SHARED EXPERIENCE, SEPARATE LEARNING: COMPANION CONFIDENCE CALIBRATION FOR LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reliable self-assessment is essential for large language models (LLMs), yet they often remain highly confident when their answers are wrong. We study *concurrent confidence calibration*, where confidence is learned alongside capability improvement rather than calibrated only after training. Reinforcement learning from verifiable rewards (RLVR) provides a natural setting for this paradigm, as it continuously produces responses paired with verifiable correctness feedback. Existing concurrent methods, however, learn both capability and confidence through reinforcement learning within shared policy parameters, potentially coupling two fundamentally different learning problems. We instead propose *shared experience but separate learning*: capability and confidence learn from the same trajectories, but through separate optimization mechanisms and parameters. Based on this principle, we introduce **CoCal (Companion Confidence Calibration)**, which trains a lightweight companion from rollout hidden states and verifier-derived correctness supervision while leaving task optimization unchanged. Experiments on Qwen3-8B and Qwen3-14B show that CoCal improves confidence estimation without sacrificing task performance, outperforming both RL-based concurrent methods and matched post-hoc calibration. The learned companion further generalizes across domains and policy shifts, while the benefits of CoCal persist at both scales.

1 INTRODUCTION

Large language models (LLMs) have become capable of solving complex tasks such as mathematical reasoning and software engineering (Guo et al., 2025; Yang et al., 2025; Jimenez et al., 2024; OpenAI, 2026; Anthropic, 2026). Yet they still struggle to recognize the limits of their own capabilities, often expressing high confidence in incorrect answers (Xiong et al., 2023; Ni et al., 2024; Damani et al., 2026; Ma et al., 2026; Liu et al., 2026; Luo et al., 2026). As LLMs take on more autonomous roles, reliable self-assessment becomes increasingly important for deciding when to answer, abstain, retrieve additional information, or allocate more inference compute.

Human self-assessment co-evolves with capability learning: as people acquire new knowledge and skills, experience continually refines their understanding of what they do and do not know (Flavell, 1979; Butler & Winne, 1995). LLM capability training provides a similar opportunity, exposing the model to a broad and evolving range of successes and failures across learning stages. In particular, early errors provide valuable evidence of capability limits but become increasingly scarce as performance improves. Moreover, capability training typically produces large volumes of responses and feedback, which can offer rich and diverse supervision for confidence learning while avoiding the additional sampling and evaluation required by post-hoc calibration (Yang et al., 2023; Ni et al., 2025c; Zhang et al., 2024). We call this paradigm-learning confidence alongside capability improvement-*concurrent confidence calibration*.

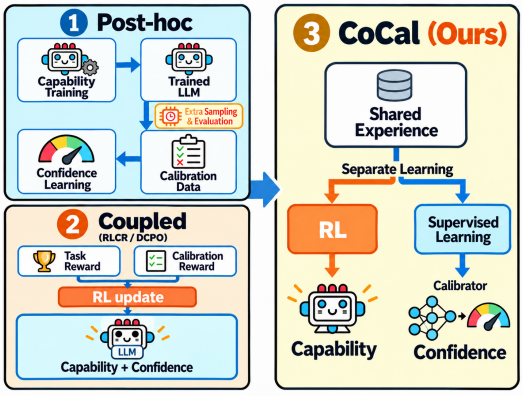

Figure 1: Comparison of post-hoc calibration, coupled concurrent training, and CoCal. CoCal learns from shared RLVR experience while separating capability and confidence learning.

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.

Extending this paradigm across the full model-training lifecycle would be prohibitively expensive. We therefore focus on RLVR (Reinforcement Learning from Verifiable Rewards) (Guo et al., 2025), a key post-training stage where responses are naturally paired with verifiable correctness feedback. Recent methods already perform concurrent calibration during RLVR, but formulate confidence learning as part of policy optimization. RLCR (Damani et al., 2026) adds confidence-specific reasoning and optimizes the full sequence with combined task and calibration rewards, while DCPO (Ma et al., 2026) reduces interference between the two through token-level objective separation. Despite these differences, both learn confidence through RL within the shared policy parameters.

We propose a different division of labor: *shared experience but separate learning* (Figure 1). Task learning needs to improve a policy over possible reasoning trajectories, for which RL is appropriate. Once a response has been generated and verified, however, its correctness provides a direct supervised signal for estimating confidence. Learning this scalar through another policy-gradient objective is not necessary for the formulation, and updating shared parameters gives the two objectives a route to interfere. We therefore reuse the policy's RLVR experience while separating how, and where, confidence is learned.

Based on this division of labor, we propose **CoCal (Companion Confidence Calibration)**. Inspired by prior findings that LLM internal representations contain signals predictive of response correctness (Su et al., 2024; Ni et al., 2025c), CoCal trains a lightweight companion on rollout hidden states and verifier-derived correctness targets. The companion learns from the same experience as the policy, but with separate parameters and direct supervision, leaving task optimization unchanged.

Experiments on Qwen3-8B and Qwen3-14B yield three main findings. 1) Compared to coupled RL-based methods, CoCal provides a stronger confidence estimation while preserving task performance; in five mathematical datasets, RLCR and DCPO trail CoCal in accuracy by 9.9 and 5.0 percentage points on average. 2) Compared with matched post-hoc calibration, CoCal achieves better confidence estimation while avoiding additional response sampling. 3) CoCal further widens its advantage over both RL-based concurrent methods and post-hoc calibration under math-to-factual QA transfer. The learned companion also transfers across policy shifts, and CoCal remains effective on both Qwen3-8B and Qwen3-14B.

2 Related Work

Unsupervised confidence estimation. Some unsupervised methods study how to estimate LLM confidence and can be broadly grouped into three categories. Probability-based methods derive confidence from token-level generation probabilities (Kadavath et al., 2022; Jiang et al., 2021). Verbalized methods directly ask models to report their confidence (Lin et al., 2022; Yin et al., 2023; Tian et al., 2023; Xiong et al., 2023; Ni et al., 2024). Consistency-based methods estimate confidence from agreement across multiple generations, including answer consistency and semantic uncertainty (Manakul et al., 2023; Zhang et al., 2023; Kuhn et al., 2023).

Post-hoc confidence calibration. Many supervised methods calibrate confidence after capability training using response correctness as supervision, either by fine-tuning the model to express calibrated confidence (Yang et al., 2023; Zhang et al., 2024) or by training separate predictors over internal representations (Azaria & Mitchell, 2023; Wang et al., 2024; Su et al., 2024; Ni et al., 2025c). Beyond correctness labels, prior work has incorporated additional signals such as self-consistency (Ni et al., 2025b), knowledge popularity (Ni et al., 2025a), and signals from other or base models (Luo et al., 2025a; Tan et al., 2026). Reinforcement learning has also been used for post-hoc calibration, including SaySelf (Xu et al., 2024) and Rewarding Doubt (Bani-Harouni et al., 2026). Despite these differences, these methods calibrate confidence after the target model has acquired its task capability.

Concurrent confidence calibration. More recent work studies confidence learning alongside capability improvement during RLVR. C^2GSPG (Liu et al., 2025) derives confidence from normalized sequence probabilities and jointly optimizes calibration and reasoning within the policy objective. RLCR (Damani et al., 2026) instead learns verbalized confidence through additional confidence-specific reasoning and a joint task–calibration reward, while DCPO (Ma et al., 2026) removes this extra reasoning and separates task and calibration updates across reasoning and confidence tokens. Despite these differences, capability and confidence remain coupled in both optimization and pa-

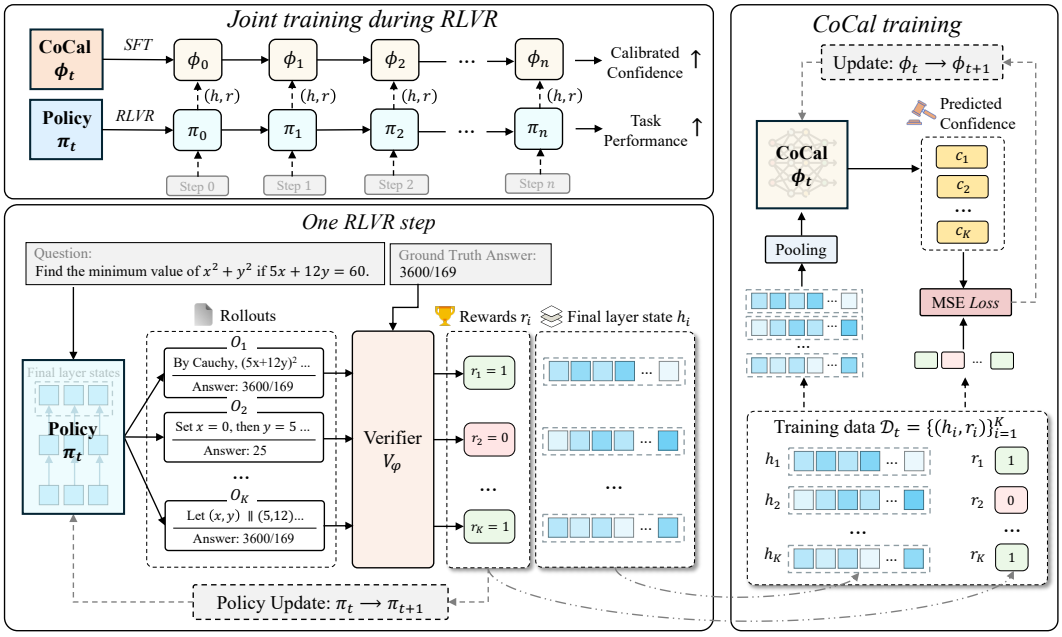

Figure 2: Overview of CoCal. Shared RLVR rollouts support task learning through policy optimization and confidence learning through direct supervision of a separate companion.

rameters: both are learned through policy optimization within the same model. CoCal follows the principle of shared experience but decoupled learning: it reuses RLVR rollouts to directly supervise a separate confidence predictor, while the policy continues to optimize task performance.

3 Preliminary

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is a representative algorithm for reinforcement learning with verifiable rewards (RLVR). Given a question q, the policy samples a group of G responses $\{o_i\}_{i=1}^G$, each receiving a reward r_i. GRPO estimates the advantage of each response from its reward relative to others in the same group:

$$A_i = \frac{r_i - \bar{r}}{\sigma_r}, \qquad \bar{r} = \frac{1}{G}\sum_{j=1}^{G} r_j, \tag{1}$$

where σ_r denotes the within-group reward standard deviation. The resulting advantage A_i is applied to the tokens of response o_i through a clipped policy-gradient objective. Following prior RLVR work (Yu et al., 2026; Damani et al., 2026; Ma et al., 2026), we use a rule-based verifier to assess final-answer correctness and use this signal as the task reward. As the policy evolves, its responses and correctness outcomes record successes and failures across different capability stages, providing a natural supervision stream for confidence calibration.

4 CoCal

CoCal implements *shared experience but decoupled learning* for concurrent confidence calibration. The policy learns task capability through RLVR, while a separate companion learns confidence through direct supervision on the same rollouts. Policy updates and companion updates use separate parameters (Figure 2).

4.1 Learning from RLVR Experience

During RLVR, the policy naturally generates rollouts together with verifier-provided correctness feedback. These signals not only support task learning, but can also provide supervision for confi-

dence learning. CoCal therefore reuses the rollouts and correctness feedback produced during policy training, avoiding the additional response sampling required by a separate post-hoc calibration stage.

At each training step t, the current policy π_t receives B questions $x_1, \ldots, x_B$ and generates K responses for each:

$$r_{i,j} \sim \pi_t(\cdot \mid x_i), \qquad y_{i,j} = V(x_i, r_{i,j}) \in \{0,1\}, \quad j = 1, \ldots, K, \tag{2}$$

where V is the verifier and $y_{i,j}$ denotes response correctness.

For task learning, the policy is updated by GRPO using response correctness as the reward. For confidence learning, we use the hidden representation $z_{i,j}$ extracted during response generation as the calibrator input and construct a confidence target $y_{i,j}^{\text{conf}}$ from the verifier feedback. To prevent confidence learning from interfering with task optimization while keeping the additional module lightweight, we introduce a separate four-layer MLP as the companion calibrator:

$$c_{i,j} = g_{\phi_t}(z_{i,j}) = \sigma(\text{MLP}_{\phi_t}(z_{i,j})) \in [0,1]. \tag{3}$$

The calibrator is trained with mean squared error:

$$\mathcal{L}_{\text{conf}}^{(t)} = \frac{1}{BK} \sum_{i=1}^{B} \sum_{j=1}^{K} \left(c_{i,j} - y_{i,j}^{\text{conf}}\right)^2. \tag{4}$$

During each chronological pass over the cached data, the companion consumes supervision in the order of the policy steps that produced it:

$$g_{\phi_0} \xrightarrow{\mathcal{S}_0} g_{\phi_1} \xrightarrow{\mathcal{S}_1} \cdots \xrightarrow{\mathcal{S}_{T-1}} g_{\phi_T}, \tag{5}$$

where $\mathcal{S}_t = \left\{\left(z_{i,j}, y_{i,j}^{\text{conf}}\right)\right\}_{i,j}$ denotes the confidence-supervision data produced at step t.

Practical implementation. Because companion optimization is fully decoupled from policy training, the two updates need not be physically interleaved. For implementation convenience, we cache rollout states and verifier labels and train the companion offline by replaying them in chronological order. This provides a practical approximation to online companion learning while preserving supervision from intermediate policy stages, and enables flexible training and evaluation without modifying the RLVR pipeline.

4.2 Response Representation and Supervision

Response representation. For a response $r = (w_1, \ldots, w_L)$, let $h_m^{(\ell)}$ denote the hidden state of its m-th token at Transformer layer ℓ. Prior work shows that deeper Transformer layers encode richer semantic and correctness-related information, with final-layer states providing strong signals for response reliability (Chuang et al., 2024; Ju et al., 2024; Su et al., 2024). We therefore extract representation from the final layer. To capture both global response-level information and fine-grained signals distributed across the response, we concatenate the final-token hidden state with the mean-pooled hidden state.

$$\bar{h}^{(\ell)} = \frac{1}{L} \sum_{m=1}^{L} h_m^{(\ell)}, \qquad z^{(\ell)} = h_L^{(\ell)} \| \bar{h}^{(\ell)}. \tag{6}$$

Our analysis in §6.4 shows that CoCal is robust to token selection and that middle-to-late layers consistently provide strong confidence signals.

Supervision construction. For constructing the confidence target $y_{i,j}^{\text{conf}}$ of a given response $r_{i,j}$, we consider two strategies:

- **Binary supervision (CoCal (binary)).** We directly supervise whether the response is correct by using the verifier-provided correctness label as the confidence target: $y_{i,j}^{\text{conf}} = y_{i,j}$. This provides direct supervision for distinguishing correct from incorrect responses.
- **Fine-grained supervision (CoCal (group)).** To teach the calibrator to distinguish consistently correct responses from occasional successes, we additionally incorporate the success rate of question x_i. Let $\bar{y}_i = \frac{1}{K} \sum_{j=1}^{K} y_{i,j}$ denote the group accuracy for question x_i. We set $y_{i,j}^{\text{conf}} = \bar{y}_i$ for a correct response ($y_{i,j} = 1$), and $y_{i,j}^{\text{conf}} = 0$ otherwise. This provides finer-grained supervision among correct responses according to how reliably the policy solves the question.

4.3 Inference and Discussion

At inference time, the calibrator predicts confidence from the internal states produced as the policy generates response r:

$$c(r) = g_\phi(z(r)).$$

This design offers several practical advantages. First, confidence learning is separated from policy optimization, avoiding interference with task learning. Second, CoCal reuses the large and diverse experience naturally produced throughout capability training, which may improve confidence estimation and generalization while eliminating the additional response sampling required by post-hoc calibration. Third, the companion is lightweight, introducing negligible training and inference overhead relative to the LLM (Table 2). The design also entails two trade-offs. CoCal requires access to internal hidden states and is therefore limited to white-box settings. Moreover, confidence is represented by an external companion rather than internalized into the policy, so CoCal does not directly improve the model's own verbalized confidence.

5 Experimental Setup

In this section, we describe the experimental setup, including datasets, models, baselines, evaluation metrics, and training protocol. More details are provided in § A.

Datasets. We organize the datasets into three groups: training data, mathematical evaluation, and out-of-distribution (OOD) evaluation.

Training. As in (Ma et al., 2026), we use DeepScaleR (Luo et al., 2025b), which contains approximately 40K mathematical problems with reference answers.

Evaluation on mathematics. We evaluate on five mathematical benchmarks spanning a range of difficulty levels. MATH500 (Lightman et al., 2024) contains 500 problems across seven mathematical subjects and five difficulty levels, providing broad coverage from relatively basic to challenging reasoning. AMC 2023 (Math-AI Team) and AMC 2024 (rawsh) contain 40 and 45 problems, respectively, drawn from high-school mathematics competitions with a more concentrated competition-level difficulty. AIME 2024 (Zhang & Math-AI, 2024) and AIME 2025 (Zhang & Math-AI, 2025) each contain 30 more challenging invitational problems that typically require deeper reasoning.

Evaluation on factual QA. We evaluate cross-domain transfer on the OOD split of HonestyBench (Ni et al., 2025b), a large-scale factual QA benchmark with over 560K training examples and about 70K evaluation examples. We use five datasets from HonestyBench-Eval (hereafter HonestyBench): the single-hop datasets SQuAD (Rajpurkar et al., 2016) and WebQuestions (Berant et al., 2013), the multi-hop datasets ComplexWebQuestions (CWQ) (Talmor & Berant, 2018) and MuSiQue (Trivedi et al., 2022), and the factual-triple-based PopQA (Mallen et al., 2022).

LLMs and baselines. We use Qwen3-8B and Qwen3-14B (Yang et al., 2025) as backbone models in non-thinking mode. We compare CoCal with seven baselines spanning training-free estimation, post-hoc calibration, and concurrent calibration. More details are provided in Appendix B.

- *Training-free estimation.* **Verbal** (Ni et al., 2024) directly prompts the model to report a confidence score together with its answer. **SelfConsis** (Manakul et al., 2023) estimates confidence from agreement across multiple sampled responses. **TokenProb** derives confidence from the token-level generation probability of the response (Kuhn et al., 2023). **P(True)** (Kadavath et al., 2022) asks the model to judge whether its own response is correct and uses the probability assigned to "True" as confidence.
- *Post-hoc calibration.* **PostCal** (Ni et al., 2025c) learns confidence after capability training. Using the final trained policy, it samples the same questions and the same number of responses per question as CoCal to construct the training data. It uses the same architecture and targets, with its training configuration detailed in Appendix B.
- *Concurrent calibration.* **RLCR** (Damani et al., 2026) adds confidence-specific reasoning after the answer and jointly optimizes task correctness and confidence calibration with a combined reward. **DCPO** (Ma et al., 2026) removes this additional reasoning and separates task and calibration

optimization at the token level, applying different advantages to reasoning-and-answer tokens and confidence tokens. Both methods learn confidence alongside capability improvement through policy optimization within the same model.

Metrics. We use accuracy to evaluate task performance, expected calibration error (ECE) (Guo et al., 2017) to measure confidence calibration, and the area under the receiver operating characteristic curve (AUROC) (Fawcett, 2006) to assess how well confidence distinguishes correct from incorrect responses. Higher accuracy and AUROC indicate better performance, while lower ECE indicates better calibration. Formal definitions are provided in Appendix A.2.

Evaluation Protocol. To reduce generation randomness, we sample ten responses per question on datasets with few questions, including AIME24/25 and AMC23/24. For all other datasets, we evaluate a single response per question. Accuracy is computed over all responses, while confidence metrics are computed only for responses with valid confidence scores. For CoCal, we independently train and evaluate the calibrator with five random seeds and report the average results.

6 Results and Analysis

In this section, we aim to answer: (1) within concurrent confidence calibration, does decoupled companion learning outperform optimizing confidence through RL within the shared policy; (2) what advantages does concurrent calibration offer over post-hoc calibration; and (3) how well does the learned companion generalize across domains and policy shifts? We further analyze the effects of model scale, token aggregation, representation layer selection, and pre-generation confidence estimation, with full results provided in Appendix §F.1–F.4.

6.1 CoCal Outperforms Existing Concurrent Methods

We first compare CoCal with training-free and RL-based concurrent confidence estimation methods. The comparison with PostCal is discussed separately in §6.2. Results are shown in Table 1. We find:

CoCal achieves strong confidence estimation with negligible additional inference overhead. As shown in Table 1, CoCal (binary) achieves the lowest ECE (0.096) among concurrent methods and an AUROC of 0.896, outperforming RLCR (0.105/0.857) and DCPO (0.139/0.831). This advantage emerges early: at step 40, CoCal already reaches an ECE of around 0.10, while RLCR and DCPO remain relatively weak and improve substantially only by step 80; even the first 10 rollout steps provide effective supervision for useful confidence estimation (Appendix Figure 7). Training-free methods are generally overconfident. SelfConsis is the strongest among them, with comparable AUROC but higher ECE and the cost of ten generations per question. As shown in Table 2, a single LLM generation takes 0.372 seconds on average, whereas CoCal adds only 0.199 ms per response.

Coupled concurrent confidence calibration can interfere with task learning. Figure 4 compares RLCR and DCPO with CoCal across training stages. RLCR and DCPO consistently achieve lower task accuracy, ending 9.9 and 5.0 percentage points below CoCal, respectively. RLCR attains stronger confidence estimation than DCPO but at a larger accuracy cost, suggesting a capability–calibration trade-off under shared policy optimization. This difference is consistent with their optimization scopes: RLCR applies confidence-related optimization to the full sequence, whereas DCPO restricts it to confidence tokens. Both methods also show declining accuracy as confidence estimation improves, while CoCal remains comparatively stable.

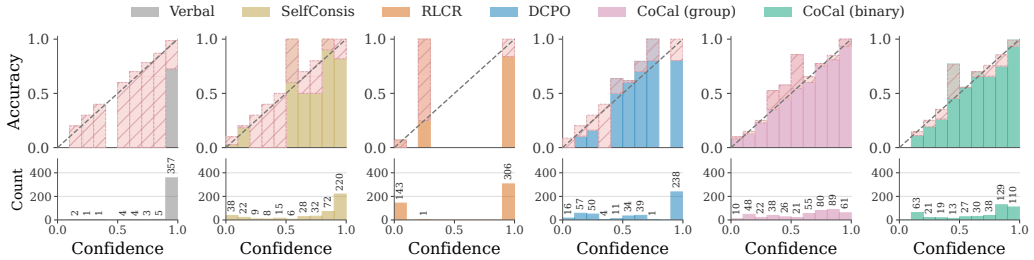

Figure 3: Reliability diagrams and confidence histograms on AMC24 after 120 training steps. Upper panels show bin accuracy and the ideal-calibration diagonal; lower panels show response counts.

Table 1: Accuracy and confidence quality on five mathematical datasets using Qwen3-8B after 120 training steps. Base + Verbal uses the pre-RL model; other training-free baselines use the GRPO-trained model. Bold and underlined values indicate the best and second-best results, respectively.

Method	MATH500				AIME24				AIME25			
	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑
Base + Verbal	79.8	0.995	0.196	0.566	25.0	0.972	0.718	0.587	18.7	0.977	0.790	0.730
Verbal	85.2	0.997	0.059	0.548	43.0	0.956	0.498	0.877	33.0	0.963	0.646	0.865
SelfConsis	86.0	0.868	**0.026**	**0.945**	**47.3**	0.576	0.118	**0.937**	**34.7**	0.603	0.257	0.869
TokenProb	86.0	0.819	0.087	0.834	**47.3**	0.668	0.246	0.873	**34.7**	0.638	0.330	0.943
P(True)	86.0	0.988	0.128	0.861	**47.3**	0.960	0.486	0.908	**34.7**	0.981	0.634	0.927
RLCR	85.6	0.908	0.064	0.797	32.3	0.338	**0.087**	0.906	28.3	0.333	**0.050**	0.965
DCPO	**86.2**	0.956	0.072	0.687	39.3	0.532	0.157	0.926	30.7	0.492	0.191	0.956
PostCal (group)	86.0	0.849	0.044	0.870	**47.3**	0.446	0.133	0.910	**34.7**	0.424	0.158	0.948
PostCal (binary)	86.0	0.887	0.049	0.754	**47.3**	0.598	0.134	0.894	**34.7**	0.566	0.245	**0.967**
CoCal (group)	86.0	0.826	0.049	0.911	**47.3**	0.429	0.100	0.910	**34.7**	0.408	0.128	0.936
CoCal (binary)	86.0	0.843	0.038	0.912	**47.3**	0.505	0.102	0.910	**34.7**	0.456	0.147	0.944

Method	AMC23				AMC24				Macro Average			
	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑
Base + Verbal	69.8	0.991	0.297	0.633	43.3	0.980	0.539	0.638	47.3	0.983	0.508	0.631
Verbal	74.0	0.984	0.209	0.633	**65.8**	0.969	0.282	0.692	60.2	0.974	0.339	0.723
SelfConsis	**85.8**	0.846	**0.064**	**0.967**	65.1	0.789	0.181	0.801	**63.8**	0.736	0.129	**0.904**
TokenProb	**85.8**	0.744	0.131	0.848	65.1	0.716	0.066	0.718	**63.8**	0.717	0.172	0.843
P(True)	**85.8**	0.991	0.134	0.770	65.1	0.982	0.331	0.830	**63.8**	0.980	0.343	0.859
RLCR	63.5	0.724	0.191	0.769	59.6	0.681	0.133	0.848	53.9	0.597	0.105	0.857
DCPO	79.0	0.814	0.122	0.779	58.9	0.714	0.152	0.806	58.8	0.702	0.139	0.831
PostCal (group)	**85.8**	0.684	0.185	0.852	65.1	0.624	0.080	0.802	**63.8**	0.605	0.120	0.876
PostCal (binary)	**85.8**	0.794	0.083	0.878	65.1	0.710	0.090	**0.867**	**63.8**	0.711	0.120	0.872
CoCal (group)	**85.8**	0.681	0.176	0.853	65.1	0.619	**0.065**	0.808	**63.8**	0.593	0.103	0.884
CoCal (binary)	**85.8**	0.740	0.118	0.866	65.1	0.676	0.077	0.851	**63.8**	0.644	**0.096**	0.896

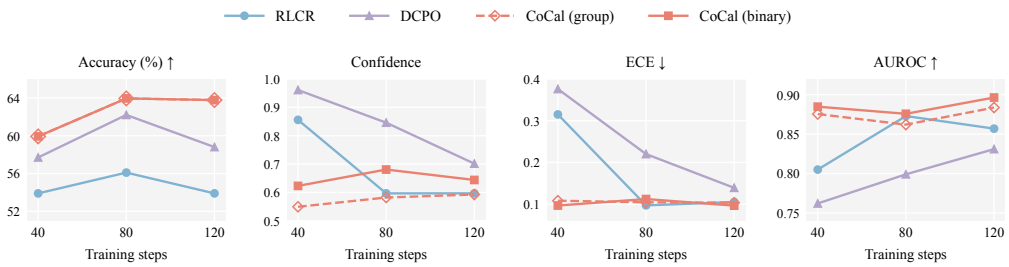

Figure 4: Task accuracy, confidence, calibration, and discrimination across policy-training stages on five mathematical datasets. CoCal results average five seeds; both variants share the same task accuracy.

Direct supervision yields more fine-grained confidence estimates. Figure 3 shows that RLCR produces highly polarized confidence estimates on AMC24, assigning over 99% of responses to the lowest or highest bin, while DCPO remains concentrated at high confidence. In contrast, both CoCal variants spread predictions across a broader range. Binary supervision directly targets response correctness and performs better overall on mathematics, whereas group supervision incorporates question-level reliability and produces more conservative estimates. Corresponding results for all five datasets are provided in Appendix Figure 8.

6.2 Benefits of CoCal over Post-hoc Calibration

We next examine whether confidence is better learned alongside capability improvement. We compare CoCal with PostCal, which uses the same calibrator architecture and supervision but learns from newly sampled responses of the final policy. Table 1 reports confidence performance on mathematics, while Table 2 compares computational cost. Detailed settings are provided in Appendix §E.

Concurrent calibration yields better confidence estimates. On the mathematical benchmarks in Table 1, both CoCal variants achieve lower ECE and higher AUROC than their PostCal counterparts. This advantage is consistent across matched training data budgets: CoCal obtains lower ECE and higher AUROC in seven of eight settings (Appendix Figure 9). The performance gap widens further under math-to-factual transfer, as discussed in §6.3, consistent with the benefit of learning confidence from broader experience across capability-learning stages.

CoCal avoids additional response sampling. CoCal reuses rollouts already produced during policy training, whereas PostCal needs to sample new additional responses from the model after capability training. This extra sampling takes 14.22 hours with vLLM (Kwon et al., 2023) on a single B200 GPU (Table 2). By contrast, the confidence predictor itself is lightweight, requiring only about 15 s of training and 0.199 ms per-response inference, compared with 0.372 s for a single LLM generation. Detailed cost settings are provided in Appendix §E.2.

Table 2: Training and per-question inference costs. LLM inference latency is averaged over MATH500 questions using vLLM on a B200 GPU.

Method	Training			Inference	
	LLM (h)	Sampling (h)	MLP (s)	LLM (s)	MLP (ms)
CoCal	16.67	–	15.1	0.372	0.199
PostCal	16.67	14.22	$\approx$ 15.1	0.372	0.199
SelfConsis	16.67	–	–	3.225	–

Table 3: Math-to-factual transfer on HonestyBench using Qwen3-8B after 120 training steps. Base + Verbal uses the pre-RL model; other training-free baselines use the GRPO-trained model. Bold and underlined values indicate the best and second-best results, respectively.

Method	SQuAD				WebQuestions				ComplexWebQuestions			
	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑
Base + Verbal	30.7	0.923	0.620	0.647	53.9	0.983	0.442	0.608	33.2	0.947	0.625	0.632
Verbal	31.1	0.907	0.603	0.656	53.7	0.978	0.441	0.623	33.0	0.918	0.596	0.677
SelfConsis	**31.7**	0.749	0.432	0.687	**57.1**	0.755	0.211	**0.712**	**34.2**	0.643	0.301	0.754
TokenProb	**31.7**	0.871	0.554	0.667	**57.1**	0.883	0.313	0.648	**34.2**	0.881	0.539	**0.759**
P(True)	**31.7**	0.845	0.554	0.666	**57.1**	0.886	0.374	0.643	**34.2**	0.809	0.501	0.694
RLCR	30.6	0.570	0.284	0.741	56.1	0.898	0.345	0.601	33.8	0.603	0.354	0.677
DCPO	31.5	0.751	0.444	0.712	54.8	0.939	0.392	0.619	33.0	0.767	0.455	0.672
PostCal (group)	**31.7**	0.344	0.143	0.668	**57.1**	0.448	0.237	0.629	**34.2**	0.315	0.155	0.645
PostCal (binary)	**31.7**	0.431	0.176	0.677	**57.1**	0.550	0.236	0.648	**34.2**	0.365	0.164	0.651
CoCal (group)	**31.7**	0.352	**0.037**	0.738	**57.1**	0.442	0.143	0.700	**34.2**	0.331	**0.032**	0.707
CoCal (binary)	**31.7**	0.391	0.074	**0.747**	**57.1**	0.544	**0.130**	0.700	**34.2**	0.352	0.044	0.710

Method	MuSiQue				PopQA				Macro Average			
	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑
Base + Verbal	**16.1**	0.849	0.770	0.468	22.1	0.953	0.732	0.543	31.2	0.931	0.638	0.580
Verbal	15.3	0.793	0.699	0.534	22.4	0.946	0.723	0.569	31.1	0.908	0.612	0.612
SelfConsis	11.3	0.503	0.390	**0.747**	21.5	0.490	0.275	**0.794**	31.1	0.628	0.322	**0.739**
TokenProb	11.3	0.846	0.733	0.722	21.5	0.870	0.655	0.788	31.1	0.870	0.559	0.717
P(True)	11.3	0.787	0.678	0.718	21.5	0.749	0.550	0.747	31.1	0.815	0.531	0.694
RLCR	13.4	0.323	0.289	0.649	22.2	0.702	0.497	0.662	31.2	0.619	0.354	0.666
DCPO	15.9	0.548	0.468	0.535	**22.8**	0.884	0.658	0.582	**31.6**	0.778	0.483	0.624
PostCal (group)	11.3	0.233	**0.127**	0.662	21.5	0.400	0.218	0.654	31.1	0.348	0.176	0.651
PostCal (binary)	11.3	0.268	0.155	0.661	21.5	0.508	0.299	0.701	31.1	0.425	0.206	0.667
CoCal (group)	11.3	0.264	0.151	0.729	21.5	0.375	**0.161**	0.751	31.1	0.353	**0.104**	0.725
CoCal (binary)	11.3	0.269	0.156	0.729	21.5	0.422	0.207	0.745	31.1	0.396	0.122	0.726

6.3 CoCal Generalizes across Domains and Policies

We assess the generalization of CoCal along two complementary dimensions. Cross-domain transfer tests whether confidence learned from mathematical reasoning transfers to factual QA, while cross-policy transfer tests whether the learned companion can be reused across different models.

CoCal gains a larger advantage under domain shift. Table 3 shows that CoCal transfers effectively from mathematical reasoning to factual QA, and its advantage over both RL-based concurrent methods and PostCal becomes substantially larger under domain shift. With group supervision, CoCal achieves an ECE of 0.104, compared with 0.354 for RLCR, 0.483 for DCPO, and 0.176 for PostCal. With binary supervision, CoCal reaches an AUROC of 0.726, compared with 0.666, 0.624, and

0.667, respectively. The widening gap relative to PostCal is consistent with the benefit of broader supervision across capability-learning stages, which may improve cross-domain generalization of the learned confidence. Among training-free methods, SelfConsis retains strong discrimination with an AUROC of 0.739, but its ECE rises to 0.322. Verbal, TokenProb, and P(True) show similar overconfidence. Overall, CoCal remains well calibrated while retaining competitive discrimination under substantial domain shift.

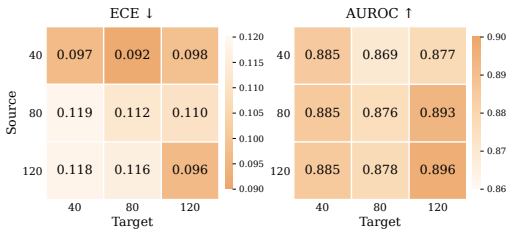

Figure 5: Cross-iteration transfer of CoCal. Source indicates the rollout-data prefix used for companion training, and target the policy used for evaluation.

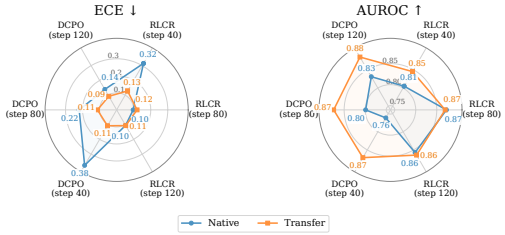

Figure 6: Cross-policy transfer of CoCal across training methods. The calibrator is applied to models under DCPO and RLCR after the same training steps.

CoCal also transfers across policy changes. Figure 5 evaluates transfer between policies from different stages of capability learning using CoCal (binary). Across the six off-diagonal source–target pairs, AUROC remains between 0.869 and 0.893 and ECE between 0.092 and 0.119, showing that the learned confidence signal remains stable as the policy evolves. Figure 6 further evaluates transfer to RLCR- and DCPO-trained policies. Transfer improves both ECE and AUROC across all three DCPO stages and for RLCR at step 40, but becomes less consistent for later RLCR policies. A likely reason is that RLCR alters the generation process more substantially by introducing confidence-specific reasoning, creating a larger shift in internal representations. Overall, CoCal transfers reliably across training stages and remains useful under independently optimized policies, although larger representation shifts can reduce transfer quality.

6.4 Further Analysis

We further analyze CoCal along four dimensions: model scale, token aggregation, representation layer, and pre-generation confidence estimation. CoCal remains effective when scaling from Qwen3-8B to Qwen3-14B and is robust to different token aggregation strategies. Middle-to-late Transformer layers provide similarly strong confidence signals, suggesting that there is no need for careful layer selection. Prompt-only states also support informative pre-generation estimates; although less accurate than post-generation confidence, they reduce time to confidence by roughly 10^2–$10^3\times$ by avoiding response generation. Due to space constraints, full results are provided in Appendix §F.1-F.4.

7 Conclusion

We introduced CoCal, a companion approach to concurrent confidence calibration built on the principle of *shared experience but separate learning*. Rather than learning confidence through policy optimization, CoCal trains a lightweight predictor directly from RLVR rollouts and verifier feedback while leaving task learning unchanged. Across Qwen3-8B and Qwen3-14B, this simple separation yields strong calibration and discrimination without sacrificing task accuracy, with useful confidence signals emerging from early training experience and transferring across domains and policy states. CoCal also generally outperforms post-hoc calibration while eliminating the need for additional response sampling after capability training. These results suggest that confidence need not be entangled with capability optimization: the two can co-evolve from the same experience through separate learning processes.

Ethics Statement

This work uses publicly available models and datasets for mathematical reasoning and factual question answering. Our research focuses on improving confidence estimation in large language models to support more reliable assessment of their responses.

Reproducibility Statement

We provide the experimental details needed to facilitate reproduction of our main results. Section 5 describes the experimental setup, baselines, and evaluation protocol, while Appendix A provides implementation details including model architecture, training hyperparameters, data splits, and random seeds. Additional experimental settings and results are provided in Appendices C–F.

AI Use Statement

Generative AI tools were used for language editing, figure presentation, manuscript organization, and refining the presentation of experimental analyses. The research motivation, methodology, experimental design, implementation, and experimental results were produced by the authors. All AI-assisted text and presentation suggestions were reviewed and verified by the authors, who take full responsibility for the final content of the paper.

References

Anthropic. Claude fable 5 and claude mythos 5 system card. https://www.anthropic.com/system-cards, June 2026. Accessed 2026-09-17.

Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying. *arXiv preprint arXiv:2304.13734*, 2023.

David Bani-Harouni, Chantal Pellegrini, Paul Stangel, Ege Özsoy, Kamilia Zaripova, Nassir Navab, and Matthias Keicher. Rewarding doubt: A reinforcement learning approach to calibrated confidence expression of large language models. In *International Conference on Learning Representations*, volume 2026, pp. 117927–117943, 2026.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1160.

Deborah L Butler and Philip H Winne. Feedback and self-regulated learning: A theoretical synthesis. *Review of educational research*, 65(3):245–281, 1995.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In *International Conference on Learning Representations*, volume 2024, pp. 54158–54183, 2024.

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 *International conference on learning representations*, volume 2026, pp. 59456–59493, 2026.

Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

John H Flavell. Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American psychologist*, 34(10):906, 1979.

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.

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.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Tianjie Ju, Weiwei Sun, Wei Du, Xinwei Yuan, Zhaochun Ren, and Gongshen Liu. How large language models encode context knowledge? a layer-wise probing study. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 8235–8246, 2024.

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.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

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. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

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

Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*, 2022.

Haotian Liu, Shuo Wang, and Hongteng Xu. C^2gspg: Confidence-calibrated group sequence policy gradient towards self-aware reasoning. *arXiv preprint arXiv:2509.23129*, 2025.

Xun Liu, Yi Evie Zhang, Vira Kasprova, Parisa Rabbani, Pardis Sadat Zahraei, Tianyu Zhang, Ali Ebrahimpour-Boroojeny, and Varun Chandrasekaran. Agentabstain: Do llm agents know when not to act? *arXiv preprint arXiv:2607.10059*, 2026.

Beier Luo, Shuoyuan Wang, Sharon Li, and Hongxin Wei. Your pre-trained LLM is secretly an unsupervised confidence calibrator. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/ef6b667990416111e3ebbeeba4a8f283-Abstract-Conference.html.

Han Luo, Bingbing Wen, and Lucy Lu Wang. Agentic abstention: Do agents know when to stop instead of act? *arXiv preprint arXiv:2606.28733*, 2026.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025b. Notion Blog.

Zhengzhao Ma, Xueru Wen, Boxi Cao, Yaojie Lu, Hongyu Lin, Jinglin Yang, Min He, Xianpei Han, and Le Sun. Decoupling reasoning and confidence: Resurrecting calibration in reinforcement learning from verifiable rewards. *arXiv preprint arXiv:2603.09117*, 2026.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*, 2022.

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.

Math-AI Team. American mathematics competitions (amc) 2023. Hugging Face dataset. URL https://huggingface.co/datasets/math-ai/amc23.

Shiyu Ni, Keping Bi, Jiafeng Guo, and Xueqi Cheng. When do llms need retrieval augmentation? mitigating llms' overconfidence helps retrieval augmentation. *arXiv preprint arXiv:2402.11457*, 2024.

Shiyu Ni, Keping Bi, Jiafeng Guo, and Xueqi Cheng. How knowledge popularity influences and enhances llm knowledge boundary perception. *arXiv preprint arXiv:2505.17537*, 2025a.

Shiyu Ni, Keping Bi, Jiafeng Guo, Minghao Tang, Jingtong Wu, Zengxin Han, and Xueqi Cheng. Annotation-efficient universal honesty alignment. *arXiv preprint arXiv:2510.17509*, 2025b.

Shiyu Ni, Keping Bi, Jiafeng Guo, Lulu Yu, Baolong Bi, and Xueqi Cheng. Towards fully exploiting llm internal states to enhance knowledge boundary perception. *arXiv preprint arXiv:2502.11677*, 2025c.

OpenAI. Gpt-6 astra system card. https://deploymentsafety.openai.com/gpt-6-astra, September 2026. Accessed 2026-09-17.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

rawsh. American mathematics competitions (amc) 2024. Hugging Face dataset. URL https://huggingface.co/datasets/rawsh/2024_AMC12.

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.

Weihang Su, Changyue Wang, Qingyao Ai, Yiran Hu, Zhijing Wu, Yujia Zhou, and Yiqun Liu. Unsupervised real-time hallucination detection based on the internal states of large language models. *arXiv preprint arXiv:2403.06448*, 2024.

Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 641–651, 2018.

Hexiang Tan, Wanli Yang, Junwei Zhang, Xin Chen, Rui Tang, Du Su, Jingang Wang, Yuanzhuo Wang, Fei Sun, and Xueqi Cheng. BaseCal: Unsupervised confidence calibration via base model signals. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2026. URL https://aclanthology.org/2026.acl-long.234/.

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. *arXiv preprint arXiv:2305.14975*, 2023.

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

Yanling Wang, Haoyang Li, Hao Zou, Jing Zhang, Xinlei He, Qi Li, and Ke Xu. Hidden question representations tell non-factuality within and across large language models. *arXiv e-prints*, pp. arXiv–2406, 2024.

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.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yuqing Yang, Ethan Chern, Xipeng Qiu, Graham Neubig, and Pengfei Liu. Alignment for honesty. *arXiv preprint arXiv:2312.07000*, 2023.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. Do large language models know what they don't know? *arXiv preprint arXiv:2305.18153*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

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. 7106–7132, 2024.

Jiaxin Zhang, Zhuohang Li, Kamalika Das, Bradley A Malin, and Sricharan Kumar. Sac3: Reliable hallucination detection in black-box language models via semantic-aware cross-check consistency. *arXiv preprint arXiv:2311.01740*, 2023.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An llm post-training framework for rl scaling. `https://github.com/THUDM/slime`, 2025. GitHub repository. Corresponding author: Xin Lv.

A Implementation Details

A.1 Policy and Companion Training

Policies are trained with `slime` (Zhu et al., 2025) for 120 steps and evaluated every 40 steps (Ma et al., 2026). Each step samples eight responses for each of 256 questions.

Feature extraction. We extract response hidden states from the final Transformer layer before the final RMSNorm, excluding prompt tokens. For calibrator training and validation, we discard responses that reach the maximum generation length, as these truncated outputs are incomplete and may provide unreliable response representations and correctness supervision. For responses that terminate normally, the generated end-of-sequence token is included in the extracted representation.

Calibrator training. For practical training, we cache the rollout hidden states and correctness labels produced at each RLVR step and replay them offline in chronological order to emulate step-wise companion learning. For the calibrator associated with policy step t, we use all data produced up to that step, ordered by their policy-training steps.

The MLP uses the widths and optimization settings in Table 4, with LayerNorm, GELU (Hendrycks & Gimpel, 2016), dropout rates of 0.1/0.1/0.05 after the first three hidden layers, and a sigmoid output. We split questions 90:10 into training and validation sets, keeping all responses to the same question in the same partition. Group targets retain the original eight-response group accuracy before truncation filtering. Checkpoints are selected by validation MSE against binary correctness.

A.2 Evaluation and Metric Definitions

The sampling parameters in Table 4 apply to stochastic generation. Test responses that reach the generation-length limit are retained, although they account for only a small fraction of the evaluation set. For methods that generate verbalized confidence, including Verbal, RLCR, and DCPO, some responses may fail to produce a valid confidence score. We therefore compute accuracy over all evaluated responses, while confidence metrics are computed only over responses with a valid confidence score in $[0, 1]$. For a confidence-valid set of N responses, let c_i and y_i denote predicted confidence and binary correctness. We divide $[0, 1]$ into ten equal-width bins $\mathcal{B}_b$: $[0, 0.1]$, $(0.1, 0.2]$, ..., $(0.9, 1]$. For each bin,

$$\mathrm{acc}(\mathcal{B}_b) = \frac{1}{|\mathcal{B}_b|} \sum_{i \in \mathcal{B}_b} y_i, \qquad \mathrm{conf}(\mathcal{B}_b) = \frac{1}{|\mathcal{B}_b|} \sum_{i \in \mathcal{B}_b} c_i. \tag{7}$$

The expected calibration error is:

$$\mathrm{ECE} = \sum_{b:\, |\mathcal{B}_b| > 0} \frac{|\mathcal{B}_b|}{N} \left| \mathrm{acc}(\mathcal{B}_b) - \mathrm{conf}(\mathcal{B}_b) \right|. \tag{8}$$

AUROC measures the ranking of correct versus incorrect responses. We first compute metrics separately for each dataset and then average them with equal weight. Results for CoCal average five independently trained companions with randomly sampled seeds 42, 137, 500, 700, and 1024.

B Baseline Details

B.1 Training-Free Confidence Estimation

Evaluation responses. CoCal, SelfConsis, TokenProb, and P(True) are evaluated on the same fixed responses generated by the GRPO-trained policy. Verbal confidence is evaluated on responses generated in its confidence-reporting format. RLCR and DCPO modify the policy generation process during training, so they are evaluated on responses generated by their own trained policies.

SelfConsis. For an evaluated response o and a pool of $M = 10$ responses $\{o^{(m)}\}_{m=1}^{M}$ sampled for the same question, confidence is computed as the fraction of responses consistent with the evaluated answer:

$$c_{\mathrm{SC}}(o) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}\Big[a(o^{(m)}) \equiv a(o)\Big], \tag{9}$$

where $a(\cdot)$ denotes the extracted final answer and $\equiv$ denotes answer equivalence. For mathematical tasks, equivalence is determined by deterministic answer matching; for factual QA, we use Qwen2.5-32B (Qwen et al., 2025) as a semantic-equivalence judge.

TokenProb. For a response $o = (o_1, \ldots, o_L)$, TokenProb uses the mean of its token-level generation probability as the confidence:

$$c_{\mathrm{Tok}}(o) = \exp\left(\frac{1}{L} \sum_{m=1}^{L} \log \pi(o_m \mid q, o_{<m}) \right). \tag{10}$$

When present, the generated end-of-sequence token is included. Probabilities are computed from the policy's raw logits rather than the temperature-adjusted sampling distribution.

Table 4: Hyperparameters for policy training, calibrator training, and evaluation. Response lengths are measured in tokens.

Setting	Value
Policy training	
Training steps	120
Questions per step	256
Responses per question	8
Maximum response length (GRPO / DCPO)	3,000
Maximum response length (RLCR)	4,096
Rollout temperature	1.0
Clip ratio (lower / upper)	0.20 / 0.28
Calibrator training	
Representation layer	Final transformer block
Input dimension (8B / 14B)	8,192 / 10,240
MLP hidden-layer widths	2,048 / 1,024 / 512 / 256
Train / validation split	90% / 10% by question
Loss	MSE
Optimizer	Adam
CoCal learning rate	4×10^{-4}
CoCal weight decay	0
CoCal training order	Chronological
Batch size	256
Checkpoint selection	Lowest binary validation MSE
Random seeds	42, 137, 500, 700, 1024
Evaluation	
Maximum response length (AIME / AMC)	16,384
Maximum response length (Other datasets)	4,096
Sampling temperature	0.7
Top-p	0.8
Top-k	20
Presence penalty	1.5

P(True). Given question q and its fixed response o, the model is prompted to judge whether the final answer is correct. Let p_{T} and p_{F} denote the next-token probabilities assigned to `True` and `False`, respectively. We compute

$$c_{\text{P(True)}}(o) = \frac{p_{\mathrm{T}}}{p_{\mathrm{T}} + p_{\mathrm{F}}}. \tag{11}$$

B.2 Training-Based Confidence Estimation

For RLCR and DCPO, we follow their original policy-optimization objectives and settings. For all policy-optimization methods, we set the KL coefficient to zero following prior work (Yu et al., 2026; Damani et al., 2026; Ma et al., 2026).

RLCR. RLCR (Damani et al., 2026) generates a solution and answer, followed by uncertainty reasoning and a confidence score. For mathematical tasks, our implementation uses the following output format:

```
Solution reasoning
\boxed{answer}
<reason>Uncertainty reasoning</reason>
CONFIDENCE: score
```

Given confidence $c_i \in [0, 1]$ and verifier correctness $y_i \in \{0, 1\}$, RLCR combines task correctness with a Brier-style calibration reward:

$$R_i^{\mathrm{RLCR}} = y_i - (c_i - y_i)^2. \tag{12}$$

The combined reward is normalized within each rollout group to obtain the GRPO advantage, which is applied to the entire generated sequence.

DCPO. DCPO (Ma et al., 2026) removes the additional confidence-specific reasoning and separates task and calibration optimization across token subsets. The task reward is response correctness,

$$R_i^r = y_i. \tag{13}$$

For confidence learning, DCPO constructs a target combining individual correctness with group accuracy:

$$y_i^{\mathrm{DCPO}} = (1 - \lambda)y_i + \lambda\bar{y}, \qquad \bar{y} = \frac{1}{G}\sum_{j=1}^{G} y_j, \tag{14}$$

and defines the calibration reward as

$$R_i^c = -\left|c_i - y_i^{\mathrm{DCPO}}\right|. \tag{15}$$

Here, λ controls the contribution of group accuracy and is set to 0.5 as in its original implementation. Task and calibration rewards are normalized separately to obtain their respective advantages. The task advantage is applied only to reasoning-and-answer tokens, while the calibration advantage is applied only to confidence tokens. Thus, DCPO separates the direct token-level learning signals, but both objectives are still optimized through the shared policy parameters.

PostCal. PostCal uses the same architecture, representations, targets, and checkpoint criterion as CoCal, but trains on fresh final-policy responses with shuffled batches. The main comparison uses the same 245,760-response raw budget as CoCal. After excluding truncated responses, PostCal retains 152,065 training and 16,968 validation responses. Table 4 lists its optimizer settings; Appendix E.1 details the budget comparison.

C Confidence Quality across Training Stages

C.1 Earlier Training Stages

Tables 5 and 6 report the 40- and 80-step results. Both CoCal variants outperform RLCR and DCPO in ECE and AUROC at step 40. At step 80, RLCR achieves the lowest ECE, while CoCal (binary) has the highest AUROC; however, RLCR's task accuracy remains substantially lower than that of CoCal.

Table 5: Accuracy and confidence quality on five mathematical datasets using Qwen3-8B after 40 training steps. Bold and underlined values indicate the best and second-best results, respectively.

Method	MATH500				AIME24				AIME25			
	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑
RLCR	85.8	0.963	0.098	0.747	37.0	0.762	0.392	0.860	25.3	0.784	0.530	<u>0.895</u>
DCPO	<u>86.6</u>	0.994	0.105	0.706	<u>37.7</u>	0.937	0.556	0.766	<u>28.0</u>	0.940	0.660	0.875
CoCal (group)	**87.0**	0.784	<u>0.093</u>	<u>0.909</u>	**39.3**	0.392	**0.068**	**0.890**	**30.3**	0.368	**0.124**	0.891
CoCal (binary)	**87.0**	0.831	**0.055**	**0.910**	**39.3**	0.478	<u>0.126</u>	<u>0.887</u>	**30.3**	0.446	<u>0.166</u>	**0.902**

Method	AMC23				AMC24				Macro Average			
	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑
RLCR	64.8	0.918	0.271	0.716	56.7	0.853	0.286	0.808	53.9	0.856	0.315	0.805
DCPO	<u>78.0</u>	0.980	0.198	0.745	<u>58.4</u>	0.955	0.362	0.718	<u>57.7</u>	0.961	0.376	0.762
CoCal (group)	**78.8**	0.639	<u>0.150</u>	<u>0.847</u>	**64.2**	0.564	<u>0.105</u>	<u>0.841</u>	**59.9**	0.549	<u>0.108</u>	<u>0.876</u>
CoCal (binary)	**78.8**	0.708	**0.081**	**0.855**	**64.2**	0.652	**0.056**	**0.870**	**59.9**	0.623	**0.097**	**0.885**

Table 6: Accuracy and confidence quality on five mathematical datasets using Qwen3-8B after 80 training steps. Bold and underlined values indicate the best and second-best results, respectively.

Method	MATH500				AIME24				AIME25			
	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑
RLCR	<u>88.0</u>	0.897	<u>0.058</u>	0.835	37.7	0.356	**0.092**	**0.936**	28.3	0.330	**0.049**	**0.964**
DCPO	87.0	0.977	0.069	0.684	<u>41.7</u>	0.754	0.338	0.870	<u>33.7</u>	0.741	0.405	0.930
CoCal (group)	**89.0**	0.822	0.075	<u>0.898</u>	**44.0**	0.414	<u>0.104</u>	<u>0.875</u>	**36.7**	0.389	<u>0.099</u>	0.929
CoCal (binary)	**89.0**	0.860	**0.051**	**0.905**	**44.0**	0.547	0.160	0.855	**36.7**	0.517	0.185	<u>0.943</u>

Method	AMC23				AMC24				Macro Average			
	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑
RLCR	68.3	0.760	0.187	0.751	58.4	0.639	0.099	**0.880**	56.1	0.597	**0.097**	<u>0.873</u>
DCPO	<u>83.8</u>	0.916	<u>0.096</u>	0.748	<u>64.9</u>	0.842	0.193	0.766	<u>62.2</u>	0.846	0.220	0.799
CoCal (group)	**84.5**	0.677	0.168	<u>0.807</u>	**65.6**	0.607	**0.075**	0.801	**63.9**	0.582	<u>0.104</u>	0.862
CoCal (binary)	**84.5**	0.771	**0.082**	**0.835**	**65.6**	0.710	<u>0.082</u>	<u>0.840</u>	**63.9**	0.681	0.112	**0.876**

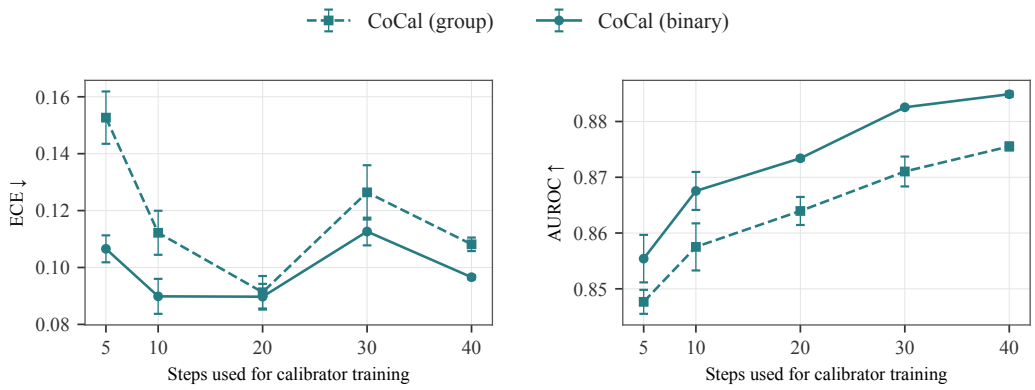

Figure 7: Confidence estimation using training data produced by early steps, evaluated on the fixed step-40 GRPO policy. Points average five seeds and five mathematical datasets; error bars show seed standard deviations.

C.2 Learning from Early Rollout Data

Figure 4 shows that RLCR and DCPO still have relatively weak confidence estimates at step 40 and improve substantially by step 80. Since policy checkpoints were saved only every 40 steps due to storage constraints, we further probe earlier learning by training CoCal on rollout prefixes from the first 5, 10, 20, 30, or 40 steps. For each prefix, both training and validation are restricted to the corresponding rollout data while preserving the question-level split, and all companions are evaluated on the same step-40 policy responses. As shown in Figure 7, CoCal (binary) already achieves an ECE of 0.090 and an AUROC of 0.868 with only the first 10 steps of rollout data, indicating that strong confidence estimation can emerge from very limited early training experience.

C.3 Full Reliability Diagrams

Figure 8 extends the main-text analysis to all five mathematical datasets and shows similar patterns: Verbal and DCPO concentrate confidence toward the high end, often exceeding empirical accuracy, while RLCR favors near-zero or near-one estimates. Both CoCal variants distribute confidence across more bins, providing finer-grained estimates, although calibration varies across datasets. CoCal uses seed 42 to display integer bin counts; empty bins are left blank.

D Cross-Iteration and Cross-Policy Transfer

In Figure 5, source X denotes the rollout steps used to train the companion, while target Y denotes the policy checkpoint on which it is evaluated. Off-diagonal cells therefore measure cross-iteration transfer without retraining, yielding ECE of 0.092–0.119 and AUROC of 0.869–0.893.

For cross-policy transfer (Figure 6), we apply GRPO-trained companions to matched-stage DCPO and RLCR policies with the same backbone. For DCPO, we retain the response endpoint when extracting features. For RLCR, we extract features only from the task response, stopping before the first `<reason>` tag and excluding uncertainty reasoning and confidence text.

E Comparison with PostCal

E.1 Matched Budgets and Training-Data Composition

We compare CoCal and PostCal under binary supervision across matched rollout-data budgets corresponding to the first 1/2/5/10/20/40/80/120 RLVR steps. For each budget, CoCal uses the rollouts collected during RLVR, while PostCal resamples responses from the final step-120 policy using the same questions and the same number of responses per question. Both methods discard truncated re-

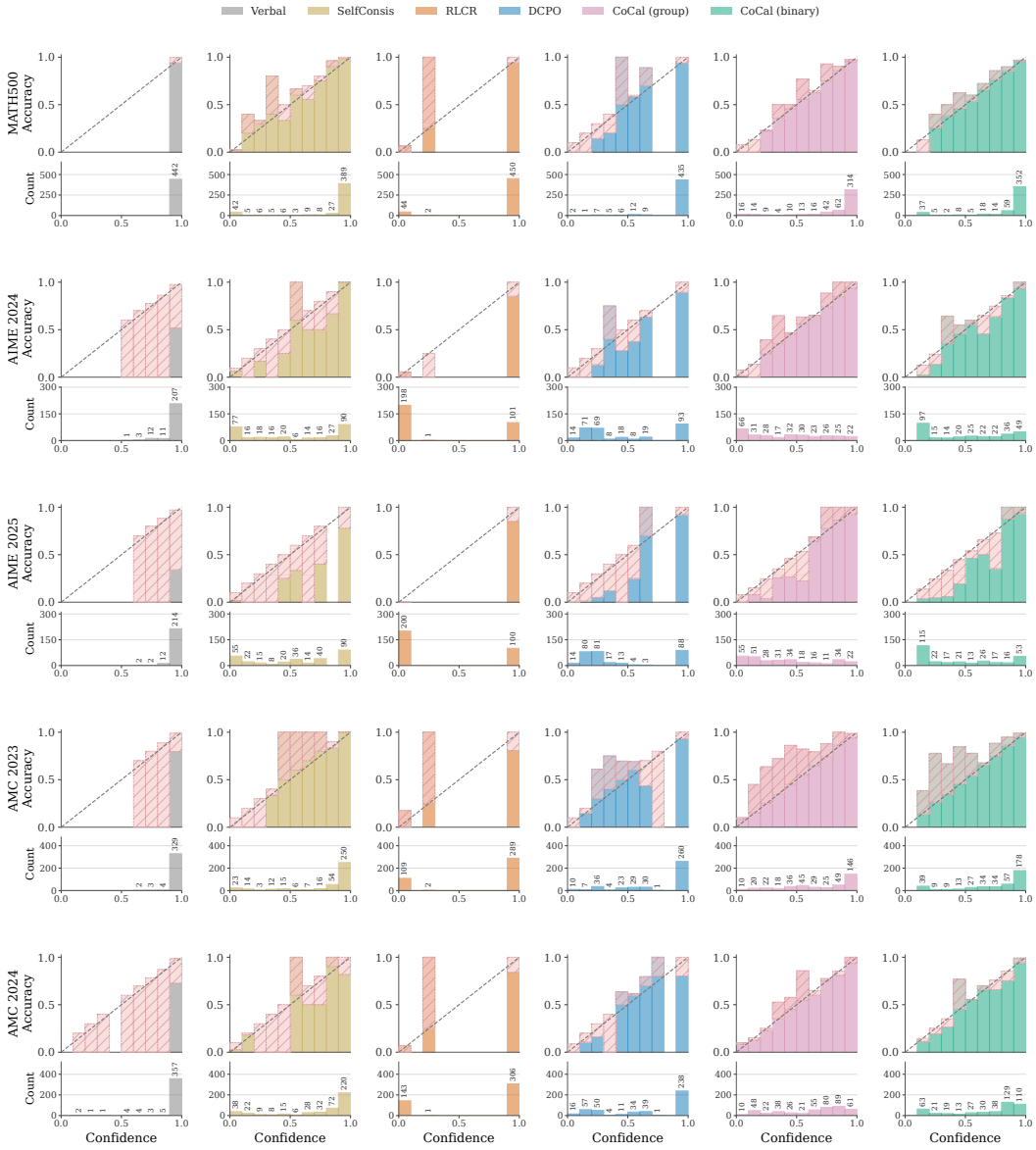

Figure 8: Reliability diagrams for Qwen3-8B after 120 policy-training steps on all five mathematical datasets. Colored bars show accuracy within nonempty confidence bins; hatched regions show the absolute gap to the actual bin-mean confidence. The dashed diagonal is an ideal-calibration reference. Lower panels report bin counts.

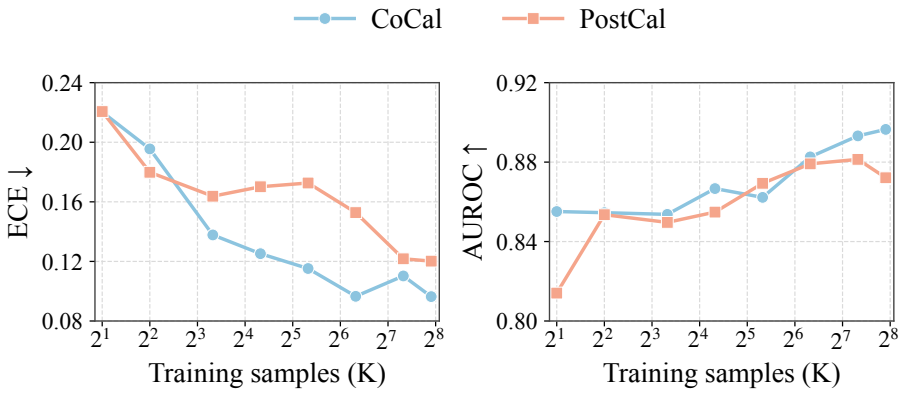

Figure 9: Confidence performance across matched rollout-data budgets. Both methods use binary supervision; results average five mathematical datasets.

sponses and are evaluated on the same step-120 policy responses. This study matches raw sampling budgets, as in the main comparison. Its full-budget binary results use the same checkpoints as the main tables. Each budget uses a fixed, question-disjoint validation pool.

As shown in Figure 9, CoCal achieves lower ECE and higher AUROC in seven of eight budgets. Its training data also contain a higher fraction of incorrect responses across budgets and generally lead to lower test confidence (Figure 10). This suggests that exposure to errors from earlier policy stages may help reduce overconfidence, although error frequency is only one of several differences between the two training-data distributions.

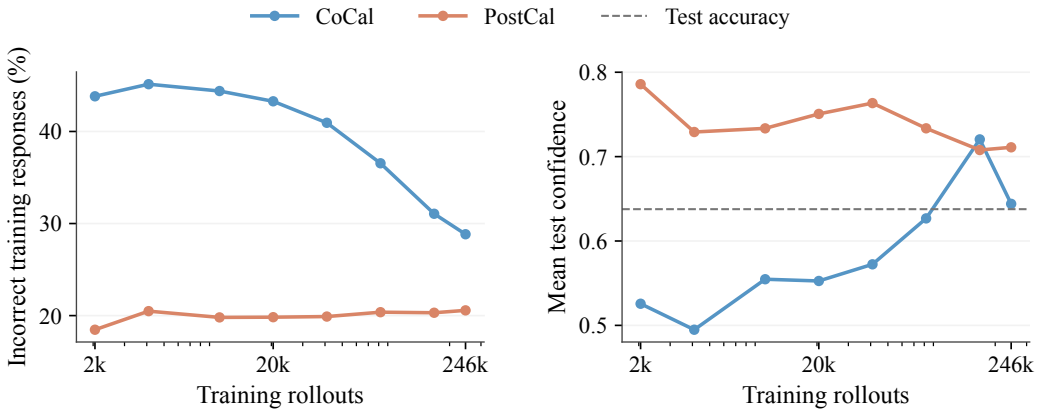

Figure 10: Training-data composition and mean test confidence under matched raw rollout budgets. Both methods use binary supervision. Left: the fraction of incorrect effective training responses, after excluding validation and truncated responses. Right: mean confidence on the fixed 120-step test responses, averaged over five companion seeds and five mathematical datasets. The dashed line marks macro task accuracy.

E.2 Computational-cost Accounting

PostCal sampling time is measured on 30,720 question occurrences from DeepScaleR using vllm on one B200 GPU, with eight responses per occurrence. Both CoCal and PostCal use the same raw rollout budget of 245,760 responses, corresponding to 30,720 question occurrences with eight responses sampled per question. We measure LLM generation time on all 500 MATH500 questions using vLLM 0.9.2 on a single NVIDIA B200 GPU with BF16 precision and tensor parallelism of 1.

All questions are submitted in one batch, with at most 64 concurrent sequences, CUDA graphs enabled, and prefix caching disabled. We report total generation wall-clock time divided by 500, using one response per question for CoCal and PostCal and ten for SelfConsis. Timing excludes model loading, compilation, warm-up, tokenization, and answer scoring, including SelfConsis answer-equivalence computation. MLP latency is averaged over 1,000 batch-size-one predictions.

F Further Analysis

F.1 Model Scale

Table 7 reports Qwen3-14B results on mathematical datasets. Both CoCal variants preserve task accuracy and outperform RLCR and DCPO in ECE and AUROC; CoCal (binary) performs best on both metrics.

Table 7: Accuracy and confidence quality on five mathematical datasets using Qwen3-14B after 120 training steps. Base + Verbal uses the pre-RL model; other training-free baselines use the GRPO-trained model. Bold and underlined values indicate the best and second-best results, respectively.

Method	MATH500				AIME24				AIME25			
	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑
Base + Verbal	84.0	0.993	0.151	0.717	30.7	0.964	0.654	0.728	23.7	0.967	0.729	0.800
Verbal	88.8	0.991	0.103	0.747	44.3	0.949	0.506	0.712	33.7	0.961	0.624	0.820
SelfConsis	**90.8**	0.925	**0.045**	0.863	**49.0**	0.565	0.096	**0.939**	**37.0**	0.548	0.178	0.868
TokenProb	**90.8**	0.958	0.050	0.813	**49.0**	0.886	0.396	0.847	**37.0**	0.867	0.497	0.897
P(True)	**90.8**	0.925	0.078	0.842	**49.0**	0.667	0.222	0.844	**37.0**	0.601	0.280	0.869
RLCR	90.2	0.938	0.060	0.754	47.0	0.507	0.094	0.921	35.0	0.407	0.068	**0.947**
DCPO	89.8	0.933	0.079	0.727	45.0	0.512	0.133	0.869	36.0	0.419	0.115	0.882
CoCal (group)	**90.8**	0.842	0.072	0.904	**49.0**	0.441	**0.071**	0.908	**37.0**	0.389	**0.063**	0.903
CoCal (binary)	**90.8**	0.866	0.050	**0.908**	**49.0**	0.550	0.074	0.907	**37.0**	0.492	0.145	0.914

Method	AMC23				AMC24				Macro Average			
	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑	Acc.↑	MeanConf	ECE↓	AUROC↑
Base + Verbal	73.8	0.986	0.247	0.704	50.9	0.978	0.464	0.712	52.6	0.978	0.449	0.732
Verbal	85.0	0.981	0.131	0.802	63.8	0.964	0.326	0.749	63.1	0.969	0.338	0.766
SelfConsis	**86.5**	0.842	**0.060**	**0.994**	**67.8**	0.796	0.152	0.778	**66.2**	0.735	0.106	0.888
TokenProb	**86.5**	0.935	0.070	0.856	**67.8**	0.924	0.246	0.748	**66.2**	0.914	0.252	0.832
P(True)	**86.5**	0.895	0.140	0.795	**67.8**	0.838	0.214	0.841	**66.2**	0.785	0.187	0.838
RLCR	86.2	0.860	0.071	0.867	63.6	0.781	0.158	0.794	64.4	0.699	0.090	0.857
DCPO	82.5	0.840	0.158	0.714	65.8	0.784	0.154	0.773	63.8	0.697	0.128	0.793
CoCal (group)	**86.5**	0.737	0.128	0.874	**67.8**	0.613	0.104	0.857	**66.2**	0.604	0.088	0.889
CoCal (binary)	**86.5**	0.794	0.071	0.882	**67.8**	0.693	**0.071**	**0.874**	**66.2**	0.679	**0.082**	**0.897**

F.2 Token Aggregation

Table 8 compares different token aggregation strategies under binary supervision. Overall, CoCal is robust to the choice of response representation, with all three variants achieving strong confidence estimation. The final-token state performs consistently well, achieving the best ECE and AUROC on Qwen3-8B and the best AUROC on Qwen3-14B. Mean pooling alone yields weaker discrimination on both backbones, while concatenating the two remains competitive and achieves the lowest ECE on Qwen3-14B. These results suggest that the final-token state already captures strong confidence signals, with response-level pooling providing complementary information in some settings.

F.3 Layer Selection

Each companion uses binary supervision and the concatenated representation in Equation 6; only the extraction layer varies. Layer indices are zero-based, ending at 35 for Qwen3-8B and 39 for Qwen3-14B. As shown in Figure 11, middle-to-late layers consistently provide strong confidence signals on both backbones, while early layers perform noticeably worse. The lowest ECE occurs

Table 8: Confidence performance with different token aggregations, averaged over five seeds and five mathematical datasets.

Feature	Qwen3-8B		Qwen3-14B	
	ECE↓	AUROC↑	ECE↓	AUROC↑
Last token	**0.093**	**0.900**	0.087	**0.898**
Response mean	0.095	0.871	0.104	0.869
Last + mean	0.096	0.896	**0.082**	0.897

at layers 30 and 38 for Qwen3-8B and Qwen3-14B, respectively, while AUROC peaks at layers 35 and 27. Performance varies only modestly across later layers, suggesting that confidence-relevant information is broadly available in deeper representations rather than concentrated at a single layer. We therefore use the final layer as a simple and robust default.

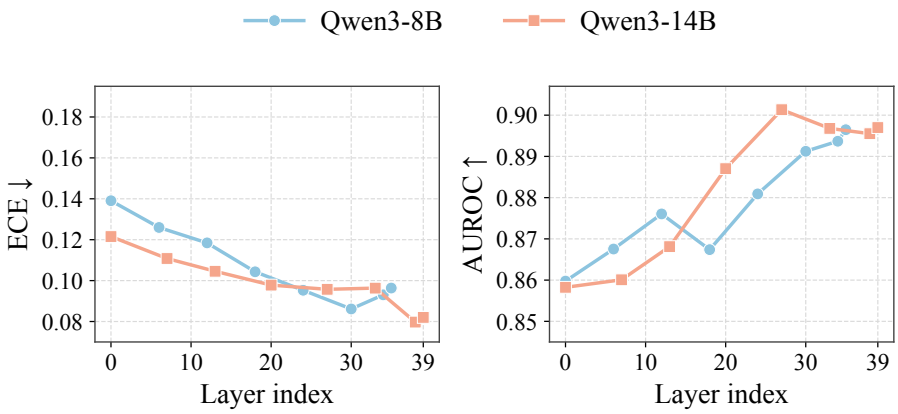

Figure 11: Average confidence estimation performance using internal states from different layers on mathematical datasets.

F.4 Pre-generation Confidence Estimation

We examine whether CoCal can estimate confidence before response generation. The predictor takes the final prompt-token state as input. Since no individual response is available at this stage, we use the mean correctness of the original $K = 8$ rollout group as the supervision target, corresponding to question-level reliability. Unlike post-generation group supervision, this target is not multiplied by individual response correctness.

To approximate historical prompt states, we use the policies trained for 40, 80, and 120 steps for rollouts from steps 1–40, 41–80, and 81–120, respectively. Training follows the same chronological protocol as CoCal, with no shuffling and question-level train/validation splits. We use the group variant's optimizer settings in Table 4, and select checkpoints by validation MSE against group accuracy.

Table 9 compares pre-generation confidence with post-generation CoCal (group) on the same evaluation responses. Prompt-only states provide informative confidence estimates, but post-generation confidence achieves substantially lower ECE and higher AUROC on both domains. This gap is expected because pre-generation confidence lacks response-specific information: it has no access to the generated reasoning or answer, and all responses to the same question receive the same confidence score, limiting discrimination between correct and incorrect generations. Post-generation CoCal, in contrast, conditions confidence on each generated response and can capture response-specific variation. Since the two settings also differ in supervision and historical-state extraction, this comparison does not isolate the effect of response information alone.

Table 9: Pre- and post-generation confidence performance of CoCal on Qwen3-8B after 120 policy-training steps. Results average five seeds and five datasets.

Setting	Math		HonestyBench	
	ECE↓	AUROC↑	ECE↓	AUROC↑
Pre-generation	0.124	0.785	0.268	0.650
Post-generation (group)	0.103	0.884	0.104	0.725

Table 10: Mean time to confidence on eight questions per dataset.

Dataset	Pre (ms)	Post (s)	Post/Pre
MATH500	38.78	58.46	1508×
AIME 2024	43.76	230.11	5258×
AIME 2025	41.78	184.56	4417×
AMC 2023	24.71	62.94	2548×
AMC 2024	25.22	79.72	3162×

Time to confidence. To fairly compare pre-generation and post-generation confidence, we use the same Transformers inference pipeline rather than using vllm for both settings, allowing the pre-generation route to stop after prompt encoding without decoding a response. We measure eight randomly sampled questions from each mathematical dataset using the same Qwen3-8B policy on a single B200 GPU with Transformers 4.53.1, BF16, SDPA, and batch size one. Pre-generation latency includes prompt encoding and MLP prediction, whereas post-generation latency includes cached autoregressive decoding, final-state processing, and MLP prediction. Model loading, tokenization, and warm-up are excluded.

Table 10 shows that pre-generation confidence is available within 25–44 ms, while post-generation confidence requires tens to hundreds of seconds depending on response length. This reduces time to confidence by roughly three to four orders of magnitude, enabling confidence-aware decisions such as abstention, retrieval, or allocating additional computation before response generation begins.