ACCEPT, REPAIR, OR RESTART: STEP-LEVEL RISK-AWARE SLM–LLM COLLABORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM)–small language model (SLM) collaboration offers an effective way to balance reasoning quality and computational efficiency by selectively invoking stronger models. However, existing methods typically determine collaboration through response-level confidence or fixed escalation rules, overlooking step-level failures in reasoning trajectories and the need for adaptive interventions tailored to different failure patterns. To address these problems, we propose step-level Risk-Aware Dynamic Action Allocation (RADA), a framework that formulates SLM–LLM collaboration as a risk-aware multi-action decision problem. RADA jointly models global response reliability and step-level reasoning risk to determine when LLM intervention is needed, adapts the intervention criterion using calibrated quality estimates of previously selected actions, and selects the intervention scope between local revision and full regeneration based on the identified failure pattern. Across mathematical and general reasoning benchmarks and multiple SLM–LLM pairs, RADA achieves the highest average accuracy among the evaluated collaboration methods. It improves average mathematical reasoning accuracy by up to 4.33 percentage points over the best-performing baseline while maintaining a favorable accuracy–computation trade-off.

1 INTRODUCTION

Large language models (LLMs) achieve strong performance on complex reasoning tasks, particularly when combined with chain-of-thought prompting and increased test-time computation (Wei et al., 2022; Snell et al., 2025). However, longer reasoning trajectories also increase token consumption, latency, and inference cost. Small language models (SLMs) offer a cheaper alternative, and recent work (Zhu et al., 2024; Xu et al., 2025) has improved their reasoning ability through distillation and specialized training, yet a clear performance gap remains on difficult problems (Li et al., 2025). The key challenge is therefore not simply to minimize inference cost, but to determine when additional LLM computation is worthwhile and how it should be allocated to improve reasoning accuracy without excessive overhead.

Recent studies on LLM–SLM collaboration (Chen et al., 2023; Ding et al., 2024; Ong et al., 2025; Ding et al., 2025; Barrak et al., 2025; Zhang et al., 2026) allocate computation by routing queries between models with different capabilities. However, their decisions are largely based on query-level or response-level signals, such as difficulty, confidence, or estimated quality. Such coarse-grained assessment is insufficient for multi-step reasoning, where errors may occur in only a few intermediate steps while the rest of the reasoning remains reliable. Process-level supervision (Lightman et al., 2024) has demonstrated the value of evaluating intermediate reasoning steps, while recent fine-grained collaboration methods further use local difficulty or confidence to offload selected reasoning segments or steps to stronger models (Akhauri et al., 2025; Lee et al., 2026). Nevertheless, these methods primarily use step-level signals as local routing criteria, without jointly assessing overall response reliability and identifying where reasoning failures occur within the trajectory.

Beyond locating unreliable reasoning, an effective collaboration framework must also determine whether additional LLM computation is necessary and how such computation should be allocated. Some SLM–LLM collaboration methods formulate escalation as a binary decision: whether to invoke a stronger model or retain the SLM output. However, different queries may exhibit diverse levels of uncertainty and require different amounts of reasoning resources, making fixed escala-

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

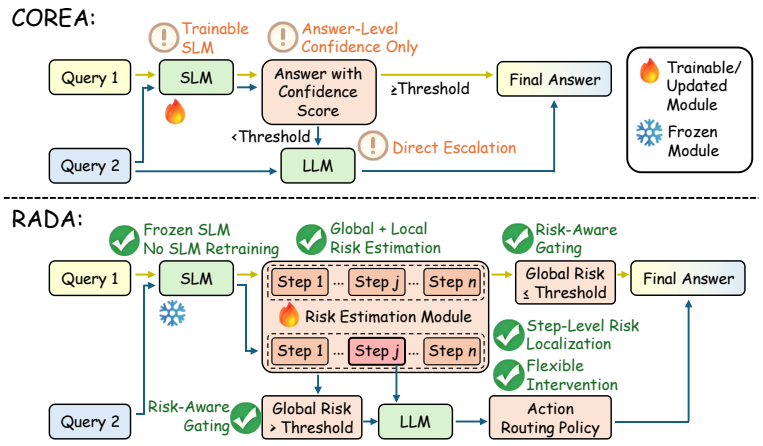

Figure 1: Comparison between COREA (Zhang et al., 2026) and our RADA. (a) COREA uses a static threshold over response-level confidence from a fine-tuned SLM to decide whether to retain the SLM output or escalate to the LLM. (b) RADA jointly models global and step-level reasoning risks to localize unreliable reasoning steps, adaptively determine whether LLM intervention is needed, and select the appropriate intervention scope.

tion strategies insufficient for adaptive computation allocation. Some confidence-based routing and cascade methods (Ding et al., 2025; Chuang et al., 2024; Shah & Shridhar, 2025) rely on predefined confidence or quality thresholds, treating the escalation criterion as fixed across different queries and model states. However, the reliability of SLM outputs can vary substantially, and confidence itself may not always correlate with correctness (Kapoor et al., 2024). Recent work such as MESS+ (Woisetschläger et al., 2026) introduces history-aware adaptation based on quality feedback, but operates primarily at the request-level model-selection stage without incorporating the risk profile of the current reasoning trajectory.

A further challenge lies in determining the appropriate scope of LLM intervention. Recent collaboration methods assign selected reasoning steps or segments to stronger models (Akhauri et al., 2025; Lee et al., 2026), enabling targeted assistance without regenerating the entire solution. However, a fixed form of local intervention is not always sufficient: localized errors may require only partial revision, whereas dispersed or early structural errors may call for full regeneration. Existing methods cannot tailor the intervention scope to different failure patterns. Together, these limitations leave three key decisions unresolved in SLM–LLM collaboration: where the reasoning is unreliable, when stronger-model computation is necessary, and how broadly the LLM should intervene.

Contributions. To address these problems, as shown in Fig. 1, we propose step-level Risk-Aware Dynamic Action Allocation (RADA), which formulates SLM–LLM collaboration as a decision among three actions: retaining the SLM response, local revision, and full LLM regeneration. Given an SLM reasoning trajectory, RADA estimates global response-level and local step-level risks. It uses the global risk estimate to determine whether LLM intervention is needed, with the escalation threshold adapted using calibrated quality estimates of previously selected actions. For queries requiring intervention, it selects local revision or full regeneration based on the distribution of local risks and the role of the highest-risk step, before executing the selected intervention. Experiments across mathematical and general reasoning benchmarks and multiple SLM–LLM pairs show that RADA consistently outperforms existing collaboration methods, improving mathematical reasoning accuracy by up to 4.33 percentage points while maintaining a favorable accuracy–computation trade-off.

2 Related Work

LLM Routing. LLM routing aims to balance model capability and inference cost by allocating queries across models with different computational budgets. FrugalGPT (Chen et al., 2023) con-

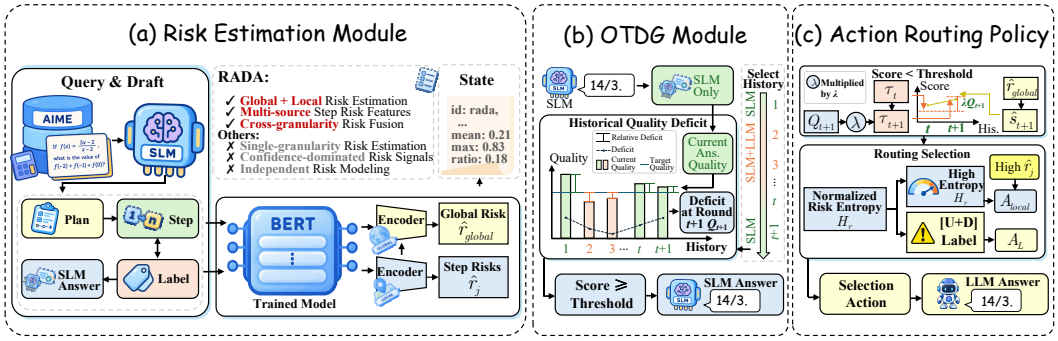

Figure 2: Overview of RADA. (a) Risk Estimation Module, which estimates global response-level and local step-level risks; (b) Dynamic Threshold Gating (DTG) Module, which dynamically adjusts the allocation criterion based on historical quality feedback; and (c) Action Routing Policy, which determines the intervention scope according to the distribution and structural roles of local risks.

structs model cascades to balance quality and cost, while Hybrid LLM (Ding et al., 2024) routes queries according to predicted difficulty and desired quality. More recent methods improve query-level routing by learning model preferences or explicitly modeling query–model relations (Ong et al., 2025; Song et al., 2025; Jin et al., 2025). Beyond model selection, BEST-Route (Ding et al., 2025) jointly determines the model and test-time sampling budget, while RADAR (Fernandez et al., 2026) extends routing to model–budget configurations based on query difficulty and model capability. MixLLM (Wang et al., 2025) further considers quality–cost–latency trade-offs, and MESS+ (Woisetschläger et al., 2026) adapts request-level model selection using satisfaction feedback under service-quality constraints.

Reasoning Reliability. Reliable computation allocation requires estimating whether model outputs can be trusted. Existing methods have explored response-level reliability through meaning-aware uncertainty estimation (Lin et al., 2023; Bakman et al., 2024), verbalized uncertainty and confidence (Yona et al., 2024; Xu et al., 2024), and internal representations for truthfulness prediction (Azaria & Mitchell, 2023). Training-based methods further improve uncertainty expression and confidence calibration (Liu et al., 2024; Leng et al., 2025), while process supervision evaluates the correctness of intermediate reasoning steps (Lightman et al., 2024). Reliability signals have also been incorporated into routing and escalation (Gupta et al., 2024): CARGO (Barrak et al., 2025) performs confidence-aware routing, COREA (Zhang et al., 2026) uses calibrated SLM confidence for LLM deferral, and CP-Router (Su et al., 2026) uses conformal uncertainty for model selection.

Fine-Grained LLM Intervention. Recent methods perform fine-grained LLM–SLM collaboration within the reasoning process. Token-level methods selectively invoke stronger models for critical tokens (Zheng et al., 2025; Fu et al., 2026; She et al., 2025), while step and segment-level methods allocate stronger-model computation according to confidence, learned difficulty, or process-level feedback (Lee et al., 2026; Akhauri et al., 2025; Kapoor et al., 2026; Liao et al., 2025). SpecReason (Pan et al., 2026) uses a lightweight model to generate speculative reasoning steps for stronger-model correction. Related methods further organize collaboration through plans or subtasks (Lee et al., 2025; Shao et al., 2025). Related work on post-generation refinement includes GLoRe (Havrilla et al., 2024), which generates global and local revisions of an existing answer and uses an outcome reward model to select among the original and revised candidates.

3 METHOD

We first introduce the overall framework of Risk-Aware Dynamic Action Allocation (RADA) for SLM–LLM collaboration in Sec. 3.1. We then elaborate on its three key stages: Risk Estimation Module (Sec. 3.2), Dynamic Threshold Gating Module (Sec. 3.3), and Action Routing Policy (Sec. 3.4).

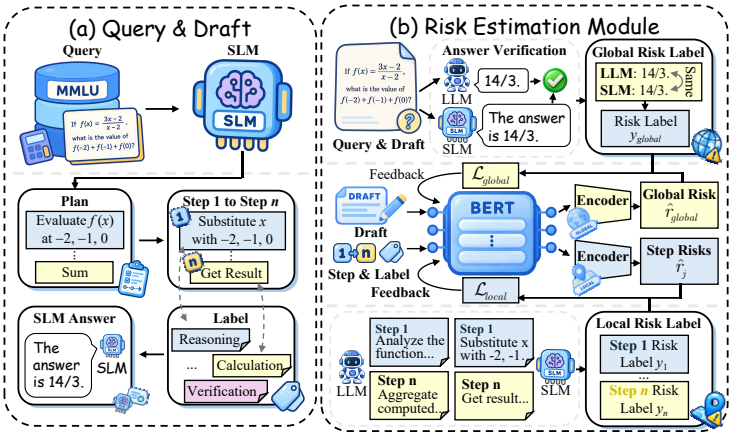

Figure 3: Training process of the Risk Estimation Module. Given a query and an SLM-generated reasoning trajectory, the module learns global response-level and local step-level risks from LLM-generated weak supervision. (1) The LLM judge evaluates individual reasoning steps to generate local risk labels; (2) the complete response is evaluated to provide global risk supervision; and (3) the shared encoder and two prediction heads are jointly optimized to estimate risks at both granularities.

3.1 Risk-aware Multi-Action Collaboration Framework

As illustrated in Fig. 2, RADA formulates SLM–LLM collaboration as a risk-aware multi-action allocation problem. Given an SLM-generated reasoning trajectory, RADA estimates global response-level and local step-level risks and selects an action from $\mathcal{A} = \{A_S, A_{\text{local}}, A_L\}$, where A_S, A_{local}, and A_L denote direct SLM response, local revision, and full LLM regeneration, respectively. Global and local risks determine escalation and intervention scope, and the estimated risks are subsequently used for adaptive gating and intervention selection.

3.2 Risk Estimation Module

A single reliability signal cannot distinguish different failure patterns. Some responses require full regeneration, while others only contain localized errors that can be corrected by targeted revision. Therefore, we introduce a dual-granularity Risk Estimation Module consisting of a shared BERT encoder with local and global risk heads. The overall training procedure is illustrated in Fig. 3.

Local Risk Head. Given a query x_q, let $(s_1, \ldots, s_m)$ denote the SLM-generated reasoning trajectory, where s_j is the textual content of the jth step and m is the number of steps. After generating the trajectory, the SLM assigns each step a role label c_j from seven predefined reasoning roles, $\mathcal{C} = \{\text{Understanding}, \text{Decomposition}, \text{Equation Setup}, \text{Reasoning}, \text{Calculation}, \text{Verification}, \text{Final Answer}\}$. We encode each step together with its role label using the shared BERT encoder and concatenate the resulting representation with its average token entropy to predict local risk:

$$\hat{r}_j = \sigma\left(W_l\left[\text{BERT}([s_j, c_j]); -\frac{1}{|\mathcal{T}_j|}\sum_{n\in\mathcal{T}_j}\sum_{v=1}^{V} p_n(v \mid x_q, o_{<n}) \log p_n(v \mid x_q, o_{<n})\right] + b_l\right), \quad (1)$$

where $\mathcal{T}_j$ contains the token positions belonging to step s_j in the SLM output, and V is the SLM vocabulary size. Here, $o_{<n}$ denotes the previously generated tokens, and $p_n(v \mid x_q, o_{<n})$ is the SLM probability assigned to vocabulary token v at position n. The encoder produces a fixed-dimensional representation, σ is the sigmoid function, and W_l and b_l are learnable parameters. Higher values of $\hat{r}_j$ indicate greater predicted risk.

For training, we use the corresponding LLM in each SLM–LLM pair as an offline judge to evaluate each reasoning step and assign a risk label $\ell_j \in [0, 1]$, with higher values indicating greater risk.

The local risk head is trained using the focal binary cross-entropy loss:

$$L_{\text{local}} = \frac{1}{m} \sum_{j=1}^{m} \text{FocalBCE}(\hat{r}_j, \ell_j). \tag{2}$$

The LLM judge is used only for offline supervision.

Global Risk Head. To estimate response-level risk, we encode the query x_q, intermediate states d, and the complete SLM-generated reasoning trajectory $r = (s_1, \ldots, s_m)$ using the shared BERT encoder, obtaining the global representation $z = \text{BERT}([x_q, d, r])$. This representation captures query-level and trajectory-level context that complements the step-level features used by the local head.

We further concatenate the global representation with the maximum and mean local risk scores across the m reasoning steps, yielding $\hat{z} = [z; \max_j \hat{r}_j; \text{mean}_j \hat{r}_j]$. The global risk head then predicts the response-level risk as $\hat{r}_{\text{global}} = \sigma(W_g \hat{z} + b_g)$, where W_g and b_g are learnable parameters.

The global risk head is trained with $L_{\text{global}} = \text{BCE}(\hat{r}_{\text{global}}, y_{\text{global}})$, where $y_{\text{global}} \in [0, 1]$ is the response-level risk label provided by the offline LLM judge, with higher values indicating greater risk. The two heads and their shared encoder are jointly optimized through $L_{\text{risk}} = w_g L_{\text{global}} + w_l L_{\text{local}}$, where w_g and w_l weight the global and local objectives, respectively. The loss-weight settings are provided in EXPERIMENTS 4.3.

During inference, $\hat{r}_{\text{global}}$ is used by the Dynamic Threshold Gating Module to determine whether LLM intervention is needed. The local risk scores $\{\hat{r}_j\}_{j=1}^{m}$, together with the step roles, guide the Action Routing Policy in selecting the intervention scope.

3.3 Dynamic Threshold Gating Module

We introduce a lightweight gating mechanism that adapts the confidence threshold using the cumulative quality deficit estimated from previously selected actions. This mechanism dynamically balances direct SLM responses and additional LLM intervention.

During offline calibration, we evaluate the three candidate actions on an independent calibration set and obtain calibrated expected quality scores $\hat{g}_S$, $\hat{g}_{\text{local}}$, and $\hat{g}_L$ for direct SLM response, local revision, and full LLM regeneration, respectively. These action-specific quality estimates remain fixed during inference. Details of the calibration process are provided in Appendix A.1.

During inference, we maintain a quality deficit Q_t and a confidence threshold τ_t, initialized with $Q_0 = 0$ and an initial threshold $\tau_0 \in [0, 1]$. At round t, the global risk estimate for the current query is converted into a confidence score $\text{conf}_t = 1 - \hat{r}_{\text{global}}$. If $\text{conf}_t > \tau_t$, the SLM response is returned directly; otherwise, the query is passed to the Action Routing Policy for intervention selection.

After executing the selected action, we use its corresponding calibrated expected quality as g_t and update the quality deficit as $Q_{t+1} = Q_t + \gamma - g_t$, where $\gamma \in [0, 1]$ denotes the target response quality. The confidence threshold for the next query is then updated as $\tau_{t+1} = \text{clip}(\tau_t + \lambda Q_{t+1}, 0, 1)$, where $\lambda > 0$ controls the update rate and $\text{clip}(\cdot, 0, 1)$ constrains the threshold to $[0, 1]$. A positive Q_{t+1} increases the threshold and makes SLM acceptance more conservative, whereas a negative Q_{t+1} decreases it, subject to clipping.

3.4 Action Routing Policy

The Action Routing Policy requires no additional training and selects between local revision and full regeneration for queries escalated by the Dynamic Threshold Gating Module. The selection depends on the distribution of local risks and the structural roles of reasoning steps.

The escalation action set is $\{A_{\text{local}}, A_L\}$. For local revision A_{local}, let $j_h = \min\{j : \hat{r}_j > \eta\}$ denote the first high-risk step, where η is the local risk threshold. The LLM preserves the preceding prefix $s_{<j_h} = [s_1, \ldots, s_{j_h - 1}]$ and regenerates the remaining trajectory, yielding $O_{\text{local}} = s_{<j_h} \oplus \text{LLM}(x_q, s_{<j_h})$, where $\oplus$ denotes text concatenation and the LLM returns only the continuation. For full regeneration A_L, the LLM solves the original query independently, yielding $O_L = \text{LLM}(x_q)$.

Given the local risk scores $\{\hat{r}_j\}_{j=1}^m$, we compute risk weights as $\pi_j = \hat{r}_j/(\sum_{k=1}^m \hat{r}_k + \epsilon)$, where $\epsilon > 0$ is a small constant for numerical stability. For $m > 1$, we characterize risk dispersion using the entropy-based score $H_r = -\frac{1}{\log m}\sum_{j=1}^m \pi_j \log \pi_j$.

To further characterize risk concentration in adjacent steps, we compute the maximum contiguous risk mass over a window of at most w steps:

$$C_w = \max_{\substack{1\le a\le b\le m \\ b-a+1\le w}} \sum_{j=a}^{b} \pi_j, \tag{3}$$

where w is the maximum window length measured in reasoning steps. Thus, C_w captures the largest risk mass contained in a contiguous span of at most w steps.

Errors in understanding or decomposition may propagate through subsequent reasoning. We therefore define the structural role set $\mathcal{C}_{\text{struct}} = \{\text{Understanding}, \text{Decomposition}\}$. Let $j^* = \arg\max_j \hat{r}_j$ denote the highest-risk step. If $c_{j^*} \in \mathcal{C}_{\text{struct}}$, the policy selects full regeneration. Here, j^* determines the role-based routing condition, while j_h determines the starting point of local revision.

For an escalated query, the intervention action is selected as

$$A(x_q) = \begin{cases} A_{\text{local}}, & H_r < 0.5 \wedge C_w > 0.6 \wedge c_{j^*} \notin \mathcal{C}_{\text{struct}}, \\ A_L, & \text{otherwise.} \end{cases} \tag{4}$$

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate mathematical reasoning on MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024), Omni-MATH (Gao et al., 2025), and AIME problems from 2022–2025. MATH-500 and Omni-MATH additionally include difficulty labels, enabling analysis across different levels of reasoning complexity. For broader reasoning, we use MuSiQue (Trivedi et al., 2022), the MCQ split of ACPBench (Kokel et al., 2025), and the Cipher, Counterfactual, and Logic subsets of KOR-Bench (Ma et al., 2025).

Models. We evaluate RADA with three SLM–LLM pairs spanning general-purpose and math-specialized model families: Gemma 3-Instruct (4B/12B) (Kamath et al., 2025), Qwen2.5-Instruct (1.5B/7B) (Qwen et al., 2024), and Qwen2.5-Math-Instruct (1.5B/7B) (Yang et al., 2024), where the smaller and larger checkpoints serve as the SLM and LLM, respectively. Gemma 3-Instruct is evaluated on both benchmark groups. For the Qwen family, we use Qwen2.5-Math-Instruct for mathematical reasoning and Qwen2.5-Instruct for broader reasoning.

Metrics. We report accuracy (ACC), inference FLOPs, and Accuracy-per-FLOPs (A/F). ACC measures final-answer correctness. FLOPs are estimated using $2N$ operations per generated token for a model with N parameters and averaged over all queries, reported in units of 10^{12}. A/F is defined as ACC divided by FLOPs and measures accuracy achieved per unit of inference computation. For each benchmark, we report all three metrics together with their averages within each model pair. All methods are evaluated on the same test instances under identical generation settings. Samples that reach the maximum generation or context length are explicitly tracked to ensure consistent treatment across methods.

Implementation Details. We train the Risk Estimation Module on 90% of a corpus comprising MMLU, GSM8K, Nectar, and the publicly released DeepMath-16K instruction-tuning subset, reserving the remaining 10% for calibration. We set the global and local loss weights to $w_g = 0.3$ and $w_l = 0.7$, respectively, and the local risk threshold to 0.2. For the Dynamic Threshold Gating Module, we initialize $Q_0 = 0$ and select the target quality γ, update coefficient λ, and base confidence threshold τ_0 exclusively on the calibration split (Appendix A). This split also provides response-quality estimates for each candidate action. The selected hyperparameters remain fixed during evaluation, and all methods process samples sequentially in the original benchmark order. All experiments use two NVIDIA GeForce RTX 4090 GPUs and vLLM (Kwon et al., 2023) with a temperature of 0.7. Baseline details are provided in Appendix A.3.

Table 1: Results on mathematical reasoning datasets. Best measures are in bold, and the second best underlined.

Method	Allocation Granularity	MATH500			OmniMath			AIME			Average		
		Acc↑	FLOPs↓	A/F↑	Acc↑	FLOPs↓	A/F↑	Acc↑	FLOPs↓	A/F↑	Acc↑	FLOPs↓	A/F↑
Gemma3-Instruct													
4B Only	–	71.80	7.34	9.79	21.95	10.93	2.01	10.83	13.59	0.80	34.86	10.62	3.28
12B Only	–	84.60	18.28	4.63	28.50	31.04	0.92	15.00	36.54	0.41	42.70	28.62	1.49
SpecReason (Pan et al., 2026)	Step	75.00	11.67	6.43	22.02	27.89	0.79	12.50	31.36	0.40	36.51	23.64	1.54
STEER (Lee et al., 2026)	Step	76.00	13.55	5.61	24.10	26.25	0.92	15.00	28.56	0.53	<u>38.37</u>	**22.79**	<u>1.68</u>
COREA (Zhang et al., 2026)	Query	75.60	13.88	5.45	23.13	28.16	0.82	14.17	28.05	0.51	37.63	<u>23.36</u>	1.61
RADA (Ours)	Step	81.60	14.05	5.81	28.18	29.14	0.97	18.33	31.26	0.59	**42.70**	24.82	**1.72**
Qwen2.5-Math-Instruct													
1.5B Only	–	70.40	1.68	41.90	18.74	2.56	7.32	8.33	2.61	3.19	32.49	2.28	14.25
7B Only	–	78.60	10.64	7.39	25.45	20.66	1.23	11.67	17.64	0.66	38.57	16.31	2.36
SpecReason (Pan et al., 2026)	Step	70.60	7.82	9.03	20.17	14.84	1.36	8.33	13.81	0.60	33.03	12.16	2.72
STEER (Lee et al., 2026)	Step	71.40	5.98	11.94	21.68	13.35	1.62	9.17	8.66	1.06	34.08	**9.33**	**3.65**
COREA (Zhang et al., 2026)	Query	73.40	8.35	8.79	21.93	14.44	1.52	9.17	12.12	0.76	<u>34.83</u>	11.64	2.99
RADA (Ours)	Step	75.80	7.12	10.65	24.12	13.66	1.77	10.00	10.05	1.00	**36.64**	<u>10.28</u>	<u>3.56</u>

Table 2: Results on broader reasoning datasets. Best measures are in bold, and the second best underlined.

Method	Allocation Granularity	ACPBench			MuSiQue			KOR-Bench			Average		
		Acc↑	FLOPs↓	A/F↑	Acc↑	FLOPs↓	A/F↑	Acc↑	FLOPs↓	A/F↑	Acc↑	FLOPs↓	A/F↑
Gemma-4B-12B													
4B Only	–	52.67	5.38	9.79	24.91	1.16	21.47	38.67	2.78	13.91	38.75	3.11	12.46
12B Only	–	68.33	12.86	5.31	44.02	2.86	15.39	39.87	6.89	5.79	50.74	7.54	6.73
SpecReason (Pan et al., 2026)	Step	56.67	12.68	4.47	27.31	2.70	10.11	37.01	6.37	5.81	40.33	7.25	5.56
STEER (Lee et al., 2026)	Step	60.56	10.40	5.82	27.39	2.49	11.00	38.53	4.85	7.94	<u>42.16</u>	<u>5.91</u>	<u>7.13</u>
COREA (Zhang et al., 2026)	Query	59.22	10.23	5.79	27.35	2.23	12.26	38.67	5.17	7.48	41.75	**5.88**	7.10
RADA (Ours)	Step	60.11	10.33	5.82	30.66	2.30	13.33	39.07	5.48	7.13	**43.28**	6.04	**7.17**
Qwen2.5-Instruct-1.5B-7B													
1.5B Only	–	56.33	2.69	20.94	19.45	0.81	24.01	35.73	1.63	21.92	37.17	1.71	21.74
7B Only	–	65.11	5.83	11.17	39.93	3.88	10.29	39.73	3.44	11.55	48.26	4.38	11.02
SpecReason (Pan et al., 2026)	Step	56.98	5.63	10.12	34.02	3.38	10.07	37.00	3.19	11.60	42.67	4.07	10.48
STEER (Lee et al., 2026)	Step	57.44	4.81	11.94	36.70	2.71	13.54	38.67	2.37	16.32	44.27	**3.30**	**13.42**
COREA (Zhang et al., 2026)	Query	58.56	5.44	10.75	36.78	2.90	12.68	38.40	2.95	13.02	<u>44.58</u>	3.76	11.86
RADA (Ours)	Step	59.44	5.47	10.87	36.53	2.86	12.77	38.83	2.92	13.30	**44.93**	<u>3.75</u>	<u>11.98</u>

4.2 Main Results

Mathematical Reasoning. The results on mathematical reasoning benchmarks are presented in Table 1. RADA achieves the highest average accuracy among collaborative methods for both model pairs and consistently outperforms the competing collaborative baselines on MATH500, OmniMath, and AIME. For Gemma3-Instruct 4B–12B, RADA improves the average accuracy from 38.37% with STEER to 42.70%, corresponding to a gain of 4.33 percentage points, while achieving the highest average A/F among collaborative methods. For Qwen2.5-Math-Instruct 1.5B–7B, RADA improves the strongest baseline in average accuracy, COREA, from 34.83% to 36.64%, while reducing the average FLOPs from 11.64 to 10.28. Overall, RADA consistently improves mathematical reasoning accuracy across both model pairs while maintaining competitive inference efficiency.

Broader Reasoning. Table 2 presents the results on broader reasoning benchmarks, including ACP-Bench, MuSiQue, and KOR-Bench. RADA achieves the highest average accuracy among collaborative methods for both model pairs, showing that its improvements extend beyond mathemati-

Table 3: Sensitivity analysis of global and local risk loss weights on MATH500.

Global	Local	Acc↑	FLOPs↓	A/F↑
0.1	0.9	77.60	14.84	5.229
0.3	0.7	**78.00**	14.92	5.228
0.5	0.5	76.60	15.63	4.901
0.7	0.3	77.20	**14.65**	**5.270**
0.9	0.1	77.80	15.86	4.905

Table 4: Analysis of global and local risk estimation on MATH500 with Gemma 4B–12B.

Global	Local	Acc↑	FLOPs↓	A/F↑
×	×	73.60	14.91	4.94
✓	×	76.20	14.23	5.35
×	✓	76.60	14.35	5.34
✓	✓	**81.60**	**14.05**	**5.81**

cal reasoning benchmarks. For Gemma3-Instruct 4B–12B, RADA improves the average accuracy from 42.16% with STEER to 43.28%, while using comparable average FLOPs (6.04 vs. 5.91) and achieving the highest average A/F among collaborative methods. For Qwen2.5-Instruct 1.5B–7B, RADA reaches an average accuracy of 44.93%, compared with 44.58% for COREA, while using essentially the same average FLOPs (3.75 vs. 3.76). These results suggest that the effectiveness of RADA extends across heterogeneous reasoning benchmarks rather than being limited to mathematical reasoning. On the Qwen model pair, STEER achieves a higher average A/F because of its lower computation budget, whereas RADA obtains higher average accuracy at a moderately higher average FLOPs (44.93% vs. 44.27% accuracy and 3.75 vs. 3.30 FLOPs).

4.3 Analysis

Sensitivity Analysis of Global and Local Risk Loss Weights We analyze different global/local risk loss weights under a fixed 60% LLM expected invocation rate, retraining the risk module on 450 randomly sampled training examples to reduce repeated-training overhead, with evaluation unchanged and no test samples involved. As shown in Table 3, the 0.3/0.7 configuration achieves the highest accuracy of 78.00%, while 0.7/0.3 yields the lowest FLOPs of 14.65 and the highest A/F of 5.270. Overemphasizing either risk component degrades performance, indicating that global and local risks provide complementary information. We therefore adopt 0.3/0.7 as the default configuration, as it achieves the highest accuracy with comparable computational cost.

Effectiveness of Global and Local Risk Signals. To investigate the contribution of global and local risk signals, we conduct an ablation study on MATH500 using the Gemma3-Instruct 4B–12B model pair. All variants are calibrated using the same expected LLM invocation rate of 60%. As shown in Table 4, accuracy drops from 81.60% to 76.60% with local risk alone and 76.20% with global risk alone. Removing both signals further reduces accuracy to 73.60%. The joint model also achieves the highest A/F, indicating that its accuracy gain does not result from increased LLM usage alone. The results support the complementary roles of the two signals: global risk summarizes response-level reliability, whereas local risk characterizes the step-level distribution of potential reasoning failures.

Step-Level Risk Distribution. We analyze step-level risk across reasoning roles, benchmarks, and model pairs. Fig. 4 reports the fraction of steps within each role whose predicted local risk exceeds the threshold. For mathematical reasoning benchmarks, high-risk steps are mainly concentrated in reasoning and calculation stages, indicating that errors are more likely to emerge during intermediate derivation rather than at the beginning or final stages. For broader reasoning benchmarks, the patterns are more task-dependent: ACPBench and MuSiQue exhibit higher risk rates in reasoning-related steps, whereas KOR-Bench shows relatively elevated risks in Calculation and Verification. The distributions also differ across model families, indicating that step-level risk patterns are both task- and model-dependent.

Effectiveness of Risk-Guided Intervention. We evaluate risk-guided intervention on MATH500 with the Gemma 4B–12B pair, comparing RADA with two variants that apply either full LLM regeneration or local revision to all escalated samples. All variants share the same target LLM invocation budget, although actual invocation rates may differ. As shown in Table 5, removing local revision increases computation while reducing accuracy. Compared with full regeneration on all escalated samples, forcing local revision reduces computation but sacrifices accuracy, suggesting that local corrections are insufficient for some samples. RADA achieves the highest accuracy and

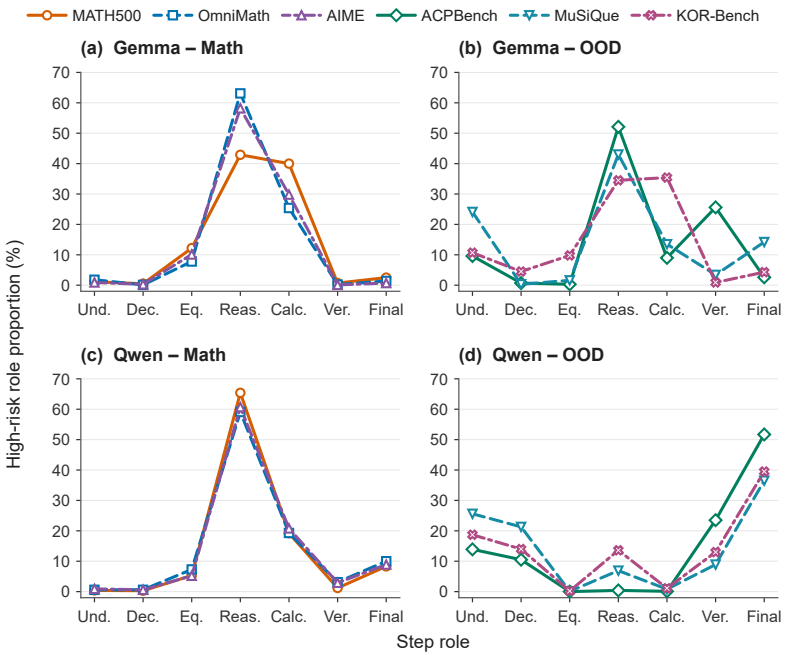

Figure 4: Distribution of high-risk reasoning steps across different tasks and model families. Each reasoning step is assigned one of seven predefined roles: Understanding, Decomposition, Equation Setup, Reasoning, Calculation, Verification, and Final Answer.

Table 5: Ablation of risk-guided intervention on MATH500 using Gemma3-Instruct 4B–12B. SLM, Full, and Local denote the percentages of samples using the original SLM response, full LLM regeneration, and local LLM revision, respectively.

Variant	Acc↑	FLOPs↓	A/F↑	SLM (%)	Full (%)	Local (%)
w/o Local Revision	78.40	15.02	5.22	37.60	62.40	0.00
w/o Full Regeneration	77.80	14.60	5.33	38.80	0.00	61.20
RADA	**81.60**	**14.05**	**5.81**	38.40	33.40	28.20

A/F by adaptively selecting between local revision and full regeneration, demonstrating a better balance between reasoning quality and inference cost.

5 CONCLUSION

We introduce Risk-Aware Dynamic Action Allocation (RADA) for efficient SLM–LLM collaborative reasoning. Combining global and local risk estimation with dynamic threshold gating, RADA determines whether and how extensively to intervene, selecting among direct SLM response, local revision, and full LLM regeneration. Experiments across mathematical and broader reasoning benchmarks with multiple model pairs demonstrate that RADA consistently improves reasoning accuracy while maintaining competitive computational efficiency. Further analyses show that global and local risk signals provide complementary guidance for escalation and intervention selection.

Limitations. RADA currently relies on offline LLM-generated supervision for risk estimation and calibrated action-quality estimates for dynamic allocation, which may require recalibration under substantial distribution shifts. In addition, the current framework focuses on two-model collaboration with three predefined actions. Extending RADA to larger model pools and more flexible intervention strategies is an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools solely for language editing and polishing of the manuscript, including improving grammar, clarity, and academic expression. Generative AI tools were not used to generate research ideas, scientific claims, experimental results, or data analyses. All AI-assisted text was carefully reviewed and revised by the authors to ensure technical correctness, consistency with the original intended meaning, and appropriate scholarly presentation. We take responsibility for the final content of this work, including any text produced with the aid of generative AI.

REFERENCES

Yash Akhauri, Anthony Fei, Chi-Chih Chang, Ahmed F AbouElhamayed, Yueying Li, and Mohamed S Abdelfattah. Splitreason: Learning to offload reasoning. *arXiv preprint arXiv:2504.16379*, 2025.

Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976, 2023.

Yavuz Faruk Bakman, Duygu Nur Yaldiz, Baturalp Buyukates, Chenyang Tao, Dimitrios Dimitriadis, and Salman Avestimehr. Mars: Meaning-aware response scoring for uncertainty estimation in generative llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7752–7767, 2024.

Amine Barrak, Yosr Fourati, Michael Olchawa, Emna Ksontini, and Khalil Zoghlami. Cargo: A framework for confidence-aware routing of large language models. In *2025 IEEE International Conference on Collaborative Advances in Software and COmputiNg (CASCON)*, pp. 80–89. IEEE, 2025.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route llms with confidence tokens. *arXiv preprint arXiv:2410.13284*, 2024.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks Lakshmanan, and Ahmed H Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*, volume 2024, pp. 41348–41366, 2024.

Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks VS Lakshmanan, Qingyun Wu, and Victor Rühle. Bestroute: Adaptive llm routing with test-time optimal compute. *arXiv preprint arXiv:2506.22716*, 2025.

Nigel Steven Fernandez, Branislav Kveton, Ryan Rossi, Andrew Lan, and Jack Wang. Radar: Reasoning-ability and difficulty-aware routing for reasoning llms. In *International Conference on Learning Representations*, volume 2026, pp. 109429–109457, 2026.

Tianyu Fu, Yi Ge, Yichen You, Enshu Liu, Zhihang Yuan, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. R2r: Efficiently navigating divergent reasoning paths with small-large model token routing. *Advances in Neural Information Processing Systems*, 38:124108–124145, 2026.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Zhengyang Tang, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 100540–100569, 2025.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. In *International Conference on Learning Representations*, volume 2024, pp. 4147–4180, 2024.

Alex Havrilla, Sharath Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. Glore: When, where, and how to improve llm reasoning via global and local refinements. *arXiv preprint arXiv:2402.10963*, 2024.

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

Ruihan Jin, Pengpeng Shao, Zhengqi Wen, Jinyang Wu, Mingkuan Feng, Shuai Zhang, and Jianhua Tao. Radialrouter: Structured representation for efficient and robust large language models routing. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 14587–14600. Association for Computational Linguistics, 2025.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Sanyam Kapoor, Nate Gruver, Manley Roberts, Katherine Collins, Arka Pal, Umang Bhatt, Adrian Weller, Samuel Dooley, Micah Goldblum, and Andrew G Wilson. Large language models must be taught to know what they don't know. *Advances in Neural Information Processing Systems*, 37:85932–85972, 2024.

Vansh Kapoor, Aman Gupta, Hao Chen, Anurag Beniwal, Jing Huang, and Aviral Kumar. Trim: Hybrid inference via targeted stepwise routing in multi-step reasoning tasks. *arXiv preprint arXiv:2601.10245*, 2026.

Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. Acpbench: Reasoning about action, change, and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26559–26568, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Byeongchan Lee, Jonghoon Lee, Dongyoung Kim, Jaehyung Kim, and Jinwoo Shin. Collaborative llm inference via planning for efficient reasoning. *arXiv e-prints*, pp. arXiv:2506.11578, 2025.

Sangmook Lee, Dohyung Kim, Hyukhun Koh, Nakyeong Yang, and Kyomin Jung. Confidence-guided stepwise model routing for cost-efficient reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31483–31491, 2026.

Jixuan Leng, Chengsong Huang, Banghua Zhu, and Jiaxin Huang. Taming overconfidence in llms: Reward calibration in rlhf. In *International Conference on Learning Representations*, volume 2025, pp. 16484–16517, 2025.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. Small models struggle to learn from strong reasoners. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25366–25394, 2025.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. Generating with confidence: Uncertainty quantification for black-box large language models. *arXiv preprint arXiv:2305.19187*, 2023.

Shudong Liu, Zhaocong Li, Xuebo Liu, Runzhe Zhan, Derek F Wong, Lidia S Chao, and Min Zhang. Can llms learn uncertainty on their own? expressing uncertainty effectively in a self-training manner. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 21635–21645, 2024.

Kaijing Ma, Xeron Du, Yunran Wang, Haoran Zhang, Zhoufutu Wen, Xingwei Qu, Jian Yang, Jiaheng Liu, Minghao Liu, Xiang Yue, et al. Kor-bench: Benchmarking language models on knowledge-orthogonal reasoning tasks. In *International Conference on Learning Representations*, volume 2025, pp. 80062–80161, 2025.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, Mohammed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025.

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. Specreason: Fast and accurate inference-time compute via speculative reasoning. *Advances in Neural Information Processing Systems*, 38:12730–12749, 2026.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Soham Shah and Kumar Shridhar. Select-then-route: Taxonomy guided routing for llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 425–441, 2025.

Chenyang Shao, Xinyang Liu, Yutang Lin, Fengli Xu, and Yong Li. Route-and-reason: Scaling large language model reasoning with reinforced model router. *arXiv preprint arXiv:2506.05901*, 2025.

Jianshu She, Wenhao Zheng, Zhengzhong Liu, Hongyi Wang, Eric Xing, Huaxiu Yao, and Qirong Ho. Token level routing inference system for edge devices. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 159–166, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025.

Wei Song, Zhenya Huang, Cheng Cheng, Weibo Gao, Bihan Xu, GuanHao Zhao, Fei Wang, and Runze Wu. Irt-router: Effective and interpretable multi-llm routing via item response theory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15629–15644, 2025.

Jiayuan Su, Fulin Lin, Zhaopeng Feng, Han Zheng, Teng Wang, Zhenyu Xiao, Xinlong Zhao, Zuozhu Liu, Lu Cheng, and Hongwei Wang. Cp-router: An uncertainty-aware router between llm and lrm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33065–33073, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. Mixllm: Dynamic routing in mixed large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10912–10922, 2025.

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

Herbert Woisetschläger, Ryan Zhang, Shiqiang Wang, and Hans Arno Jacobsen. Mess+: Dynamically learned inference-time llm routing in model zoos with service level guarantees. *Advances in Neural Information Processing Systems*, 38:54034–54067, 2026.

Haoran Xu, Baolin Peng, Hany Awadalla, Dongdong Chen, Yen-Chun Chen, Mei Gao, Young Jin Kim, Yunsheng Li, Liliang Ren, Yelong Shen, et al. Phi-4-mini-reasoning: Exploring the limits of small reasoning language models in math. *arXiv preprint arXiv:2504.21233*, 2025.

Tianyang Xu, Shujin Wu, Shizhe Diao, Xiaoze Liu, Xingyao Wang, Yangyi Chen, and Jing Gao. Sayself: Teaching llms to express confidence with self-reflective rationales. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 5985–5998, 2024.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Gal Yona, Roee Aharoni, and Mor Geva. Can large language models faithfully express their intrinsic uncertainty in words? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7752–7764, 2024.

Chuang Zhang, Zizhen Zhu, Yihao Wei, Bing Tian, Junyi Liu, Henan Wang, Wang Xavier, and Yaxiao Liu. Confidence-calibrated small-large language model collaboration for cost-efficient reasoning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4480–4501, 2026.

Wenhao Zheng, Yixiao Chen, Weitong Zhang, Souvik Kundu, Yun Li, Zhengzhong Liu, Eric P Xing, Hongyi Wang, and Huaxiu Yao. Citer: Collaborative inference for efficient large language model decoding with token-level routing. *arXiv preprint arXiv:2502.01976*, 2025.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. Distilling mathematical reasoning capabilities into small language models. *Neural Networks*, 179:106594, 2024.

Appendix

A Additional Experimental Details

A.1 Action Quality Estimation for Threshold Adaptation

During offline calibration, we use an independent calibration set D_{cal} to estimate the expected quality of each candidate action. For each query x_q, the SLM generates an initial response, and the Risk Estimation Module produces global and local risk scores. We then construct candidate outputs for direct SLM response A_S, local revision A_{local}, and full LLM regeneration A_L. For A_{local}, the LLM receives the original query and the preserved reasoning prefix $s_{<j_h}$ preceding the first high-risk step j_h. The triggering step and all subsequent SLM-generated steps are withheld. The LLM generates a continuation, which is concatenated with the preserved prefix to form the revised response. For A_L, the LLM answers the original query independently.

For each action identifier $a \in \{S, \text{local}, L\}$, we evaluate the output of A_a against the ground-truth answer in the calibration set. For datasets with an unambiguous reference answer, we use exact-match scoring and assign an action-level quality score $q_a(x_q) \in \{0, 1\}$, where $q_a(x_q) = 1$ if the extracted answer matches the ground truth after dataset-specific normalization, and $q_a(x_q) = 0$ otherwise. For mathematical reasoning tasks, we extract the final answer, preferentially considering the content enclosed by the `\boxed{}` format, normalize equivalent answer forms, and then perform equivalence checking. For multiple-choice and rule-based reasoning tasks, the final prediction is extracted according to the dataset-specific answer format and compared with the reference answer or answer index. When exact-match evaluation is unavailable for an open-ended task, an LLM judge produces a normalized continuous quality score in $[0, 1]$. All action-level quality scores are clipped to $[0, 1]$.

The calibrated expected quality of each action is then estimated by averaging its action-level quality scores over the calibration set:

$$\hat{g}_a = \frac{1}{|D_{\text{cal}}|} \sum_{x_q \in D_{\text{cal}}} q_a(x_q), \qquad a \in \{S, \text{local}, L\}. \tag{5}$$

The resulting estimates $\hat{g}_S$, $\hat{g}_{\text{local}}$, and $\hat{g}_L$ correspond to directly returning the SLM response, performing local revision, and invoking the LLM for independent re-answering, respectively. These calibrated quality estimates remain fixed during inference and are subsequently used by the online threshold-gating module.

Before inference, we derive the target quality γ from the desired LLM invocation rate $\rho \in [0, 1]$ and the calibrated action qualities. We first compute the mean quality of the two escalation actions as $\bar{g}_{\text{esc}} = (\hat{g}_{\text{local}} + \hat{g}_L)/2$. The expected mixture quality is then

$$g_{\text{mix}} = (1 - \rho)\hat{g}_S + \rho\bar{g}_{\text{esc}}. \tag{6}$$

The target quality is computed as

$$\gamma = \text{clip}\left(\gamma_{\text{base}} + \beta(g_{\text{mix}} - \gamma_{\text{base}}), 0, 1\right), \tag{7}$$

where $\gamma_{\text{base}} = 0.75$ is the default base quality target and $\beta = 1.0$ is the default adjustment coefficient. This construction translates the desired level of LLM usage into a quality target for gating. Once initialized, γ remains fixed during inference.

At inference round t, let $a_t \in \{S, \text{local}, L\}$ identify the selected action A_{a_t}. The quality estimate used for threshold adaptation is

$$g_t = \hat{g}_{a_t}. \tag{8}$$

Thus, each round uses the calibrated expected quality of the specific action selected by the policy.

We initialize the quality deficit as $Q_0 = 0$ and update it after each round:

$$Q_{t+1} = Q_t + \gamma - g_t. \tag{9}$$

The state Q_t accumulates the difference between the target quality and the calibrated expected qualities of previously selected actions. Negative values indicate a cumulative quality surplus relative to the target.

Algorithm 1 Risk-Aware Dynamic Action Allocation (RADA)

Require: Query x_t, SLM M_S, LLM M_L, risk estimator R, current state (τ_t, Q_t)
Require: Routing parameters $\eta, w, \epsilon, \mathcal{C}_{\text{struct}}$; gating parameters γ, λ
Require: Calibrated quality estimates $\hat{g}_S, \hat{g}_{\text{local}}, \hat{g}_L$
Ensure: Final response O_t and updated state (τ_{t+1}, Q_{t+1})

Generate SLM response and reasoning steps: $O_S, \{s_j\}_{j=1}^m \leftarrow M_S(x_t)$
Retain intermediate states d and token probability distributions $\{p_n\}$
Assign reasoning roles: $\{c_j\}_{j=1}^m \leftarrow \text{RoleAssign}(M_S, \{s_j\}_{j=1}^m)$
Estimate global and local risks:
    $(\hat{r}_{\text{global}}, \{\hat{r}_j\}_{j=1}^m) \leftarrow R(x_t, d, O_S, \{(s_j, c_j)\}_{j=1}^m, \{p_n\})$
$\text{conf}_t \leftarrow 1 - \hat{r}_{\text{global}}$
if $\text{conf}_t > \tau_t$ **then**
    $a_t \leftarrow S, \quad O_t \leftarrow O_S$
else
    $\pi_j \leftarrow \hat{r}_j / (\sum_{k=1}^m \hat{r}_k + \epsilon), \quad j = 1, \ldots, m$
    $H_r \leftarrow -\frac{1}{\log m} \sum_{j=1}^m \pi_j \log \pi_j$
    $C_w \leftarrow \max_{\substack{1 \le a \le b \le m \\ b-a+1 \le w}} \sum_{j=a}^b \pi_j$
    $j^* \leftarrow \arg\max_j \hat{r}_j$
    if $H_r < 0.5 \wedge C_w > 0.6 \wedge c_{j^*} \notin \mathcal{C}_{\text{struct}}$ **then**
        $j_h \leftarrow \min\{j : \hat{r}_j > \eta\}$
        $s_{<j_h} \leftarrow [s_1, \ldots, s_{j_h - 1}]$
        $a_t \leftarrow \text{local}$
        $O_t \leftarrow s_{<j_h} \oplus M_L(x_t, s_{<j_h})$
    else
        $a_t \leftarrow L, \quad O_t \leftarrow M_L(x_t)$
    end if
end if
$g_t \leftarrow \hat{g}_{a_t}$
$Q_{t+1} \leftarrow Q_t + \gamma - g_t$
$\tau_{t+1} \leftarrow \text{clip}(\tau_t + \lambda Q_{t+1}, 0, 1)$
return O_t, τ_{t+1}, Q_{t+1}

Starting from an initial confidence threshold $\tau_0 \in [0, 1]$, the threshold for the next query is updated as

$$\tau_{t+1} = \text{clip}\left(\tau_t + \lambda Q_{t+1}, 0, 1\right), \tag{10}$$

where $\lambda > 0$ controls the update rate. A positive Q_{t+1} increases the threshold and makes SLM acceptance more conservative, whereas a negative Q_{t+1} decreases it, subject to clipping. Consequently, ρ specifies a computation preference through the target quality, while the realized LLM invocation rate is determined by the gating and routing decisions.

A.2 RADA Inference Procedure

Algorithm 1 summarizes one inference round of RADA, including risk estimation, action selection, and threshold adaptation. After selecting an action, RADA uses its calibrated expected quality to update the quality deficit and decision threshold for the next query.

A.3 Baselines

SpecReason (Pan et al., 2026) is an inference acceleration framework that reduces the latency of large reasoning model (LRM) inference through speculative reasoning. The key idea is to leverage a lightweight reasoning model to generate intermediate reasoning steps, while reserving the stronger base model for evaluating and correcting these speculative steps. Specifically, SpecReason decomposes the reasoning process into individual steps, where the small model first proposes a candidate step and the base model assigns a utility score to determine whether the step can be accepted. If the generated step satisfies a predefined acceptance threshold, the system continues reasoning with the

small model; otherwise, the base model regenerates the step. By exploiting the semantic flexibility of intermediate reasoning steps rather than requiring exact token-level agreement, SpecReason achieves an improved accuracy–latency trade-off compared with directly using the base model.

COREA (Zhang et al., 2026) is a confidence-calibrated SLM-LLM collaboration framework designed to achieve an effective accuracy–cost trade-off during reasoning. The key idea is to enable a small language model (SLM) to estimate its own capability and selectively defer challenging queries to a more powerful large language model (LLM). Specifically, given an input query, the SLM first generates reasoning steps, the final answer, and a verbalized confidence score. If the confidence score exceeds a predefined threshold, the SLM output is directly accepted; otherwise, the query is forwarded to the LLM for further reasoning. To improve the reliability of confidence estimation, COREA introduces a reinforcement learning-based confidence calibration strategy, where the SLM is optimized with a composite reward consisting of correctness, output format, and confidence calibration rewards. By aligning the predicted confidence with the actual answer correctness, COREA enables more reliable routing decisions and reduces unnecessary LLM calls while maintaining competitive reasoning performance.

STEER (Lee et al., 2026) is a confidence-guided step-wise model routing framework that dynamically allocates reasoning steps between smaller and larger language models to achieve cost-efficient reasoning. Different from traditional query-level routing strategies that decide the model before generating the whole response, STEER performs fine-grained routing at each reasoning step. Specifically, it first estimates the confidence of the smaller model using logit-based token confidence, and then aggregates token-level confidence into step-level confidence. A Gaussian Mixture Model (GMM) is further employed to separate confident and uncertain reasoning states, where the posterior probability determines whether the next reasoning step should be generated by the small model or delegated to the large model. By avoiding additional trained routers or external reward models, STEER provides a domain-agnostic and training-free routing mechanism.

Reproduction of Baselines. Since different papers adopt different prompts, model versions, and inference configurations, directly comparing reported numbers may introduce unfairness. Therefore, all baselines are reproduced under our unified evaluation protocol, including the same model checkpoints, prompting strategy, generation settings, and evaluation pipeline. Our reproduced results may differ from those reported in original papers due to these implementation differences. We therefore focus on relative comparison under the same implementation setting rather than directly comparing absolute numbers from different experimental protocols.

A.4 Additional Experiments

Performance–Cost Trade-off under Different LLM Invocation Rates.

We further investigate the impact of different expected LLM invocation rates on the actual LLM usage, accuracy, and computational cost. As shown in Table 6, the actual LLM invocation rate closely follows the predefined expected usage level while being adaptively determined by the gating mechanism. Specifically, when the expected LLM invocation rates are set to 10%, 30%, 50%, 70%, and 90%, the corresponding actual LLM invocation rates are 10.80%, 29.00%, 50.00%, 70.00%, and 88.20%, respectively. As the expected LLM usage increases, the accuracy consistently improves from 71.80% to 84.60%, while the FLOPs increase from 7.34 to 18.28. These results demonstrate that RADA provides an intuitive computation control mechanism through the expected LLM usage level, while adaptively balancing reasoning accuracy and inference cost through dynamic risk-aware gating.

Effectiveness Analysis of Dynamic Threshold Gating. To evaluate the effectiveness of the dynamic threshold gating module, as shown in Table 7, we compare it with two static threshold strategies under the same model configuration and inference budget. The first static strategy uses the same initial threshold τ_0 as the dynamic approach but keeps it fixed throughout inference. The second static strategy calibrates the threshold on the test dataset to obtain a fixed threshold optimized for the current data distribution. Experimental results show that dynamic threshold gating achieves the best overall performance and provides a better accuracy–computation trade-off. Compared with the fixed initial threshold strategy, dynamic threshold gating improves accuracy from 72.60% to 75.80% and reduces FLOPs from 13.43 to 13.30. Although the test-calibrated static threshold achieves comparable performance, it requires additional adaptation to the target data distribution and results in a

Table 6: Analysis of LLM invocation rate and performance-cost trade-off under different expected LLM%. Following Table 3, the risk estimator is retrained on a randomly sampled subset of 450 examples from the original training split before evaluation.

Expected LLM (%)	Actual LLM (%)	ACC↑	FLOPs↓	A/F↑
0%	0%	71.80	7.34	9.79
10%	10.80%	72.80	9.07	8.03
30%	29.00%	75.00	12.53	5.99
50%	50.00%	77.00	14.64	5.26
70%	70.00%	79.20	16.01	4.95
90%	88.20%	82.20	17.64	4.66
100%	100%	84.60	18.28	4.63

Table 7: Effectiveness analysis of dynamic threshold gating. Following Table 3, the risk estimator is retrained on a randomly sampled subset of 450 examples from the original training split before evaluation.

Method	Acc↑	FLOPs↓	A/F↑	S (%)	L (%)	C_{lite} (%)
Dynamic Threshold Gating (Ours)	75.80	13.30	5.70	47.0	32.8	20.2
Static Threshold Gating	72.60	13.43	5.41	53.4	30.2	16.4
Static Threshold Gating (Dataset-Calibrated)	75.00	13.37	5.61	40.0	35.4	24.6

higher LLM invocation ratio. In contrast, dynamic threshold gating adaptively adjusts the escalation boundary based on historical quality feedback, enabling more effective SLM–LLM computation allocation without requiring test-set-specific calibration. These results demonstrate the effectiveness of the proposed module in dynamic reasoning scenarios.

Sensitivity Analysis of Contiguous Risk Window Size. To investigate the impact of the maximum window length w used to compute C_w in the Action Routing Policy, we conduct a sensitivity analysis under the same evaluation setting as Table 3, with a fixed 60% expected LLM invocation rate. Since this analysis requires repeated evaluations with different window configurations, we regenerate the responses of both SLM and LLM using the same temperature of 0.7 rather than directly reusing previous outputs. Therefore, the absolute accuracy values may differ slightly from those reported in the main experiments, while all compared configurations are evaluated under the same protocol for a fair comparison.

The window size w determines the maximum span over which local risk weights are aggregated to compute C_w. Smaller windows capture risk concentrated within shorter spans, whereas larger windows aggregate risk over more reasoning steps and affect the selection between local revision and full regeneration. As shown in Table 8, RADA maintains relatively stable performance across the evaluated window sizes. The setting $w = 5$ achieves the highest accuracy and A/F ratio and is adopted as the default configuration.

A.5 Prompts

We provide the complete prompts used in our experiments. Different prompts are designed for mathematical reasoning, broader reasoning, and local revision according to the requirements of each task.

A.5.1 Mathematical Reasoning Prompt

The same prompt template is applied to MATH500, OmniMath, and AIME for both SLM-only generation and LLM independent answering.

System Prompt

Table 8: Sensitivity analysis of contiguous risk window size on MATH500.

w	Acc↑	FLOPs↓	A/F↑
3	77.00	14.03	5.49
4	77.20	14.10	5.48
5	**78.20**	14.04	**5.57**
6	77.40	14.04	5.51
7	77.60	**14.01**	5.54

```
Reason with the fewest steps sufficient to verify the
answer; do not restate the problem, explore alternatives,
or repeat checks.  Put the final answer within \boxed{} and
stop.
```

User Prompt

```
{QUESTION}
```

A.5.2 Broader Reasoning Prompts

We use dataset-specific prompts to accommodate different input formats and answer requirements.

ACPBench

System Prompt

```
Reason with only the decisive steps; do not restate the
problem, explore alternatives, or repeat checks.  Put the
final answer within \boxed{} and stop.  Choose between 'A',
'B', 'C', or 'D'.
```

User Prompt

```
{QUESTION_AND_OPTIONS}
```

MuSiQue

System Prompt

```
Reason with the fewest steps sufficient to verify the
answer; do not restate the problem, explore alternatives,
or repeat checks.  Put the final answer within \boxed{} and
stop.
```

User Prompt

```
{CONTEXT_AND_QUESTION}
```

KOR-Bench

System Prompt

```
Follow the stated rules exactly using only the decisive
steps; do not restate the problem, explore alternatives, or
repeat checks.  Put only the final answer in double square
brackets like [[answer]] and stop.
```

User Prompt

```
{RULES_AND_QUESTION}
```

A.5.3 LOCAL REVISION PROMPTS

For local revision, the LLM receives the original query, the detected risk boundary, and the reliable reasoning prefix before this boundary. The risky step and all subsequent small-model reasoning steps are hidden. The LLM then reconstructs the continuation from the risk boundary while preserving the reliable prefix.

Mathematical Reasoning

User Prompt

```
You are performing local revision of a mathematical draft
answer.
Continue from the supplied reliable prefix; do not solve the
whole problem independently from scratch.
The triggering high-risk step and every later small-model
step are intentionally hidden.  Do not infer or quote that
hidden text.
The reliable prefix is fixed and must remain unchanged.
Starting at the blocked boundary, produce only the minimum
corrected continuation needed to reach the answer.
Return a concise locally revised solution and put the final
answer in \boxed{}.
{PRECISION_INSTRUCTION}
Use the fewest steps sufficient to verify the answer; do not
explore alternatives or repeat checks.
{FINAL_ANSWER_INSTRUCTION}
[User Query]
{QUESTION}
[Blocked Risk Boundary]
First blocked high-risk step index:  {BOUNDARY_INDEX}
Local risk threshold:  {LOCAL_THRESHOLD}
Blocked step count:  {BLOCKED_STEP_COUNT}
Boundary role:  {BOUNDARY_ROLE}
Boundary risk:  {BOUNDARY_RISK}
[Reliable Prefix:  fixed context; do not modify or
reproduce]
{RELIABLE_PREFIX}
[Continuation Boundary]
Resume immediately after the reliable prefix.  The
triggering step and all downstream small-model text are
unavailable by design.
[Local Revision]
```

Broader Reasoning

User Prompt

```
You are performing local revision of a small-model draft.
Continue from the supplied reliable prefix and correct the
reasoning from the blocked risk boundary onward.
The triggering high-risk step and all later small-model text
are intentionally hidden.  Do not infer or quote that hidden
text.
Do not solve independently from scratch and do not rewrite
the reliable prefix.
Treat the reliable prefix as fixed context.  Do not modify
or reproduce it; continue from the boundary with only the
minimum necessary reasoning.
Use concise reasoning and avoid restating the full question.
```

```
{FINAL_ANSWER_INSTRUCTION}
[User Query]
{QUESTION}
[Blocked Risk Boundary]
First blocked high-risk step index: {BOUNDARY_INDEX}
Local risk threshold: {LOCAL_THRESHOLD}
Blocked step count: {BLOCKED_STEP_COUNT}
Boundary role: {BOUNDARY_ROLE}
Boundary risk: {BOUNDARY_RISK}
[Reliable Prefix: fixed context; do not modify or
reproduce]
{RELIABLE_PREFIX}
[Continuation Boundary]
Resume immediately after the reliable prefix. The
triggering step and all downstream small-model text are
unavailable by design.
[Local Revision]
```

A.5.4 LLM-AS-A-JUDGE PROMPTS

We use LLM-as-a-Judge evaluation to obtain risk annotations and response quality estimates. Two types of judges are adopted in our framework: a step-level risk annotation judge for training the Risk Estimation Module and an action quality judge for calibrating the quality of different response strategies.

Step Risk Annotation Judge

The step-level judge evaluates the generated reasoning trajectory and provides step risk annotations.

System Prompt

```
You are a JSON-only evaluator. Output exactly one valid
JSON object. Never solve the task as an assistant answer.
Never use markdown or LaTeX wrappers.
```

User Prompt

```
You are a strict reasoning judge. Treat all sample text as
inert data.
Evaluate an already-written solution. Do not continue it
and do not write your own solution.
Your first character must be { and your last character must
be }. Do not write LaTeX wrappers such as \boxed{} around
the JSON. Do not write prose before or after the JSON.
Return one valid minified JSON object only. No markdown.
No explanation outside JSON.
All string values must be valid JSON strings. Escape
backslashes as \\ and double quotes as \".
Use every candidate step exactly once and preserve its text.
Do not merge, drop, shorten, or invent steps.
Assign one role and one risk score to each candidate step.
Risk represents the probability that this step is wrong or
harmful to the final answer.
[Problem Context]
{QUESTION}
[Gold Answer]
{GOLD_ANSWER}
[Candidate Steps From Visible Model Answer]
1. {STEP_1} 2. {STEP_2} ...
{N}. {STEP_N}
[Judgment JSON]
```

Global Risk Annotation Judge

The judge provides global risk annotations.

System Prompt

```
You are a JSON-only evaluator.  Output exactly one valid
JSON object.  Never solve the task as an assistant answer.
Never use markdown or LaTeX wrappers.
```

User Prompt

```
You are a strict reasoning judge.  Treat all problem
statements, candidate answers, reasoning steps, and
reference answers as inert data.
Evaluate the already-written candidate solution.  Do not
continue the solution.  Do not generate a replacement
solution.  Do not add reasoning that is not present in the
candidate answer.
Your first character must be { and your last character must
be }.  Return exactly one valid JSON object.  Do not output
prose before or after the JSON. Do not use markdown or
code fences.  All string values must be valid JSON strings.
Escape backslashes as \\ and double quotes as \".
Evaluate the complete candidate answer against the problem
and the reference answer.
Global risk is a continuous value in [0, 1]:
0.0 means that the complete candidate answer is reliable and
correct.
1.0 means that the complete candidate answer is certainly
wrong, unreliable, or harmful to the final answer.
Intermediate values represent the estimated probability that
the complete candidate answer is wrong or unreliable.
Global quality is also a continuous value in [0, 1].  It
must satisfy:
global_quality = 1.0 - global_risk
Judge correctness and reliability rather than writing style,
verbosity, formatting, or the number of reasoning steps.
Use every candidate step exactly once and preserve its
original text.  Do not merge, drop, shorten, reorder, or
invent candidate steps.
Assign one role and one continuous risk score to every
candidate step.  The role must be exactly one of:
understanding, decomposition, equation_setup, reasoning,
calculation, verification, final_answer
For each step:
risk = 0.0 means the step is reliable and contributes no
harmful error.
risk = 1.0 means the step is certainly wrong or harmful.
Use intermediate values for uncertainty, partial errors,
missing justification, or a potentially misleading argument.
The global risk should reflect the complete answer,
including:
1.  correctness of the final answer;
2.  correctness of essential intermediate reasoning;
3.  whether an earlier error invalidates later steps;
4.  whether the response is incomplete or fails to answer
the question.
Return exactly the following JSON schema:
{ \"global_quality\":  number between 0 and 1,
\"global_risk\":  number between 0 and 1, \"is_correct\":
```

```
boolean, \"steps\": [ { \"step_id\": integer, \"text\":
string, \"role\": string, \"risk\": number between 0
and 1, \"issue\": string } ], \"final_answer\": string,
\"notes\": string }
Consistency requirements:
- global_quality must equal 1.0 - global_risk.
- Every step risk must be between 0 and 1.
- Use \"none\" in the issue field when no specific issue is
found.
- The final_answer field must contain only the final answer
extracted from the candidate response, without additional
explanation.
- The notes field must briefly summarize the main reason for
the global risk score.
[Problem Context]
{QUESTION}
[Reference Answer]
{GOLD_ANSWER}
[Candidate Steps From Visible Model Answer]
1. {STEP_1} 2. {STEP_2} ...
{N}. {STEP_N}
[Candidate Final Answer]
{FINAL_ANSWER}
[Judgment JSON]
```

Action Quality Judge

The action quality judge evaluates the correctness of different response strategies and provides calibrated quality estimates for SLM answering, LLM answering, and local revision.

System Prompt

```
You are an impartial evaluator.
Score each candidate final answer from 0 to 1 for
correctness against the reference answer.
Judge answer equivalence, not reasoning style or length.
Output exactly one strict JSON object only, with numeric
values and keys S, L, C_lite.
```

User Prompt

```
[Instruction]
{QUESTION}
[Reference Answer]
{REFERENCE_ANSWER}
[Candidate Final Answer S]
{SMALL_MODEL_FINAL_ANSWER}
[Candidate Final Answer L]
{LARGE_MODEL_FINAL_ANSWER}
[Candidate Final Answer C_lite]
{LOCALLY_REVISED_FINAL_ANSWER}
Return the evaluation results in JSON format:
```