging	87.9	1345	45.6	7627
DARE-Linear	88.1	1578	40.6	7354
SVD	88.7	1694	43.3	7892
Ties-SVD	87.6	1625	40.0	7515

Table 11: Comparison of different adapter merging methods on MATH500 and AIME25.

Module	Latency (ms)	Ratio
Backbone Generation	9520.0	\sim 99.9%
Router Inference	8.4	\sim 0.1%

Table 12: Latency comparison between router inference and backbone model generation.

formance based on the DeepSeek-R1-Distill-Qwen-7B model on the MATH500 and AIME25 datasets. As shown in Table 11, while there are minor fluctuations in accuracy and average response length across different methods, the overall performance differences are not significant.

B.7 Efficiency Analysis

We conducted two sets of experiments to verify the computational efficiency of the AdaMix framework: (1) Router Latency Analysis, and (2) Grouped Inference Efficiency. All experiments were conducted on the MATH500 dataset using a single NVIDIA A6000 GPU.

Router Latency Analysis. We first measured the overhead introduced by the routing module itself. We compared the average inference time of the router against the average generation time of the backbone model (DeepSeek-R1-Distill-Qwen-7B) with a batch size of 1.

As shown in Table 12, the router requires significantly less time compared to the generation process. The introduced latency is approximately 8.4 ms, which is less than 1% of the total inference time. This confirms that the routing step does not create a bottleneck in the overall pipeline.

Efficiency of Grouped Inference. To further validate the efficiency of our request grouping strategy in high-throughput scenarios, we compared the inference throughput under two serving settings with a batch size of 128:

Strategy	Throughput (req/s)	Speedup
Naive Serving	5	1.0×
Grouped Serving	39	7.9×

Table 13: Throughput comparison between Naive and Grouped serving strategies (Batch Size = 128).

Method	Training FLOPs
Single-model DPO	1.683
AdaMix-Short Adapter	0.701
AdaMix-Long Adapter	1.753
AdaMix-Router	0.005
AdaMix (Total)	2.460

Table 14: Comparison of training FLOPs (DeepSeek-R1-Distill-Qwen-7B).

- **Naive Serving:** Requests are processed sequentially, switching the adapter parameters for each request based on its target weight $\hat{\alpha}$.
- **Grouped Serving:** Requests are first sorted and grouped by their target weight $\hat{\alpha}$, allowing sub-batches to be processed in parallel using a shared adapter.

As shown in Table 13, the Grouped Serving strategy achieves a throughput that is significantly higher than the Naive approach. This confirms that grouping effectively mitigates the overhead of multi-adapter management.

B.8 Training Efficiency.

To clarify the computational overhead of our decoupled strategy, we compare the training FLOPs of AdaMix against a single-model multi-objective DPO baseline. While AdaMix incurs higher total training costs due to its dual-adapter design, it consistently achieves a superior accuracy-efficiency trade-off as demonstrated in our main results. Furthermore, AdaMix offers better scalability; adapting to new domains typically requires only a lightweight router update, whereas DPO often necessitates a full retraining of the entire model.

C Future Works

While AdaMix has demonstrated effectiveness in optimizing the trade-off between reasoning efficiency and accuracy, several intriguing directions remain for future exploration:

- **Universal difficulty perception.** Our current routing module relies on domain-specific supervi-

sion to assess problem complexity. A promising direction is to investigate unsupervised difficulty estimation metrics, such as model uncertainty or intrinsic perplexity variance. This would enable the router to assess problem complexity in a zero-shot manner, eliminating the need for recalibration when transferring to new domains.

- **Continuous and dynamic modulation.** The current framework utilizes a discrete set of fusion weights. Future iterations could employ regression-based routers or hypernetworks to generate continuous mixing coefficients in real-time. This would allow for smoother, fine-grained control over the model’s reasoning behaviors, potentially finding optimal operating points that discrete selection might miss.
- **Scalability and multimodal extension.** While this work validates AdaMix on standard-scale models, we plan to apply it to significantly larger backbones (e.g., 70B+ parameters) to verify its scalability in high-compute regimes. Furthermore, we aim to adapt the decoupled framework to Multimodal Large Language Models (MLLMs), exploring how distinct adapters can optimize efficiency in complex reasoning tasks involving both visual and textual inputs.

D Qualitative Examples

This section qualitatively compares the reasoning trajectories generated by AdaMix against the baseline across different difficulty levels. On simple problems, the baseline often exhibits “overthinking” with verbose steps, whereas AdaMix activates the short adapter to produce a concise and direct solution. For medium-difficulty tasks, AdaMix selects a mixed adapter that strikes a balance, generating a path less verbose than the baseline’s but retaining all critical logic. On complex problems where the baseline may fail, AdaMix leverages the long adapter to perform deeper, more exhaustive exploration. This often results in a longer reasoning path but enables AdaMix to solve challenging problems that are intractable for the baseline, thereby improving accuracy.

Easy Question: Jerry has two pools, both with leaks emptying them out at 4 gallons/minute. 4 minutes ago, the big pool had twice as much water as the small pool. Now the big pool has four times as much water as the small pool. How much water does the small pool have now? (Answer: 8)

AdaMix (reasoning intensity = 0)

Okay, so Jerry has two pools, both leaking at 4 gallons per minute. Let me denote the current amount of water in the small pool as S gallons and the big pool as B gallons. Four minutes ago, the big pool had twice as much water as the small one. So, four minutes ago, the big pool had $B + 4 \times 4 = B + 16$ gallons, and the small pool had $S + 16$ gallons. According to the problem, $B + 16 = 2 \times (S + 16)$. Now, the big pool has four times as much as the small one, so $B = 4S$. Substituting $B = 4S$ into the first equation: $4S + 16 = 2(S + 16) \implies 4S + 16 = 2S + 32 \implies 2S = 16 \implies S = 8$. So the small pool has 8 gallons now.

Answer: ✓ Tokens:

DeepSeek-R1-Distill-Qwen

First, I'll define the variables to represent the current amount of water in each pool. Let S be the current amount of water in the small pool, and B be the current amount of water in the big pool. Four minutes ago, the big pool had twice as much water as the small pool. Since both pools are leaking at 4 gallons per minute, four minutes ago, the big pool had $B + 16$ gallons and the small pool had $S + 16$ gallons. According to the problem, four minutes ago, the big pool had twice as much water as the small pool. This gives the equation: $B + 16 = 2(S + 16)$. Simplifying this equation: $B + 16 = 2S + 32$, $B = 2S + 16$. Now, we know that currently, the big pool has four times as much water as the small pool: $B = 4S$. Substituting $B = 4S$ into the previous equation: $4S = 2S + 16$, $2S = 16$, $S = 8$. Therefore, the small pool currently has 8 gallons of water.

Let's solve the problem step by step. Given: both pools are leaking at a rate of 4 gallons per minute; four minutes ago, the big pool had twice as much water as the small pool; currently, the big pool has four times as much water as the small pool. Let's define: S = current amount of water in the small pool (gallons), B = current amount of water in the big pool (gallons). Four minutes ago: small pool had $S + 4 \times 4 = S + 16$ gallons, big pool had $B + 4 \times 4 = B + 16$ gallons. According to the problem: $B + 16 = 2(S + 16)$. Simplify the equation: $B + 16 = 2S + 32$, so $B = 2S + 16$ (1). Also, currently: $B = 4S$ (2). Substitute (2) into (1): $4S = 2S + 16$, $4S - 2S = 16$, $2S = 16$, $S = 8$.

Answer: ✓ Tokens:

Medium Question: Find the constant term in the expansion of $(10x^3 - \frac{1}{2x^2})^5$ (Answer: -125)

AdaMix (reasoning intensity = 0.3)

Okay, so I need to find the constant term in the expansion of $(10x^3 - \frac{1}{2x^2})^5$. Hmm, I remember that the binomial theorem can be used here. Each term in the expansion is of the form $\binom{5}{k} \cdot (10x^3)^{5-k} \cdot (-\frac{1}{2x^2})^k$. I need to find the term where the exponent of x is zero because that's the constant term. Let's calculate the exponent of x in each term: Exponent from $(10x^3)^{5-k}$ is $3 \times (5 - k)$, and from $(-\frac{1}{2x^2})^k$ is $-2k$. So the total exponent is $3 \times (5 - k) - 2k = 15 - 3k - 2k = 15 - 5k$. Set $15 - 5k = 0 \Rightarrow 5k = 15 \Rightarrow k = 3$. So the constant term is when $k = 3$. Now compute the term: $\binom{5}{3} \cdot (10x^3)^2 \cdot (-\frac{1}{2x^2})^3 \cdot \binom{5}{3} = 10 \cdot (10x^3)^2 = 100x^6 \cdot (-\frac{1}{2x^2})^3 = -\frac{1}{8x^6}$. Multiply them together: $10 \times 100x^6 \times (-\frac{1}{8x^6}) = 10 \times 100 \times (-\frac{1}{8}) = 10 \times (-12.5) = -125$. So the constant term is -125 .

Answer: ✓ Tokens:

DeepSeek-R1-Distill-Qwen

To find the constant term in the expansion of $(10x^3 - \frac{1}{2x^2})^5$, I'll use the binomial theorem. First, I'll identify the general term in the expansion, which is given by: $\binom{5}{k} \cdot (10x^3)^{5-k} \cdot (-\frac{1}{2x^2})^k$. Next, I'll simplify the exponents of x in the general term to find the value of k that results in a constant term. Setting the exponent equal to zero: $3(5 - k) - 2k = 0 \Rightarrow 15 - 5k = 0 \Rightarrow k = 3$. Now, I'll substitute $k = 3$ into the general term to find the constant term: $\binom{5}{3} \cdot (10x^3)^2 \cdot (-\frac{1}{2x^2})^3 = 10 \cdot 100x^6 \cdot (-\frac{1}{8x^6}) = -125$. Therefore, the constant term in the expansion is -125 . To find the constant term in the expansion of $(10x^3 - \frac{1}{2x^2})^5$, we'll use the binomial theorem, which states: $(a + b)^n = \sum_{k=0}^n \binom{n}{k} a^{n-k} b^k$. In this case, $a = 10x^3$, $b = -\frac{1}{2x^2}$, and $n = 5$. ### Step 1: Identify the General Term The general term in the expansion is given by: $\binom{5}{k} \cdot (10x^3)^{5-k} \cdot (-\frac{1}{2x^2})^k$ ### Step 2: Simplify the Exponents of x First, simplify the exponents of x in the general term: $(10x^3)^{5-k} = 10^{5-k} \cdot x^{3(5-k)} = 10^{5-k} \cdot x^{15-3k}$ $(-\frac{1}{2x^2})^k = (-1)^k \cdot (\frac{1}{2})^k \cdot x^{-2k} = (-1)^k \cdot \frac{1}{2^k} \cdot x^{-2k}$ Combine these: $10^{5-k} \cdot (-1)^k \cdot \frac{1}{2^k} \cdot x^{15-3k-2k} = 10^{5-k} \cdot (-1)^k \cdot \frac{1}{2^k} \cdot x^{15-5k}$ ### Step 3: Find the Constant Term A constant term occurs when the exponent of x is zero: $15 - 5k = 0 \Rightarrow k = 3$ ### Step 4: Substitute $k = 3$ into the General Term Now, substitute $k = 3$ into the general term: $\binom{5}{3} \cdot 10^{5-3} \cdot (-1)^3 \cdot \frac{1}{2^3} \cdot x^{15-5 \cdot 3}$ Simplify each part: $\binom{5}{3} = 10$ $10^{5-3} = 10^2 = 100$ $(-1)^3 = -1$ $\frac{1}{2^3} = \frac{1}{8}$ $x^{15-15} = x^0 = 1$ Multiply these together: $10 \cdot 100 \cdot (-1) \cdot \frac{1}{8} = 10 \cdot 100 \cdot (-\frac{1}{8}) = 10 \cdot (-12.5) = -125$ ### Final Answer The constant term in the expansion is:

Answer: ✓ Tokens:

Hard Question: Let ABCDEF be a convex equilateral hexagon in which all pairs of opposite sides are parallel. The triangle whose sides are extensions of segments AB, CD, and EF has side lengths 200, 240, and 300. Find the side length of the hexagon. (Answer: 80)

AdaMix (reasoning intensity = 1)

Okay, so I have this geometry problem about a convex equilateral hexagon where all pairs of opposite sides are parallel. The triangle formed by extending the sides AB, CD, and EF has side lengths 200, 240, and 300. [Omitted] Wait, but is this triangle itself a right triangle? Let me check: $20^2 + 24^2 = 400 + 576 = 976$, which is not equal to $30^2 = 900$, so it's not a right triangle. [Omitted] Wait, maybe the formula is $s = (a \cdot b \cdot c) / (a + b + c)$, but that doesn't seem right. Alternatively, perhaps $1/s = 1/a + 1/b + 1/c$. Wait, but $1/200 + 1/240 + 1/300 \approx 0.005 + 0.004166 + 0.003333 = 0.0125$, so $s = 80$. So, numerator = $72000 + 60000 + 48000 = 180000$. Denominator = $200 \cdot 240 \cdot 300 = 14,400,000$. Thus, $1/s = 180000/14,400,000 = 1/80$, so $s = 80$. So, perhaps the side length of the hexagon is 80. Hmm, that seems possible. [Omitted] The formula involves the reciprocal of the sum of the reciprocals of the triangle's sides: $\frac{1}{s} = \frac{1}{200} + \frac{1}{240} + \frac{1}{300}$. Calculating each term: $\frac{1}{200} = 0.005$, $\frac{1}{240} \approx 0.0041667$, $\frac{1}{300} \approx 0.0033333$. Adding these together: $0.005 + 0.0041667 + 0.0033333 = 0.0125$. Thus, the side length s is: $s = \frac{1}{0.0125} = 80$. Therefore, the side length of the hexagon is 80.

Answer: 80 ✓ Tokens: 7442

DeepSeek-R1-Distill-Qwen

Alright, so I'm trying to solve this problem about a convex equilateral hexagon with opposite sides parallel. The triangle formed by extending sides AB, CD, and EF has side lengths 200, 240, and 300. I need to find the side length of the hexagon. The semi-perimeter, s_T , is $(200 + 240 + 300)/2 = 740/2 = 370$. Then the area is $\sqrt{370(370 - 200)(370 - 240)(370 - 300)} = \sqrt{370 \cdot 170 \cdot 130 \cdot 70}$, which approximately equals 23,924. [Omitted] I tried to model the problem by considering that the hexagon divides the triangle into three smaller, similar triangles. If the scaling factors are k_a, k_b, k_c , then $s = 200(1 - 2k_a) = 240(1 - 2k_b) = 300(1 - 2k_c)$. However, solving for the scaling factors using these equations resulted in identities (like $s = s$), indicating I need a different relationship or formula. [Omitted] I tested specific values for the side length s . For $s = 120$, the calculated area of the hexagon $A_{hex} = A_T - \sum A_{small}$ did not match the expected area of a regular hexagon. I also tried $s = 100$ and $s = 150$, but they were also inconsistent. [Omitted] Finally, I considered the formula involving the sum of reciprocals: $1/s = 1/a + 1/b + 1/c$. Calculating $1/200 + 1/240 + 1/300 \approx 0.0125$, which leads to $s = 1/0.0125 = 80$. Although earlier area checks were confusing, this result appears to be the intended solution for this specific configuration of an equilateral hexagon formed by extending triangle sides. The side length of the equilateral hexagon is 80.

Answer: 80 ✓ Tokens: 18396