RelayLLM: Efficient Reasoning via Collaborative Decoding

Anonymous ACL submission

Abstract

Deploying Large Language Models (LLMs) for complex reasoning is often hindered by high computational costs and latency, while resource-efficient Small Language Models (SLMs) typically lack the necessary reasoning capacity. Existing collaborative approaches, such as cascading or routing, operate at a coarse granularity by offloading entire queries to LLMs, resulting in significant computational waste when the SLM is capable of handling the majority of reasoning steps. To address this, we propose ***RelayLLM***, a novel framework for efficient reasoning via token-level collaborative decoding. Unlike routers, *RelayLLM* empowers the SLM to act as an active controller that dynamically invokes the LLM only for critical tokens via a special command, effectively "relaying" the generation process. We introduce a two-stage training framework, including supervised warm-up and Group Relative Policy Optimization (GRPO) to teach the SLM to balance independence with strategic help-seeking. Empirical results across six benchmarks demonstrate that *RelayLLM* improves the average accuracy from 42.5% to 49.52%, effectively bridging the performance gap between the two models. Notably, this is achieved by invoking the LLM for only **1.07%** of the total generated tokens, offering a 98.2% cost reduction compared to performance-matched random routers.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in complex reasoning and problem-solving (Comanici et al., 2025; Yang et al., 2025a; Achiam et al., 2023). However, their deployment is often constrained by high computational costs and latency. In contrast, Small Language Models (SLMs) are resource-efficient options but typically struggle with hard reasoning tasks due to their limited capacity (Kaplan et al., 2020). This trade-off has motivated the development of collaborative systems that combine the reasoning capabilities of LLMs with the efficiency of SLMs (Hakimov et al., 2025).

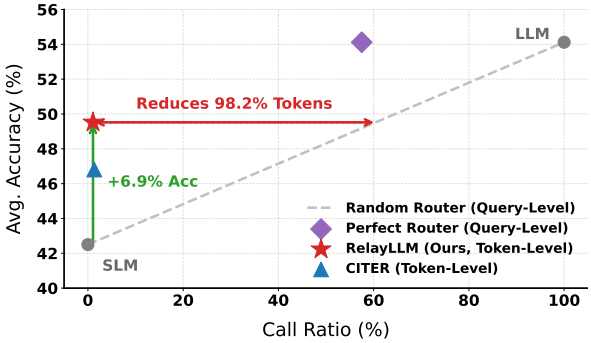

Figure 1: Results are averaged across six mathematical benchmarks. The "Random Router" baseline randomly directs questions to either the SLM or the LLM. The "Perfect Router" baseline directs only the questions the SLM cannot solve to the LLM. The x-axis represents the Call Ratio (percentage of tokens generated by the LLM), and the y-axis denotes the average accuracy on six mathematical reasoning benchmarks.

Existing approaches to different-sized model collaboration often rely on "cascading" or "routing" mechanisms, where a router determines the difficulty of a query and directs it to either an SLM or an LLM (Ding et al., 2024; Hu et al., 2024; Ong et al., 2024). While effective to some extent, these methods typically operate at a coarse granularity by offloading the entire generation task to the LLM once a query seems difficult. This "all-or-nothing" strategy leads to significant computational waste, as the SLM often possesses the competence to handle the majority of the reasoning steps, requiring LLM intervention only at specific critical positions (Lin et al., 2024; Ruan et al., 2025).

To address these problems, we propose *RelayLLM*, a novel framework for efficient reasoning via token-level collaborative decoding (Shen et al., 2024) without an additional controller. Un-

like static routers, *RelayLLM* empowers the SLM to act as both a problem solver and an active controller that dynamically requests assistance from the LLM only when necessary.

Inspired by tool-use agents (Wölflein et al., 2025; Zheng et al., 2025a), we introduce an interleaved generation process where the SLM generates a special command token (`<call>`) to pause its own generation and invoke the LLM for a specified number of tokens. The SLM then receives the LLM's guidance and resumes reasoning, effectively "relaying" the output process between models. We propose a two-stage framework to equip the SLM with this strategic delegation capability to train the *RelayLLM*. We first employ a supervised warm-up phase to teach the model the syntactic structure of calling commands. This is followed by a reinforcement learning stage using Group Relative Policy Optimization (GRPO) (Shao et al., 2024) training, where we design a context-aware reward that guides the model to balance independence with necessary help-seeking, penalizing both wasted costs and avoidable errors.

Empirical results on six benchmarks demonstrate the effectiveness of our approach. As illustrated in Figure 1, *RelayLLM* achieves an average accuracy of 49.52% on six benchmarks, largely recovering the performance gap between the SLM and the LLM. Remarkably, this gain is achieved with minimal cost, as *RelayLLM* invokes the LLM for only 1.07% of the total generated tokens. Furthermore, in comparison to a resource-equivalent Random Router, *RelayLLM* yields a substantial 6.9% accuracy improvement. These results confirm that *RelayLLM* effectively identifies critical reasoning steps for LLM intervention, reducing token costs by 98.2% compared to a performance-matched router. Surprisingly, evaluations in an LLM-free setting reveal that the model internalizes effective reasoning patterns during collaboration, enabling it to surpass baselines on easier benchmarks even without LLM assistance.

2 *RelayLLM* Inference

As illustrated in Fig. 2, we consider a hybrid inference setting involving a primary, resource-efficient Small Language Model (SLM), denoted as $\mathcal{M}_S$, and a powerful but computationally expensive Large Language Model (LLM), denoted as $\mathcal{M}_L$. Given an input query $\mathbf{x}$, the system aims to generate a high-quality response $\mathbf{y}$ through dynamic collaboration.

Different from standard cascading methods that simply offload the entire remaining task to a larger model, we define an interleaved generation process where the SLM acts as both reasoner and central controller. The process operates as follows:

2.1 SLM Generation

By default, $\mathcal{M}_S$ generates tokens autoregressively based on the current context history as a normal language model. $\mathcal{M}_S$ is augmented with a special control capability: it can actively request assistance by generating a specific command pattern:

$$\mathcal{C}_{\text{cmd}}(n) = \texttt{<call>} \oplus n \oplus \texttt{</call>}$$

where $\oplus$ denotes string concatenation, and $n \in \mathbb{Z}^+$ represents the number of tokens requested from the LLM.

2.2 LLM Intervention

When this trigger pattern is detected, the generation by $\mathcal{M}_S$ pauses. The current context (including the query and tokens generated by the SLM) is forwarded to $\mathcal{M}_L$. Crucially, to maintain compatibility with the LLM's standard input distribution, we strip the special command tokens ($\mathcal{C}_{\text{cmd}}$) from the context provided to $\mathcal{M}_L$. The LLM then generates the next n tokens (or stops early if an end-of-sequence "[EOS]" token is reached), providing a high-quality continuation or reasoning step.

2.3 Iterative Relay of LLM and SLM

After $\mathcal{M}_L$ completes the generation, control is returned to the SLM. The context is updated with the new tokens generated by $\mathcal{M}_L$ added into the existing context. Unlike the LLM, $\mathcal{M}_S$ retains the full history, including its own generated command tokens (`<call>n</call>`). This allows $\mathcal{M}_S$ to maintain a trace of its active delegation decisions. With the updated context, $\mathcal{M}_S$ resumes generation, digesting the LLM's guidance to continue the reasoning process.

3 *RelayLLM* Training

To enable the SLM $\mathcal{M}_S$ to determine *when* and *how long* to invoke the LLM, we propose a two-stage framework including a supervised warm-up phase followed by a reinforcement learning phase.

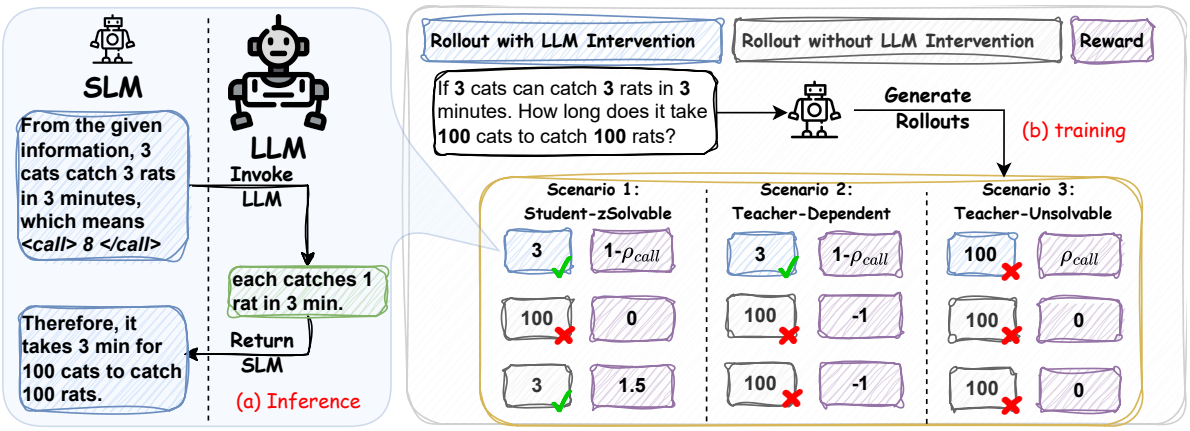

Figure 2: Overview of the *RelayLLM* framework. **(Left)** ***RelayLLM* Inference:** The SLM acts as a central controller. During generation, it can actively trigger an "intervention" by generating a `<call>` command. The LLM then generates the specified number of tokens, after which control returns to the SLM to complete the reasoning. **(Right) Difficulty-Aware Reward Design:** During GRPO training, we sample a group of rollouts (both with and without LLM intervention) and classify the query difficulty into three scenarios: *SLM-Solvable*, *LLM-Dependent*, and *LLM-Unsolvable*, to guide the policy optimization. Distinct reward designs are applied to each scenario to align the SLM's behavior with the optimal strategy.

3.1 Cold Start via Supervised Warm-up

The SLM $\mathcal{M}_S$ may not naturally generate the command pattern $\mathcal{C}_{\text{cmd}}$ if we train it directly with reinforcement learning. We employ a supervised warm-up phase as cold start to initialize the model with the necessary structural knowledge. We construct a synthetic dataset $\mathcal{D}_{\text{warm}}$ to teach the model the calling command without distribution shifts.

To prevent distribution shifts of the SLM $\mathcal{M}_S$, we first generate base sequences $\mathbf{y}$ by sampling directly from vanilla $\mathcal{M}_S$ rather than relying on external corpora. This ensures that the training contexts align perfectly with the SLM's own distribution. Within these self-sampled sequences, we insert command tokens at random indices t at the token level. By avoiding rigid restrictions to sentence or paragraph boundaries, we enable the model to learn to trigger assistance at the precise moment a reasoning gap occurs during the inference time. This data construction strategy ensures effective on-policy training data to prevent data distribution shift of the SLM, while simulating various calling scenarios.

Furthermore, we explicitly simulate varying degrees of reliance on the LLM by synthesizing delegation lengths across multiple orders of magnitude. We randomly sample the target length $n_{\text{sample}} = d \times 10^k$, where $d \in \{1, \ldots, 9\}$ and $k \in \{0, \ldots, 3\}$. To guaranty validity, the final request length is clipped to the available response via $n = \min(n_{\text{sample}}, L_{\text{rem}})$, where L_{rem} is the number of the tokens after the command.

We fine-tune $\mathcal{M}_S$ on this dataset using standard cross-entropy loss. The warm-up only aims to teach $\mathcal{M}_S$ to emit valid commands $\mathcal{C}_{\text{cmd}}$; the subsequent RL stage then optimizes *when* and *for how long* to invoke $\mathcal{M}_L$.

3.2 Policy Refinement with Reinforcement Learning

In this stage, we employ Reinforcement Learning to align the model's behavior with our core objective: maximizing response quality while strictly minimizing collaborative cost. We proceed by detailing our optimization framework.

3.2.1 GRPO Training with RLVR

We leverage Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to refine the policy π_θ of $\mathcal{M}_S$ by adopting the Reinforcement Learning with Verifiable Reward (RLVR) paradigm (Lambert et al., 2024). This paradigm is well-suited for domains where the quality of generated responses can be deterministically verified. In its standard form, RLVR employs a rule-based verifier $v : \mathcal{X} \to \{0, 1\}$ that assigns binary rewards. This binary scheme is effective for tasks with unambiguous success criteria, such as mathematical problem-solving. In our setting, we design a corresponding rule-based reward to verify model-generated relations, described in Sec. 3.2.3.

To optimize the policy using these rewards, GRPO samples a group of outputs $\{o_i\}_{i=1}^{G}$ for each

query q from the old policy $\pi_{\theta_{\text{old}}}$ and evaluates them relative to the group average. The training objective is formulated as:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim\mathcal{D}}\left[\frac{1}{G}\sum_{i=1}^{G}(\mathcal{M}_i - \beta\mathbb{D}_{\text{KL}})\right], \tag{1}$$

where $\mathbb{D}_{\text{KL}} = D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}})$ is the regularization term. The surrogate objective $\mathcal{M}_i$ is computed as $\min(\rho_i A_i, \text{clip}(\rho_i, 1-\epsilon, 1+\epsilon)A_i)$, where $\rho_i = \frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}$. The advantage A_i is derived from the group-normalized rewards defined below:

$$A_i = \frac{r_i - \text{mean}(\{r_j\})}{\text{std}(\{r_j\}) + \varepsilon_{\text{stab}}}, \tag{2}$$

where $\varepsilon_{\text{stab}}$ is a small constant for stability. This formulation encourages the model to generate responses that outperform the group average.

3.2.2 Data Filtering

Since our method leverages the LLM to generate reasoning paths or feedback, it is essential to identify the subset of data where such intervention is useful. If the LLM consistently fails to solve a query, calling it during training yields no positive gain. Therefore, we preprocess the dataset to filter out instances that are too hard for the LLM. We sample 10 responses per query and only preserve those with a pass rate of $\geq 50\%$. This step ensures that the training data lies in the competence boundary of the LLM and the responses can contribute effectively. We provide an ablation study on this filtering mechanism in Table 3.

3.2.3 Reward Design

We formulate the optimization objective using two distinct reward signals, a simple reward and our designed difficulty-aware reward. Let y be the response, a be the final answer parsed from y, g be the ground truth, and $\rho(y) \in [0, 1]$ be the *call ratio* (the ratio of LLM-generated tokens to the total response length).

Simple Reward. We define a straightforward reward to encourage both accuracy and efficiency:

$$r_{\text{simple}}(y) = \mathbb{1}(a = g) - \rho(y). \tag{3}$$

where $\mathbb{1}(\cdot)$ is the indicator function, thus the responses are scored by their correctness and penalized by the cost of calling the LLM.

Difficulty-Aware Reward. To capture the relative difficulty of each query, we define the reward based on the collective performance of the sampled group $\mathcal{G}$. We categorize each query into three scenarios based on its difficulty (correctness of responses in $\mathcal{G}$). As illustrated in Figure 2 (Right), we provide concrete examples of how these rewards are assigned across different categories.

Scenario 1: SLM-Solvable (Encouraging Independence). This scenario applies when the SLM is capable of solving the query independently, without assistance from the LLM. It is identified if there exists at least one sample in the group $\mathcal{G}$ that answers correctly *without* invoking the LLM. In this case, calling the LLM is deemed redundant. Consequently, we assign a boosted bonus ($r = 1.5$) for independent success to promote efficiency, while dependent success ($\rho(y) > 0$) still receives the simple reward r_{simple} in Eq. (3), and incorrect responses receive zero reward.

Scenario 2: LLM-Dependent (Penalizing Stubbornness). This scenario represents challenging queries where correct answers appear *only* in samples that call the LLM. Here, the SLM's independent reasoning is insufficient. To discourage blind guessing, we impose a penalty on samples that fail to call the LLM ($r = -1.0$ when $\rho(y) = 0$). Conversely, effective LLM calling that leads to a correct answer is rewarded with r_{simple} in Eq. (3).

Scenario 3: LLM-Unsolvable (Incentivizing Exploration). This scenario occurs when no sample in $\mathcal{G}$ yields the correct answer, indicating that the query is extremely difficult or the LLM's guidance was ineffective. Rather than providing zero training signal for all responses, we assign a small exploration reward ($r = \rho(y)$) to samples that attempted to call the LLM. This reinforces the tendency to seek help in highly uncertain situations. This piecewise design aligns the policy with an optimal strategy: *solve independently when possible, seek help when necessary, and avoid costly errors.*

4 Experiments

4.1 Experimental Setup

4.1.1 Models

To evaluate the effectiveness of *RelayLLM*, we utilize the Qwen3 model family (Yang et al., 2025a) due to its consistent architectural scaling and strong performance across various sizes. We se-

Model	Minerva	MATH500	GSM8K	Olympiad	AIME25	AIME24	Average	Avg. Call Ratio
Qwen3-0.6B								
Base Model	15.81	54.00	64.82	26.22	1.04	1.15	27.17	–
GRPO	17.65	58.60	65.50	29.04	5.42	3.23	29.91	–
CITER	19.29	58.80	67.78	29.60	5.93	3.24	30.77	0.98%
RelayLLM (Simple)	20.96	**60.20**	69.14	32.15	**7.19**	**3.85**	32.25	0.31%
RelayLLM (Difficulty-Aware)	**23.53**	60.00	**71.95**	**32.74**	6.15	**3.85**	**33.04**	0.77%
Qwen3-1.7B								
Base Model	33.82	74.60	82.64	43.11	8.75	12.08	42.50	–
GRPO	35.66	75.60	81.73	45.04	10.73	15.62	44.06	–
CITER	38.63	80.24	82.26	51.20	11.96	16.58	46.81	1.34%
R2R	39.12	80.46	83.15	50.84	12.14	16.92	47.11	1.42%
RelayLLM (Simple)	43.01	**83.40**	86.13	51.56	**13.44**	**18.23**	49.30	0.43%
RelayLLM (Difficulty-Aware)	**43.75**	81.40	**86.28**	**55.70**	12.71	17.29	**49.52**	1.07%
Qwen3-8B	48.16	83.20	93.63	56.89	17.92	24.90	54.12	100%

Table 1: Performance comparison on six benchmarks. We compare the effectiveness of *RelayLLM* using Qwen3-0.6B and Qwen3-1.7B as SLMs across different methods: the standard Base model, the GRPO-tuning baseline, CITER and *RelayLLM* (Simple-Reward and Difficulty-Aware-Reward). The Qwen3-8B LLM performance is provided for reference. We report avg@32 for the challenging AIME datasets and standard pass@1 (greedy decoding) for all other benchmarks. The "Avg. Call Ratio" denotes the percentage of tokens generated by the LLM during the collaborative inference process. The best results within each model group are highlighted in bold.

lect Qwen3-0.6B and Qwen3-1.7B as our primary SLMs ($\mathcal{M}_S$) to investigate how our framework scales with model capacity at the sub-2B parameter level. For the LLM ($\mathcal{M}_L$), we utilize Qwen3-8B. Selecting models from the same family ensures that the generation style, token distribution, vocabulary, and tokenizer are consistent, making collaboration more stable. To optimize training and inference efficiency, we run the models in **non-thinking** mode throughout. To show that *RelayLLM* is not tied to a single model family, we additionally report a cross-family configuration (Gemma-3-1B SLM with a Qwen3-8B LLM) in App. F.

4.2 Evaluation Setup

To evaluate the effectiveness of *RelayLLM*, we conduct experiments on six reasoning benchmarks, and compare our approach against the standard GRPO baseline. We compare against two recent token-level collaborative decoding baselines: CITER (Zheng et al., 2025b), which trains an additional MLP controller to route every token, and R2R (Fu et al., 2025), a contemporaneous SLM–LLM token-routing method. The benchmarks include Minerva (Lewkowycz et al., 2022), MATH-500 (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), Olympiad-Bench (He et al., 2024), AIME-2024, and AIME-2025. We use GPT-4o-mini as a semantic judge to verify the model's output against the ground truth (Zhao et al., 2025). For the difficult AIME datasets, we report the avg@32 metric to ensure a robust evaluation. For other benchmarks, we report standard accuracy (pass@1) using greedy decoding.

Full training details, including dataset, framework, and the switching mechanism, are deferred to App. L.

4.3 Main Results

As presented in Table 1, *RelayLLM* demonstrates a superior trade-off between reasoning capability and inference efficiency. First, our method achieves substantial performance improvements across all benchmarks while maintaining a negligible collaborative cost (less than 1% token overhead). For instance, on the challenging Minerva benchmark, the Qwen3-0.6B model with Difficulty-Aware-Reward improves from a base score of 15.81% to 23.53%, representing a relative improvement of approximately 48.8% while invoking the LLM for only 0.77% of the total tokens. Compared to CITER, our method demonstrates superior performance despite CITER's more computationally intensive design. CITER relies on an external MLP to estimate a score every token, which introduces substantial latency and computational overhead. In contrast, *RelayLLM* achieves better results with a significantly more efficient mechanism at the cost of only several additional tokens. We further benchmark against R2R (Fu et al., 2025), a recent token-routing baseline: on Qwen3-1.7B, R2R reaches 47.11% average accuracy with a 1.42% call ratio, while *Re-*

layLLM achieves 49.52% at a lower 1.07% ratio, indicating that an integrated controller learned end-to-end is more effective than an external router for sparse LLM calling.

Second, comparing optimization strategies, the Difficulty-Aware-Reward mechanism outperforms the Simple-Reward in performance, with a marginal increase in token consumption. For the Qwen3-1.7B model, the Difficulty-Aware-Reward strategy achieves a higher average accuracy of 49.52% compared to 49.30% for the Simple-Reward, which correlates with a slight increase in the average call ratio from 0.43% to 1.07%, suggesting that the difficulty-based signal better incentivizes the model to seek help in complex scenarios.

Finally, *RelayLLM* effectively bridges the capability gap between SLMs and LLMs using minimal tokens. Remarkably, the Qwen3-1.7B (Difficulty-Aware) recovers approximately 60% of the performance gap between the base SLM (42.50%) and the LLM (54.12%), highlighting that sparse, strategic intervention at critical reasoning steps is sufficient to unlock a significant portion of the LLM's potential. A complementary scaling study with an 8B SLM and a 32B LLM (App. G), as well as a cross-family experiment pairing a Gemma-3-1B SLM with a Qwen-3-8B LLM (App. F), confirm that the same trend holds at larger scales and across model families.

5 Analysis

In this section, we conduct a series of in-depth analyzes to better understand the behavior and effectiveness of *RelayLLM* framework.

5.1 *RelayLLM* Generalizes to Unseen Reasoning Domains

To verify the generalization capability of *RelayLLM*, we extended our evaluation to general reasoning domains that were unseen during training. Although our model was trained exclusively on the mathematical DAPO dataset, we tested it on three diverse benchmarks out of the math domain: Big-Bench Hard (BBEH) (Kazemi et al., 2025), MMLU-Pro (Wang et al., 2024), and SuperGPQA (Du et al., 2025). As shown in Table 2, *RelayLLM* consistently outperforms baseline methods despite the domain shift. For instance, using Qwen3-1.7B, our method achieves 59.03% on MMLU-Pro, significantly surpassing the GRPO baseline (49.76%) and CITER (53.38%). These results demonstrate that our framework effectively helps the SLM acquire a generalized help-seeking behavior; even when facing unfamiliar inputs, the SLM successfully recognizes its knowledge gaps and invokes the LLM, leading to substantial performance gains in out-of-distribution tasks.

Model	BBEH	MMLU-Pro	SuperGPQA
Qwen3-0.6B			
Base Model	7.19	30.03	17.22
GRPO	7.82	32.15	19.91
CITER	8.16	33.12	20.34
RelayLLM (Simple)	8.32	35.61	**21.35**
RelayLLM (Difficulty-Aware)	**8.56**	**35.87**	20.88
Qwen3-1.7B			
Base Model	9.91	46.90	24.46
GRPO	10.89	49.76	26.01
CITER	11.67	53.38	28.25
RelayLLM (Simple)	**12.67**	58.76	29.85
RelayLLM (Difficulty-Aware)	12.46	**59.03**	**29.93**
Qwen3-8B	15.31	66.46	36.21

Table 2: Performance comparison on reasoning and general knowledge benchmarks. We evaluate the effectiveness of *RelayLLM* using Qwen3-0.6B and Qwen3-1.7B as SLMs across different settings. The Qwen3-8B performance is provided for reference. The best results within each model group are highlighted in bold.

5.2 Ablation Study

To investigate the distinct contribution of each component in *RelayLLM* , we conducted an ablation study using the Qwen3-1.7B model in Table 3.

Data filtering prevents wasteful calls where the LLM fails. We show that removing the data filtering mechanism results in a tripled call ratio with decreased accuracy; this confirms that filtering out queries intractable for the LLM is crucial to avoid cost that yields no performance gain. Filtering out the hardest data also saves time and resources during the training stage. The filter retains $\sim$37% of training examples ($\sim$17k $\rightarrow$ $\sim$6.35k). To rule out the filter itself as the source of improvement, we re-train the GRPO baseline on the same filtered subset and find a negligible difference vs. full-data GRPO (App. J), confirming that the gains stem from collaborative decoding, not data curation.

Encouraging independence reduces reliance on the LLM and improves efficiency. We remove the independence incentive (the boosted correctness reward from 1 to 1.5 for solvable queries), and this causes the call ratio to spike to 4.10%. This demonstrates that specifically rewarding independent success is crucial to prevent the SLM

Method	Avg. Acc. (%)	Call Ratio (%)
RelayLLM	**49.52**	1.07
w/o Data Filtering	48.76	3.30
w/o Indep. Incentive	49.34	4.10
w/o Explor. Reward	47.56	0.65

Table 3: Ablation study on data filtering and reward design strategies using Qwen3-1.7B. "w/o Data Filtering" denotes training on the unfiltered dataset including LLM-failed queries. "w/o Indep. Incentive" removes the bonus reward (from 1.5 to 1) for independent success (Scenario 1). "w/o Explor. Reward" removes the exploration reward (from ρ to 0) for seeking help in unsolvable queries (Scenario 3).

Method	Easy (%)	Hard (%)
GRPO Baseline	59.51	13.18
RelayLLM (Simple)		
Standard Inference	66.03	15.84
w/o LLM	61.12	13.13
RelayLLM (Difficulty-Aware)		
Standard Inference	66.78	15.00
w/o LLM	60.26	11.93

Table 4: Evaluation of intrinsic reasoning capability. We disable the LLM during inference ("w/o LLM") by masking the call tokens. "Hard" refers to AIME24 and AIME25, while "Easy" refers to the remaining four benchmarks.

Method	Avg. Acc. (%)	Call Ratio (%)
Fixed-20	49.41	1.32
Fixed-100	49.56	2.87
Fixed-500	**51.17**	5.37
RelayLLM	49.52	**1.07**

Table 5: Comparison between dynamic length prediction (*RelayLLM*) and fixed delegation length strategies ("Fixed-k": models retrained with a fixed call length of k).

from becoming over-reliant on the LLM for tasks it could solve alone.

Exploration reward effectively increases accuracy. When we remove the exploration reward (for unsolvable queries), this leads to a significant accuracy drop to 47.56%, indicating that the exploration reward is necessary to encourage the SLM to ask for help from the LLM in highly uncertain scenarios. The accuracy gap between Simple and Difficulty-Aware rewards at convergence is small but the difficulty-aware design accelerates learning on hard queries during early training; we report training-step-level dynamics on AIME24 in App. I. We also study the sensitivity of *RelayLLM* to the cost-penalty weight in r_{simple} (App. H): setting the penalty to zero collapses the policy to always-call (100% call ratio), whereas our default value of 1.0 retains extreme sparsity.

5.3 Intrinsic Reasoning Capability

To investigate whether *RelayLLM* improves the SLM's intrinsic reasoning or merely learns to offload tasks, we evaluate the models in an "LLM-Free" setting (implemented via `bad_words=["<call>", "</call>"]` during inference). Note that the gradient is *not* propagated through LLM-generated tokens during GRPO training (they are masked from the loss), so this setting does not reduce to distillation. Table 4 shows that on Easy datasets, the LLM-Free *RelayLLM* (Simple) still reaches 61.12%, surpassing the GRPO baseline—indicating internalized reasoning gains—while on Hard datasets the performance drops sharply (e.g., $15.00\% \rightarrow 11.93\%$ for Difficulty-Aware), confirming continued LLM dependence on complex queries. The Simple-Reward variant retains more independent capability than the Difficulty-Aware-Reward variant, consistent with its lower call ratio.

5.4 Dynamic Token-Length Calling Minimizes Computational Cost

We investigate whether dynamically predicting the calling length n requested from the LLM is superior to using rigid, pre-defined lengths. To ensure a fair evaluation, we **retrained** separate variations of the SLM where the call command is hard-constrained to a fixed token count $k \in \{20, 100, 500\}$ during both training and inference. As shown in Table 5, *RelayLLM* demonstrates superior efficiency compared to these specialized fixed-length models. Specifically, compared to the **Fixed-100** model, *RelayLLM* achieves a similar accuracy but reduces the call ratio from 2.87% to 1.07%. This indicates that while the fixed-length model is forced to consume a set budget even for simple queries, *RelayLLM* effectively learns to request "just enough" tokens to bridge the reasoning gap, thereby minimizing computational waste without compromising performance. We provide a more detailed results in Appendix B. We further analyze the empirical distribution of n across benchmarks in App. D, and verify that n positively correlates with the amount of useful reasoning pro-

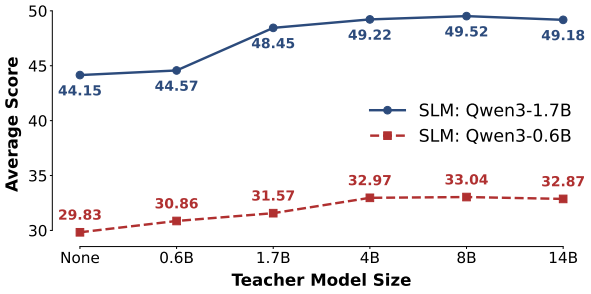

Figure 3: Impact of LLM size on SLM performance. We evaluate two SLMs (Difficulty-Aware) across six benchmarks. The x-axis represents the size of the LLM used during the collaborative inference process, ranging from "None" (as in Sec. 5.3, where the SLM is prevented from calling) to 14B. Scores are averaged across all six datasets.

duced by $\mathcal{M}_L$ in App. E.

5.5 Distributional Alignment

To determine whether the SLM has acquired generalized reasoning capabilities or merely overfitted to the specific patterns and tokens of the LLM seen at training time, we perform a cross-LLM evaluation by substituting the training-time LLM with different models in the inference phase. The results, illustrated in Figure 3, reveal two critical insights. First, consistency between training and inference LLMs yields optimal performance. The accuracy peaks when the inference-time LLM matches the one used at training, reaching 49.52% for the 1.7B SLM. Notably, replacing it with a larger model results in a slight performance decline. This indicates that the distribution shift between training-time and inference-time LLMs can outweigh the benefits of the larger model's superior reasoning capabilities.

Second, even employing a relatively weak LLM that is smaller than the SLM itself (e.g., 0.6B or 1.7B) consistently outperforms the "None" baseline. This suggests that the trained SLM has become accustomed to the presence of external assistance, adapting its generation dynamics to effectively leverage such interventions rather than relying solely on its intrinsic capabilities. Furthermore, excluding the distribution shift at the 8B training-time LLM, there is a positive correlation between LLM size and SLM performance, confirming that the SLM effectively leverages the stronger reasoning signals provided by more capable LLMs.

6 Related Work

6.1 Model Collaboration

Model collaboration (Feng et al., 2025a) spans weight-level merging (Wortsman et al., 2022; Huang et al., 2023), logits-level ensembling (Liu et al., 2024; Li et al., 2023), and text-level interaction. For navigating the SLM–LLM efficiency trade-off, prior work follows two directions: *speculative reasoning*, which uses judges or verifiers as dynamic routers to validate SLM outputs or to inject LLM guidance at inference time (Bachmann et al., 2025; Pan et al., 2025; Yang et al., 2025b; Zhang et al., 2025); and *collaborative decoding* (Shen et al., 2024), where interleaved generation is governed by a separate controller or joint policy (Shen et al., 2024; Sun et al., 2024; Li et al., 2025a; Feng et al., 2025b; Fu et al., 2025). *RelayLLM* differs by internalizing the controller within the SLM itself, eliminating any external routing module.

6.2 RL for LLM Reasoning

Reinforcement learning has emerged as a key technique for LLM reasoning, with success across math and code (Guo et al., 2025; Wang et al., 2025a), multi-modal (Huang et al., 2025b; Wang et al., 2025b; Li et al., 2025b), and structured environments (Shi et al., 2025; Tang et al., 2025). Concurrent work refines RL via new paradigms such as self-play (Liu et al., 2025; Huang et al., 2025a; Yu et al., 2025b) and improved algorithms like DAPO (Yu et al., 2025a), VAPO (Yue et al., 2025), and high-entropy guided optimization (Dai et al., 2025; Wang et al., 2025c; Zhou et al., 2025), on which *RelayLLM* builds.

7 Conclusion

We presented *RelayLLM*, addressing the inefficiency of "all-or-nothing" offloading in routing systems. By treating the LLM as an on-demand tool rather than a fallback generator, our approach demonstrates that the SLM can handle the vast majority of reasoning steps if supported at specific critical positions. The success of our GRPO-based training strategy confirms that help-seeking behaviors can be effectively learned and optimized. Our results show that *RelayLLM* not only outperforms resource-equivalent random routers by 6.9% but also achieves comparable reasoning accuracy to the LLM with negligible computation.

Limitations

First, our default configuration pairs an SLM with an LLM from the same model family to maintain a shared vocabulary and consistent chat templates. While our cross-family experiment (App. F) shows that *RelayLLM* still transfers via text-level re-tokenization, direct collaboration between different architectures suffers from a measurable performance gap without additional alignment mechanisms. Second, restricted by computational resources, we use an 8B-parameter LLM during training rather than scaling to the strongest available models (e.g., 70B+ or closed-source models); our larger-scale study in App. G only extends to a 32B LLM. The potential of leveraging state-of-the-art reasoning engines inside the training loop remains an open question that we aim to explore in future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, et al. 2025. Judge decoding: Faster speculative sampling requires going beyond model alignment. *arXiv preprint arXiv:2501.19309*.

Karl Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, et al. 2021. Training verifiers to solve math word problems. *ArXiv preprint*, abs/2110.14168.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Runpeng Dai, Linfeng Song, Haolin Liu, Zhenwen Liang, Dian Yu, et al. 2025. Cde: Curiosity-driven exploration for efficient reinforcement learning in large language models. *arXiv preprint arXiv:2509.09675*.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, et al. 2024. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*.

Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, et al. 2025. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*.

Shangbin Feng, Wenxuan Ding, Alisa Liu, Zifeng Wang, Weijia Shi, et al. 2025a. When one llm drools, multi-llm collaboration rules. *arXiv preprint arXiv:2502.04506*.

Shangbin Feng, Wenhao Yu, Yike Wang, Hongming Zhang, Yulia Tsvetkov, et al. 2025b. Don't throw away your pretrained model. *arXiv preprint arXiv:2510.09913*.

Tianyu Fu, Yi Ge, Yichen You, Enshu Liu, Zhihang Yuan, et al. 2025. R2r: Efficiently navigating divergent reasoning paths with small-large model token routing. *arXiv preprint arXiv:2505.21600*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Sherzod Hakimov, Roland Bernard, Tim Leiber, Karl Osswald, Kristina Richert, et al. 2025. The price of thought: A multilingual analysis of reasoning, performance, and cost of negotiation in large language models. *arXiv preprint arXiv:2510.08098*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, et al. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Annual Meeting of the Association for Computational Linguistics*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *Proc. of ICLR*.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, et al. 2024. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, et al. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, et al. 2025a. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, et al. 2025b. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, et al. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, et al. 2025. Big-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26473–26501.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, et al. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Nathan Lambert, Jacob Daniel Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, et al. 2024. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, et al. 2023. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 12286–12312.

Ziheng Li, Hengyi Cai, Xiaochi Wei, Yuchen Li, Shuaiqiang Wang, et al. 2025a. Efficient thought space exploration through strategic intervention. *arXiv preprint arXiv:2511.10038*.

Zongxia Li, Wenhao Yu, Chengsong Huang, Rui Liu, Zhenwen Liang, et al. 2025b. Self-rewarding vision-language model via reasoning decomposition. *arXiv preprint arXiv:2508.19652*.

Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Lin, Xing Wang, et al. 2024. Critical tokens matter: Token-level contrastive estimation enhances llm's reasoning capability. *arXiv preprint arXiv:2411.19943*.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, et al. 2024. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*.

Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, et al. 2025. Spiral: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. *arXiv preprint arXiv:2506.24119*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, et al. 2024. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*.

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, et al. 2025. Specreason: Fast and accurate inference-time compute via speculative reasoning. *arXiv preprint arXiv:2504.07891*.

Zhiwen Ruan, Yixia Li, He Zhu, Yun Chen, Peng Li, et al. 2025. Enhancing large language model reasoning via selective critical token fine-tuning. *arXiv preprint arXiv:2510.10974*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. 2024. Learning to decode collaboratively with multiple language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12974–12990.

Yucheng Shi, Wenhao Yu, Zaitang Li, Yonglin Wang, Hongming Zhang, et al. 2025. Mobilegui-rl: Advancing mobile gui agent through reinforcement learning in online environment. *arXiv preprint arXiv:2507.05720*.

Hao Sun, Jiayi Wu, Hengyi Cai, Xiaochi Wei, Yue Feng, et al. 2024. Adaswitch: Adaptive switching between small and large agents for effective cloud-local collaborative learning. *arXiv preprint arXiv:2410.13181*.

Yinxu Tang, Chengsong Huang, Jiaxin Huang, and William Yeoh. 2025. Unirel-r1: Rl-tuned llm reasoning for knowledge graph relational question answering. *Preprint*, arXiv:2512.17043.

Haozhe Wang, Long Li, Chao Qu, Fengming Zhu, Weidi Xu, et al. 2025a. To code or not to code? adaptive tool integration for math language models via expectation-maximization. *arXiv preprint arXiv:2502.00691*.

Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, et al. 2025b. Vl-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning. *arXiv preprint arXiv:2504.08837*.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, et al. 2025c. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, et al. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelovic, and Jakob Nikolas Kather. 2025. Llm agents making agent tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26092–26130.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Wang Yang, Xiang Yue, Vipin Chaudhary, and Xiaotian Han. 2025b. Speculative thinking: Enhancing small-model reasoning with large model guidance at inference time. *arXiv preprint arXiv:2504.12329*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, et al. 2025a. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Wenhao Yu, Zhenwen Liang, Chengsong Huang, Kishan Panaganti, Tianqing Fang, et al. 2025b. Guided self-evolving llms with minimal human supervision. *arXiv preprint arXiv:2512.02472*.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, et al. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*.

Haozhen Zhang, Tao Feng, and Jiaxuan You. 2025. Router-r1: Teaching llms multi-round routing and aggregation via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yulai Zhao, Haolin Liu, Dian Yu, S. Y. Kung, Haitao Mi, et al. 2025. One token to fool llm-as-a-judge. volume abs/2507.08794.

Tong Zheng, Hongming Zhang, Wenhao Yu, Xiaoyang Wang, Xinyu Yang, et al. 2025a. Parallel-r1: Towards parallel thinking via reinforcement learning. *arXiv preprint arXiv:2509.07980*.

Wenhao Zheng, Yixiao Chen, Weitong Zhang, Souvik Kundu, Yun Li, et al. 2025b. Citer: Collaborative inference for efficient large language model decoding with token-level routing. *arXiv preprint arXiv:2502.01976*.

Yaowei Zheng, Shenzhi Wang Junting Lu, Zhangchi Feng, and Yuwen Xiong Dongdong Kuang. 2025c. Easyr1: An efficient, scalable, multi-modality rl training framework. https://github.com/hiyouga/EasyR1.

Yujun Zhou, Zhenwen Liang, Haolin Liu, Wenhao Yu, Kishan Panaganti, et al. 2025. Evolving language models without labels: Majority drives selection, novelty promotes variation. *arXiv preprint arXiv:2509.15194*.

A Case Study

Here we show two case studies from Qwen3-1.7B (Difficulty-Aware) to illustrate how invoking LLM can help solve the questions in Figure 4.

B Detailed Analysis of Delegation Length Strategies

In the main text, we demonstrated that RelayLLM achieves comparable average accuracy to a fixed 100-token delegation strategy while consuming significantly fewer tokens. Table 6 presents the granular performance breakdown across all six mathematical benchmarks.

We observe distinct trends across datasets of varying difficulty:

Efficiency on Standard Benchmarks. On standard reasoning datasets such as MATH500 and GSM8K, RelayLLM matches the performance of the Fixed-100 strategy almost exactly (e.g., 81.40% on MATH500 for both methods). However, it achieves this parity with a drastically lower call ratio (1.07% vs. 2.87%). This indicates that for the majority of queries in these datasets, the SLM only requires short, precise interventions rather than long-context guidance.

Adaptability on Complex Benchmarks. The advantage of dynamic prediction becomes more evident on the challenging Minerva benchmark. *RelayLLM* outperforms the Fixed-20 significantly (43.75% vs. 39.71%), suggesting that our model correctly identifies the need for longer generation lengths when facing harder problems. While the Fixed-500 strategy yields the highest accuracy on Minerva (44.49%) and AIME, this marginal gain comes at a prohibitive cost: it incurs a 5.37% call ratio, which is over $5\times$ the computational overhead of RelayLLM.

C Prompt Templates

This section presents the exact prompt templates used for the models.

C.1 Inference Prompt

Solver Prompt Template

System Message:
Please reason step by step, and put your final answer within \boxed{}.
User Message:
{problem_statement}

Note: `{problem_statement}` *is a placeholder for the actual math problem.*

C.2 GPT-4o-mini Judge Prompt

To programmatically evaluate the correctness of answers on mathematical benchmarks where the final answer can be complex (e.g., simplified expressions), we use GPT-4o-mini as a judge. The exact prompt and configuration used for this evaluation are detailed below.

Configuration for GPT-4o as Judge

- **Model**: `gpt-4o`
- **Temperature**: 0.1

System Message:

You are a math answer checker.

User Message Template:

Hi, there is an answer: `{answer}`,

and the ground truth answer is: `{response}`,

please check whether the answer is correct or not, and return the **only** Yes or No.

Note: `{answer}` *is a placeholder for the model-generated solution, and* `{response}` *is the ground-truth answer from the benchmark.*

D Call-Length Distribution Across Benchmarks

To investigate whether the SLM's request length n in the command pattern $\mathcal{C}_{\text{cmd}}(n)$ is physically meaningful (i.e., whether it reflects the difficulty of the underlying query), we analyze the empirical distribution of generated call lengths across all six benchmarks. We bucket calls into three ranges: *short* (1–10 tokens), *medium* (11–100 tokens), and

Case Study: A list of positive integers has the following properties:
• The sum of the items in the list is 30.
• The unique mode of the list is 9.
• The median of the list is a positive integer that does not appear in the list itself.Find the sum of the squares of all the items in the list.

We are given a list of **positive integers** with the following properties:
1. **Sum of the items is 30.**
2. **The unique mode is 9.**
3. **The median is a positive integer that does not appear in the list.**
We are to find the **sum of the squares of all the items** in the list.
—
Step 1: Understand the properties - **Mode is 9**: This means 9 appears more times than any other number in the list. - **Unique mode is 9**: No other number appears as frequently as 9.
- **Median is a positive integer that does not appear in the list**: So the median is some number m, and m is not in the list.
- **Sum of the items is 30**.
—
Step 2: Consider the structure of the list Let's denote the list as $a_1, a_2, \ldots, a_n$, sorted in non-decreasing order. #### Key observations: - The **median** is the middle value. If n is odd, it is the $\frac{n+1}{2}$-th element. If n is even, it is the average of the two middle elements. - Since the **median is a positive integer that does not appear in the list**, it must be **greater than the largest number in the list** (if the list has only one element, it's trivial, but we'll assume $n > 1$). - The **mode is 9**, so 9 appears more times than any other number.
—
Step

```
<call> 300 </call>
3: Try small values of $ n $
Let's test small values of $ n $ to find a valid list that satisfies all
    conditions.
#### Try $ n = 5 $ (odd):
- Then the **median is the 3rd element**.
- Suppose the list is $ [a, b, m, c, d] $, where $ m $ is the median.
- $ m $ is not in the list.
- The **mode is 9**, so 9 must appear at least twice, and more than any other
    number.
Let's try to construct such a list:
Try $ [9, 9, 1, 1, 10] $ (sum = 9+9+1+1+10 = 30)
Check the properties:
- **Sum = 30 **
- **Mode = 9** (appears twice) - but are there other numbers with same
    frequency?
- 1 appears 3 times -> **mode is 1**, not 9 correct
Try $ [9, 9, 1, 2, 9]$ -> sum = 9+9+1+2+9=30
- Mode is 9 (3 times), 1 and 2 once -> correct
- Median is 1 (the middle number) -> 1 is in the list incorrect
Try $ [9, 9

(end call)
```

, 1, 2, 8] $ -> sum = 9+9+1+2+8 = 29 incorrect Try $[9, 9, 1, 3, 8]$ -> sum = 30 - Mode is 9 - Median is 1 (not in the list) - 1 is not in the list So this is a valid list:

$$[9, 9, 1, 3, 8]$$

Check sum: $9 + 9 + 1 + 3 + 8 = 30$ Check mode: 9 appears twice, others once -> **unique mode is 9** Check median: middle element is 3 $\rightarrow$ 3 is not in the list
—
Step 4: Compute the sum of squares Compute $9^2 + 9^2 + 1^2 + 3^2 + 8^2 = 81 + 81 + 1 + 9 + 64 = 236$
—
$\boxed{236}$

Figure 4: A qualitative case study showcases that our model invoke LLM to show how to validate the final answer, then solve the problem by itself.

Table 6: Full performance breakdown of different delegation length strategies across six benchmarks. "Fixed-k" indicates models retrained with a fixed call length of k tokens. We report mean@32 for AIME datasets (AIME24, AIME25) and pass@1 for others. Best accuracy values are bolded, and the most efficient call ratio is highlighted.

Method	Minerva	MATH500	GSM8K	Olympiad	AIME25	AIME24	Average	Call Ratio
Fixed-20	39.71	81.40	86.96	55.74	14.11	18.54	49.41	1.32%
Fixed-100	40.44	81.40	86.50	56.78	14.42	17.81	49.56	2.87%
Fixed-500	**44.49**	**81.80**	**88.25**	**57.96**	**15.05**	**19.48**	**51.17**	5.37%
RelayLLM	43.75	81.40	86.28	55.70	12.71	17.29	49.52	**1.07%**

long ($>$ 100 tokens). Table 7 summarizes the results.

Benchmark	Diff.	Short	Medium	Long
GSM8K	Std.	88%	11%	$<1\%$
MATH500	Std.	75%	22%	3%
Minerva	Hard	45%	42%	13%
Olympiad	Hard	40%	45%	15%
AIME 24	Extr.	35%	46%	19%
AIME 25	Extr.	32%	48%	20%

Table 7: Empirical distribution of call lengths n produced by *RelayLLM* (Qwen3-1.7B, Difficulty-Aware) across benchmarks of varying difficulty.

We observe a distinct, difficulty-dependent shift in delegation behavior. On standard reasoning tasks (GSM8K, MATH500), the SLM predominantly issues short calls, using the LLM for precise micro-corrections or arithmetic verification. On more complex benchmarks (Minerva, Olympiad, AIME), the distribution flattens substantially, with a non-trivial fraction of calls requesting long-context interventions ($>$ 100 tokens). This confirms that *RelayLLM* does not learn a static pattern but instead acts as an adaptive controller, conserving tokens on easy queries and correctly identifying reasoning bottlenecks that require extended LLM guidance.

E Call-Length and Reasoning Quality Correlation

To verify that the requested length n positively correlates with the actual amount of useful reasoning produced by $\mathcal{M}_L$, we conduct a controlled study on AIME25 using Qwen3-0.6B as the SLM. We fix the LLM to generate the first n tokens of each response and let the SLM complete the remainder, sweeping n to obtain a range of effective call-token ratios.

As shown in Table 8, accuracy increases monotonically as the LLM produces more tokens, demonstrating that LLM-provided tokens contain reasoning information that directly improves final-answer quality. The positive correlation confirms

Call Token Ratio (%)	Avg. Score
0.00 (Base Model)	1.23
0.31 (*RelayLLM*)	**7.19**
20.00	3.42
50.00	6.85
80.00	15.93
100.00 (Full LLM)	**17.92**

Table 8: Effect of call-token ratio on accuracy (AIME25, Qwen3-0.6B SLM, Qwen3-8B LLM).

that n has a clear physical meaning and is a reliable proxy for the depth of LLM intervention. Notably, *RelayLLM* with a learned, sparse 0.31% call ratio already substantially outperforms naively pre-fixing 20% of the response to the LLM, suggesting that *where* the calls happen is as important as *how many* tokens are delegated.

F Cross-Family Generalization

A natural concern is whether *RelayLLM* relies on a shared vocabulary between $\mathcal{M}_S$ and $\mathcal{M}_L$. While same-family pairings minimize tokenizer friction (a standard practice in efficient inference, e.g., speculative decoding), *RelayLLM* interacts at the *text* level rather than the logits level, making cross-family collaboration feasible via re-tokenization. We empirically validate this with a heterogeneous pair: Gemma-3-1B as the SLM and Qwen-3-8B as the LLM.

Method	Min.	MATH	GSM	Oly.	AIME25	AIME24	Avg.
Base Gemma-3-1B	17.21	48.04	62.83	27.46	1.48	1.82	26.47
+ *RelayLLM*	**21.02**	**50.87**	**65.64**	**30.78**	**4.06**	**3.24**	**29.27**

Table 9: Cross-family results with Gemma-3-1B SLM and Qwen-3-8B LLM (call ratio 1.14%).

RelayLLM consistently improves the Gemma SLM by invoking the Qwen LLM for only 1.14% of tokens, with notable gains on challenging tasks (e.g., +3.81 on Minerva, +2.58 on AIME25). Gains are smaller than in the same-family setting, which we attribute to text-level distribution mismatch, but

the result confirms that the SLM learns a *universal* help-seeking policy that transfers across architectures.

G Scaling to Larger LLMs

To assess how *RelayLLM* scales with model size, we train an 8B SLM (Qwen3-8B) paired with a 32B LLM (Qwen3-32B). Training with closed-source LLMs such as GPT-4 or Gemini is currently infeasible because the GRPO loop requires real-time LLM rollouts during training (latency and API cost prohibitive); we therefore focus on the open-weight scaling axis.

Setting	Min.	MATH	GSM	Oly.	AIME25	AIME24	Avg.
Qwen3-8B (base)	48.16	83.20	93.63	56.89	17.92	24.90	54.11
Qwen3-32B (full)	55.40	87.40	95.80	64.50	29.10	20.90	58.85
RelayLLM (8B+32B)	**52.30**	85.80	94.90	61.20	**24.50**	23.10	**56.96**

Table 10: Scaling results with Qwen3-8B SLM and Qwen3-32B LLM (call ratio 1.15%). *RelayLLM* recovers a substantial portion of the gap between the 8B base and the 32B full model.

The trend mirrors our sub-2B results: *RelayLLM* closes a meaningful portion of the gap between the SLM and the full LLM, while invoking the LLM for only $\sim$1% of tokens. This indicates that the learned help-seeking behavior is not specific to sub-2B SLMs and generalizes upward in scale.

H Cost Penalty Sensitivity

The simple reward $r_{\text{simple}}(y) = \mathbb{1}(a = g) - \rho(y)$ in Eq. (3) uses an implicit cost-penalty weight of 1.0 on $\rho(y)$. We study the effect of varying this weight on Qwen3-1.7B.

Penalty Weight	Avg. Acc. (%)	Call Ratio (%)
1.0 (default)	49.52	1.07
0.5	51.37	3.43
0.0	54.12	100.00

Table 11: Sensitivity to cost-penalty weight.

Reducing the penalty to 0.5 raises accuracy by $\sim$2 points at the cost of tripling the call ratio. Setting the penalty to 0 collapses the policy: the SLM delegates all generation to the LLM, recovering full-LLM accuracy (54.12%) but eliminating any efficiency benefit. Our default weight of 1.0 was chosen to prioritize extreme token efficiency while still achieving significant reasoning gains; practitioners can tune this weight to navigate the accuracy–cost trade-off based on deployment constraints.

I Difficulty-Aware Reward Training Dynamics

The accuracy gap between the Simple and Difficulty-Aware reward variants at convergence is small (49.30% vs. 49.52% on Qwen3-1.7B), which might suggest the difficulty-aware design is unnecessary. However, the gap during *early training* on the hardest benchmark (AIME24) is substantially larger, indicating that the difficulty-aware design accelerates the acquisition of help-seeking behavior on hard queries:

Training Step	Difficulty-Aware	Simple
10	11.4	10.4
20	12.5	10.8
30	13.9	12.9
40	14.5	13.9

Table 12: Early-training accuracy on AIME24 (Qwen3-1.7B) for Simple vs. Difficulty-Aware reward.

As training progresses, queries that initially fall into S2/S3 (LLM-Dependent / LLM-Unsolvable) gradually shift into S1 (SLM-Solvable) as the SLM's own capability improves; once most queries are in S1, the difficulty-aware reward reduces to the simple reward. Thus the difficulty-aware design is most beneficial at the beginning of training, where the explicit penalty on stubbornness (S2) and exploration bonus (S3) accelerate the discovery of when to seek help.

J Data Filtering Statistics

The LLM-success filter described in Sec. 3.2.2 reduces the training set from $\sim$17k examples to $\sim$6.35k examples ($\sim$37% retained). To isolate the contribution of this filter from *RelayLLM* itself, we re-train the GRPO baseline on the same filtered subset:

Setting	Qwen3-1.7B	Qwen3-0.6B
GRPO (full data)	44.06	29.91
GRPO (filtered data)	44.15	29.83

Table 13: GRPO baseline trained on the full vs. filtered dataset (average over six benchmarks).

The difference is negligible (< 0.1 point), confirming that the filtering itself yields no measurable benefit for the GRPO baseline—hard queries on

which the LLM fails simply do not provide learning signal under standard RLVR. Consequently, the gains of *RelayLLM* over GRPO are attributable to the collaborative decoding mechanism rather than to data curation.

K Native Function Calling

An alternative to introducing a custom `<call>` command is to leverage the native function-calling API of recent instruction-tuned models (e.g., Qwen3's tool-use interface). We chose a custom command for three reasons. First, native function calling is trained to invoke external tools with structured JSON arguments, which is unnecessarily verbose for our setting (a single integer n). Second, the function-calling format introduces additional control tokens that the SLM must emit reliably, increasing the syntactic burden during cold-start. Third, our minimal command pattern integrates cleanly with vLLM's stop-sequence interface, allowing inference without modifying the underlying engine. We expect native function calling to be a viable alternative for deployments that already rely on it, and leave a direct comparison to future work.

L Training Details

We conduct our experiments using the DAPO dataset (Yu et al., 2025a). Our implementation is built upon the EasyR1 framework (Zheng et al., 2025c) using its default hyperparameter configurations (shown in App. M). All models are trained for a single epoch to ensure a fair comparison. Regarding data usage, the GRPO baseline is trained on the full dataset, whereas our method utilizes the filtered subset as described in Section 3.2.2. To enable efficient interaction with the LLM, we serve the LLM via the vLLM inference engine (Kwon et al., 2023). We implement the switching mechanism as a stop sequence in the sampling parameters: when the SLM generates the calling command token, generation halts, and the system invokes the LLM via the API.

M Hyperparameter

We utilize the Group Relative Policy Optimization (GRPO) algorithm for post-training. The model is optimized using `AdamW` with a constant learning rate of 1×10^{-6} and a weight decay of 1×10^{-2}. The global batch size is set to 32.

To support extensive reasoning steps required for complex mathematical problems, we configure the maximum sequence length to allow for long Chain-of-Thought (CoT) generation. Specifically, the maximum prompt length is set to 4096 tokens, and the maximum response length is extended to 8192 tokens.

For the GRPO specific configurations, we set the group size $G = 8$, meaning that 8 outputs are sampled for each prompt to estimate the baseline. To ensure training stability and prevent the policy from deviating excessively from the reference model, we apply a KL divergence coefficient β of 0.01. During the rollout phase, we use a sampling temperature of 1.0 to encourage diverse reasoning paths.

Table 14: Hyperparameter settings for GRPO training.

Hyperparameter	**Value**
Optimizer	AdamW
Learning Rate	1×10^{-6}
Weight Decay	1×10^{-2}
Global Batch Size	32
Max Prompt Length	4096
Max Response Length	8192
Temperature	1.0
Group Size (G)	8
KL Coefficient (β)	0.01