CAPABILITY SELF-ASSESSMENT IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The ability to recognize one's own limitations and decide whether to solve a problem or seek help is fundamental to reliable intelligent systems. Yet modern large language models tend to overestimate their competence and attempt queries they cannot solve. We study how models can learn Capability Self-Assessment (CSA) while preserving their problem-solving ability. Because the same model both assesses and solves queries, self-assessment training can change the capability it must recognize. We systematically compare supervised fine-tuning (SFT) and reinforcement learning (RL), evaluating each trained model's self-assessment against its updated capability and measuring how well it retains its solving ability. Across model families, scales, and domains, RL provides the strongest assessment–retention trade-off. We also find that CSA transfers to the out-of-distribution settings we evaluate, and demonstrate its practical value for local–cloud routing at inference time and targeted data selection during training.

1 INTRODUCTION

An ideal intelligent system not only solves problems well, but also recognizes its limits. It asks for help when a problem exceeds its ability or uses failure to identify what it should learn next, as humans do (Madras et al., 2018). However, large language models (LLMs) often elect to answer queries they cannot solve reliably, leading to hallucinated outputs (Farquhar et al., 2024) and wasted compute (Shirkavand et al., 2025). Prior work has studied whether LLMs can assess answer correctness and recognize unanswerable questions (Kadavath et al., 2022; Yin et al., 2023), and trained models to estimate confidence (Kapoor et al., 2024) or abstain when uncertain (Zhang et al., 2024).

We ask whether a generative model can be post-trained to predict its own query-level success, *while preserving its problem-solving ability*. We use Capability Self-Assessment (CSA) to denote a pre-answer choice between SELF-SOLVE, where the model attempts the query itself, and DELEGATE, where the query is handed to a stronger system. Because the same model both assesses and solves queries, self-assessment training can shift the capability boundary it must recognize. We therefore study how to learn and evaluate model-specific self-assessment policies while accounting for changes in problem-solving ability.

To investigate this question, we compare supervised fine-tuning (SFT) with reinforcement learning with verifiable rewards (RLVR) (Shao et al., 2024; Guo et al., 2025). We evaluate both approaches within a common CSA framework, measuring self-assessment quality alongside changes in the model's original capabilities. Across model families, scales, and domains, we find that SFT can improve self-assessment at the expense of task performance, whereas RLVR can improve CSA while preserving the model's problem-solving ability. We observe transfer across domains and datasets, suggesting that the learned assessment policy remains useful beyond its training distribution.

We then explore two practical uses of this learned decision signal. At inference time, CSA improves local–cloud routing by identifying queries that require a stronger model. During training, CSA can be used to identify examples that may lie near or beyond the model's current competence, providing a targeted signal for data selection. In this way, learned CSA can play a dual role analogous to competence assessment in effective problem solving, guiding when to act or delegate at inference time, and guiding what to learn next during training.

In summary, our contributions are:

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.

Table 1: Comparison with representative capability-assessment and uncertainty-estimation methods. *Trained for assessment* means model parameters are updated specifically for assessment. *Assessment before answering* means the assessment is made without generating any candidate answer. *Same model assesses and solves* allows an auxiliary assessment head on the solving model. *Solving ability evaluated after training* denotes an explicit comparison of solving performance before and after assessment training. ✓ indicates an explicitly studied feature. — indicates that it is not part of the listed setting.

Method	Trained for assessment	Assessment before answering	Same model assesses and solves	Solving ability evaluated after training
Self-consistency (Wang et al., 2023)	—	—	—	—
Semantic entropy (Farquhar et al., 2024)	—	—	—	—
$P(\text{IK})$ (Kadavath et al., 2022)	✓	✓	✓	—
Internal Confidence (Chen et al., 2026)	—	✓	✓	—
Calibration tuning (Kapoor et al., 2024)	✓	—	—	—
Initial self-assessment (Barkan et al., 2026)	—	✓	✓	—
RL-trained confidence (Damani et al., 2026)	✓	—	✓	Partial
Ours (CSA training)	✓	✓	✓	✓

1. We evaluate pre-answer capability assessment jointly with the effects of assessment training on the model. We use both aggregate task performance and query-level capability stability to distinguish assessment gains from changes in the capability being assessed.
2. We systematically compare training-free methods, SFT, and RLVR for acquiring CSA. Our experiments show that RLVR can improve self-assessment while preserving problem-solving ability, with evidence of transfer across domains and datasets.
3. We demonstrate the practical value of CSA for local–cloud routing and targeted data selection, guiding both when to seek help and what to learn next.

2 Related Work

Pre-answer capability prediction. Kadavath et al. (2022) distinguish assessment of a generated answer through P(True) from prediction of whether a model knows an answer through P(IK). Their learned P(IK) predictor and cross-task evaluations are direct precedents for our setting. Chen et al. (2026) introduce a training-free query-level uncertainty signal and demonstrate adaptive inference, including model cascading. Barkan et al. (2026) study predictions of task success and their relationship to overconfidence and decision making. We build on this literature by comparing assessment-training methods while measuring their effects on the model and evaluating predictions against its post-training capability.

Confidence, abstention, and deferral. Confidence estimation, abstention, and learning to defer provide closely related objectives and decision rules (Madras et al., 2018; Kapoor et al., 2024; Zhang et al., 2024; Wen et al., 2025). Sampling-based methods estimate uncertainty from multiple generated outputs (Wang et al., 2023; Manakul et al., 2023; Farquhar et al., 2024; Kossen et al., 2024). Gul et al. (2026) and Damani et al. (2026) use RL to train help-seeking and calibrated confidence, but neither extensively evaluates per-query capability retention.

Our experiments compare these signals and training approaches within the assessment–retention setting. Moreover, unlike much prior work that treats self-knowledge as a safeguard against hallucination, using confidence or abstention to avoid unreliable answers, we explore the model's own capability boundary as a signal for adaptation. The same learned decision determines when the model acts or seeks help at inference time and which examples it learns from during training. Table 1 provides a comparison of our work and existing studies. Appendix A provides a broader discussion.

3 CSA through Training

Our prompt-only study shows that prompting alone does not elicit reliable CSA in the evaluated settings (Appendix D.1). We therefore compare training strategies for improving this assessment. We formulate CSA learning as policy optimization with a capability-preservation goal and evaluate how well different training methods meet it.

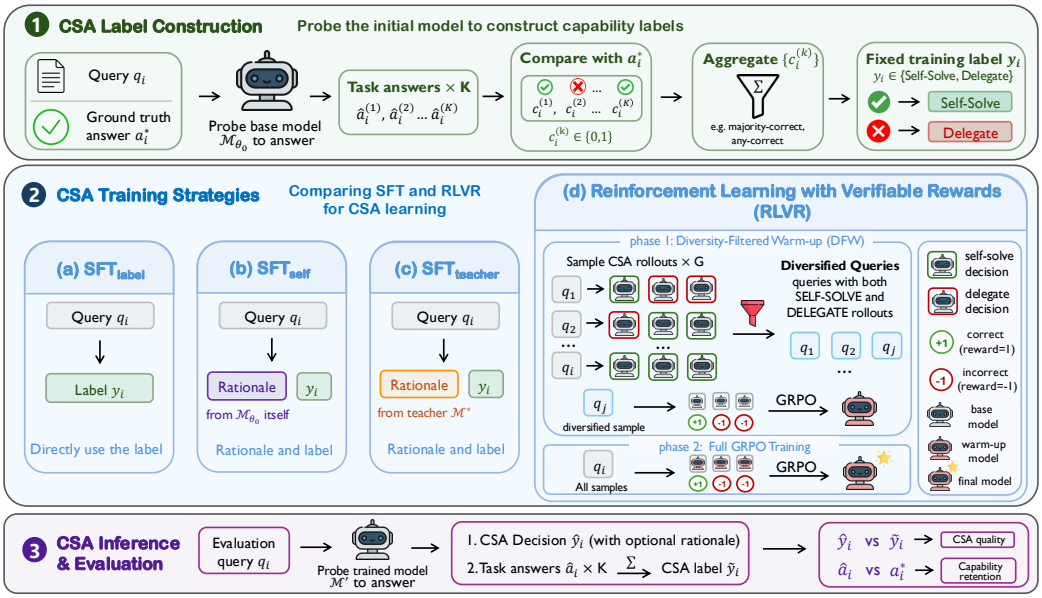


Figure 1: Overview of the CSA training and evaluation framework. **(1) Label construction**: aggregate K solution attempts from the initial model into fixed capability labels y_i. **(2) Training**: learn self-assessment using these labels via three SFT variants (label-only, self-rationale, and teacher-rationale) or RLVR. **(3) Inference and Evaluation**: predict SELF-SOLVE or DELEGATE before answering and evaluate decisions against updated capability labels $\tilde{y}_i$.

3.1 CSA FORMULATION

Given a query q_i, a model makes a binary decision between SELF-SOLVE and DELEGATE before attempting an answer. Let y_i denote its capability label under a specified task and evaluation protocol. The policy π_θ^{CSA} takes $x_i = [p; q_i]$ as input, where p is a CSA instruction, and predicts a decision $\hat{y}_i$ without observing a candidate solution.

We formulate CSA learning as improving this decision policy while preserving the model's underlying problem-solving ability. Let θ_0 denote the initial model parameters, $\Delta_{\text{cap}}(\theta, \theta_0)$ measure capability change under the same evaluation protocol, and $u(\hat{y}_i, y_i)$ measure agreement between the predicted decision $\hat{y}_i$ and the capability label y_i. The learning objective is

$$\theta^\star = \arg\max_\theta \mathbb{E}_{q_i \sim \mathcal{D},\, \hat{y}_i \sim \pi_\theta^{\text{CSA}}(\cdot|x_i)} \left[u(\hat{y}_i, y_i)\right] \quad \text{s.t.} \quad \Delta_{\text{cap}}(\theta, \theta_0) \le \varepsilon, \tag{1}$$

where ε specifies the tolerated capability change. The choice of Δ_{cap} determines whether preservation is assessed through aggregate task performance or finer-grained capability profiles. We use fixed capability labels from the initial model as training targets. Because training may change the capability being assessed, we re-probe each trained model for evaluation and separately compare its solving performance with the initial checkpoint. The preservation constraint expresses the desired learning goal, and our training methods do not explicitly enforce it. Figure 1 summarizes the training and evaluation framework.

3.2 CSA LABEL CONSTRUCTION

We construct training labels by probing the initial model $\mathcal{M}_{\theta_0}$ under a fixed task and evaluation protocol. Given a dataset $\mathcal{D} = \{(q_i, a_i^*)\}_{i=1}^N$, we sample K solution attempts $\{\hat{a}_i^{(k)}\}_{k=1}^K$ for each query. Let $c_i^{(k)} = \text{Correct}(\hat{a}_i^{(k)}, a_i^*) \in \{0, 1\}$ indicate whether an attempt passes the task-specific correctness check. We aggregate these outcomes into a binary capability label:

$$y_i = \begin{cases} \text{SELF-SOLVE}, & \text{if } g\big(\{c_i^{(k)}\}_{k=1}^K\big) = 1, \\ \text{DELEGATE}, & \text{otherwise}, \end{cases} \tag{2}$$

where g is an aggregation rule, such as any-correct or majority-correct. Thus, y_i denotes the initial model's empirical capability label for query q_i under the chosen sampling and aggregation protocol.

We construct these labels once from the initial model and use them across all training strategies. They provide a common training target, but may no longer describe the model after training. We therefore evaluate each trained policy against freshly probed capability labels and separately measure problem-solving retention.

3.3 CSA Training and Evaluation

We compare two post-training paradigms for learning the CSA policy: supervised fine-tuning (SFT) and reinforcement learning with verifiable rewards (RLVR). Both use the same fixed initial capability labels and query inputs. Each input combines a CSA prompt p with the query, $x_i = [p; q_i]$. See Appendix E for all details.

Within SFT, we compare three forms of supervision.

(a) $\text{SFT}_{\text{Label}}$ (Zhang et al., 2024; Kapoor et al., 2024). The target is the bare capability label, $o_i^{\text{label}} = y_i$. The model learns to emit the decision directly.

(b) SFT_{Self} (Kadavath et al., 2022; Chaudhry et al., 2024). The initial model generates a rationale conditioned on the query input and capability label, $r_i^{\text{self}} \sim \mathcal{M}_{\theta_0}(\cdot \mid x_i, y_i)$. The training target is the rationale followed by the label, $o_i^{\text{self}} = [r_i^{\text{self}}; y_i]$.

(c) $\text{SFT}_{\text{Teacher}}$ (Hager et al., 2025). A stronger teacher model generates the rationale under the same conditioning, $r_i^{\text{teacher}} \sim \mathcal{M}^*(\cdot \mid x_i, y_i)$. The training target is $o_i^{\text{teacher}} = [r_i^{\text{teacher}}; y_i]$.

All three SFT variants minimize the same negative log-likelihood objective:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{i=1}^{N} \log p_\theta(o_i \mid x_i), \qquad o_i \in \{o_i^{\text{label}}, o_i^{\text{self}}, o_i^{\text{teacher}}\}, \tag{3}$$

(d) RLVR. RLVR generates a rationale and a decision, $o_i^{(g)} = [r_i^{(g)}; \hat{y}_i^{(g)}]$, and rewards agreement between the decision and the fixed capability label: $R_i^{(g)} = +1$ if $\hat{y}_i^{(g)} = y_i$, and -1 otherwise. The reward evaluates the CSA decision, not a generated task answer. We optimize this policy using Group Relative Policy Optimization (GRPO) (Shao et al., 2024).

Training has two stages. *Diversity-Filtered Warm-up (DFW)* runs GRPO on queries whose initial G rollouts contain both SELF-SOLVE and DELEGATE decisions. *Full GRPO training* then continues from the warm-up checkpoint on the full dataset. Appendices B and D.2.2 provide more details. Both stages use the following objective:

$$\mathcal{L}_{\text{GRPO}}(\theta; \mathcal{B}) = -\,\mathbb{E}_{\substack{x_i \sim \mathcal{B} \\ \{o_i^{(g)}\} \sim \pi_{\theta_{\text{old}}}}} \left[\frac{1}{G} \sum_{g=1}^{G} \min\Big(\rho_i^{(g)}(\theta) A_i^{(g)}, \text{clip}\big(\rho_i^{(g)}(\theta), 1-\epsilon, 1+\epsilon\big) A_i^{(g)} \Big) \right] + \beta\, \text{KL}[\pi_\theta \,\|\, \pi_{\text{ref}}], \tag{4}$$

where $\rho_i^{(g)}(\theta) = \pi_\theta(o_i^{(g)} \mid x_i) / \pi_{\theta_{\text{old}}}(o_i^{(g)} \mid x_i)$, ϵ is the clipping threshold, β is the KL strength, and $A_i^{(g)} = (R_i^{(g)} - \text{mean}_{g'} R_i^{(g')}) / \text{std}_{g'} R_i^{(g')}$, which is the group-normalized advantage within each rollout group. $\mathcal{B}$ is a mini-batch from $\mathcal{D}_{\text{div}}$ (Stage 1) or $\mathcal{D}$ (Stage 2).

Connection to Eq. 1. A Lagrangian relaxation of Eq. 1 produces a regularized objective with a capability penalty $\Delta_{\text{cap}}(\theta, \theta_0)$. RLVR (Eq. 4) uses KL regularization as a heuristic approximation to this penalty. SFT (Eq. 3) maximizes target-output likelihood and can optionally include the same KL penalty; we use vanilla SFT by default, following common practice.

Inference and Evaluation. At inference time, the trained model receives $[p; q]$ and outputs a CSA decision, preceded by a rationale for all methods except $\text{SFT}_{\text{Label}}$. We re-probe each trained model on the evaluation queries using the sampling protocol and aggregation rule in Eq. (2), obtaining updated capability labels $\{\tilde{y}_i\}$. We measure CSA quality by comparing the predicted decisions $\hat{y}_i$ with these labels. Separately, we assess capability retention by comparing task performance before and after training.

Table 2: CSA performance across training strategies and domains (higher is better). **Bold** and underline mark the best and second-best trained models. —: CDS is unavailable due to highly imbalanced decisions.

		Math					Science				
Model	**Metric**	Vanilla	$\text{SFT}_{\text{Label}}$	SFT_{Self}	$\text{SFT}_{\text{Teacher}}$	RLVR	Vanilla	$\text{SFT}_{\text{Label}}$	SFT_{Self}	$\text{SFT}_{\text{Teacher}}$	RLVR
Qwen3-1.7B	M-F1	0.53	0.40	0.68	0.69	**0.80**	0.46	0.49	0.61	0.55	**0.67**
	CDS	5.15	3.41	7.67	7.86	**17.76**	3.02	5.30	6.39	2.75	**10.12**
	CR	100	10.5	91.9	73.1	**98.8**	100	37.3	89.5	81.1	**102.1**
Qwen3-4B	M-F1	0.51	0.41	0.65	0.64	**0.80**	0.42	0.31	0.60	0.64	**0.64**
	CDS	—	5.51	6.14	5.15	**17.17**	—	4.47	5.92	7.38	**8.07**
	CR	100	13.3	95.0	96.4	**100.7**	100	20.7	86.9	87.6	**101.9**
Qwen3-8B	M-F1	0.49	0.76	0.66	0.68	**0.77**	0.45	0.50	0.58	0.62	**0.62**
	CDS	—	11.10	6.08	6.35	**14.13**	—	**10.73**	4.19	6.34	6.72
	CR	100	89.2	98.8	**102.3**	101.4	100	49.1	91.5	94.0	**101.5**
Qwen3-14B	M-F1	0.51	0.73	0.67	0.60	**0.75**	0.45	0.20	**0.61**	0.60	0.57
	CDS	—	9.31	7.05	3.43	**10.94**	—	2.25	**5.57**	5.12	5.25
	CR	100	84.3	98.2	**106.2**	100.9	100	6.7	92.5	95.1	**100.3**
OLMo2-7B	M-F1	0.74	0.30	0.84	0.83	**0.89**	0.55	0.44	0.66	0.65	**0.69**
	CDS	10.87	3.63	19.07	18.26	**26.60**	3.44	0.00	7.83	7.08	**10.94**
	CR	100	2.1	96.6	92.8	**101.9**	100	0.6	88.5	76.8	**106.7**
OLMo2-13B	M-F1	0.67	0.28	0.78	0.79	**0.89**	0.54	0.42	0.61	0.64	**0.66**
	CDS	7.24	0.90	16.06	15.02	**22.73**	6.13	0.00	5.70	6.99	**8.40**
	CR	100	3.0	49.7	72.2	**100.8**	100	5.0	76.4	61.2	**99.6**

4 Experiments and Analysis

4.1 Setup

We conduct experiments on the **Qwen3** (Yang et al., 2025) family (1.7B, 4B, 8B, 14B), as well as **OLMo2** (OLMo et al., 2024) (7B, 13B). Evaluation spans two domains: **Math** (GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2023), and AIME (Veeraboina, 2024)) and **Science** (MMLU-Pro (Wang et al., 2024) across the biology, chemistry, health, and physics categories). We use **Macro F1 (M-F1)** as the primary metric of CSA quality and report **Capability Discrimination Score (CDS)** as an auxiliary metric. We use **Capability Ratio (CR)** to measure average retention of the model's problem-solving ability. Full experimental details appear in Appendix C.

4.2 Main Results

For each query, we instantiate Eq. (2) with $K = 5$ independently sampled task answers. Repeated attempts reduce reliance on a single stochastic outcome. We use *any-correct* for open-ended Math: at least one answer must be correct. For multiple-choice Science, we use *majority-correct*: at least three answers must be correct, since guessing an option is more likely to succeed. These labels define capability under the sampling protocol, rather than single-attempt accuracy. Appendix C.1.4 explains the construction, and Appendix D.2.1 varies K and the aggregation threshold at $K = 5$.

RLVR outperforms SFT in balancing assessment quality and capability retention. Table 2 shows that, across the evaluated models and tasks, RLVR outperforms the SFT baselines in terms of M-F1 in most cases. It also retains approximately 100% of the original task performance on average. Mean CR is 100.8% on Math and 102.0% on Science, compared with only 33.7% and 19.9% for $\text{SFT}_{\text{Label}}$. The rationale-based SFT_{Self} and $\text{SFT}_{\text{Teacher}}$ retain 82.6–90.5% on average across methods and domains. Appendix D.3 illustrates this with training dynamics on Qwen3-4B Science, where RLVR's CSA predictions remain aligned with its current problem-solving ability, whereas $\text{SFT}_{\text{Label}}$ increasingly fits the original labels despite its declining ability.

Furthermore, Table 3 shows that this advantage persists when supervised training is modified to preserve capability, using LoRA, KL regularization, and task mixing. RLVR achieves the highest M-F1 for both model sizes on Math. The results suggest that RLVR's advantage is not solely attributable to the capability loss induced by standard SFT. See Appendix D.4 for more details.

Table 3: RLVR's advantage persists over capability-preserving training baselines on Math. Columns report M-F1 and CR for Qwen3-1.7B and 4B, averaged over three seeds. Full Math and Science results appear in Appendix D.4.

	1.7B		4B	
Method	M-F1	CR	M-F1	CR
Base	0.53	100.0	0.51	100.0
SFT with smaller lr	0.64	72.4	0.67	97.3
LoRA SFT	0.73	99.0	0.62	100.1
KL SFT	0.63	90.6	0.66	97.5
Task-mixed SFT	0.63	94.6	0.75	100.2
DFW-matched SFT	0.69	72.2	0.71	97.4
Query Probe	0.70	100.0	0.71	100.0
RLVR	**0.84**	100.3	**0.85**	102.8

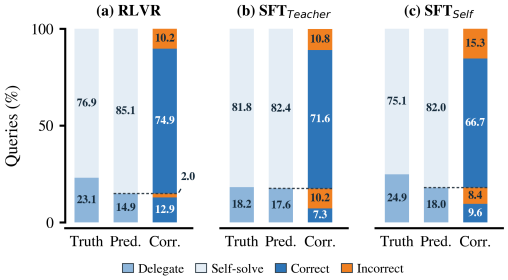


Figure 2: Qwen3-4B on Math: (a) RLVR, (b) $SFT_{Teacher}$, (c) SFT_{Self}. Pred.: predicted; Corr.: correctness. The two regions of Corr. correspond to predicted DELEGATE (bottom) and SELF-SOLVE (top), each split into correct and incorrect decisions.

A closer look at individual decisions helps explain this remaining advantage. SFT can closely match the overall self-solve rate without accurately identifying which queries the model can solve. In Figure 2, $SFT_{Teacher}$ predicts a SELF-SOLVE rate of 82.4%, close to its true rate of 81.8%, yet decision accuracy is only 79.0% for $SFT_{Teacher}$ and 76.3% for SFT_{Self}, compared with 87.8% for RLVR. A policy can match the overall rate yet choose the wrong queries to solve or delegate. RLVR's higher decision accuracy indicates better discrimination between these queries, beyond matching how often to self-solve.

RLVR vs. training-free baselines. In Figure 3, CSA-based RLVR outperforms both training-free query-based methods and methods that require generated answers on Qwen3-4B Math. It achieves M-F1 of 0.80, compared with at most 0.57 for query-based baselines, 0.76 for minimum token probability from one answer, and 0.78 and 0.75 for self-consistency and semantic entropy using five answers. Appendix D.5 provides the full baseline setup and more experimental results. Compared with competitive answer-based methods, CSA avoids generating full task solutions before deciding whether to solve or delegate, using about 151 rationale and decision tokens per query in this setting (Appendix D.6). A decision-first format can further reduce token use by stopping as soon as the SELF-SOLVE or DELEGATE decision is emitted. Appendix D.2.3 compares this format with analysis-first output across three settings and finds similar CSA performance.

4.3 CAPABILITY RETENTION ANALYSIS

Why retention matters. We could assess the solver with a separate adapter or probe, leaving its solving ability untouched. But these methods are less accurate: they score 0.62–0.73 M-F1 on Math and 0.51–0.56 on Science, versus 0.84–0.85 and 0.66–0.67 for RLVR (Tables 3 and 15). LoRA SFT also performs worse than full-parameter SFT in three of four settings, suggesting insufficient capacity for this task. The best assessment in our experiments comes from changing the solver's own weights, so preserving its solving ability matters. This also fits practical efficiency, where one updated checkpoint handles both solving and assessment. See Appendix D.4 for more.

The CR results in Tables 2 and 3 show that RLVR largely preserves aggregate task performance, but this can hide changes in which individual queries the model can solve. We therefore compare query-level capability before and after CSA training across eight mathematics and coding settings. As shown in Figure 4, on average 4.5% of capability labels change, compared with 5.4% disagreement expected from repeated finite-sample measurements of the same capability. Agreement between the pre- and post-training labels remains high ($\kappa = 0.895$), while the mean Capability Ratio is 101.0%. These results suggest that CSA training preserves both aggregate performance and the model's query-level capability boundary within the resolution of our evaluation. Full details are provided in Appendix D.7.

4.4 GENERALIZATION OF CSA

Within-Source Discrimination A Math policy could exploit benchmark source identity as a shortcut instead of assessing its ability to solve each query. To test this explanation, we shuffle decisions within GSM8K, MATH500, and AIME while preserving each source's routing counts and

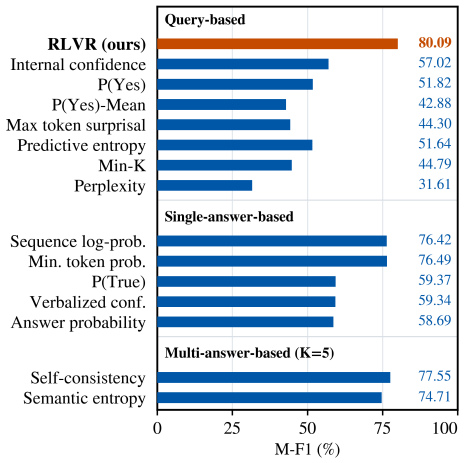


Figure 3: M-F1 on Qwen3-4B Math. **Query-based** methods assess capability from the query without generating a task answer. **Single-answer-based** methods use a single answer and its likelihood or confidence. **Multi-answer-based** methods use agreement or entropy across five sampled answers.

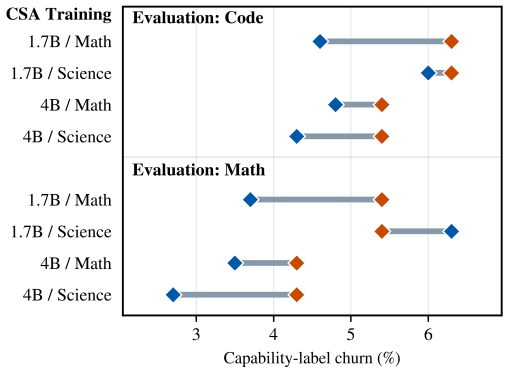


Figure 4: RLVR demonstrates query-level capability stability. Connected diamonds compare observed label churn (blue) with sampling noise (orange). Groups identify the evaluation domain; rows give model size / CSA training domain. Mean churn is 4.5%, versus 5.4% sampling noise.

the model's post-training capability labels. This preserves any advantage from source-level routing but removes the association between individual decisions and query solvability. Figure 5 shows that all six models exceed their null means by 3.44–9.61 M-F1 percentage points. Thus, learned CSA captures query-level differences in solvability within benchmark sources, beyond exploiting benchmark identity. Appendix D.8 gives the experimental protocol and permutation details.

Math–Science Transfer We assess out-of-distribution generalization by evaluating math-trained policies on science problems and science-trained policies on math problems. Table 4 shows that RLVR achieves the highest M-F1 in three of four cross-domain settings, including both transfer directions for Qwen3-1.7B and Science-to-Math transfer for Qwen3-4B. These results show that its assessment advantage extends beyond the training domain.

Generalization to Omni-MATH We further evaluate on Omni-MATH (Gao et al., 2025), a held-out mathematics dataset with annotated difficulty levels. Evaluating within a single benchmark lets us examine whether CSA responds to difficulty without relying on differences between benchmark sources. We apply the Qwen3-4B CSA policy trained on Math. Figure 6 shows that its SELF-SOLVE rate decreases from 90.7% in the easiest interval to 5.8% in the hardest, with a strong linear association between self-solve rate and difficulty ($R^2 = 0.938$). The policy also achieves query-level M-F1 of 0.67. Together, these results show that learned CSA transfers to unseen Math queries. It also adjusts its willingness to solve with difficulty while retaining query-level discrimination.

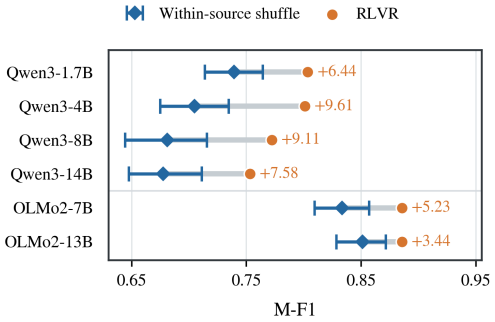


Figure 5: Math M-F1 versus within-source shuffling: 95% null intervals and gains in percentage points.

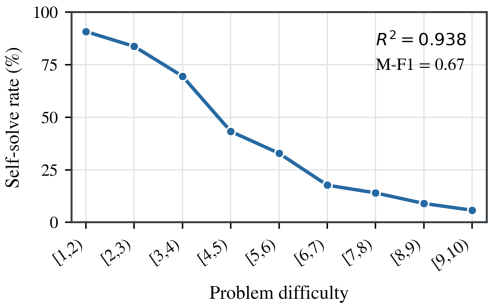


Figure 6: Qwen3-4B-Math on Omni-MATH: self-solve rate decreases with problem difficulty.

Table 4: **RLVR shows better cross-domain transfer between science and math.** Each entry reports M-F1. Models are trained on one domain and evaluated on another (e.g., Math → Science: trained on Math, evaluated on Science).

Model	SFT_{Label}	SFT_{Self}	$SFT_{Teacher}$	RLVR
Science → Math				
Qwen3-1.7B	0.68	0.66	<u>0.73</u>	**0.79**
Qwen3-4B	0.75	0.67	<u>0.75</u>	**0.78**
Math → Science				
Qwen3-1.7B	0.40	<u>0.53</u>	0.51	**0.63**
Qwen3-4B	0.39	**0.53**	<u>0.52</u>	0.51

Table 5: **RLVR shows zero-shot transfer to coding.** Policies trained on mathematics or science are evaluated without further training on 722 Python problems. Pass@1 measures coding ability, SSR the self-solve rate, and M-F1 assessment quality. Results are averaged over three decoding seeds.

Model	CSA training	Pass@1	SSR	M-F1
Qwen3-1.7B	None	0.56	1.00	0.36
	Math	0.58	0.83	0.64
	Science	0.57	0.87	0.58
Qwen3-4B	None	0.68	1.00	0.41
	Math	0.68	0.92	0.57
	Science	0.68	0.89	0.63

Transfer to Coding Table 5 tests a larger distribution shift: policies trained only on mathematics or science are evaluated on 722 Python programming problems. For Qwen3-1.7B, the best transferred policy raises coding-assessment M-F1 from 0.36 to 0.64. For Qwen3-4B, it rises from 0.41 to 0.63. Coding ability is preserved, with pass@1 changing from 0.562 to 0.566–0.578 for the 1.7B model and from 0.675 to 0.676 for the 4B model. Together, these results show that CSA transfers to a previously unseen domain and task format. Full details are provided in Appendix D.9.

4.5 CSA Tracks Model-Specific Capability

The alignment between CSA decisions and problem difficulty does not by itself establish model-specific self-assessment. A policy could learn which problems are generally difficult for language models. We test this alternative explanation on 722 coding problems using OLMoE-1B-7B (Muennighoff et al., 2025) as an independent comparator. For each Qwen policy's own model and OLMoE, we estimate per-problem competence in the same way by sampling $K = 32$ solutions.

Table 6 shows that CSA decisions track the policy model's own competence more strongly than OLMoE's competence. After controlling for OLMoE's pass rate, the partial correlation between the CSA decision and the policy model's own pass rate ranges from 0.224 to 0.322. In the reverse direction, after controlling for the policy model's own pass rate, the partial correlation with OLMoE ranges from only 0.042 to 0.093.

Table 6: **CSA tracks model-specific competence.** Partial correlations $r_{x|y} = r(\mathrm{dec}, x \mid y)$ and M-F1 versus an OLMoE-based rule at matched delegation rates. Pass rates use $K = 32$ attempts per query.

Model	Training	$r_{\mathrm{own}\|\mathrm{OLMoE}}$	$r_{\mathrm{OLMoE}\|\mathrm{own}}$	M-F1 CSA	M-F1 OLMoE rule
Qwen3-1.7B	Math	0.256	0.093	**0.639**	0.576
	Science	0.224	0.067	**0.580**	0.521
Qwen3-4B	Math	0.280	0.042	**0.571**	0.522
	Science	0.322	0.063	**0.629**	0.576

We also construct an OLMoE-based difficulty rule that delegates exactly as many problems as each CSA policy, selecting those with the lowest OLMoE pass rates. CSA achieves 0.05–0.06 higher Macro-F1 than this rule across all four settings, even though CSA observes only the problem text while the OLMoE rule uses 32 executed solutions per problem. Thus, the learned policy captures information about which problems its own model can solve beyond a generic estimate of problem difficulty.

Cross-scale policy swapping provides a complementary test. A policy trained for Qwen3-1.7B nearly matches that solver's aggregate capability, with a coverage–accuracy gap of $+0.003$, whereas attaching a policy trained for the stronger Qwen3-8B model to the 1.7B solver produces an $+0.110$ gap. Across all pairings, policies matched to their solver have a smaller mean absolute coverage–accuracy gap than mismatched policies (0.049 versus 0.072). While these cross-scale results describe aggregate alignment, the query-level comparisons provide more direct evidence of model-specific assessment. Full results are reported in Appendix D.10.

4.6 Applications

4.6.1 Self-Routing

A model with accurate CSA is a natural router in a local–cloud inference system. When the local model chooses to self-solve, the query is answered on-device. When it chooses to delegate, the

Table 7: Math routing to Qwen3-235B-A22B.

Method	Qwen3-1.7B	Qwen3-4B
Routing Accuracy		
Local only	0.60	0.69
Cloud only	0.86	0.86
Vanilla	0.64	0.70
Random	0.65	0.72
CSA	0.73	0.80
Routing Overhead		
CSA cloud calls	96	73
Time (s) / Tokens	0.53 / 304.9	0.39 / 150.7

Table 8: Data selection results on MEDMCQA.

	Qwen3-1.7B			Qwen3-4B		
Method	5%	10%	20%	5%	10%	20%
Eval only	*— 0.40 —*			*— 0.42 —*		
Train on all	*— 0.44 —*			*— 0.48 —*		
Random	0.40	0.45	0.44	0.43	0.43	0.44
Random-band	0.41	0.45	0.44	0.44	0.44	0.41
Dev Similarity	0.40	0.39	0.40	0.43	0.45	0.34
High Loss	0.27	0.26	0.27	0.24	0.24	0.25
Low Loss	0.41	0.43	0.42	0.49	0.51	0.49
CSA-Based	**0.43**	**0.45**	**0.45**	**0.57**	**0.56**	**0.56**

query is routed to a powerful cloud model. The local model serves as its own router, offering a practical cost–accuracy trade-off between local-only and cloud-only inference. Table 7 presents the Math routing results using Qwen3-1.7B and Qwen3-4B as local models and Qwen3-235B-A22B as the cloud model. **Accuracy.** CSA routing reaches 0.73 and 0.80 accuracy for the 1.7B and 4B models, outperforming local-only and vanilla routing and improving over random routing by about 8.2 percentage points at matched cloud-call budgets. **Efficiency.** CSA routing is fast, taking an average of 0.53 seconds per query for Qwen3-1.7B and 0.39 seconds for Qwen3-4B. This enables prompt selection between local answering and cloud delegation with little delay from the routing decision. Appendix D.12.1 provides a detailed illustration and evaluation setup. Appendix D.12.2 further studies single-attempt routing with an expected-outcome reward that accounts for both answer accuracy and delegation cost.

4.6.2 SELF-GUIDED DATA SELECTION

We further test whether CSA can serve as a data-selection signal. Starting from a base model M_0, we train an RL-based CSA model M_{RLVR} on MMLU-Pro (Wang et al., 2024) and apply it to a fresh candidate pool from MedMCQA (Pal et al., 2022). We select examples based on a metric which prioritizes examples that the CSA model is likely to DELEGATE while filtering for cases that are uncertain but not extreme loss outliers, and fine-tune the model on this subset. We then compare against standard selection baselines, including random, loss-based, and dev-similarity selection, to assess whether CSA identifies examples that more effectively improve the model's downstream performance.

Table 8 shows that CSA-based selection generally outperforms the baselines across model sizes and selection budgets. For Qwen3-1.7B, CSA selection improves over the eval-only model at every budget and slightly exceeds training on the full 50K candidate pool when using only 10–20% of the data. The gains are especially pronounced for Qwen3-4B. CSA-selected subsets achieve 55.8–57.4% accuracy, substantially outperforming both the eval-only baseline (42.2%) and training on all examples (47.6%). In contrast, high-loss selection performs poorly across all settings, indicating that simply selecting the hardest examples is not sufficient and can over-emphasize noisy or unlearnable samples. Low-loss selection is the strongest baseline for Qwen3-4B, but remains well below CSA-based selection. These results suggest that the learned CSA signal identifies examples that are not merely difficult or similar to the development set, but are especially useful for improving the model under a limited fine-tuning budget. See Appendix D.12.3 for the full details.

5 CONCLUSION

We studied how LLMs can acquire CSA while retaining the solving ability being assessed. Through a systematic study, we showed RLVR training provides the strongest overall assessment–retention trade-off among the evaluated training strategies. We also demonstrated out-of-distribution generalization of CSA. More broadly, CSA shifts self-knowledge from a safeguard against errors to a signal for adaptation, guiding both when a model acts and what it learns next. We believe CSA is a promising building block for more adaptive, trustworthy, and responsible AI systems. Ultimately, intelligence is not only about solving problems, but also about recognizing the limits of one's own competence, and adapting when needed.

AI USE STATEMENT

In this work, we used generative AI tools to revise writing and to support the retrieval of related work. Writing assistance included improving clarity, grammar, and organization. Retrieval assistance supplemented our manual literature search by suggesting relevant papers and search terms. We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims or assist with proofs, propose or refine hypotheses, design or provide feedback on the research methodology or experiments, implement our methods, translate text, clean or reformat datasets, support qualitative or thematic data analysis, or interpret results. We reviewed all AI-assisted text and independently checked the relevance and bibliographic details of every suggested reference. We take responsibility for the final content, claims, and results of this work.

ETHICS STATEMENT

This work uses only publicly available models and benchmarks and involves no human subjects or personal data.

REPRODUCIBILITY STATEMENT

All models and datasets used in this work are publicly available. Appendix C.1 specifies the data splits (Table 9), sampling settings, capability-label construction, and evaluation metrics; Appendix B gives the complete two-stage GRPO procedure (Algorithm 1); Appendix C.2 lists all training hyperparameters and hardware (Tables 10–11); Appendices C.3, D.5, and D.12 describe the baseline and application setups; and Appendix E provides every prompt template. We will release our code upon publication.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. Automix: Automatically mixing language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=e6WrwIvgzX.

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*, 2024. URL https://arxiv.org/abs/2402.16827.

Anastasios N Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023. URL https://doi.org/10.1561/2200000101.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

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. URL https://arxiv.org/abs/2304.13734.

Casey O. Barkan, Sidney Black, and Oliver Sourbut. Do large language models know what they are capable of? In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=EO6WtJ0q6G.

Peter L. Bartlett and Marten H. Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(59):1823–1840, 2008. URL http://jmlr.org/papers/v9/bartlett08a.html.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009. URL https://doi.org/10.1145/1553374.1553380.

Felix Jedidja Binder, James Chua, Tomek Korbak, Henry Sleight, John Hughes, Robert Long, Ethan Perez, Miles Turpin, and Owain Evans. Looking inward: Language models can learn about themselves by introspection. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=eb5pkwIB5i.

Margarida Campos, António Farinhas, Chrysoula Zerva, Mário AT Figueiredo, and André FT Martins. Conformal prediction for natural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 12:1497–1516, 2024. URL https://arxiv.org/abs/2405.01976.

Iván Vicente Moreno Cencerrado, Arnau Padrés Masdemont, Anton Gonzalvez Hawthorne, David Demitri Africa, and Lorenzo Pacchiardi. No answer needed: Predicting llm answer accuracy from question-only linear probes. *arXiv preprint arXiv:2509.10625*, 2025. URL https://arxiv.org/abs/2509.10625.

Arslan Chaudhry, Sridhar Thiagarajan, and Dilan Gorur. Finetuning language models to emit linguistic expressions of uncertainty, 2024. URL https://arxiv.org/abs/2409.12180.

Lihu Chen, Gerard de Melo, Fabian M. Suchanek, and Gaël Varoquaux. Query-level uncertainty in large language models. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2506.09669.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=cSimKw5p6R. Featured Certification.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

John J Cherian, Isaac Gibbs, and Emmanuel J Candès. Large language model validity via enhanced conformal prediction methods. *Advances in Neural Information Processing Systems*, 37:114812–114842, 2024. URL https://arxiv.org/abs/2406.09714.

Prateek Chhikara. Mind the confidence gap: Overconfidence, calibration, and distractor effects in large language models. *arXiv preprint arXiv:2502.11028*, 2025. URL https://arxiv.org/abs/2502.11028.

C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1):41–46, 1970. URL https://doi.org/10.1109/TIT.1970.1054406.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=dYur3yabMj.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route LLMs with confidence tokens. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=U08mUogGDM.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenfeld, Leshem Choshen, Yoon Kim, and Jacob Andreas. Beyond binary rewards: Training LMs to reason about their uncertainty. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ASQ649zdHm.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=02f3mUtqnM.

Yu Du, Fangyun Wei, and Hongyang Zhang. Anytool: Self-reflective, hierarchical agents for large-scale api calls. *arXiv preprint arXiv:2402.04253*, 2024. URL https://arxiv.org/abs/2402.04253.

Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(53):1605–1641, 2010. URL https://www.jmlr.org/beta/papers/v11/el-yaniv10a.html.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024. URL https://doi.org/10.1038/s41586-024-07421-0.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025. URL https://arxiv.org/abs/2504.11536.

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. URL https://arxiv.org/abs/2410.07985.

Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. In *International conference on machine learning*, pp. 2151–2159. PMLR, 2019. URL https://proceedings.mlr.press/v97/geifman19a.html.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning LLMs on new knowledge encourage hallucinations? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7765–7784, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.444. URL https://aclanthology.org/2024.emnlp-main.444/.

Gemma Team and others. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Isaac Gibbs, John J Cherian, and Emmanuel J Candès. Conformal prediction with conditional guarantees. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 87(4):1100–1126, 2025. URL https://arxiv.org/abs/2305.12616.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007. URL https://sites.stat.washington.edu/people/raftery/Research/PDF/Gneiting2007jasa.pdf.

Google DeepMind. Gemini 3.1 Pro model card. Technical report, Google DeepMind, 2 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/.

Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Tobias Groot and Matias Valdenegro-Toro. Overconfidence is key: Verbalized uncertainty evaluation in large language and vision-language models. In *Proceedings of the 4th Workshop on Trustworthy Natural Language Processing (TrustNLP 2024)*, pp. 145–171, 2024. URL https://arxiv.org/abs/2405.02917.

Yuxian Gu, Li Dong, Hongning Wang, Yaru Hao, Qingxiu Dong, Furu Wei, and Minlie Huang. Data selection via optimal control for language models. *arXiv preprint arXiv:2410.07064*, 2024. URL https://arxiv.org/abs/2410.07064.

Zhengyao Gu and Mark Hopkins. On the evaluation of neural selective prediction methods for natural language processing. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7888–7899, 2023. URL https://aclanthology.org/2023.acl-long.437/.

Mustafa Omer Gul, Claire Cardie, and Tanya Goyal. Mash: Modeling abstention via selective help-seeking, 2026. URL https://arxiv.org/abs/2510.01152.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017. URL https://arxiv.org/abs/1706.04599.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Sophia Hager, David Mueller, Kevin Duh, and Nicholas Andrews. Uncertainty distillation: Teaching language models to express semantic confidence, 2025. URL https://arxiv.org/abs/2503.14749.

Jiatong Han, Neil Band, Muhammed Razzak, Jannik Kossen, Tim GJ Rudner, and Yarin Gal. Simple factuality probes detect hallucinations in long-form natural language generation. *Findings of the Association for Computational Linguistics: EMNLP*, pp. 16209–16226, 2025. URL https://aclanthology.org/2025.findings-emnlp.880.pdf.

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

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025. URL https://arxiv.org/abs/2311.05232.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, et al. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*, 2023. URL https://arxiv.org/abs/2310.03128.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025. URL https://arxiv.org/abs/2403.07974.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023. URL https://doi.org/10.1145/3571730.

Ziwei Ji, Delong Chen, Etsuko Ishii, Samuel Cahyawijaya, Yejin Bang, Bryan Wilie, and Pascale Fung. LLM internal states reveal hallucination risk faced with a query. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 88–104, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.blackboxnlp-1.6. URL https://aclanthology.org/2024.blackboxnlp-1.6/.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022. URL https://arxiv.org/abs/2207.05221.

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. URL https://arxiv.org/abs/2406.08391.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. Semantic entropy probes: Robust and cheap hallucination detection in llms. *arXiv preprint arXiv:2406.15927*, 2024. URL https://arxiv.org/abs/2406.15927.

Bhawesh Kumar, Charlie Lu, Gauri Gupta, Anil Palepu, David Bellamy, Ramesh Raskar, and Andrew Beam. Conformal prediction with large language models for multi-choice question answering. *arXiv preprint arXiv:2305.18404*, 2023. URL https://arxiv.org/abs/2305.18404.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Minjae Lee, Kyungmin Kim, Taesoo Kim, and Sangdon Park. Selective generation for controllable language models. *Advances in Neural Information Processing Systems*, 37:50494–50527, 2024. URL https://arxiv.org/abs/2307.09254.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023. URL https://arxiv.org/abs/2305.01210.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. URL https://arxiv.org/abs/1711.05101.

David Madras, Toniann Pitassi, and Richard Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer, 2018. URL https://arxiv.org/abs/1711.06664.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2002.07650.

Potsawee Manakul, Adian Liusie, and Mark Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 9004–9017, 2023. URL https://arxiv.org/abs/2303.08896.

Sabrina J Mielke, Arthur Szlam, Emily Dinan, and Y-Lan Boureau. Reducing conversational agents' overconfidence through linguistic calibration. *Transactions of the Association for Computational Linguistics*, 10:857–872, 2022. URL https://arxiv.org/abs/2012.14983.

Sören Mindermann, Jan Brauner, Muhammed Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N. Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt, 2022. URL https://arxiv.org/abs/2206.07137.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. In *International Conference on Learning Representations*, volume 2025, pp. 62061–62121, 2025. URL https://arxiv.org/abs/2409.02060.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021. URL https://arxiv.org/abs/2112.09332.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024. URL https://arxiv.org/abs/2501.00656.

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. URL https://arxiv.org/abs/2406.18665.

OpenAI. GPT-5 system card. Technical report, OpenAI, 8 2025. URL https://openai.com/index/gpt-5-system-card/.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pp. 248–260. PMLR, 2022. URL https://proceedings.mlr.press/v174/pal22a.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S Jaakkola, and Regina Barzilay. Conformal language modeling. *arXiv preprint arXiv:2306.10193*, 2023. URL https://arxiv.org/abs/2306.10193.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3505–3506, 2020. URL https://doi.org/10.1145/3394486.3406703.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023. URL https://arxiv.org/abs/2302.04761.

Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(12):371–421, 2008. URL https://jmlr.csail.mit.edu/papers/v9/shafer08a.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7HNRYT4V44.

Reza Shirkavand, Shangqian Gao, Peiran Yu, and Heng Huang. Cost-aware contrastive routing for LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=4Qe2Hga43N.

Aviv Slobodkin, Omer Goldman, Avi Caciularu, Ido Dagan, and Shauli Ravfogel. The curious case of hallucinatory (un) answerability: Finding truths in the hidden states of over-confident large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3607–3625, 2023. URL https://arxiv.org/abs/2310.11877.

Jiayuan Su, Jing Luo, Hongwei Wang, and Lu Cheng. Api is enough: Conformal prediction for large language models without logit-access. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 979–995, 2024. URL https://arxiv.org/abs/2403.01216.

Fengfei Sun, Ningke Li, Kailong Wang, and Lorenz Goette. Large language models are overconfident and amplify human bias. *arXiv preprint arXiv:2505.02151*, 2025. URL https://arxiv.org/abs/2505.02151.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5433–5442, 2023. URL https://arxiv.org/abs/2305.14975.

Hemish Veeraboina. Aime problem set 1983-2024, 2024. URL https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer, 2005. URL https://link.springer.com/book/10.1007/b106715.

Kaishen Wang, Hengrui Gu, Meijun Gao, and Kaixiong Zhou. DAMO: Decoding by accumulating activations momentum for mitigating hallucinations in vision-language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=JUr0YOMvZA.

Kaishen Wang, Tong Zheng, Xuehao Cui, Ruibo Chen, Tianyi Xiong, and Heng Huang. Mitigating factual hallucination in large reasoning models via mixed-mode advantage regularization, 2026. URL https://arxiv.org/abs/2607.05861.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2203.11171.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

Bingbing Wen, Jihan Yao, Shangbin Feng, Chenjun Xu, Yulia Tsvetkov, Bill Howe, and Lucy Lu Wang. Know your limits: A survey of abstention in large language models. *Transactions of the Association for Computational Linguistics*, 13:529–556, 2025. URL https://arxiv.org/abs/2407.18418.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023. URL https://arxiv.org/abs/2305.10429.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. The art of abstention: Selective prediction and error regularization for natural language processing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1040–1051, 2021. URL https://cs.uwaterloo.ca/~y328yu/publication/xin-tyl-21-b/xin-tyl-21-b.pdf.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063*, 2023. URL https://arxiv.org/abs/2306.13063.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*, 2023. URL https://arxiv.org/abs/2305.16504.

An Yang et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Daniel Yang, Yao-Hung Hubert Tsai, and Makoto Yamada. On verbalized confidence scores for llms. *arXiv preprint arXiv:2412.14737*, 2024a. URL https://arxiv.org/abs/2412.14737.

Yuqing Yang, Ethan Chern, Xipeng Qiu, Graham Neubig, and Pengfei Liu. Alignment for honesty. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 63565–63598. Curran Associates, Inc., 2024b. doi: 10.52202/079017-2030. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/7428e6db752171d6b832c53b2ed297ab-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022. URL https://arxiv.org/abs/2210.03629.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuan-Jing Huang. Do large language models know what they don't know? In *Findings of the association for Computational Linguistics: ACL 2023*, pp. 8653–8665, 2023. URL https://aclanthology.org/2023.findings-acl.551/.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Hanning Zhang, Shizhe Diao, Yong Lin, Yi Fung, Qing Lian, Xingyao Wang, Yangyi Chen, Heng Ji, and Tong Zhang. R-tuning: Instructing large language models to say 'i don't know'. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7113–7139, 2024. URL https://arxiv.org/abs/2311.09677.

Liu Ziyin, Zhikang Wang, Paul Pu Liang, Ruslan Salakhutdinov, Louis-Philippe Morency, and Masahito Ueda. Deep gamblers: Learning to abstain with portfolio theory. *Advances in Neural Information Processing Systems*, 32, 2019. URL https://arxiv.org/abs/1907.00208.

Appendix Contents

A Related Work . . . 19
A.1 Self-Evaluation and Calibration of Correctness in LLMs . . . 19
A.2 Selective Prediction, Abstention, and Risk-Coverage Evaluation . . . 19
A.3 Hallucination Detection . . . 20
A.4 Conformal Prediction and Statistical Risk Control for LLMs . . . 20
A.5 Learning Decision Policies with RL . . . 21
A.6 Data Selection and Curriculum Learning . . . 21
A.7 Relation to Adjacent Formulations . . . 21

B RLVR Training Algorithm . . . 21

C Experimental Setup . . . 22
C.1 Main Experimental Setup . . . 22
C.1.1 Models . . . 22
C.1.2 Datasets . . . 22
C.1.3 CSA Inference Setup . . . 23
C.1.4 Label Construction . . . 23
C.1.5 Evaluation Metrics . . . 24
C.2 Training Settings . . . 25
C.3 Capability-Preserving Training Baseline Setup . . . 25

D Additional Results . . . 27
D.1 Prompt-Only CSA Study . . . 27
D.2 Ablation Studies . . . 28
D.2.1 Capability-Label Construction . . . 28
D.2.2 Diversity-Filtered Warm-up . . . 28
D.2.3 CSA Output Order . . . 29
D.3 Training Dynamics of SFT and RLVR . . . 29
D.4 Capability-Preserving Training Baselines . . . 29
D.5 Training-Free Baselines . . . 31
D.6 CSA Inference Efficiency . . . 33
D.7 Query-Level Capability Stability . . . 33
D.8 Within-Source Discrimination on Math . . . 37
D.9 Zero-Shot Transfer to Coding . . . 37
D.10 CSA Is Model Specific . . . 39
D.11 Case Studies . . . 40
D.12 Applications of CSA . . . 40
D.12.1 Self-Routing . . . 40
D.12.2 Cost-Aware Routing . . . 42
D.12.3 Self-Guided Data Selection . . . 43

E Prompt Templates . . . 46
E.1 CSA Prompt Template . . . 46
E.2 Self-Rationale Prompt (SFT_{Self}) . . . 46
E.3 Teacher-Rationale Prompt ($\text{SFT}_{\text{Teacher}}$) . . . 47

A Related Work

Our work draws on research on self-evaluation and calibration, selective prediction and abstention, uncertainty estimation, learning to defer, and decision-making with external tools. These areas have shown that LLMs can expose useful signals about correctness and answerability and can use such signals to abstain or seek assistance. We focus on a particular coupling within this broader literature: the same generative model is trained to assess its own query-level success before answering, even though this training may alter the capability being assessed. We therefore study self-assessment quality together with capability retention and evaluate whether the resulting behavior transfers beyond its training distribution.

A.1 Self-Evaluation and Calibration of Correctness in LLMs

Kadavath et al. (2022) study whether models can estimate the correctness of their own outputs through quantities such as P(True) and P(IK). That work shows that larger models can exhibit useful self-evaluation signals when queried in the right format, and it helped establish a widely used framing of LLM self-knowledge as a calibration (Gneiting & Raftery, 2007; Guo et al., 2017) problem. Follow-up work has stressed both the promise and limits of this perspective. Yin et al. (2023) ask whether LLMs know what they do not know, focusing on unanswerable questions and explicit uncertainty recognition, while Kapoor et al. (2024) argue that prompting alone is not enough and show that uncertainty estimation benefits substantially from explicit fine-tuning. Zhang et al. (2024) show models can be instruction-tuned to say I don't know. Barkan et al. (2026) study whether LLMs can predict their own task success and find all tested LLMs are overconfident. Damani et al. (2026) jointly train for answer correctness and calibrated confidence. Together, these studies show that uncertainty awareness is partly learnable and can be expressed through scores, verbal judgments, or abstention behavior.

A large practical literature studies how to extract usable confidence from aligned systems (Chuang et al., 2025). Tian et al. (2023) show that for RLHF-tuned models, verbalized confidences (numbers or linguistic markers produced as tokens) can be more calibrated than raw conditional probabilities on several QA benchmarks, and they catalog prompting/temperature-scaling strategies for eliciting confidence. Mielke et al. (2022) propose linguistic calibration for dialogue agents to better align the style of uncertainty expression with correctness, highlighting that calibration is not only numeric but also communicative.

A parallel line scrutinizes verbalized uncertainty and its failure modes. Sun et al. (2025) find five LLMs overestimate the probability that their answer is correct by 20–60% on reasoning problems. Chhikara (2025) analyze overconfidence as mismatch between predicted confidence and true correctness across QA datasets. Xiong et al. (2023) show that LLMs can be systematically overconfident when asked to verbalize confidence, though calibration and error prediction improve with scale and with specific elicitation strategies (e.g., consistency-based approaches). Groot & Valdenegro-Toro (2024) further evaluate verbalized uncertainty across LLMs/VLMs and emphasize overconfidence as a recurring concern. Yang et al. (2024a) synthesize why empirical findings about verbalized confidence disagree across papers (prompting styles, evaluation setups), reinforcing the need for decision-centric evaluation rather than relying on any single elicitation recipe.

These methods provide natural signals and baselines for CSA, since confidence estimates can be converted into answer-or-abstain decisions through thresholding. Our focus is therefore not a categorical distinction between scores and policies, but the shared-parameter setting in which the same model learns a pre-answer, model-specific decision while retaining the problem-solving ability being assessed. Cross-domain evaluation tests whether this behavior extends beyond its training distribution.

A.2 Selective Prediction, Abstention, and Risk-Coverage Evaluation

The classical foundation for abstention (Ziyin et al., 2019) is the reject option, often traced to Chow (1970), who derives optimal decision rules trading misclassification error against rejection cost. Modern selective prediction (El-Yaniv & Wiener, 2010) formalizes this as optimizing a risk–coverage trade-off—achieving low error on the subset of instances the model elects to answer.

In deep learning, Geifman & El-Yaniv (2019) treat selective classification as choosing a subset to minimize selective risk subject to a coverage constraint. The broader "learning with rejection" literature develops surrogate losses and theoretical analyses for reject option learning (Bartlett & Wegkamp, 2008).

Within NLP specifically, Xin et al. (2021) revisit selective prediction for language tasks, comparing confidence estimators and introducing regularization tricks to improve confidence without heavy compute. Gu & Hopkins (2023) provide a methodological audit of selective prediction evaluation in NLP and propose additional metrics (e.g., refinement) to separate the quality of confidence scores from downstream accuracy.

For LLMs, a current synthesis is Wen et al. (2025), which organizes abstention methods by lifecycle stage (pretraining, alignment (Yang et al., 2024b), inference), and emphasizes that abstention must be evaluated not only technically but also relative to user values and deployment context.

A.3 Hallucination Detection

Hallucination detection addresses related failures in which a model produces unsupported or incorrect content, sometimes with unwarranted confidence (Ji et al., 2023; Huang et al., 2025; Wang et al., 2026; 2025; Ji et al., 2024; Gekhman et al., 2024). A prominent black-box detection approach is SelfCheckGPT (Manakul et al., 2023) which detects hallucinated sentences by sampling multiple outputs and measuring disagreement/contradiction—motivated by the intuition that knowledgeable generations are more self-consistent. A complementary statistically grounded approach is semantic entropy: Farquhar et al. (2024) propose uncertainty estimators at the level of meaning, showing that entropy over semantic equivalence classes can detect a subset of hallucinations (confabulations) and generalize to unseen questions. Kossen et al. (2024) then introduce semantic entropy probes designed to make such uncertainty signals cheaper while retaining detection quality.

Internal representations can carry correctness/answerability signals even when outputs are overconfident (Binder et al., 2025). Azaria & Mitchell (2023) show that linear classifiers on hidden states can predict statement truthfulness better than sequence probability alone, suggesting that internal representations contain information relevant to output reliability. Slobodkin et al. (2023) focus on unanswerability: they find indications that models encode answerability, even when they still generate hallucinated answers for unanswerable questions. Newer probe-based hallucination detectors (Han et al., 2025) push this toward practical deployment (e.g., linear probes for long-form generation) while reducing the cost relative to multi-sampling. A related direction (Cencerrado et al., 2025) predicts answer correctness from question-only activations, indicating that some correctness signals may be present before decoding the full answer—useful for early deferral/abstention.

A.4 Conformal Prediction and Statistical Risk Control for LLMs

Conformal prediction (CP) methods usually provide post-hoc risk control (thresholding, filtering, prediction sets) rather than directly modifying the LLM's parameters. CP provides distribution-free coverage guarantees under exchangeability assumptions (Angelopoulos & Bates, 2023; Shafer & Vovk, 2008; Vovk et al., 2005). Because exact conditional coverage is generally impossible in finite samples, modern work develops controlled relaxations and conditional targets; Gibbs et al. (2025) formalize a spectrum between marginal and conditional validity that is useful when risk differs by topic or subgroup.

Several recent works adapt CP to LMs and text. Quach et al. (2023) propose Conformal Language Modeling, producing sets/subsets of outputs (or filtered segments of text) with validity guarantees for language model generations. Kumar et al. (2023) apply conformal prediction to multiple-choice QA with LLMs, arguing CP produces useful uncertainty sets correlated with accuracy. Because many deployed LLMs are API-only, Su et al. (2024) propose CP methods that do not require access to logits, improving applicability in closed models.

For open-ended generation (Campos et al., 2024), correctness-labeling and scoring functions become the bottleneck: CP guarantees depend on a nonconformity score that must detect errors in generated text. Lee et al. (2024) address this by using textual entailment to define correctness of generated sequences and propose selective generation algorithms controlling false discovery rates (FDR) over entailment relations. Cherian et al. (2024) propose enhanced conformal methods for validity

guarantees on LLM outputs, explicitly noting issues of topic-dependent reliability and the utility loss from overly aggressive filtering; they propose conditional-conformal-inspired adaptations and methods to improve the scoring function.

A.5 Learning Decision Policies with RL

RL with verifiable rewards can optimize terminal decisions while allowing intermediate reasoning to emerge from interaction with the reward signal (Guo et al., 2025; Shao et al., 2024). A broader literature studies tool use and help-seeking through prompting, self-supervised training, supervised fine-tuning, and reinforcement learning (Yao et al., 2022; Schick et al., 2023; Nakano et al., 2021; Huang et al., 2023; Xu et al., 2023; Feng et al., 2025; Du et al., 2024; Gul et al., 2026). These methods typically learn whether an external resource improves downstream task utility. Our setting instead isolates prediction of the assessed model's own success and evaluates how learning this decision affects the model's underlying problem-solving ability. We use RL as one approach to optimizing this decision and compare it directly with supervised alternatives.

A.6 Data Selection and Curriculum Learning

Albalak et al. (2024) survey data selection for language models across goals such as quality, diversity, difficulty, and efficiency, and highlight that naively training on all available data may be suboptimal or infeasible (Mindermann et al., 2022). A prominent practical method is DoReMi (Xie et al., 2023), which optimizes domain mixture weights via distributionally robust optimization on a proxy model and then scales training on a reweighted mixture, improving efficiency and average downstream performance. More recently, data selection has been formulated as an optimal control problem over training dynamics (Gu et al., 2024), providing language to treat data scheduling as a sequential decision problem. Classical curriculum learning shows that organizing examples by difficulty can improve optimization and generalization (Bengio et al., 2009).

Our application uses CSA scores as a model-specific signal for data selection. Examples receiving stronger delegation scores are likely to be difficult for the current model, while loss-based filtering removes examples that are already too easy or appear excessively difficult or noisy. This connects CSA to difficulty-aware data selection.

A.7 Relation to Adjacent Formulations

Our distinction from related work concerns the joint setting in which the target is the assessed model's own success and self-assessment training may alter the capability being assessed. Moreover, unlike much prior work that treats self-knowledge as a safeguard against hallucination, using confidence or abstention to avoid unreliable answers, we explore the model's own capability boundary as a signal for adaptation. The same learned decision determines when the model acts or seeks help at inference time and which examples it learns from during training.

B RLVR Training Algorithm

We provide details and pseudocode for the two-stage GRPO procedure summarized in Section 3.3. The binary reward is $+1$ when a rollout predicts the ground-truth CSA label and -1 otherwise. Prompted base models often predict SELF-SOLVE on nearly every query (Appendix D.1). Applying GRPO directly can therefore yield rollout groups with a single label, vanishing reward variance, and no useful policy-gradient signal.

Stage 1: Diversity-Filtered Warm-up (DFW). We construct a fixed diversified subset $\mathcal{D}_{\text{div}} \subseteq \mathcal{D}$ by retaining only queries whose G rollouts from the initial policy π_{θ_0} contain both SELF-SOLVE and DELEGATE. Under our decision reward, these screening groups contain both reward values. We then run T_{warm} GRPO steps on fresh rollouts from $\mathcal{D}_{\text{div}}$, concentrating early training on queries that exhibited decision diversity and producing checkpoint θ_{warm}.

DFW is related to DAPO's dynamic sampling (Yu et al., 2025), which filters uniformly rewarded rollout groups while constructing training batches. Both use within-query reward variation to identify

useful training opportunities. DFW instead selects a fixed query subset using the initial policy, trains on fresh rollouts from that subset for a limited warm-up period, and then returns to the full dataset.

Stage 2: Full GRPO Training. Starting from θ_{warm}, we train for another T_{full} steps on the full dataset $\mathcal{D}$, where the now-informative advantages drive learning across both solvable and unsolvable queries. Algorithm 1 gives the complete procedure.

Algorithm 1 Two-Stage GRPO with Diversity-Filtered Warm-up.

Require: Dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, initial policy π_{θ_0}, reference policy π_{ref}, group size G, warm-up steps T_{warm}, full-phase steps T_{full}
Ensure: Trained policy $\pi_{\theta_{\text{final}}}$

// Stage 1: Diversity-Filtered Warm-up (DFW)
$\mathcal{D}_{\text{div}} \leftarrow \{x_i \in \mathcal{D} : G$ rollouts from π_{θ_0} yield both SELF-SOLVE and DELEGATE$\}$ ▷ filter on initial policy
$\theta_{\text{warm}} \leftarrow$ TRAIN-GRPO$(\theta_0, \mathcal{D}_{\text{div}}, T_{\text{warm}})$

// Stage 2: Full GRPO Training
$\theta_{\text{final}} \leftarrow$ TRAIN-GRPO$(\theta_{\text{warm}}, \mathcal{D}, T_{\text{full}})$ ▷ continue on full data
return $\pi_{\theta_{\text{final}}}$

procedure TRAIN-GRPO$(\theta_{\text{init}}, \mathcal{B}_{\text{src}}, T)$
  $\theta \leftarrow \theta_{\text{init}}$
  for $t = 1, \ldots, T$ **do**
    $\theta_{\text{old}} \leftarrow \theta$
    Sample mini-batch $\mathcal{B} \subset \mathcal{B}_{\text{src}}$
    for each $x_i \in \mathcal{B}$ **do**
      Draw G candidates $\{o_i^{(g)}\}_{g=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x_i)$, each with predicted label $\hat{y}_i^{(g)}$
      Compute reward $R_i^{(g)} = +1$ if $\hat{y}_i^{(g)} = y_i$, else -1
      Compute advantage $A_i^{(g)} = \big(R_i^{(g)} - \text{mean}_{g'} R_i^{(g')}\big)/\text{std}_{g'} R_i^{(g')}$
    end for
    Update θ by minimizing $\mathcal{L}_{\text{GRPO}}(\theta; \mathcal{B})$ ▷ Eq. (4)
  end for
  return θ
end procedure

C Experimental Setup

C.1 Main Experimental Setup

C.1.1 Models

Our prompt-only CSA evaluation (Appendix D.1) spans the Qwen3 (Yang et al., 2025), Gemma3 (Gemma Team and others, 2025), Llama3 (Grattafiori et al., 2024), OLMo2 (OLMo et al., 2024), GPT-5 (OpenAI, 2025), and Gemini 3.1 (Google DeepMind, 2026) families, covering both open-weight and frontier closed-weight models.

For CSA through training experiments, we experiment with two model families. From the Qwen3 family (Yang et al., 2025), we use four scales: 1.7B, 4B, 8B, and 14B parameters, to study how CSA varies with model capacity within a single family. We additionally include OLMo2 (OLMo et al., 2024) at 7B and 13B parameters to assess generalization across model families. For the SFT Teacher strategy, we use Qwen3-235B-A22B as the teacher model.

C.1.2 Datasets

We evaluate on two domains: **Math** and **Science**. The prompt-only study in Appendix D.1 uses the Math test set. Our main results in Section 4 cover both domains.

Math. We aggregate problems from three sources spanning a wide difficulty spectrum: GSM8K (Cobbe et al., 2021) (grade-school arithmetic word problems), MATH500 (Lightman et al.,

2023) (competition-style problems across algebra, geometry, number theory, etc.), and AIME (Veeraboina, 2024) (American Invitational Mathematics Examination problems from 1983 to 2024). All problems have open-ended numerical or symbolic answers. Table 9a summarizes the size of each source and the train/test split used in our experiments.

Science. We use multiple-choice questions from MMLU-Pro (Wang et al., 2024), restricted to four science-relevant categories: `biology`, `chemistry`, `health`, and `physics`. Each question has about 10 answer options, and we feed the question together with all options as the query to the model. Table 9b reports the per-category counts and the overall train/test split.

Table 9: Dataset composition.

(a) Math

Source	#Train	#Test	#Total
GSM8K	1000	300	1300
MATH500	400	100	500
AIME	860	90	950
Total	**2260**	**490**	**2750**

(b) Science

Category	#Train	#Test	#Total
Biology	590	127	717
Chemistry	932	200	1132
Health	674	144	818
Physics	1070	229	1299
Total	**3266**	**700**	**3966**

C.1.3 CSA Inference Setup

We use vLLM (Kwon et al., 2023) for inference across all evaluated models. Sampling parameters are kept consistent across models:

- Temperature: 1.0
- Top-p: 1.0
- Top-k: -1
- Number of samples per query: 1

For closed-weight models (GPT-5, Gemini 3.1), we query the official APIs provided by OpenAI and Google.

C.1.4 Label Construction

A single sampled answer can succeed or fail because of decoding variability. We therefore sample K independent task solutions and aggregate their correctness into a binary capability label. Repeated attempts provide evidence about the model's ability under a specified sampling protocol, rather than treating one failed generation as proof that the query is unsolvable. We use $K = 5$ to obtain this evidence at a moderate evaluation cost. Increasing K also changes the meaning of an any-correct label, so K is part of the target definition rather than only a variance-reduction parameter. Appendix D.2.1 examines other values of K and aggregation thresholds.

Let $c_i^{(k)} \in \{0, 1\}$ indicate whether attempt k answers query i correctly. We assign SELF-SOLVE when the relevant condition below holds, and DELEGATE otherwise.

Math: any-correct. For open-ended Math, the condition is $\sum_{k=1}^{K} c_i^{(k)} \geq 1$: at least one of the five attempts must be correct. Guessing an exact open-ended answer is generally less likely than guessing a multiple-choice option. A correct attempt therefore provides useful evidence that the model can solve the problem, even if its sampled solutions are not consistently successful. This label measures success within K attempts, not reliable success on a single attempt. If the single-attempt success probability is p, independent sampling gives an any-correct probability of $1 - (1 - p)^K$.

Science: majority-correct. For multiple-choice Science, we require $\sum_{k=1}^{K} c_i^{(k)} > K/2$, or at least three correct answers out of five. A single correct option can arise by guessing, so a majority provides stronger evidence of consistent success. We independently shuffle the option order for each attempt

to reduce reliance on option position. Neither majority voting nor option shuffling eliminates all chance successes; the rule makes the target more stringent than any-correct for this answer format.

Training labels are constructed from the initial model and held fixed. For evaluation, we repeat the same procedure with each trained model to assess decisions against its current capability. The single-attempt routing objective is discussed separately in Appendix D.12.2.

C.1.5 EVALUATION METRICS

Macro F1 (M-F1). Our primary assessment metric, M-F1, is the unweighted mean of the per-class F1 scores for the SELF-SOLVE and DELEGATE classes,

$$M - F1 = \tfrac{1}{2}\left(\mathrm{F1}_{\text{SELF-SOLVE}} + \mathrm{F1}_{\text{DELEGATE}}\right), \tag{5}$$

where each per-class F1 is the harmonic mean of precision and recall on that class:

$$\mathrm{F1}_c = \frac{2\,\mathrm{Prec}_c \cdot \mathrm{Rec}_c}{\mathrm{Prec}_c + \mathrm{Rec}_c}, \qquad c \in \{\text{SELF-SOLVE}, \text{DELEGATE}\}. \tag{6}$$

The two classes receive equal weight, so performance on both contributes to the score. Class prevalence and errors on a rare class can still affect M-F1.

Example: on a test set of 100 queries with 60 true SELF-SOLVE and 40 true DELEGATE, suppose the model predicts SELF-SOLVE on 70 queries (50 correct, 20 false) and DELEGATE on 30 queries (20 correct, 10 false). Then $\mathrm{Prec}_{\text{SELF-SOLVE}} = 50/70 \approx 0.71$, $\mathrm{Rec}_{\text{SELF-SOLVE}} = 50/60 \approx 0.83$, giving $\mathrm{F1}_{\text{SELF-SOLVE}} \approx 0.77$. Similarly, $\mathrm{Prec}_{\text{DELEGATE}} = 20/30 \approx 0.67$, $\mathrm{Rec}_{\text{DELEGATE}} = 20/40 = 0.50$, giving $\mathrm{F1}_{\text{DELEGATE}} \approx 0.57$. Thus $M - F1 = \frac{1}{2}(0.77 + 0.57) \approx 0.67$.

Capability Discrimination Score (CDS). We report CDS as an auxiliary measure of the separation in capability-label proportions between queries assigned SELF-SOLVE and those assigned DELEGATE. Concretely, we partition the test set by the model's predicted label and let p_S and p_D denote the fractions of queries labeled solvable within the SELF-SOLVE and DELEGATE groups, respectively, with group sizes n_S and n_D. These fractions use the same $K = 5$ capability labels as M-F1. CDS is the (unpooled) two-proportion z-statistic

$$CDS = \frac{p_S - p_D}{\sqrt{\dfrac{p_S(1-p_S)}{n_S} + \dfrac{p_D(1-p_D)}{n_D}}}, \tag{7}$$

which normalizes the difference in solvable-label proportions by its estimated standard error. Its magnitude depends on sample size and group balance, and it can be unstable when a group is small or its accuracy is near zero or one. We therefore use M-F1 for our main assessment comparisons and do not interpret CDS as a comparable effect size across datasets.

Example: suppose the policy assigns $n_S = 200$ queries to SELF-SOLVE and $n_D = 100$ to DELEGATE. Under the $K = 5$ capability-label protocol, 170 queries in the first group and 20 in the second are labeled solvable. Thus $p_S = 170/200 = 0.85$ and $p_D = 20/100 = 0.20$, giving

$$CDS = \frac{0.85 - 0.20}{\sqrt{\dfrac{0.85 \cdot 0.15}{200} + \dfrac{0.20 \cdot 0.80}{100}}} \approx 13.7.$$

Capability Ratio (CR). To verify that CSA training preserves the model's original capabilities, we report CR, defined as the percentage of the original model's problem-solving accuracy retained after CSA training:

$$CR = \frac{\mathrm{Acc}_{\text{post}}}{\mathrm{Acc}_{\text{pre}}} \times 100\%, \tag{8}$$

where $\mathrm{Acc}_{\text{pre}}$ and $\mathrm{Acc}_{\text{post}}$ denote the model's problem-solving accuracy before and after CSA training, respectively. To reduce variance from stochastic decoding, both $\mathrm{Acc}_{\text{pre}}$ and $\mathrm{Acc}_{\text{post}}$ are averaged over 5 independent runs. We compute ratios and aggregate statistics from unrounded values and round only for presentation.

Example: if the model solves 60% of the test queries before CSA training ($\mathrm{Acc}_{\text{pre}} = 0.60$) and 57% after ($\mathrm{Acc}_{\text{post}} = 0.57$), then $CR = 0.57/0.60 \times 100\% = 95\%$.

C.2 Training Settings

Table 10 reports the hyperparameters for both SFT and GRPO. SFT and GRPO both use full-parameter updates, implemented with the TRL library[1] on top of DeepSpeed ZeRO-3 (Rasley et al., 2020).

Table 10: Training hyperparameters for SFT and GRPO. Both use full-parameter updates.

Hyperparameter	SFT	GRPO
Optimization		
Optimizer	AdamW	AdamW
Learning rate	1×10^{-5}	1×10^{-6}
LR schedule	cosine	cosine
Warmup ratio	0.1	0.1
Weight decay	0.01	0.01
Epochs	1 / 3[2]	1 / 3[2]
Per-device batch size	8	16
Gradient accumulation	1	1
Precision	bf16	bf16
GRPO-specific		
Rollouts per prompt (G)	—	16
KL coefficient (β)	—	0.01
Clipping threshold (ϵ)	—	0.2
Sampling temperature	—	1.0
Top-p	—	1.0
Rollout backend	—	vLLM
Training framework		
Framework	TRL	TRL
Distributed strategy	DeepSpeed ZeRO-3	DeepSpeed ZeRO-3
CPU offload	False	False

DFW Hyperparameters. For DFW training, we use the same hyperparameter setup as in Table 10, except for the number of training steps, which we report in Table 11 for each model. Because the size of the diversified subset varies across models, the training schedule differs accordingly: smaller models yield larger subsets and are typically early-stopped before a full epoch, while larger models, with smaller subsets, are trained for 1–2 epochs. This keeps the total number of training steps roughly comparable across model sizes.

Training Pipeline across Model Families. Another point worth noting is the difference in training pipelines across model families. For Qwen3 models, we follow the full two-stage RLVR pipeline described in Section 3.3. For OLMo2 models, we omit the DFW stage, since they already produce sufficiently diverse responses during rollout as shown in Table 11. This is consistent with Figure 7 in Appendix D.1: while Qwen3 models exhibit SELF-SOLVE rates close to 100%, OLMo2 models show noticeably lower SELF-SOLVE rates, indicating greater rollout diversity without explicit intervention.

Hardware. All main experiments are run on NVIDIA B200 GPUs. SFT runs use a single B200, except for Qwen3-14B and OLMo2-13B, which use 2 $\times$ B200s. All GRPO runs use 4 $\times$ B200s.

C.3 Capability-Preserving Training Baseline Setup

This experiment extends the main training comparison in Table 2 by testing whether RLVR's advantage persists when supervised training better preserves capability or uses the same example-selection curriculum. Table 12 summarizes the controls; results appear in Table 3 and Appendix D.4. This is a separate controlled comparison on two model sizes, not a rerun of every cell in Table 2.

[1] `https://github.com/huggingface/trl`

[2] We train for 1 epoch on Math and 3 epochs on Science.

Table 11: Subset sizes and training steps for DFW across different models on Math and Science, where *Subset Size* denotes the number of samples selected by DFW from the full training set.

	Math		Science	
Model	Subset Size	Training Steps	Subset Size	Training Steps
Qwen3-1.7B	544	150	1781	100
Qwen3-4B	183	46	111	150
Qwen3-8B	165	42	302	152
Qwen3-14B	148	57	171	129
OLMo2-7B	1832	—	3131	—
OLMo2-13B	2035	—	2977	—

Table 12: Training changes relative to the original $\text{SFT}_{\text{Teacher}}$ setup. Supervised variants retain teacher-rationale supervision and the same capability labels unless stated otherwise.

Method	**Changes relative to $\text{SFT}_{\text{Teacher}}$**
SFT with smaller lr	LR: 10^{-5} → best of $\{10^{-5}, 10^{-6}\}$, including RLVR's LR.
LoRA SFT	Full-parameter updates → rank-16 LoRA; LR: best of $\{10^{-4}, 10^{-6}\}$.
KL SFT	SFT loss → SFT loss + forward-KL penalty to the initial model (weight 0.05).
Task-mixed SFT	CSA-only supervision → CSA + task-solving rehearsal, mixed 1:1.
DFW-matched SFT	Full-data training → RLVR's diversified-subset curriculum followed by full-data training.
Query Probe	Generative SFT → a classifier on frozen query representations; no solver updates.

C.3.1 Models and data

We compare capability self-assessment methods using Qwen3-1.7B and Qwen3-4B, training separate models for math and science. The math dataset contains 2,260 training and 490 test questions from GSM8K, MATH500, and AIME. The science dataset contains 3,266 training and 700 test questions from MMLU-Pro biology, chemistry, health, and physics. For each model and domain, we reserve 300 training questions for validation using a split stratified by the initial model's correctness label.

We obtain supervision by sampling five task answers per question from the initial model. A math question is labeled solvable if any sample is correct, while a science question is labeled solvable if a majority of five option-shuffled attempts are correct. Solvable and unsolvable questions receive `SELF_SOLVE` and `DELEGATE` labels, respectively. For SFT, the teacher model generates one rationale conditioned on each label. All generative methods use an analysis-first response format consisting of an `<analysis>` block followed by a `<decision>` block.

C.3.2 Baselines

We include the prompted, untrained model; full-parameter SFT with smaller lr; LoRA (Hu et al., 2021) SFT with rank 16, scaling factor 32, and dropout 0.05 on attention and MLP projections; KL-regularized full SFT with a forward $D_{\text{KL}}(\pi_{\text{initial}} \| \pi)$ penalty of weight 0.05. We also train an additional SFT baseline with a 1:1 mixture of capability self-assessment examples and task-solving rehearsal examples. Each rehearsal target is the initial model's shortest correct generation for a training question, paired with the exact prompt that produced it. For science, this preserves the sampled option ordering. The two example types receive equal loss weight. We also compare with SFT with the same example-selection curriculum as DFW RLVR, and a supervised classifier on frozen, question-only model representations.

C.3.3 Optimization and selection

For the generative methods, we use an effective batch size of 64 questions, yielding 31 updates for math and 47 for science. SFT uses completion-only loss and a 4,096-token sequence limit. Training uses four GPUs, bfloat16, AdamW (Loshchilov & Hutter, 2017), weight decay 0.01, a cosine

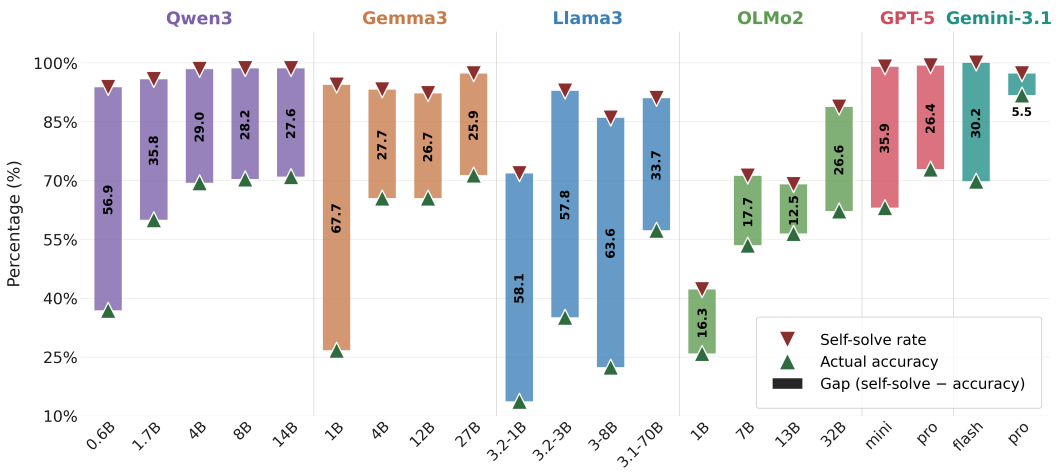


Figure 7: Prompted self-solve rates and measured task accuracy across model families and sizes on a combined math test set. The ▼ marker denotes the *self-solve rate*, and the ▲ marker denotes task accuracy under the solving protocol. The shaded gap captures the model's aggregate tendency to attempt more queries than it solves successfully.

learning-rate schedule, and 10% warm-up. We select the best config between learning rates of 10^{-6} and 10^{-5} for full-parameter SFT, 10^{-6} and 10^{-4} for LoRA SFT, and 10^{-6} for RLVR.

D Additional Results

D.1 Prompt-Only CSA Study

We examine whether prompting alone elicits decisions that reflect a model's measured ability. Prior work reports overconfidence in verbalized confidence (Xiong et al., 2023) and predictions of task success (Barkan et al., 2026). Here we examine this issue through the SELF-SOLVE / DELEGATE decision used in our CSA formulation.

Experimental setup. We evaluate the Qwen3, Gemma3, Llama3, OLMo2, GPT-5, and Gemini 3.1 families, covering open-weight and closed-weight models at multiple scales (Appendix C.1.1). All models use the same combined Math test set of 490 queries: 300 from GSM8K, 100 from MATH500, and 90 from AIME. Dataset details appear in Appendix C.1.2. No CSA training is applied in this study.

For each query, we prompt the model to assess its own capability and choose SELF-SOLVE or DELEGATE before generating a task solution. The prompt requests an analysis followed by a structured decision (Appendix E.1). We obtain one assessment per query. Separately, we measure task accuracy using one solution attempt per query and compare the answer with ground truth. Task accuracy is measured on the full test set, including queries the model chooses to delegate. Open-weight inference uses vLLM, while closed-weight models are accessed through their official APIs. The assessment sampling settings are reported in Appendix C.1.3.

Measurements. The *self-solve rate* is the fraction of queries assigned SELF-SOLVE. The *task accuracy* is the fraction answered correctly in the separate solving evaluation. We compare these two rates for each model. This preliminary study uses one task attempt per query. The main training experiments instead construct capability labels from five attempts, as described in Appendix C.1.4.

Results and interpretation. Figure 7 shows that the self-solve rate exceeds task accuracy for every evaluated model. Under this protocol, models choose to answer more queries than they successfully solve. This pattern is consistent with capability overestimation and extends the prior observations to our explicit decision setting. Some models choose SELF-SOLVE on nearly every query, leaving little scope for selective help-seeking.

The gap measures aggregate mismatch, rather than whether the model identifies the right queries to solve or delegate. Matching the two rates alone would not establish accurate self-assessment: a model could select the correct number of queries but select the wrong ones. A single solution attempt also provides a noisy estimate of query-level capability. We therefore use repeated attempts and query-level assessment metrics in the main experiments, alongside capability retention. This study motivates learning CSA beyond the behavior elicited by the evaluated prompt. It does not establish that all prompting strategies fail.

D.2 Ablation Studies

D.2.1 Capability-Label Construction

We examine how the number of samples K and the aggregation threshold affect Math CSA performance. Results are broadly similar across the tested settings.

Figure 8a varies K under any-correct aggregation, with M-F1 ranging from 0.78 to 0.82. Figure 8b fixes $K = 5$ and varies the required number of correct answers. All tested thresholds remain effective, with the three-correct rule reaching M-F1 of 0.83. Overall, performance does not depend narrowly on a particular sample count or aggregation threshold.

Because these thresholds define different capability targets, we also compare the policies under a common single-attempt evaluation protocol. Each query receives a capability label based on one task answer from the corresponding trained model, and results are averaged over five trials. Figure 8c shows that the policies remain effective under this protocol. Overall, performance is broadly similar across the tested thresholds, supporting robustness to the aggregation threshold.

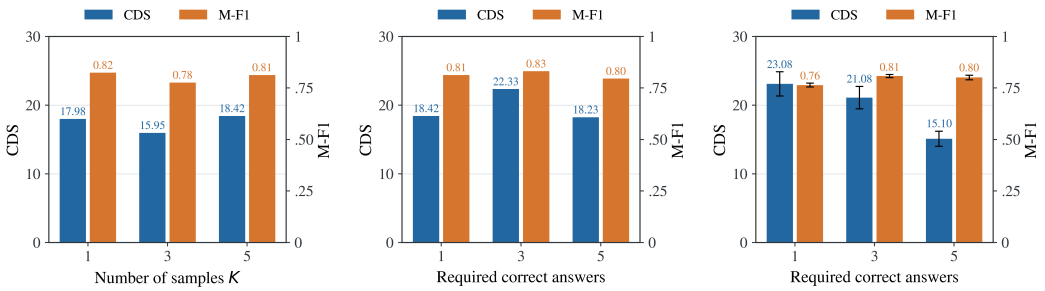
D.2.2 Diversity-Filtered Warm-up

Table 13 confirms the importance of DFW: without it, the policy collapses to almost always predicting SELF-SOLVE (441/490 at 4B and 484/490 at 8B), and both M-F1 and CDS drop substantially. The reward standard deviation also falls sharply, showing that the rollout reward is too uniform to drive meaningful policy updates without DFW.

Beyond this ablation, we find that stronger models yield smaller diversified subsets and that rollout diversity emerges primarily on queries the model should delegate. We examine both patterns next.

Two findings on the diversified subset. **Finding 1: Stronger models yield smaller diversified subsets.** Figure 9 shows the number of Math queries that pass the diversification filter as a function of model size. The subset shrinks sharply with scale: the 1.7B policy contributes roughly 550 diversified samples, and the 4B/8B/14B policies all fall below 200. This is consistent with the diversification criterion acting as an *uncertainty filter*: as the policy becomes stronger, its 16 rollouts on a given query are more likely to agree (all SELF-SOLVE or all DELEGATE), and fewer queries survive the "both labels present" condition. Larger models are simply more internally consistent about what they can and cannot do, and the per-step cost of DFW decreases naturally with scale.

Finding 2: Rollout diversity emerges primarily on queries the model should delegate. Figure 10 contrasts the label distribution of the diversified subset (solid) against the full training set (hatched). The comparison against the full set is what makes this finding meaningful: since the diversified subset is by construction a subset of the full training data, if the model had no preference about *which* queries trigger rollout disagreement, the SELF-SOLVE / DELEGATE proportions in the two distributions should roughly match. Instead, we observe a clear and systematic shift—across all four model sizes, the diversified subset is skewed toward DELEGATE (62%–70%), compared to a 20%–34% DELEGATE rate in the full data. This shift indicates that rollout diversity does not arise uniformly across the data, but emerges preferentially on queries that lie *beyond* the model's reliable solving capability. On clearly-within- and clearly-beyond-capability queries the model's 16 rollouts tend to agree. The boundary cases—where the model *should* delegate but is occasionally tempted to attempt—are exactly what surface in the diversified subset, and exactly where DFW's training signal is most needed.



(a) Number of samples K under any-correct aggregation.

(b) Required correct answers with $K = 5$.

(c) Threshold comparison under common single-draw evaluation.

Figure 8: Sensitivity of Math RLVR to capability-label construction. Panels (a) and (b) use original-model capability labels. Panel (c) evaluates routers trained with $K = 5$ and one-, three-, or five-correct thresholds under a common single-draw capability-label rule. Bars show means across five trials and error bars show sample standard deviations. Blue bars report CDS and orange bars report M-F1.

Table 13: Ablation of DFW on Math. n_S/n_D: SELF-SOLVE/DELEGATE prediction counts.

	Qwen3-4B		**Qwen3-8B**	
Metric	w/o	w/	w/o	w/
M-F1	0.70	**0.80**	0.49	**0.77**
CDS	12.10	**17.17**	4.08	**14.13**
n_S / n_D	441/49	**417/73**	484/6	**426/64**
Reward std	0.133	**0.214**	0.106	**0.232**

D.2.3 CSA Output Order

Because the models are autoregressive, the position of the `<decision>` tag determines how soon a routing decision is available. Under analysis-first the decision is emitted only after the rationale, a median of 81–384 tokens into the response depending on model and task. Under decision-first, inference can stop once the decision field is decoded, so routing requires only the prompt and decision tokens. Table 14 tests this change on three settings: decision-first CDS differs from analysis-first by 0.05, 1.41, and 0.05, respectively, while M-F1 is lower by 0.3–4.7 percentage points, computed before rounding the displayed means. These checkpoints were trained with analysis preceding the decision, so decision-first prompting also changes their expected output format. Across three random seeds, the standard deviation never exceeds 0.8 percentage points in M-F1 or accuracy and 0.51 in CDS.

D.3 Training Dynamics of SFT and RLVR

As shown in Figure 11, we track Qwen3-4B over three training epochs on Science, and the two methods exhibit sharply different trends. For $\text{SFT}_{\text{Label}}$, M-F1 against the original labels keeps rising while M-F1 against the re-probed labels drops, and problem-solving accuracy collapses sharply after the first epoch. Thus, the model learns a fixed delegation rule frozen at the base-model snapshot while its underlying capability changes. For RLVR, the two M-F1 curves rise together and problem-solving accuracy stays at the base level, indicating that self-assessment remains aligned with the model's current capability. This contrast supports the assumption in Section 3.1: acquiring CSA should not come at the cost of the underlying problem-solving ability.

D.4 Capability-Preserving Training Baselines

Experimental settings are given in Appendix C.3. Table 15 provides the full mean $\pm$ standard deviation results for the supervised baselines summarized in Table 3. The results separate two effects that are confounded in standard SFT: learning the CSA decision and preserving the solver whose

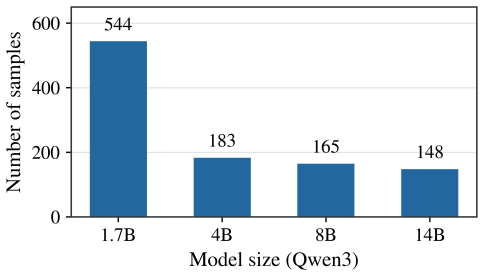


Figure 9: Number of samples per model size on Math whose 16 rollouts during DFW yield both SELF-SOLVE and DELEGATE responses.

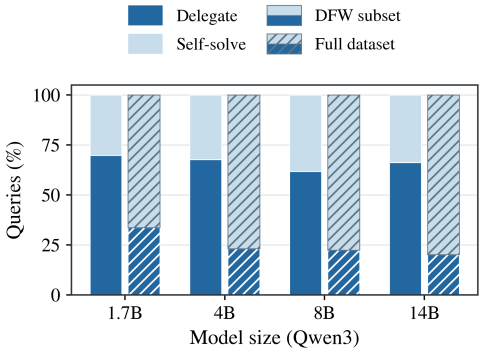


Figure 10: Ground-truth label distribution (SELF-SOLVE vs. DELEGATE) in the diversified subset during DFW (solid) compared to the full training set (hatched).

Table 14: CSA output-order ablation. We evaluate analysis-first CSA checkpoints prompted in both output orders on the held-out test split. Values are mean ± standard deviation over three random seeds.

Model / Task	Output order	M-F1	CDS
Qwen3-1.7B / Math	Analysis-first	0.80 ± 0.00	16.13 ± 0.22
	Decision-first	0.80 ± 0.01	16.08 ± 0.20
Qwen3-1.7B / Science	Analysis-first	0.67 ± 0.01	10.72 ± 0.51
	Decision-first	0.62 ± 0.00	9.31 ± 0.36
Qwen3-4B / Science	Analysis-first	0.65 ± 0.01	8.30 ± 0.50
	Decision-first	0.61 ± 0.01	8.25 ± 0.09

capability is being assessed. These results are averaged across three runs; Table 2 reports the main training comparison.

SFT with smaller lr improves Macro-F1 over the Base model, but can substantially alter the underlying solver. This effect is most pronounced for Qwen3-1.7B, where CR falls to 72.4 on Math and 84.5 on Science. Qwen3-4B is more robust on Math (97.3 CR), but still loses substantial capability on Science (88.6). Thus, improved CSA accuracy under SFT with smaller lr does not necessarily correspond to faithful self-assessment of the original capability boundary.

LoRA SFT largely removes this failure mode: CR remains approximately 99–100% in all four settings. However, preserving capability alone is not sufficient to obtain consistently strong CSA. LoRA reaches a relatively high M-F1 of 0.73 for Qwen3-1.7B on Math, but drops to 0.51 on Science. For Qwen3-4B, it obtains 0.62 and 0.54 on Math and Science, respectively. The relatively large variation for some LoRA settings, such as 0.73 ± 0.07 on Qwen3-1.7B Math, also indicates greater run-to-run sensitivity than RLVR in these experiments.

KL-regularized SFT provides an intermediate trade-off. Relative to SFT with smaller lr, it generally improves capability retention—for example, from 72.4 to 90.6 CR on Qwen3-1.7B Math and from 84.5 to 92.0 on Science—but its CSA performance remains below RLVR. Task-mixed SFT similarly reduces capability degradation in several settings and performs strongly on Qwen3-4B Math, reaching 0.75 M-F1 with 100.2 CR. However, its behavior is not uniform across domains: on Qwen3-1.7B Science, capability falls to 82.0 ± 3.0 despite mixing task-solving supervision into training.

In comparison, RLVR achieves the highest M-F1 in every model–domain setting: 0.84 and 0.66 for Qwen3-1.7B on Math and Science, and 0.85 and 0.67 for Qwen3-4B. At the same time, its CR remains close to or above the Base model in every case (100.3–102.8%). The standard deviations are also small for M-F1 (0.01–0.02 across the four settings). Taken together, these results indicate that RLVR's advantage over supervised training is not merely due to the severe capability loss of naive SFT: the same pattern persists against LoRA, KL regularization, and task-mixed training, which

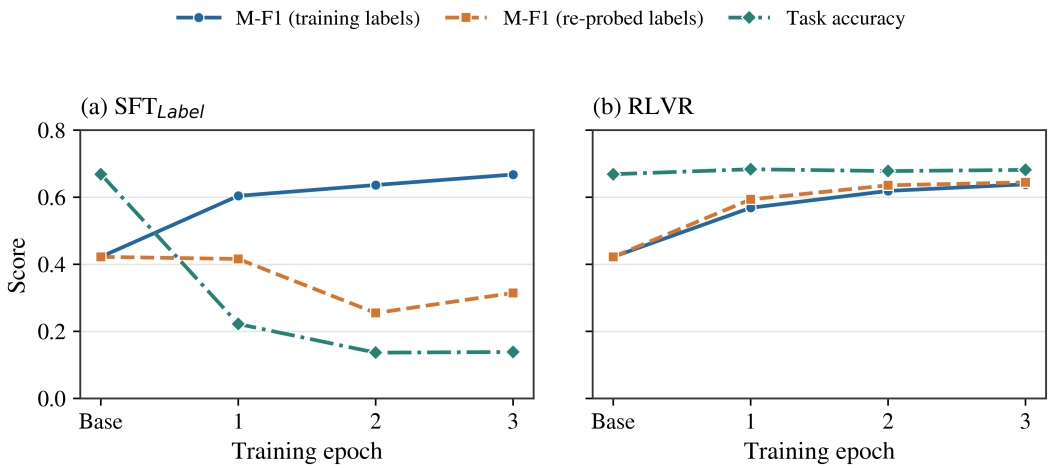


Figure 11: Training dynamics of Qwen3-4B on Science. M-F1 is evaluated against the original and freshly re-probed labels. Accuracy measures original task performance. **(a)** SFT$_{\text{Label}}$. **(b)** RLVR.

Table 15: Full comparison of capability self-assessment training methods on Math and Science for Qwen3-1.7B and Qwen3-4B. We report Macro-F1 (M-F1) for self-assessment quality and Capability Ratio (CR) for preservation of the model's original task capability. Higher is better for both metrics. Values are mean ± standard deviation across three runs. Base denotes the untrained model.

	Qwen3-1.7B		**Qwen3-4B**	
Method	M-F1	CR	M-F1	CR
Math				
Base	0.53	100.0	0.51	100.0
SFT with smaller lr	0.64 ± 0.01	72.4 ± 0.1	0.67 ± 0.01	97.3 ± 1.9
LoRA SFT	0.73 ± 0.07	99.0 ± 1.0	0.62 ± 0.05	100.1 ± 0.2
KL SFT	0.63 ± 0.02	90.6 ± 0.0	0.66 ± 0.01	97.5 ± 0.2
Task-mixed SFT	0.63 ± 0.01	94.6 ± 0.2	0.75 ± 0.07	100.2 ± 0.2
DFW-matched SFT	0.69 ± 0.02	72.2 ± 0.2	0.71 ± 0.01	97.4 ± 1.6
Query Probe	0.70 ± 0.01	100	0.71 ± 0.01	100
RLVR	**0.84 ± 0.01**	**100.3 ± 0.0**	**0.85 ± 0.01**	**102.8 ± 0.6**
Science				
Base	0.46	100.0	0.42	100.0
SFT with smaller lr	0.58 ± 0.02	84.5 ± 2.6	0.60 ± 0.06	88.6 ± 1.3
LoRA SFT	0.51 ± 0.04	100.0 ± 1.2	0.54 ± 0.03	99.5 ± 0.1
KL SFT	0.60 ± 0.00	92.0 ± 2.2	0.59 ± 0.05	90.4 ± 0.1
Task-mixed SFT	0.55 ± 0.01	82.0 ± 3.0	0.61 ± 0.06	96.4 ± 0.2
DFW-matched SFT	0.59 ± 0.02	82.1 ± 2.9	0.65 ± 0.04	89.0 ± 0.9
Query Probe	0.55 ± 0.01	100	0.56 ± 0.01	100
RLVR	**0.66 ± 0.02**	**101.3 ± 1.8**	**0.67 ± 0.02**	**101.5 ± 0.2**

explicitly or implicitly constrain changes to the base solver. RLVR preserves solving ability and beats DFW-matched SFT, suggesting that neither retention nor DFW's training curriculum alone explains its gains. See Chu et al. (2025) and Shenfeld et al. (2026) for related discussions.

D.5 Training-Free Baselines

We describe the baselines in the order used in Table 16, grouped by the task outputs available when making the assessment. Baseline references are included in the table.

Query-based baselines. These methods assess a query before generating a task answer. RLVR is the trained reference: it generates a CSA rationale and a binary decision. The remaining methods do not update model parameters. We describe the implemented scoring rules below, retaining the method names used in Table 16.

Internal confidence. We prompt the model to indicate whether it can answer correctly using Yes or No, without solving the problem. At each of the last six prompt positions, we apply the model's output head to each returned hidden state, including the embedding state and transformer-layer states. We normalize the Yes and No logits to obtain $q_{t,l} = \exp(z_{t,l}^{\text{Yes}})/[\exp(z_{t,l}^{\text{Yes}}) + \exp(z_{t,l}^{\text{No}})]$. Internal confidence averages these values with normalized Gaussian weights that favor the final token and final layer. With locality parameter 1, the unnormalized weight is $\exp[-(T-t)^2 - (L-l)^2]$. No Yes/No response needs to be generated.

P(Yes). This uses only $q_{T,L}$, the normalized Yes probability at the final prompt position and final layer. It measures the model's explicit pre-answer judgment of its own ability.

P(Yes)-Mean. This takes the uniform mean of $q_{t,l}$ over the last six prompt positions and all returned hidden states. Unlike internal confidence, it does not emphasize later positions or layers.

The next four scores use the likelihood of the input question itself, with answer options appended for Science. Let $\ell_t = \log p(x_t \mid x_{<t})$ for input tokens after the first. The implementation floors probabilities at 10^{-8} and orients all scores so that larger values indicate greater confidence.

Max token surprisal. We measure uncertainty using the largest input-token surprisal, $\max_t(-\ell_t)$. The confidence score is its negative, $-\max_t(-\ell_t) = \min_t \ell_t$, so larger scores indicate greater confidence. This tests whether any input token is particularly unexpected to the model.

Predictive entropy. The implemented score is $\sum_t \ell_t$, the negative total surprisal of the observed question. This is an input-likelihood score, not a full-vocabulary entropy calculation, and it is sensitive to question length.

Min-K. We select the five input tokens with the lowest log probabilities, or all available scored tokens if fewer than five exist. The score is the negative perplexity of this subset, $-\exp[-\operatorname{mean}_{t\in I_5} \ell_t]$. Here the five-token subset is unrelated to the number of sampled task answers used to construct capability labels.

Perplexity. We use negative question perplexity, $-\exp[-\operatorname{mean}_t \ell_t]$. Unlike Min-K, this averages over all scored input tokens; unlike the predictive-entropy score above, it normalizes by input length.

Single-answer-based baselines. These methods generate one task answer before assessing it. Let $a_1, \ldots, a_m$ denote the generated solution tokens.

Sequence log-probability. The confidence score is the geometric mean of the generated tokens' conditional probabilities, $\exp[m^{-1} \sum_j \log p(a_j \mid x, a_{<j})]$. This measures average likelihood of the sampled solution rather than its verified correctness.

Minimum token probability. The score is $\min_j p(a_j \mid x, a_{<j})$. It treats the least likely generated token as a signal of uncertainty, in contrast to averaging likelihood over the entire solution.

P(True). After producing a solution, the model receives the question and candidate answer and judges whether the answer is correct. The probability assigned to the affirmative judgment is used as confidence. This assesses a particular generated answer, whereas P(Yes) above asks about capability before an answer is available.

Verbalized confidence. The model is prompted to provide an answer together with a numerical probability that it is correct. We parse the stated probability and use it as the score, rather than reading confidence directly from token likelihoods.

Answer probability. We suppress reasoning and score the answer to which the model directly commits. For Science, the score is the largest option-letter probability after normalization over valid choices. For Math, it is the length-normalized likelihood of the directly generated answer span. It therefore excludes the likelihood of a full reasoning trace.

Multi-answer-based baselines. Both methods use $K = 3, 5, 10$ independently sampled task solutions. Here K controls the evidence available to the assessment method.

Self-consistency. We extract a final answer from each solution and use the number of votes for the most frequent valid answer divided by K. Greater agreement yields higher confidence. Unparseable generations remain in the denominator but do not count as valid votes.

Semantic entropy. We group the same samples by final-answer equivalence, using exact-match answer strings for Math and option letters for Science. With empirical cluster proportions f_j, we compute $H = -\sum_j f_j \log f_j$ and use $1 - H/\log K$ as confidence. Concentrated answers give high confidence; dispersed answers give low confidence. Unparseable generations form an additional cluster. This implementation uses answer equivalence rather than an entailment model, and agreement among unparseable outputs can inflate its score.

Simple reference strategies. *Always SELF-SOLVE* assigns every query to the local model, while *Always DELEGATE* assigns every query to delegation. They show how the label distribution affects constant-decision Macro-F1. *Random @ matched SSR* assigns decisions randomly at the target self-solve rate, providing a reference that matches decision frequency without using query information.

Evaluation setting. Figure 3 reports Qwen3-4B Math. Table 16 reports all four model-domain settings, including $K = 3, 5, 10$ for self-consistency and semantic entropy. Figure 3 uses the $K = 5$ results and reports M-F1 as a percentage. Solutions use temperature 1.0, top-p 1.0, no top-k truncation, and a maximum length of 4096 tokens. The auxiliary read-outs disable thinking. Direct-answer and verbalized-confidence generations use greedy decoding with maximum lengths of 64 and 128 tokens, respectively.

Labels and thresholds. Training-free baselines are evaluated against the base model's capability labels, and RLVR is evaluated against its own post-training labels. Query-based and single-answer-based baselines and simple strategies use the $K = 5$ evaluation labels. For each model, domain, method, and K, we select the threshold that maximizes Macro-F1 on 500 calibration questions sampled from the training split with seed 42. Calibration questions are deduplicated and exclude all test questions. The selected threshold is then applied unchanged to the test set. The random reference's self-solve rate is also set from calibration labels. Unresolved read-outs receive DELEGATE decisions. Unavailable CDS results are marked with a dash.

D.6 CSA Inference Efficiency

Table 17 shows that CSA assessments are fast in the measured settings: mean latency ranges from 0.39 to 1.32 seconds per query across four Qwen3 scales and two domains, with all Math settings below one second. We measure inference using vLLM on a single NVIDIA B200 GPU with batch size 1. Each query produces one non-thinking CSA response, with prefix caching disabled. Time is the mean wall-clock duration of generation requests after warm-up and excludes model loading and subsequent task solving. Generated-token counts come directly from vLLM token IDs and exclude prompt tokens.

D.7 Query-Level Capability Stability

Capability Ratio measures aggregate problem-solving performance, but it can conceal changes in which individual queries a model can solve. For example, CSA training could improve performance on some queries while degrading it on others, leaving average accuracy unchanged. We therefore evaluate whether CSA training changes the model's query-level capability boundary.

Evaluation setting. The analysis covers eight model–evaluation–policy settings formed by two evaluation domains (mathematics and code), two model sizes (Qwen3-1.7B and Qwen3-4B), and two CSA policies for each model (one trained on mathematics and one trained on science). The column "CSA policy" in Table 18 denotes the domain used to train the CSA policy, while "Evaluation" denotes the domain on which problem-solving capability is measured. For example, "Code / Qwen3-1.7B / Math" evaluates the coding capability of Qwen3-1.7B after CSA training on mathematics.

Table 16: Training-free baselines across model-domain settings. Entries are M-F1 / CDS on the $[0, 1]$ M-F1 scale. Query-based and single-answer-based baselines and simple strategies use the $K = 5$ evaluation labels. For self-consistency and semantic entropy, K specifies the number of sampled solutions.

	Qwen3-4B		OLMo2-7B	
Method	Math	Science	Math	Science
Query-based baselines				
RLVR (ours)	0.80 / 17.17	0.64 / 8.07	0.89 / 26.60	0.69 / 10.94
Internal confidence (Chen et al., 2026)	0.57 / 2.89	0.51 / 1.15	0.56 / 3.31	0.63 / 6.85
P(Yes) (Chen et al., 2026)	0.52 / 1.26	0.53 / 1.72	0.55 / 2.89	0.63 / 6.46
P(Yes)-Mean (Chen et al., 2026)	0.43 / -3.62	0.50 / 0.02	0.66 / 7.05	0.63 / 7.52
Max token surprisal (Chen et al., 2026)	0.44 / -2.75	0.50 / 0.83	0.52 / 2.35	0.60 / 5.30
Predictive entropy (Chen et al., 2026)	0.52 / 0.74	0.54 / 1.92	0.57 / 3.66	0.53 / 1.52
Min-K (Chen et al., 2026)	0.45 / -2.49	0.55 / 2.57	0.50 / 1.85	0.53 / 1.78
Perplexity (Chen et al., 2026)	0.32 / -8.61	0.52 / 1.20	0.36 / -3.93	0.42 / -1.28
Single-answer-based baselines				
Sequence log-probability (Malinin & Gales, 2021)	0.76 / 12.30	0.61 / 6.86	0.72 / 10.46	0.59 / 5.66
Min. token probability (Manakul et al., 2023)	0.76 / 11.57	0.58 / 4.17	0.72 / 10.48	0.68 / 9.23
P(True) (Kadavath et al., 2022)	0.59 / 3.95	0.63 / 6.98	0.68 / 8.55	0.61 / 6.01
Verbalized confidence (Xiong et al., 2023)	0.59 / 6.48	0.57 / 3.56	0.47 / -0.91	0.58 / 6.39
Answer probability (Kadavath et al., 2022)	0.59 / 3.56	0.54 / 5.01	0.68 / 9.13	0.67 / 9.22
Multi-answer-based baselines				
Self-consistency (Wang et al., 2023)				
$K = 3$	0.78 / 14.21	0.76 / 14.58	0.81 / 17.21	0.71 / 11.03
$K = 5$	0.78 / 13.75	0.75 / 14.84	0.81 / 15.95	0.76 / 13.83
$K = 10$	0.74 / 10.31	0.77 / 18.45	0.84 / 18.20	0.79 / 20.44
Semantic entropy (Farquhar et al., 2024)				
$K = 3$	0.76 / 11.48	0.76 / 14.58	0.81 / 17.21	0.71 / 10.71
$K = 5$	0.75 / 12.14	0.75 / 14.84	0.82 / 17.11	0.75 / 13.36
$K = 10$	0.74 / 9.83	0.78 / 17.39	0.84 / 18.53	0.79 / 17.64
Simple reference strategies				
Always SELF-SOLVE	0.43 / –	0.41 / –	0.39 / –	0.22 / –
Always DELEGATE	0.20 / –	0.24 / –	0.26 / –	0.42 / –
Random @ matched SSR	0.47 / –	0.50 / –	0.46 / –	0.49 / –

– indicates unavailable results.

Table 17: CSA generation efficiency across model scales and domains. Time is mean generation latency per example at batch size 1. Tokens are generated completion tokens per example.

	Math		**Science**	
Qwen3	**Time (s)**	**Tokens**	**Time (s)**	**Tokens**
1.7B	0.53	304.9	0.93	540.2
4B	0.39	150.7	0.47	184.9
8B	0.74	198.4	1.32	359.9
14B	0.93	156.7	1.16	195.2

The mathematics evaluation contains 490 queries and contributes $4 \times 490 = 1{,}960$ query–policy pairs. The coding evaluation contains 722 queries and contributes $4 \times 722 = 2{,}888$ pairs. The complete analysis therefore contains $4{,}848$ query–policy pairs.

For each setting, we compare the original base checkpoint with the corresponding CSA-trained checkpoint. Both checkpoints use the same solving prompt, sampling parameters, answer parser, and correctness criterion. A mathematics response is correct when its extracted final answer matches the reference answer. A coding response is correct when the generated program passes the benchmark's unit tests.

Capability labels and high-sample probes. We construct the reported query-level capability labels from $K = 5$ independent solution attempts per query and checkpoint. Under the any-correct rule, a query receives the label SELF-SOLVE if at least one of the five attempts is correct and receives DELEGATE otherwise.

Because a binary label constructed from five stochastic attempts is itself noisy, we additionally collect $K = 32$ attempts per query for both the base and CSA-trained checkpoints. These high-sample probes are used to estimate the label disagreement expected from finite sampling and to test for changes in each query's underlying success probability. They are not used to redefine the reported $K = 5$ capability labels.

Capability-label churn. Let y_q^{base} and y_q^{CSA} denote the $K = 5$ capability labels assigned to query q by the base and CSA-trained checkpoints. For an evaluation set containing N queries, we define capability-label churn as

$$\text{Churn} = \frac{1}{N} \sum_{q=1}^{N} \mathbf{1} \left[y_q^{\text{base}} \neq y_q^{\text{CSA}} \right]. \tag{9}$$

Churn is the fraction of queries that cross the binary capability boundary after CSA training. For example, a churn value of 0.05 means that the pre- and post-training labels differ on 5% of the evaluation queries. Nonzero churn does not necessarily imply that the underlying capability changed, because two finite stochastic probes of the same checkpoint can produce different labels.

Cohen's κ. We complement churn with Cohen's κ, which measures agreement between the base and CSA-trained labels after accounting for agreement expected from their overall label frequencies. Let

$$p_o = 1 - \text{Churn}$$

be the observed fraction of queries on which the two checkpoints agree. Let s_{base} and s_{CSA} denote the fractions of queries labeled SELF-SOLVE by the base and CSA-trained checkpoints, respectively. The agreement expected from these marginal label frequencies is

$$p_e = s_{\text{base}} s_{\text{CSA}} + (1 - s_{\text{base}})(1 - s_{\text{CSA}}). \tag{10}$$

Cohen's κ is

$$\kappa = \frac{p_o - p_e}{1 - p_e}. \tag{11}$$

A value of $\kappa = 1$ denotes perfect agreement. A value of $\kappa = 0$ means that the observed agreement equals the agreement expected from the two checkpoints' marginal label frequencies. This correction is useful when one label is common: if both checkpoints label almost every query SELF-SOLVE, their raw agreement can be high even when it contains little query-specific information. In our experiments, mean raw agreement is 95.5% and mean κ is 0.895, showing that the base and CSA-trained checkpoints have highly similar query-level capability labels even after accounting for their overall label distributions.

Capability Ratio. We also report Capability Ratio (CR) to measure aggregate capability preservation:

$$\text{CR} = \frac{\text{Acc}_{\text{CSA}}}{\text{Acc}_{\text{base}}} \times 100\%, \tag{12}$$

where Acc_{base} and Acc_{CSA} are the problem-solving accuracies of the two checkpoints under the same evaluation protocol. Churn and κ measure query-level stability, whereas CR measures aggregate performance retention.

Finite-sampling null. The $K = 5$ capability label can change across repeated probes even when the underlying checkpoint is unchanged. We estimate this finite-sampling variability using the $K = 32$ probe of the base checkpoint.

For query q, let c_q be the number of correct responses among the 32 base-checkpoint attempts. Under the any-correct rule, the probability that a random subset of five attempts receives a SELF-SOLVE label is

$$\pi_q = 1 - \frac{\binom{32-c_q}{5}}{\binom{32}{5}}. \tag{13}$$

Table 18: Query-level capability stability after CSA training. **CR** is the Capability Ratio. **Churn** is the percentage of queries whose $K = 5$ capability label differs between the base and CSA-trained checkpoints. **Null** is the expected disagreement between two $K = 5$ probes of an unchanged base checkpoint, estimated from $K = 32$ attempts per query. **Excess** is churn minus the sampling-noise null, reported in percentage points with a paired-bootstrap 95% confidence interval. κ is Cohen's chance-corrected agreement. **BH discoveries** is the number of per-query comparisons rejected after Benjamini–Hochberg correction at FDR 0.05.

Evaluation	Model	CSA policy	CR	Churn (%)	Null (%)	Excess in pp (95% CI)	κ	BH discoveries
Code	Qwen3-1.7B	Math	102.7%	4.6	6.3	-1.7 $[-3.4, -0.1]$	0.907	3
		Science	100.7%	6.0	6.3	-0.3 $[-2.1, +1.6]$	0.879	7
	Qwen3-4B	Math	100.2%	4.8	5.4	-0.6 $[-2.3, +1.1]$	0.889	11
		Science	100.2%	4.3	5.4	-1.1 $[-2.8, +0.4]$	0.902	11
Math	Qwen3-1.7B	Math	100.6%	3.7	5.4	-1.7 $[-3.4, +0.1]$	0.909	0
		Science	102.6%	6.3	5.4	$+0.9$ $[-0.9, +2.9]$	0.841	0
	Qwen3-4B	Math	101.3%	3.5	4.3	-0.8 $[-2.2, +0.8]$	0.903	0
		Science	99.7%	2.7	4.3	-1.6 $[-3.1, -0.0]$	0.927	0
Mean: CSA policies (8 cells)			**101.0%**	**4.5**	**5.4**	$\mathbf{-0.9}$	**0.895**	**32/4,848**

The probability that two independently sampled $K = 5$ labels disagree is then

$$\nu_q = 2\pi_q(1 - \pi_q). \tag{14}$$

We define the sampling-noise null for an evaluation cell as

$$\text{Null} = \frac{1}{N}\sum_{q=1}^{N} \nu_q, \tag{15}$$

and define excess churn as

$$\text{Excess} = \text{Churn} - \text{Null}. \tag{16}$$

Positive excess churn indicates more pre/post label disagreement than expected from finite sampling alone. Excess churn near or below zero indicates that the observed disagreement is comparable to the variability of repeated $K = 5$ measurements.

Bootstrap uncertainty. We estimate uncertainty in excess churn using a paired bootstrap over queries. Within each evaluation cell, we sample N queries with replacement and recompute both observed churn and the sampling-noise null on the same resampled queries. We record their difference for each of 2000 bootstrap replicates and use the 2.5th and 97.5th percentiles of the resulting distribution as the paired-bootstrap 95% confidence interval.

Aggregate stability results. Table 18 shows that query-level capability remains stable within the resolution of our evaluation. Across the eight settings, mean observed label churn is 4.5%, compared with 5.4% disagreement expected from finite sampling. Mean excess churn is therefore -0.9 percentage points. No evaluation cell has a statistically positive excess over its sampling-noise null. Agreement between the pre- and post-training labels also remains high, with mean $\kappa = 0.895$, while the mean Capability Ratio is 101.0%.

Per-query changes in success probability. Label churn measures changes in a thresholded $K = 5$ capability label. We additionally test whether CSA training changes each query's underlying success probability using the $K = 32$ high-sample probes.

For each query–policy pair, we construct the following 2×2 contingency table:

	Correct	Incorrect
Base	c_q^{base}	$32 - c_q^{\text{base}}$
CSA	c_q^{CSA}	$32 - c_q^{\text{CSA}}$.

(17)

We apply a two-sided Fisher exact test to each of the 4,848 query–policy pairs and control the false discovery rate at 0.05 using the Benjamini–Hochberg procedure across the complete family of tests.

Only 32 of the 4,848 comparisons (0.66%) are identified as changes after correction. All 32 occur in the coding evaluation: 3 and 7 for the two Qwen3-1.7B policies, and 11 for each Qwen3-4B policy. None of the 1,960 mathematics comparisons is identified as a change.

Measurement resolution and limitations. The churn and κ analyses concern the binary capability labels induced by the specified $K = 5$ aggregation rule. They do not establish that the checkpoints have identical continuous success probabilities on every query. Likewise, a per-query comparison that is not rejected should not be interpreted as proof that the two success probabilities are equal. With 32 attempts per checkpoint, small changes may remain unresolved.

Each evaluation cell contains approximately 500 to 700 queries. At this sample size, the paired-bootstrap confidence intervals rule out aggregate excess label churn larger than approximately 2–3 percentage points. Our conclusion is therefore that CSA training preserves the model's query-level capability boundary within the resolution of this evaluation.

D.8 Within-Source Discrimination on Math

Figure 5 tests whether Math CSA decisions identify solvable queries beyond benchmark source identity. We evaluate the six Math-trained RLVR models from Table 2 on the same 490 questions: 300 GSM8K, 100 MATH500, and 90 AIME. Each decision is matched to the same query's post-training capability label, constructed by any-correct aggregation over five task attempts. We exclude invalid decisions: three for OLMo2-13B and none for the other models.

For each model, we hold the capability labels fixed and independently permute the binary decisions within each source. Every permutation preserves that model's exact numbers of SELF-SOLVE and DELEGATE decisions in each source. Thus, a preference for delegating AIME is preserved, while any association between a decision and the particular query's solvability within AIME is removed. We compute pooled Macro-F1 across the three sources after each permutation, rather than averaging three source-specific F1 scores.

We generate $B = 100{,}000$ null draws with random seed 20260924. The implementation samples the number of true positives within each source from its conditional hypergeometric distribution, which is equivalent to uniformly shuffling binary decisions while fixing their counts. The blue markers in Figure 5 show the null mean; the bars show its 2.5th and 97.5th percentiles. Orange markers show the observed score, and annotations give its improvement over the null mean in percentage points. These bars describe the permutation null, not confidence intervals for the observed score. A one-sided Monte Carlo p-value is $(1 + \#\{M - F1_b \geq M - F1_{\text{obs}}\})/(B + 1)$. The saved analysis applies Holm correction jointly to the six Math tests and an additional exploratory Omni-MATH test; all six Math results remain below 0.002 under this seven-test correction.

Table 19 reports the numerical values underlying Figure 5. Observed Macro-F1 exceeds the source-preserving null by 3.44–9.61 percentage points. Source-dependent routing frequencies alone therefore do not explain the observed performance. This pooled test demonstrates discrimination within sources overall; it does not imply equally strong discrimination in every individual source.

Table 19: Numerical results for Figure 5. M-F1 and null intervals are on the 0–100 scale; gains are percentage points computed before rounding. N counts valid decisions. The 95% interval describes the within-source permutation null. Adjusted p-values use the Holm correction described in this section.

Model	N	RLVR M-F1	Null mean	95% null interval	Gain	Adjusted p
Qwen3-1.7B	490	80.35	73.91	[71.37, 76.42]	+6.44	7.00×10^{-5}
Qwen3-4B	490	80.09	70.48	[67.48, 73.46]	+9.61	7.00×10^{-5}
Qwen3-8B	490	77.23	68.11	[64.42, 71.53]	+9.11	7.00×10^{-5}
Qwen3-14B	490	75.32	67.74	[64.74, 71.09]	+7.58	1.00×10^{-4}
OLMo2-7B	490	88.55	83.32	[80.92, 85.69]	+5.23	7.00×10^{-5}
OLMo2-13B	487	88.56	85.12	[82.85, 87.13]	+3.44	1.78×10^{-3}

D.9 Zero-Shot Transfer to Coding

We test whether CSA learned from math or science transfers to Python programming without further training. Our evaluation contains 722 problems from HumanEval+ (Chen et al., 2021; Liu et al., 2023), MBPP+ (Austin et al., 2021; Liu et al., 2023), and a LiveCodeBench (Jain et al., 2025) subset with organizer-provided difficulty labels. To reduce contamination risk, we include only LiveCodeBench

Table 20: **Zero-shot CSA transfer to code.** Policies trained on math or science are evaluated, without any further training, on 722 Python problems (HumanEval+ / MBPP+ / LiveCodeBench). Mean over 3 decoding seeds. Seed-level sd is ≤ 0.02 for M-F1 and ≤ 0.9 for $z_{\text{M-F1}}$. *Solve* is each checkpoint's own pass@1, re-measured per policy. *SSR* is the self-solve rate and n_{del} the number of delegated problems. p_S / p_D are solve rates within the queries a policy self-solves and delegates. $z_{\text{M-F1}}$ scores M-F1 against an SSR-matched permutation null. All four CSA cells have $p < 0.001$. The base policy delegates on at most 2 of 722 problems, so p_D, CDS and $z_{\text{M-F1}}$ are undefined for it.

Model	CSA policy	Solve	SSR	n_{del}	p_S	p_D	M-F1	CDS	$z_{\text{M-F1}}$
Qwen3-1.7B	none (base)	0.562	1.000	0.3	0.563	—	0.36	—	—
	CSA-Math	0.578	0.828	124	0.667	0.150	**0.64**	13.83	**10.59**
	CSA-Science	0.566	0.873	92	0.630	0.127	0.58	12.78	8.98
Qwen3-4B	none (base)	0.675	0.998	1.3	0.676	—	0.41	—	—
	CSA-Math	0.676	0.920	58	0.717	0.203	0.57	9.39	7.83
	CSA-Science	0.676	0.888	81	0.738	0.182	**0.63**	12.10	**9.95**

Table 21: **Transferred policies track code difficulty they were never shown.** Self-solve rate and true solve rate per difficulty tier. Every CSA policy's SSR falls monotonically with the tier (Spearman $= 1.00$ against solve rate in all four cases), while the base policy stays at ceiling even on hard problems it solves 8–17% of the time.

		Easy (n=60)		Medium (n=60)		Hard (n=60)	
Model	CSA policy	SSR	Solve	SSR	Solve	SSR	Solve
Qwen3-1.7B	none (base)	0.999	0.648	1.000	0.183	1.000	0.083
	CSA-Math	0.950	0.673	0.317	0.117	0.111	0.083
	CSA-Science	0.962	0.653	0.594	0.167	0.256	0.100
Qwen3-4B	none (base)	1.000	0.761	1.000	0.317	0.978	0.167
	CSA-Math	0.991	0.761	0.800	0.317	0.333	0.183
	CSA-Science	0.973	0.764	0.644	0.333	0.283	0.133

contests held after the models' reported pretraining cutoff. We assess solution correctness by running each generated program against the benchmark's unit tests in a sandbox. Both trained policies are compared with the base model using the same CSA prompt. We also measure each checkpoint's coding ability separately, so its assessment decisions are evaluated against its own post-training performance.

Transfer performance and capability retention. As shown in Table 20, CSA training improves coding assessment: Macro-F1 rises from 0.36 to 0.64 for the 1.7B model and from 0.41 to 0.63 for the 4B model, with similar gains from math and science training. The transferred policies choose SELF-SOLVE on 95–99% of easy problems, but only 11–33% of hard problems. The prompted base models, by comparison, delegate at most two of the 722 problems, even though they solve fewer than 17% of hard problems. These assessment gains accompany Capability Ratios of 100–103%, indicating that overall coding ability is retained. Formatting failures are also rare, occurring in only 0.5–1.1% of attempts.

Does the policy identify which problems to delegate? An increase in Macro-F1 could result simply from delegating more often. To test whether the policies identify the right problems to delegate, we randomly shuffle each policy's decisions across problems within sources. This preserves the number of SELF-SOLVE and DELEGATE decisions but removes their association with individual problems. As shown in Table 20, the policies' observed Macro-F1 scores are 7.8–10.6 standard deviations above this shuffled baseline ($p < 0.001$). Thus, their gains reflect an ability to distinguish solvable from unsolvable problems beyond what a change in delegation frequency alone would achieve. This provides evidence that CSA transfers to coding, a domain not included in CSA training.

Reporting CDS. We omit CDS for the base models because they delegate only one or two problems. Such small delegation groups make the statistic unstable and unsuitable for comparison.

Table 22: **CSA decisions track their own model's competence more than an independent model's.** $\Delta = r(\text{decision}, \text{own pass rate} \mid \text{OLMoE}) - r(\text{decision}, \text{OLMoE} \mid \text{own})$, computed over all 722 code problems with bootstrap CIs averaged across 3 decoding seeds. Both models' per-problem competence is measured the same way, by sampling $K{=}32$ solutions and running the unit tests, so neither side is favoured by having the better-measured signal. All four policies are positive with CIs excluding zero. Split-half reliability of the competence estimates is 0.985 / 0.990 / 0.958 (Qwen3-1.7B / Qwen3-4B / OLMoE), so the effect is not an artefact of a noisier comparator.

Policy model	CSA policy	$r(\text{dec}, \text{own} \mid \text{OLMoE})$	$r(\text{dec}, \text{OLMoE} \mid \text{own})$	Δ	95% CI	$r(\text{own}, \text{OLMoE})$
Qwen3-1.7B	CSA-Math	0.256	0.093	$+0.163$	$[+0.079, +0.240]$	0.666
	CSA-Science	0.224	0.067	$+0.157$	$[+0.080, +0.225]$	0.666
Qwen3-4B	CSA-Math	0.280	0.042	$+0.238$	$[+0.171, +0.304]$	0.528
	CSA-Science	0.322	0.063	$+0.258$	$[+0.181, +0.333]$	0.528

Table 23: **CSA beats a difficulty rule built from an independent model.** The rule delegates exactly as many problems as the CSA policy, choosing the ones OLMoE is least likely to solve (measured, $K{=}32$). z_{residual} scores CSA against a permutation null confined to bins of OLMoE's pass rate, so it measures discrimination among problems an independent model rates as equally hard. CSA wins in all four cells despite seeing only the problem text, while the rule uses 32 executed solution samples per problem.

Policy model	CSA policy	M-F1 CSA	M-F1 OLMoE rule	Δ	AUC (OLMoE score)	z_{residual}
Qwen3-1.7B	CSA-Math	**0.64**	0.58	$+0.06$	0.835	4.53
	CSA-Science	**0.58**	0.52	$+0.06$	0.836	3.79
Qwen3-4B	CSA-Math	**0.57**	0.52	$+0.05$	0.795	3.97
	CSA-Science	**0.63**	0.58	$+0.05$	0.794	5.44

D.10 CSA Is Model Specific

To test whether CSA teaches a signal specific to the model carrying it, we compared each Qwen policy against OLMoE-1B-7B (Muennighoff et al., 2025), from a different lab with a different training corpus. We measured per-problem competence for both the policy's own model and OLMoE by sampling 32 solutions each on the same 722 coding problems and running the unit tests, so both sides are estimated with identical precision. The question is then whether a policy's decisions follow its own model's measured success rate or CSA is just tracking problem difficulty.

All four policies follow their own model as shown in Table 22. The partial correlation with own competence is consistently stronger than the correlation with OLMoE's, and every bootstrap confidence interval excludes zero, with effects ranging from +0.157 to +0.258. Table 23 gives the same conclusion in operational terms: delegating the problems OLMoE is least likely to solve (using 32 executed samples per problem) is a worse policy than what CSA produces from the problem text alone, in all four cells. CSA also discriminates among problems that OLMoE rates as equally hard (residual-null z of 3.79 to 5.44). Two checks rule out the obvious artifacts: the competence estimates are near-equally reliable (split-half 0.985 / 0.990 / 0.958), and OLMoE's pass rates correlate substantially with the Qwen models' (r = 0.666 and 0.528), so it is a relevant comparator rather than a noisy or irrelevant one.

Table 24 examines aggregate alignment when each of three CSA policies is paired with each solver. Policy coverage increases with the size of the model it was trained for (0.785, 0.891, and 0.892). Pairing the 8B policy with the 1.7B solver yields a coverage–accuracy gap of approximately 11 percentage points. Matched pairings have a smaller mean absolute coverage–accuracy gap than mismatched pairings (0.049 versus 0.072). The 1.7B matched pair has a gap of +0.003, while the 4B and 8B policies have nearly identical coverage. This aggregate comparison is suggestive and does not establish query-level model specificity on its own.

Figure 12 further shows that RLVR decisions adapt to model capability: weaker models DELEGATE more, while stronger models SELF-SOLVE more.

Table 24: **Cross-scale transfer of CSA models.** Rows are CSA policies trained for one Qwen3 solver, applied to all three. The gap is coverage minus the solver's accuracy (parenthesised). Matched pairs (bold) have a smaller mean absolute gap (0.049 vs. 0.072 mean |gap|).

		Coverage–accuracy gap vs. solver		
CSA Policy (trained for)	Coverage	1.7B (0.782)	4B (0.818)	8B (0.820)
CSA for Qwen3-1.7B	0.785	**+0.003**	−0.034	−0.036
CSA for Qwen3-4B	0.891	+0.110	**+0.073**	+0.071
CSA for Qwen3-8B	0.892	+0.110	+0.074	**+0.072**
Mean |gap|		diagonal **0.049**	vs.	off-diagonal 0.072

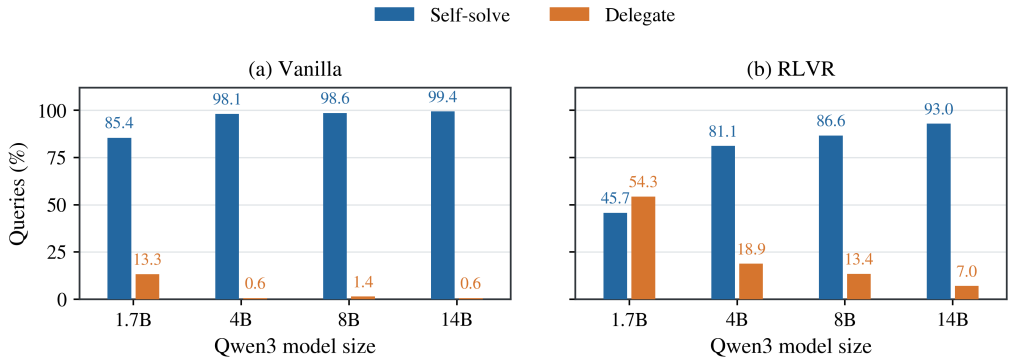


Figure 12: Qwen3 self-routing distributions on Science: (a) Vanilla (b) RLVR. Paired bars show self-solve and delegate rates among all 700 queries. Invalid Vanilla decisions (1.3% for 1.7B and 4B) are excluded from both bars.

D.11 Case Studies

To illustrate concretely how RLVR shapes CSA, we present case studies in Math and Science.

Math Problem As shown in Figure 13, we provide the complete reasoning traces for the math case study. The Vanilla Qwen3-4B model confidently chooses to SELF-SOLVE, but produces an incorrect answer across all five sampled responses. In contrast, the RLVR-trained model recognizes that the problem is beyond its reliable capability and chooses to DELEGATE. This example illustrates how RLVR teaches the model to recognize when a query exceeds its competence.

Science Problem Figure 14 presents an additional case study on a science question, where the same pattern emerges: the vanilla model answers confidently but incorrectly, while the RLVR-trained model recognizes its uncertainty and delegates.

D.12 Applications of CSA

D.12.1 Self-Routing

We provide illustrative figures for *self-routing* in Figure 15. Formally, given a query x, the local model first produces a CSA decision $\hat{y} = \mathcal{M}_{\text{local}}^{\text{CSA}}(x) \in \{\text{SELF-SOLVE}, \text{DELEGATE}\}$, and the routed answer is

$$\hat{a}(x) = \begin{cases} \mathcal{M}_{\text{local}}(x), & \hat{y} = \text{SELF-SOLVE}, \\ \mathcal{M}_{\text{cloud}}(x), & \hat{y} = \text{DELEGATE}. \end{cases} \tag{18}$$

This setup exposes a direct cost–accuracy trade-off between the two extremes of local-only inference (cheap but limited by the local model's capability) and cloud-only inference (accurate but expensive on every query). An ideal CSA policy delegates exactly the queries the local model cannot solve, approaching cloud-only accuracy at a fraction of the cost.

Test Case

Question: Find the number of positive integers $n \leq 600$ whose value can be uniquely determined when the values of $\lfloor n/4 \rfloor$, $\lfloor n/5 \rfloor$, and $\lfloor n/6 \rfloor$ are given.

Ground truth: ✔ 80

Model Attempts (×5)**:** ✘ 360 ✘ 204 ✘ 232 ✘ 504 ✘ 336

Vanilla

Decision: SELF_SOLVE

Analysis: I believe I can solve this problem by myself. The question involves determining how many positive integers $n \leq 600$ can be uniquely determined based on the values of $\lfloor n/4 \rfloor$, $\lfloor n/5 \rfloor$, and $\lfloor n/6 \rfloor$. This requires analyzing the relationships between these floor functions and determining for which values of n the combination uniquely identifies n. I can approach this by: (1) understanding the behavior of floor functions; (2) examining ranges of n where the floor values are constant; (3) determining for which n the combination of the three floor values uniquely determines n; (4) counting such n values up to 600. This is a problem that can be solved with logical reasoning and computational verification, which I can handle directly.

✘ *Failed to recognize own inability to solve this problem*

RLVR

Decision: DELEGATE

Analysis: The problem involves determining how many positive integers $n \leq 600$ can be uniquely determined based on the floor values of $n/4$, $n/5$, and $n/6$. This requires analyzing how different values of n map to these floor values and identifying which n correspond to a unique combination. This is a complex combinatorial and number-theoretic problem that involves checking for uniqueness in a range of values. It requires careful analysis of floor functions and their behavior, which can be quite involved. While it is possible to approach this problem with careful reasoning and computation, it is non-trivial and may benefit from systematic computation or more powerful logical reasoning capabilities. Therefore, this problem is best solved with a more powerful model that can handle the detailed computations and logical deductions required.

✔ *Correctly recognizes difficulty and delegates*

Figure 13: Full reasoning traces for the math case study: vanilla Qwen3-4B vs. RLVR-trained.

Test Case

Question: The complete resynthesis of phosphocreatine after very high intensity exercise normally takes:

Options:

A. about 20 minutes. B. about 10 minutes. C. about 5 seconds.
D. about 2 minutes. E. about 4 minutes. F. about 30 seconds.
G. about 1 minute. H. about 15 minutes. I. about 5 minutes. J. about 10 seconds.

Ground truth: ✔ E: about 4 minutes

Model attempts (×5)**:** ✘ D ✘ D ✘ D ✘ D ✘ D

Vanilla

Decision: SELF_SOLVE

Analysis: I believe I can solve this by myself because the question is related to physiological knowledge about energy metabolism in the body, specifically the resynthesis of phosphocreatine after intense exercise. Phosphocreatine is a high-energy phosphate compound that is rapidly used and replenished during short bursts of high-intensity activity. The resynthesis process is primarily dependent on the ATP-PCr system, which can quickly regenerate phosphocreatine from ATP and creatine. Based on standard physiological knowledge, complete resynthesis after very high intensity exercise typically takes about 15–30 seconds. Therefore, the correct answer is likely (F) about 30 seconds.

✘ *Confidently self-solves, but all five attempts are incorrect*

RLVR

Decision: DELEGATE

Analysis: The question asks about the time required for complete phosphocreatine resynthesis after very high intensity exercise. Phosphocreatine is rapidly used and replenished during short, high-intensity activities, and its resynthesis depends on ATP availability and creatine kinase activity. Most replenishment occurs within a few minutes, but determining the precise time requires specialized knowledge of exercise physiology. Since this is a specific domain fact, it requires a more powerful model to ensure accuracy.

✔ *Correctly recognizes uncertainty and delegates*

Figure 14: Case study on a science question: vanilla Qwen3-4B vs. RLVR-trained.

Random routing at a matched cloud-call budget. For the Math routing comparison in Table 7, random routing selects a uniformly random subset of exactly m out of $N = 490$ queries for cloud inference and answers the rest locally. We set m to the corresponding CSA policy's cloud-call count: 96 for Qwen3-1.7B and 73 for Qwen3-4B. Let $L_i, C_i \in \{0, 1\}$ indicate correctness of the stored single local and cloud answers to query i. The exact expected accuracy over random subsets is

$$\mathbb{E}[\mathrm{Acc}_{\mathrm{random}}] = \frac{1}{N} \sum_{i=1}^{N} \left[(1 - \rho) L_i + \rho C_i \right], \qquad \rho = \frac{m}{N}. \tag{19}$$

We use the same stored branch outputs to evaluate CSA and random routing. The expected random-routing accuracies are 0.6504 and 0.7182, compared with CSA accuracies of 0.7327 and 0.8000. These are exact expectations conditional on the recorded answers, not estimates from a single random subset. The comparison isolates query selection at the same cloud-call budget. The time and token

measurements in Table 7 come from a separate generation run of each CSA model at batch size 1. They measure assessment overhead, excluding local and cloud answer generation. The routing accuracies and cloud-call counts come from the stored routing evaluation.

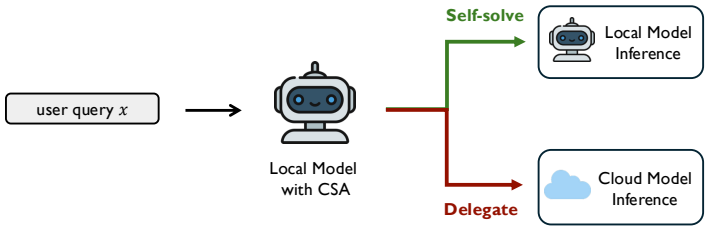


Figure 15: Illustration of *self-routing*. Upon receiving a user query x, a local model equipped with CSA assesses whether the query falls within its reliable solving capability and decides between two actions: SELF-SOLVE, where the local model answers directly, or DELEGATE, where the query is routed to a cloud model for inference.

D.12.2 COST-AWARE ROUTING AS AN EXTENSION OF RLVR-TRAINED CSA

Matching the single-attempt deployment setting. The Math any-correct label measures whether a model succeeds in at least one of five attempts. It does not represent single-attempt success: under independent attempts, a query with success probability 0.2 receives a SELF-SOLVE label with probability $1-(1-0.2)^5 \approx 0.67$. A deployed router that obtains only one answer from the selected solver instead benefits from predicting each action's expected outcome. The experiment below addresses this practical setting using an expected-outcome reward. Five sampled answers estimate each branch's single-call success probability. Evaluation averages these outcomes under the same routing protocol for every method, without selecting the best of five answers. We keep the local and cloud solvers fixed so that policy comparisons use the same branch outcomes.

From capability assessment to routing utility. The CSA policy decides whether a model should attempt a query or seek help. We extend this policy to a local–cloud setting in which the selected action is evaluated by the answer it produces and delegation incurs a cost (Chen et al., 2024; Ding et al., 2024; Ong et al., 2025; Aggarwal et al., 2024). The action space, assessment prompt, and RLVR training framework remain unchanged, and only the reward is replaced by downstream routing utility. For a query q, let $p_{\mathrm{L}}(q)$ and $p_{\mathrm{C}}(q)$ denote the probabilities that the local and cloud models answer correctly. The utility of action a is

$$\begin{cases} p_{\mathrm{L}}(q), & a = \text{SELF-SOLVE}, \\ p_{\mathrm{C}}(q) - \lambda C, & a = \text{DELEGATE}, \end{cases} \tag{20}$$

where C is the cost of a cloud call and λ controls the cost–accuracy trade-off. The corresponding oracle delegates exactly when

$$p_{\mathrm{C}}(q) - p_{\mathrm{L}}(q) > \lambda C. \tag{21}$$

This objective generalizes the original CSA decision. If local capability is binary, delegation always succeeds, and delegation is free, Eq. (20) recovers the rule of delegating precisely when the local model cannot solve the query. In the more realistic setting, delegation is rewarded only when its expected improvement exceeds its cost.

Models and datasets. We use Qwen3-1.7B as the local model and Qwen3-32B as the cloud model. We evaluate on the same Science domain as in the main experiments, comprising the biology, chemistry, health, and physics categories of MMLU-Pro with 3,266 training and 700 test queries. We set $C = 1$, so λ directly represents the penalty assigned to one cloud call.

Estimating action outcomes. For each query, we sample $K = 5$ answers independently from both the local and cloud models and compute empirical single-call success rates

$$\frac{1}{K}\sum_{k=1}^{K}\mathbf{1}\left[\hat{a}_m^{(k)}(q) = a^*(q)\right], \qquad m \in \mathrm{L}, \mathrm{C}. \tag{22}$$

We use these pass rates in Eq. (20).

For each value of λ, we apply the same two-stage RLVR procedure used for CSA. Diversity-Filtered Warm-up first retains training queries whose 16 policy rollouts contain both actions. Training then continues on the full distribution. GRPO uses 16 rollouts per prompt, a global batch size of 64, one epoch, a learning rate of 10^{-6}, a KL coefficient of 0.01, and full-parameter optimization with ZeRO-3 on four NVIDIA L40S GPUs. During training, we use the empirical expected-outcome reward

$$\begin{cases}\hat{p}_{\mathrm{L}}(q), & a = \textsc{Self-SOLVE},\\ \hat{p}_{\mathrm{C}}(q) - \lambda C, & a = \textsc{DELEGATE}.\end{cases} \tag{23}$$

This is the expectation of a sampled correctness reward under the empirical branch outcomes. Using the expectation removes within-action sampling variance, so the GRPO advantage reflects the quality of the routing decision. Unparseable policy outputs receive a reward of -1, below the reward of either valid action.

Cost grid. Because $K = 5$, each empirical success rate is quantized in increments of 0.2. We use

$$\lambda \in \{0.1, 0.3, 0.5, 0.7\}, \tag{24}$$

which places each cost between adjacent achievable outcome differences and avoids ties in Eq. (20). We train a separate policy for each value and report its realized delegation rate rather than assuming that the grid produces evenly spaced operating points.

Evaluation protocol. We evaluate every routing policy using frozen copies of the local and cloud solvers. Freezing both branches keeps the outcome estimates stationary and ensures that differences between policies arise from which queries they delegate rather than changes in solving ability. Policy decisions are sampled at temperature 1.0 over three random seeds. Let $d_\pi(q) = 1$ when policy π delegates and 0 otherwise. Its routed accuracy and delegation rate are

$$\mathrm{Acc}(\pi) = \frac{1}{N}\sum_{q}\left[\big(1 - d_\pi(q)\big)\hat{p}_{\mathrm{L}}(q) + d_\pi(q)\hat{p}_{\mathrm{C}}(q)\right], \tag{25}$$

$$\rho(\pi) = \frac{1}{N}\sum_{q} d_\pi(q). \tag{26}$$

We report the resulting cost–accuracy frontier in Figure 16.

The figure shows that outcome-aware RLVR is the strongest deployable routing method for Qwen3-1.7B on science. The trained RLVR policies lie above the baseline cluster.

The two dashed curves clarify why outcome-aware training helps. The upper oracle knows the benefit of delegation, $p_{\mathrm{cloud}} - p_{\mathrm{local}}$, whereas the lower oracle knows only whether the local model will succeed. The local model fails on 49.1% of queries, but the cloud model can rescue only 29.9 percentage points. On the remaining 19.2 points, both models fail. A capability-only signal cannot distinguish these cases and may therefore delegate queries for which a cloud call provides no benefit. By optimizing the outcomes of both actions, RLVR learns a substantially better routing policy.

The operating point still matters. With larger penalties, the policies converge to never delegating, partly because many training groups provide no decision-level learning signal. Nevertheless, the clear gains at matched delegation rates show that RLVR learns which queries benefit from cloud assistance substantially better than the available baselines.

D.12.3 Self-Guided Data Selection

Our main data-selection experiment tests whether examples preferred by an RLVR-trained CSA model are better fine-tuning data than examples selected by standard data-selection baselines. We

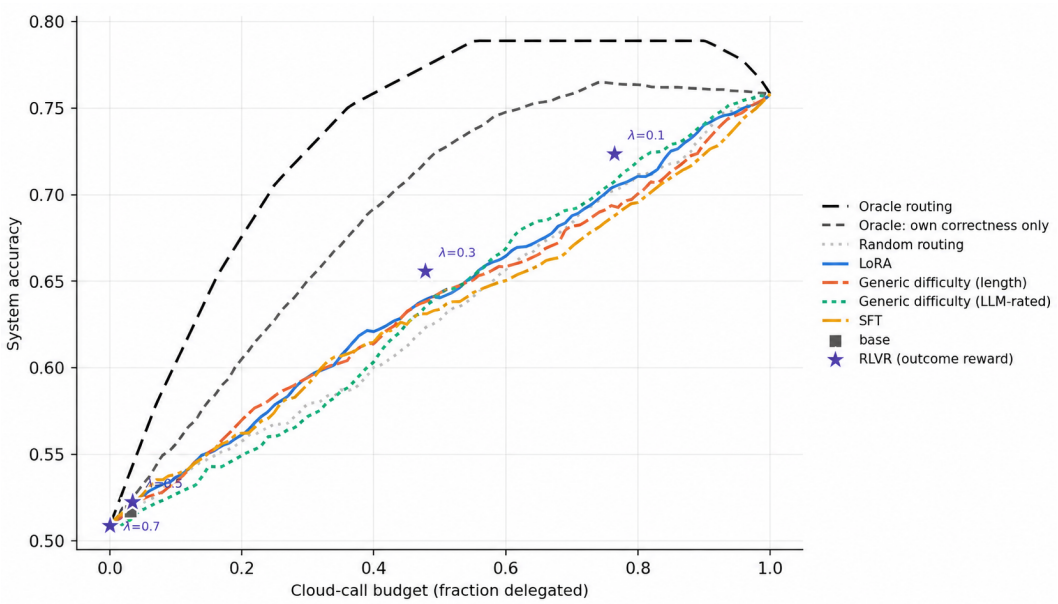


Figure 16: **Routing from Qwen3-1.7B to Qwen3-32B on science** ($n = 700$). Accuracy as a function of the cloud-call budget for RLVR and the baselines. Stars denote RLVR policies trained with different values of λ. The dashed envelopes show true-gain oracle routing (upper) and oracle routing based only on local-model correctness (lower). Their separation captures the additional value of knowing whether the cloud model can improve on the local model. RLVR outperforms all deployable baselines.

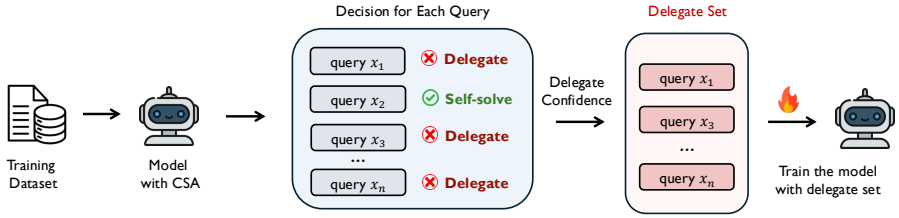


Figure 17: Overview of CSA-based data selection. A CSA-equipped model scores a fresh candidate pool, prioritizing examples it would delegate while filtering out extreme outliers. The selected subset is then used to fine-tune the model.

begin with a base model M_0 and train a CSA model M_{RLVR} to decide whether a problem should be answered directly or delegated. After training, the model is used as a scorer: it is applied to a fresh candidate training pool

$$\mathcal{D}_{\mathrm{pool}} = \{(x_i, y_i)\}_{i=1}^{N}, \tag{27}$$

and assigns each candidate a delegation-derived score. Let z_i^{self} and z_i^{del} denote the logits assigned by M_{RLVR} to the SELF_SOLVE and DELEGATE decision tokens for example x_i. We convert these logits into a normalized delegation probability

$$p_i^{\mathrm{del}} = \frac{\exp(z_i^{\mathrm{del}})}{\exp(z_i^{\mathrm{del}}) + \exp(z_i^{\mathrm{self}})}. \tag{28}$$

Intuitively, p_i^{del} estimates how strongly the trained CSA policies believe that M_0 should not solve x_i directly.

A naive delegate-only selector would choose examples with the largest p_i^{del}. However, this can over-select extreme examples that are far beyond the model's current capabilities or are noisy, ambiguous, or mismatched to the target distribution. To avoid this, we use a *delegate-based loss-bandpass* selector.

First, we compute the base model's supervised loss on each candidate:

$$\ell_i = -\frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \log p_{M_0}(y_{i,t} \mid x_i, y_{i,<t}). \tag{29}$$

We then retain only examples whose loss lies within a middle band:

$$\mathcal{B}_{\alpha,\beta} = \left\{ i : Q_\alpha(\{\ell_j\}_{j=1}^N) \le \ell_i \le Q_\beta(\{\ell_j\}_{j=1}^N) \right\}, \tag{30}$$

where Q_α and Q_β are empirical loss quantiles. This removes examples that are already too easy for M_0 as well as extreme high-loss outliers. Among the remaining examples, we rank by delegation probability:

$$s_i^{\mathrm{DLB}} = p_i^{\mathrm{del}} \cdot \mathbf{1}\{i \in \mathcal{B}_{\alpha,\beta}\}. \tag{31}$$

For a target selection budget k, the selected subset is

$$\mathcal{S}_{\mathrm{DLB}}(k) = \mathrm{TopK}_{i \in \{1,\dots,N\}} \left(s_i^{\mathrm{DLB}}, k \right). \tag{32}$$

We then fine-tune the model on $\mathcal{S}_{\mathrm{DLB}}(k)$. Figure 17 provides an overview of the method.

For the science-pool experiments, we train M_0 on MMLU-Pro (Wang et al., 2024) to get M_{RLVR}. The candidate pool is MedMCQA (Pal et al., 2022). We randomly sample $N = 50{,}000$ examples from the MedMCQA training split and score them using our method and all baselines. We conduct experiments on Qwen3-1.7B and Qwen3-4B (Yang et al., 2025). We use loss-bandpass quantiles $Q_\alpha = 0.2, Q_\beta = 0.8$. We compare delegate-based loss-bandpass against random selection, high-loss selection, low-loss selection, and development-set similarity selection. We also include an all-samples condition, which fine-tunes on the full 50,000-example candidate pool. For development-set similarity, we reserve a held-out set

$$\mathcal{D}_{\mathrm{dev}} = \{(x_j^{\mathrm{dev}}, y_j^{\mathrm{dev}})\}_{j=1}^m \tag{33}$$

of $m = 500$ randomly selected MedMCQA training examples that are excluded from the candidate pool.

The baseline selectors are defined as follows. Random selection samples a subset uniformly without replacement:

$$\mathcal{S}_{\mathrm{rand}}(k) \sim \mathrm{Unif}\left(\{\mathcal{S} \subset \mathcal{D}_{\mathrm{pool}} : |\mathcal{S}| = k\} \right). \tag{34}$$

High-loss selection ranks examples by the base model loss in Eq. (29) and selects the largest-loss examples:

$$\mathcal{S}_{\mathrm{high}}(k) = \mathrm{TopK}_i \left(\ell_i, k \right). \tag{35}$$

This baseline tests whether our method is simply performing hard-example mining. Low-loss selection selects the examples with the smallest base-model loss:

$$\mathcal{S}_{\mathrm{low}}(k) = \mathrm{TopK}_i \left(-\ell_i, k \right). \tag{36}$$

This baseline tests whether improvements can instead be obtained by emphasizing examples that are already easy and likely to be clean.

For development-set similarity, we embed each candidate x_i and each development example x_j^{dev} using a fixed sentence encoder $f(\cdot)$. We compute the maximum cosine similarity between each candidate and the held-out development set:

$$s_i^{\mathrm{sim}} = \max_{1 \le j \le m} \frac{f(x_i)^\top f(x_j^{\mathrm{dev}})}{\|f(x_i)\|_2 \, \|f(x_j^{\mathrm{dev}})\|_2}. \tag{37}$$

The selected subset is then

$$\mathcal{S}_{\mathrm{sim}}(k) = \mathrm{TopK}_i \left(s_i^{\mathrm{sim}}, k \right). \tag{38}$$

This baseline controls for the possibility that delegate-based selection works only because it selects examples similar to the evaluation distribution.

Finally, the all-samples condition uses

$$\mathcal{S}_{\text{all}} = \mathcal{D}_{\text{pool}}, \tag{39}$$

and therefore measures the performance obtained when no data selection is performed. All selected subsets of size k are fine-tuned using the same optimization hyperparameters, number of epochs, and training budget, so differences in downstream performance can be attributed to the selection rule rather than to changes in compute or training procedure.

Given the scored candidate pool, we run fixed-budget SFT experiments. For each selection method, we fine-tune on one of three budget fractions of the 50,000-example pool: 5%, 10%, and 20% corresponding to 2,500, 5,000, and 10,000 selected examples. Fine-tuning is performed with full-parameter training on NVIDIA L40S nodes. The main hyperparameters are a learning rate of 5×10^{-5}, per-device batch size 4, maximum sequence length 4096, maximum training steps 2000, and bfloat16 training. Evaluation is performed on the MedMCQA validation split rather than the official test split. Each run evaluates on 500 randomly sampled validation examples with a maximum generation length of 256 tokens. The evaluation set is identical for all methods. The primary metric is validation accuracy after fine-tuning.

E Prompt Templates

E.1 CSA Prompt Template

We use the following prompt template to elicit the model's Self-Solve / Delegate decision on each query. The template instructs the model to assess its own capability, and to emit its decision in a structured format consisting of an `<analysis>` block followed by a `<decision>` block. The same template is used for all subsequent CSA inference throughout the paper.

CSA Prompt Template

```
Decide whether you can answer
the user's query.
- Choose SELF_SOLVE if you believe you can solve it by yourself.
- Choose DELEGATE if you believe it requires a more powerful model.

# Output Format
<analysis>
Explain why you chose SELF_SOLVE or DELEGATE.
</analysis>

<decision>
SELF_SOLVE or DELEGATE
</decision>

# Query:
{query}
```

We parse the model's response by extracting the content of the `<decision>` block and matching it against Self-Solve or Delegate.

E.2 Self-Rationale Prompt (SFT_{Self})

The training model itself is prompted to produce a self-assessment justifying the ground-truth label. The placeholders `{query}` and `{decision}` are replaced with the query q_i and the label $y_i \in \{\text{Self_Solve}, \text{Delegate}\}$, respectively.

Self-Rationale Prompt Template

```
You are given a user query, along with a **ground-truth label** that
correctly reflects your capability.
```

```
The label is:
- SELF_SOLVE: you can solve the query by yourself.
- DELEGATE: the query requires a more powerful model.

For the following query: {query}

The ground-truth label is: {decision}

Write a coherent self-assessment explaining
**why this query should be classified as {decision}**.

You do NOT need to actually solve the query.
```

E.3 TEACHER-RATIONALE PROMPT ($\text{SFT}_{\text{TEACHER}}$)

A stronger teacher model $\mathcal{M}^*$ is prompted to produce a rationale on behalf of the target model. The teacher is instructed to write in the first-person perspective of the target model so that the resulting rationale stylistically resembles a self-assessment, even though it is generated externally.

Teacher-Rationale Prompt Template

```
You are analyzing the behavior of a target language model.

You are given a user query, along with a **ground-truth label** that
correctly reflects the target model's capability:
- SELF_SOLVE: the target model can solve the query by itself.
- DELEGATE: the query requires a more powerful model.

For the following user query:
{query}

The correct label is: {decision}

Your goal is to explain **why this query should be classified as
{decision} for the target model**.

Write the explanation from the **first-person perspective of the
target model**, as if it is assessing its own capability.

You do NOT need to actually solve the query.
```