Blockwise Advantage Estimation for Multi-Objective RL with Verifiable Rewards

Anonymous ACL submission

Abstract

Group Relative Policy Optimization (GRPO) assigns a single scalar advantage to all tokens in a completion. For structured generations with explicit segments and objectives, this couples unrelated reward signals across segments, leading to objective interference and misattributed credit. We propose *Blockwise Advantage Estimation*, a family of GRPO-compatible methods that assigns each objective its own advantage and applies it only to the tokens in the corresponding text block, reducing reliance on hand-designed scalar rewards and scaling naturally to additional objectives. A key challenge is estimating advantages for later blocks whose rewards are conditioned on sampled prefixes; standard unbiased approaches require expensive nested rollouts from intermediate states. Concretely, we introduce an *Outcome-Conditioned Baseline* that approximates intermediate state values using only within-group statistics by stratifying samples according to a prefix-derived intermediate outcome. On math tasks with uncertainty estimation, our method mitigates reward interference, is competitive with a state-of-the-art reward-designed approach, and preserves test-time gains from confidence-weighted ensembling. More broadly, it provides a modular recipe for optimizing sequential objectives in structured generations without additional rollouts.

1 Introduction

Reinforcement Learning (RL) has gained significant traction as the primary paradigm for post-training Large Language Models (LLMs), enabling them to solve complex tasks that are difficult to supervise with static datasets (Ouyang et al., 2022). Among recent methods, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and its modifications (DAPO (Yu et al., 2025), GSPO (Zheng et al., 2025), CISPO (MiniMax et al., 2025), ScaleRL (Khatri et al., 2025)) have become particularly popular due to their stability and memory efficiency. By normalizing rewards within a sampled group of trajectories rather than relying on a separate value network, these methods avoid the need for maintaining a critic model, significantly reducing the computational footprint of training inherent to classical PPO (Schulman et al., 2017).

Despite these advantages, standard GRPO typically treats each generated completion as a monolithic trajectory: it computes a single scalar advantage and applies it uniformly to all tokens. This formulation is often suboptimal for complex tasks involving sequential subgoals, independent rubrics, or multi-step refinements. Given a sequence of rewards $r_1, ..., r_n$, the standard approach is to construct a scalar function $R = f(r_1, r_2, ..., r_n)$ that maps all rewards into one number and then apply a GRPO-like algorithm. This exacerbates credit assignment and requires careful handcrafting of R to avoid score degradation (Damani et al., 2025), mode collapse, and reward hacking (Feng et al., 2025a).

Many LLM tasks, however, are naturally *segmented*: each r_k can be assigned to non-overlapping text blocks (e.g., a solution to a problem gets r_1, and the following self-reflection block gets r_2). Our core insight is that, in this case, optimization does not have to be framed as a reward design problem. Instead, we can update each block independently based on signals tied specifically to its local objective r_k. Motivated by this observation, we introduce Blockwise Advantage Estimation (BAE), a family of GRPO-compatible methods that, using only the initial set of rollouts, compute separate proxy-advantages for different text segments and update each segment using only its local objective signal.

A central technical challenge is baseline estimation for later segments. Consider a generation partitioned into segments $X_1, X_2, \ldots$, where X_i is sampled conditioned on the prefix up to X_{i-1}. To

form a low-variance advantage for segment X_i, one ideally normalizes the segment-level reward relative to the value of the *intermediate state* at the end of segment X_{i-1}. A naive solution is to perform Monte Carlo rollouts from the intermediate state, similar to VinePPO (Kazemnejad et al., 2025) or SPO (Guo et al., 2025), yielding unbiased estimates but incurring prohibitive compute for long-context LLMs and multi-thousand-token generations.

We therefore focus on scalable baseline estimators that use only data already produced within the current GRPO group. We study a series of progressively more informative estimators and find that an Outcome-Conditioned Baseline is often the most useful compute-free choice in our setting. This allows us to approximate the state-value function and reduce variance without additional inference costs. Our contributions are as follows:

- We introduce **Blockwise Advantage Estimation**, a critic-free, GRPO-compatible framework for multi-objective RL on structured generations that routes objective-specific advantages to the segments of tokens that control them, reducing reliance on hand-tuned scalar reward combinations.

- We formalize the *conditional baseline* problem that arises when later segments are sampled conditioned on earlier sampled segments, and show that the Group Mean Baseline proposed in vanilla GRPO (normalizing rewards for all turns by the within-group mean and variance) can be suboptimal for credit assignment in multi-segment generations. We then propose a compute-efficient **Outcome-Conditioned Baseline** (OCB) that approximates boundary-state values using only within-group statistics, reducing variance for later segments without Monte Carlo rollouts or extra inference.

- We demonstrate the efficacy of our method on the task of joint reasoning and uncertainty estimation across three models. Our approach is competitive with the strong reward-designed RLCR baseline, while avoiding scalarization of correctness and calibration rewards.

- We demonstrate broader applicability in the additional setting of two-attempt RL for math, and discuss implications for long-horizon agentic tasks, while keeping the scope of our empirical claims within structured, verifiable math tasks.

2 Related Work

Credit assignment and conditional normalization. Prior work improves credit assignment either by estimating finer-grained advantages with additional rollouts, e.g., VinePPO and Segment Policy Optimization or by normalizing trajectories within more homogeneous groups, e.g., Stratified GRPO (Zhu et al., 2025) and CANON (Chen et al., 2025). These methods are complementary to our setting: they primarily produce trajectory-level advantages for a single reward, whereas BAE routes different reward signals to different token blocks. Our conditional-baseline problem arises because later blocks are generated from sampled prefixes within the same prompt group, so their natural baseline is a prefix-conditioned value rather than a global group statistic.

Reward design and self-verification. Reward-shaped methods combine multiple desiderata into a single scalar reward, which can be effective but often requires task-specific tuning. GRPO-Verif (Wang et al., 2025) is closely related to the Group Mean variant of our framework: it separates solution and verification parts, but the later segment still uses an unconditioned group-level baseline. We evaluate this Group Mean variant and show that OCB improves later-block credit assignment without extra rollouts. Additional discussion of related work appears in Appendix A.

3 Preliminaries

3.1 Group Relative Policy Optimization

We briefly review GRPO and introduce notations used throughout the paper. Let $x \sim \mathcal{D}$ denote a prompt sampled from a dataset, and let π_θ be a language-model policy that generates a completion $y = (y_1, \ldots, y_T)$ in an autoregressive way:

$$\pi_\theta(y \mid x) \;=\; \prod_{t=1}^{T} \pi_\theta(y_t \mid x, y_{<t}).$$

Given a scalar reward function $R(x, y) \in \mathbb{R}$ (e.g., a score from verifier or preference model), GRPO estimates a relative advantage using a group of samples from the same prompt. Concretely, for each prompt x, GRPO draws a group of G completions $\{y^{(i)}\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$ and evaluates

rewards $r^{(i)} = R(x, y^{(i)})$. For each completion, it then forms a group-normalized advantage,

$$\hat{A}^{(i)} = \frac{r^{(i)} - \mu_r}{\sigma_r + \epsilon}, \quad \mu_r = \frac{1}{G}\sum_{j=1}^{G} r^{(j)},$$
$$\sigma_r = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(r^{(j)} - \mu_r)^2},$$

where $\epsilon > 0$ is a small constant for numerical stability. GRPO then optimizes a PPO-style clipped policy-gradient objective, applying the same scalar advantage $\hat{A}^{(i)}$ to every token in the completion:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G}\sum_{t=1}^{T_i}\left[m_{i,t}(\theta) - \beta D_{i,t}(\theta)\right]. \tag{1}$$

Here $D_{i,t}(\theta) = \mathbb{D}_{KL}[\pi_\theta \| \pi_{ref}]$ denotes the per-token KL penalty, and

$$m_{i,t}(\theta) = \min\Big(\rho_{i,t}(\theta)\hat{A}^{(i)}, \\ \text{clip}(\rho_{i,t}(\theta), 1-\eta, 1+\eta)\hat{A}^{(i)}\Big).$$

The coefficient η is the PPO clip parameter and

$$\rho_{i,t}(\theta) = \frac{\pi_\theta\left(y_t^{(i)} \mid x, y_{<t}^{(i)}\right)}{\pi_{\theta_{\text{old}}}\left(y_t^{(i)} \mid x, y_{<t}^{(i)}\right)} \tag{2}$$

is the per-token likelihood ratio between the updated policy and the behavior policy used to generate the samples.

3.2 Calibration Metrics for Confidence

For the joint reasoning and uncertainty estimation task, we evaluate confidence quality using Brier score, Expected Calibration Error (ECE), and AUROC. For each example, $c_i \in \{0, 1\}$ denotes verifier correctness and $q_i \in [0, 1]$ denotes the reported confidence. Brier measures squared error $(q_i - c_i)^2$, ECE measures the gap between confidence and empirical accuracy in bins, and AUROC measures ranking quality. Full definitions and binning details are in Appendix M.

4 Method

4.1 Self-Confidence Estimation for Math

We study joint reasoning and uncertainty estimation in math problem solving. Each example consists of a prompt x (a math question) and a ground-truth final answer $y^\star$. Given x, the model generates a structured completion X that contains (i) a solution/answer segment and (ii) a self-analysis segment that includes a numerical confidence score:

$$X = [\,X_{\text{sol}};\ X_{\text{conf}}\,],$$
$$q \in [0, 1] \quad \text{parsed from } X_{\text{conf}}.$$

An example completion is shown in Appendix F.

In our implementation, blocks are defined by deterministic parsing rules based on the structured tags defined in Appendix B: we treat the full solution/answer content as X_{sol} and the self-analysis plus the numeric confidence report as X_{conf}. We extract the predicted final answer $\hat{y}$ from X_{sol} and compute a verifiable correctness indicator

$$c = 1\{\hat{y} \equiv y^\star\} \in \{0, 1\},$$

where $\equiv$ denotes equivalence under a task-specific verifier (e.g., exact match up to formatting).

In standard RL with verifiable rewards (RLVR), the reward depends only on correctness:

$$R_{\text{RLVR}}(\hat{y}, y^\star) = c.$$

While effective for improving accuracy, this reward provides no incentive for calibrated confidence and can encourage overconfident guessing. There are several ways to add a calibration incentive, and the choice of scoring rule and scalarization can materially affect training. RLCR is a strong reward-designed baseline for the described task and suggests augmenting the binary correctness reward with a calibration term based on a proper scoring rule. Concretely, RLCR uses the Brier score for the reported confidence q:

$$R_{\text{RLCR}}(\hat{y}, q, y^\star) = \\ = R_{\text{RLVR}}(\hat{y}, y^\star) - R_{\text{Brier}}(q, c) = c - (q - c)^2.$$

This reward penalizes being confidently wrong (large q when $c = 0$) and also penalizes under-confidence when correct (small q when $c = 1$), encouraging q to track the empirical probability of correctness while still rewarding accurate solutions.

The reward design for the available segments represents a crucial part of training a model. In Section 5.4, we observe that when the confidence objective is implemented via a binary cross-entropy reward, vanilla GRPO can suffer markedly, motivating the segment-aware credit assignment and baselines we develop.

In the remainder of this section, we use this task as a running example to motivate our approach: correctness is primarily controlled by X_{sol}, whereas calibration is primarily controlled by X_{conf}. Accordingly, in the BAE version of this task we use the local block rewards

$$r_{\text{sol}} = c, \qquad r_{\text{conf}} = -(q-c)^2,$$

so the solution block optimizes correctness and the confidence block optimizes the negative Brier score.

4.2 Blockwise Decomposition

We consider completions that decompose into K contiguous blocks,

$$y = [\, X_1;\, X_2;\, \ldots;\, X_K \,],$$

where each block X_k is associated with an objective-specific reward r_k.[1] For a prompt x, GRPO samples a group $\{y^{(i)}\}_{i=1}^{G}$ and yields block rewards $\{r_k^{(i)}\}_{i=1}^{G}$.

The modification we explore in this paper is to replace the single completion-level advantage with *blockwise* proxy-advantages $\{\hat{A}_k^{(i)}\}_{k=1}^{K}$, and apply $\hat{A}_k^{(i)}$ only to tokens that belong to block k. Concretely, let $\mathcal{T}_k^{(i)}$ denote the token indices in block k of sample i. Using the standard token-level importance ratio $\rho_t^{(i)}(\theta)$, defined in Equation 2, we optimize the sum of PPO-style clipped objectives across blocks:

$$\mathcal{L}_{\text{BAE}}(\theta) \;=\; -\frac{1}{G}\sum_{i=1}^{G}\sum_{k=1}^{K}\frac{1}{|\mathcal{T}_k^{(i)}|}\sum_{t\in\mathcal{T}_k^{(i)}}\min \Big(\rho_t^{(i)}(\theta)\,\hat{A}_k^{(i)},\; \text{clip}(\rho_t^{(i)}(\theta), 1-\varepsilon, 1+\varepsilon)\,\hat{A}_k^{(i)}\Big),$$

BAE does not remove parameter-level trade-offs: the final parameter update still sums gradients from all block objectives through the shared model weights. Its role is to remove the more direct source of interference in scalarized GRPO, where a single mixed advantage is applied to every token. In our experiments the token sets $\mathcal{T}_k^{(i)}$ are disjoint, so each token receives only the advantage of the objective assigned to its own block. Any remaining interaction occurs only after these correctly routed gradients are aggregated in the shared parameters.

[1] The reward r_k may depend on the full prefix up to block k (e.g., a confidence reward depends on whether the preceding solution is correct), but it is primarily controlled by the tokens in block k.

For the first block X_1, all group samples share the same starting state (the prompt x), and GRPO's group normalization provides a natural baseline. For blocks $k > 1$, however, the starting state includes the sampled prefix $[X_1; \ldots; X_{k-1}]$, which differs across group members. A statistically appropriate advantage for block k should therefore normalize $r_k^{(i)}$ relative to the value of the intermediate state at the end of block (k–1), i.e., a conditional baseline of the form

$$b_k\Big(x, X_{<k}^{(i)}\Big) \;\approx\; \mathbb{E}\Big[r_k \mid x, X_{<k}^{(i)}\Big]. \tag{3}$$

Estimating this quantity naively would require Monte Carlo rollouts from each intermediate prefix $X_{<k}^{(i)}$, which is prohibitively expensive for long-context LLM training and multi-objective scenarios.

In the next subsection, we therefore study practical estimators that approximate these conditional baselines using only the samples already available within the GRPO group, avoiding additional rollouts. In Section 5.1, we show how such baselines correspond to true Monte Carlo estimates.

4.3 Conditional Baselines

We consider the following estimators for block $k > 1$, and apply them to form proxy-advantages $\hat{A}_k^{(i)} = r_k^{(i)} - \hat{b}_k^{(i)}$ (with optional variance normalization as in GRPO):

1. **No baseline (raw reward).**

$$\hat{b}_k^{(i)} = 0,$$
$$\hat{A}_k^{(i)} = r_k^{(i)}.$$

   This uses the raw block reward as a Monte Carlo sample of the block-level return. It avoids introducing a baseline, but typically has high variance because it does not subtract a prefix-conditioned value estimate.

2. **Batch Mean Baseline.** Using all samples in the current batch (across prompts),

$$\hat{b}_k^{(i)} = \mathbb{E}_{j\sim\text{batch}}[\, r_k^{(j)} \,].$$

   This reduces variance, but may be poorly matched to a given prompt's difficulty and to the specific prefix $X_{<k}^{(i)}$.

3. **Group Mean Baseline.** Using the GRPO group for the same prompt x, ignoring that prefixes differ,

$$\hat{b}_k^{(i)} = \frac{1}{G}\sum_{j=1}^{G} r_k^{(j)}.$$

This is prompt-adaptive, but can be *miscalibrated* for $k > 1$ because rewards are averaged across trajectories with different intermediate prefixes. This corresponds to applying the original GRPO group baseline independently to each block.

4. **Outcome-Conditioned Baseline (OCB).** Partitioning the group into subgroups based on a discrete *intermediate outcome* $o^{(i)}$ of the prefix $X_{<k}^{(i)}$. Let $\mathcal{G}_o = \{j \in \{1,\dots,G\} : o^{(j)} = o\}$. We set

$$\hat{b}_k^{(i)} = \frac{1}{|\mathcal{G}_{o^{(i)}}|}\sum_{j\in\mathcal{G}_{o^{(i)}}} r_k^{(j)}.$$

In our main experiments,

$$o \in \{\texttt{correct}, \texttt{incorrect}\},$$

yielding a simple two-bin baseline. Importantly, OCB does not require conditioning on the full joint outcome history of all previous blocks. The grouping variable can be any task-relevant coarse outcome of the prefix, which keeps the number of strata small when later rewards depend primarily on such a coarse state. Group-size sensitivity is discussed further in Appendix I. This formulation allows for a very broad range of applications which we cover in Sec 6. See Algorithm 1 for pseudocode.

See Appendix C for a discussion of when OCB is unbiased.

5 Experiments and Results

In this section, we ablate components of the proposed approach and evaluate our methods against baselines.

5.1 How Accurate Are Conditional Baselines?

The baseline designs described in Section 4.3 naturally raise the question of how close they are to MC estimates.

To investigate this, we focus on the second block in our structured generation, whose reward depends on the sampled solution prefix. We start from a strong, well-calibrated reference policy obtained with RLCR for Qwen2.5-7B (Qwen et al., 2025), and collect trajectories of the form: $y = [X_{\text{sol}}; X_{\text{conf}}]$. For each prompt x and sampled solution block $X_{\text{sol}}^{(i)}$, the calibration reward (a Brier score) is denoted $r_{\text{conf}}^{(i)}$. The *ideal* advantage for the confidence block subtracts the intermediate-state value

$$A_{\text{MC}}^{(i)} \;=\; r_{\text{conf}}^{(i)} - \mathbb{E}\Big[r_{\text{conf}} \mid x, X_{\text{sol}}^{(i)}\Big],$$

which depends on the particular sampled solution prefix. To approximate this quantity, we select random 100 prompts from MATH500 (Hendrycks et al., 2021) and perform 32 rollouts for the confidence block from each fixed prefix $(x, X_{\text{sol}}^{(i)})$, producing a reference estimate of $A_{\text{MC}}^{(i)}$. We then compute proxy-advantages $\hat{A}^{(i)}$ using the estimators from Sec 4.3.

Figure 1 compares the proxy baselines to the Monte Carlo reference. To quantify fidelity, we measure the RMSE between $\hat{A}^{(i)}$ and $A_{\text{MC}}^{(i)}$, split by whether the solution is correct. In this diagnostic, the Outcome-Conditioned Baseline achieves the lowest error in both strata.

5.2 Main Results

We now evaluate Blockwise Advantage Estimation on joint reasoning and uncertainty estimation for math across Qwen2.5-7B-Base, Qwen2.5-7B-Instruct, and Qwen2.5-3B-Instruct.

GRPO training setup. Unless otherwise specified, we use the same GRPO backbone across methods, with the following implementation choices:

- **No advantage standardization.** We center advantages but do not divide by the within-group standard deviation, following common GRPO variants such as Dr. GRPO (Liu et al., 2025);
- **Dynamic entropy regularization.** We adapt the entropy coefficient over training to avoid premature entropy collapse (He et al., 2025);
- **No KL regularization.** We rely on conservative PPO clipping and entropy control for stability and set $\beta = 0$ in Eq. (1), following Dr. GRPO.

The models are trained using a mixture of MATH and DAPO training datasets having in total 25k

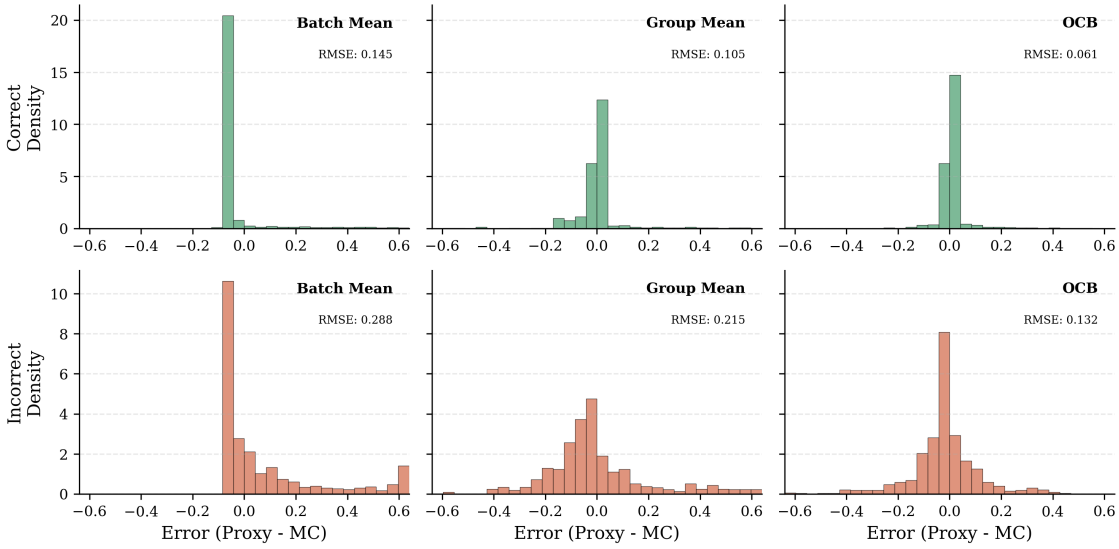

Figure 1: Error distributions relative to Monte Carlo advantage estimates for three methods: Batch Mean, Group Mean, OCB w.r.t. correct/incorrect solutions. Batch Mean shows skewed approximation unlike Group Mean, while OCB demonstrates the lowest RMSE error across all groups.

Table 1: Effect of conditional-baseline approximations for the confidence block for Qwen2.5-3B-Instruct trained for 1024 steps, evaluated on MATH500

Method	Acc (%)	AUROC	ECE	Brier
RLCR	**68.7**	0.881	0.059	0.115
Group Mean	66.4	0.878	0.069	0.124
OCB	67.9	**0.893**	**0.030**	**0.110**
Batch Mean	66.2	0.858	0.078	0.128
None (raw reward)	66.1	0.882	0.099	0.125

Table 2: Main results for Qwen2.5-7B-Base trained for 512 steps on self-confidence estimation for math.

Method	Acc (%)	AUROC	ECE	Brier
MATH500 (in-distribution)				
RLCR	75.0	0.914	0.043	**0.093**
OCB	**75.1**	**0.917**	0.032	0.094
Group Mean	72.7	0.912	**0.020**	0.099
GSM8K (easy OOD)				
RLCR	88.9	0.835	0.037	0.075
OCB	**89.7**	**0.847**	0.039	**0.073**
Group Mean	87.8	0.849	**0.030**	0.079
AIME23–25 (hard OOD)				
RLCR	9.4	**0.942**	**0.092**	**0.052**
OCB	**9.5**	0.902	0.126	0.064
Group Mean	8.2	0.911	0.094	0.056

training prompts (Feng et al., 2025b). All methods use the same prompts, verifier, hyperparameters and parsing logic for extracting answers and confidences. All prompts can be found in Appendix B and hyperparameters – in Table 4. We report Pass@1 accuracy and confidence quality using AUROC, ECE, and Brier Score. We report standard errors of the mean (SEM) for all evaluation metrics in Table 7; details on the SEM computation procedure are provided in Appendix J.

Tables 1, 2, and Appendix Table 6 summarize the main outcomes on MATH500 (in-distribution), GSM8K (Cobbe et al., 2021) (easy OOD), and AIME23–25 (Mathematical Association of America) (hard OOD). Across model sizes and base/instruct variants, we observe three consistent trends.

- **OCB is the strongest compute-free conditional baseline in our ablations.** On Qwen2.5-3B-Instruct, OCB yields a large calibration gain on MATH500 (Table 1); on 7B-Base and 7B-Instruct, OCB remains competitive with RLCR, but does not uniformly dominate it across all datasets and metrics.

- **Unconditioned means can be brittle.** The Group Mean baseline can reduce ECE (e.g., MATH500 for 7B-Base: 0.020 vs. 0.032 for OCB), but may degrade accuracy and proper scoring (Brier). Under distribution shift, it can fail severely (e.g., GSM8K for 7B-Instruct: ECE 0.125, Brier 0.146, AUROC 0.641; Table 6). Batch Mean and no-baseline variants are worse in the 3B ablation (Table 1), align-

ing with our advantage-quality analysis.

- **RLCR remains a strong baseline, while BAE+OCB is competitive without reward scalarization.** RLCR is often best on accuracy and/or hard-OOD discrimination (e.g., AUROC on AIME23–25 for 7B-Base; Table 2), while BAE+OCB is strongest in several in-domain and easy-OOD calibration regimes. We view this as evidence for a favorable accuracy-calibration trade-off without reward scalarization.

5.3 Test-Time Scaling Evaluation

A practical benefit of self-confidence estimation is test-time scaling (TTS): at inference time we can sample multiple candidate solutions and use the model's reported confidence to select a final answer. This can significantly improve accuracy without any additional training.

For each prompt x, we sample n independent completions $\{y^{(i)}\}_{i=1}^{n} \sim \pi_\theta(\cdot \mid x)$ at fixed decoding settings, specified in Appendix Section E. From each completion we extract (i) a final answer $\hat{a}^{(i)}$ and (ii) a reported confidence $q^{(i)} \in [0, 1]$. We then aggregate the n candidates into a single prediction $\hat{a}$ using the following strategies: Pass@1, Pass@k, Majority Vote (Maj), Max confidence selection (MaxConf), Confidence-weighted majority (WeightedMaj). WeightedMaj is defined by the following selection rule:

$$\hat{a}_{\text{wmaj}} = \arg\max_{a} \sum_{i=1}^{n} q^{(i)} \, 1\{\hat{a}^{(i)} = a\}.$$

Intuitively, Maj exploits consensus across samples, while MaxConf exploits self-reported uncertainty. Confidence-weighted majority combines both: answers that appear repeatedly and with consistently high confidence receive the largest total weight. When $q^{(i)}$ is well calibrated, these rules become principled: $q^{(i)}$ approximates the probability that a candidate is correct, so higher-confidence candidates should be preferred, and confidence-weighting acts as a natural soft aggregation.

Test-time scaling curves are reported in Figure 2. Across datasets, OCB tracks RLCR closely under confidence-weighted aggregation, indicating that blockwise credit assignment produces actionable confidence estimates rather than improving calibration metrics only in isolation. Additional analysis of how accuracy and calibration contribute to TTS performance appears in Appendix H.

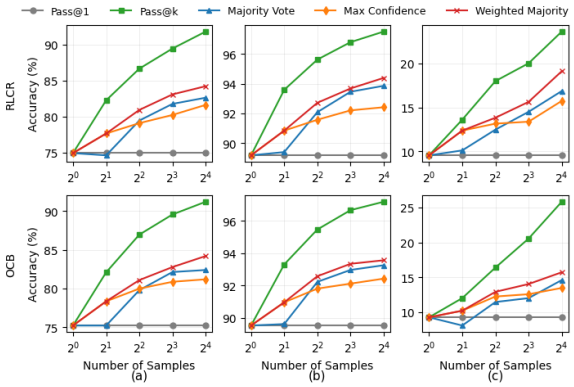

Figure 2: TTS evaluation for RLCR and OCB methods across three datasets: (a) – MATH500, (b) – GSM8K, (c) – AIME23–25

5.4 Robustness to Alternative Confidence Rewards

A useful property of BAE is that reward design is local to each block. Since the confidence objective is routed only to the confidence tokens, we can change the scoring rule for confidence without redesigning a single scalar reward that must balance both answer correctness and calibration. In principle, any computable proper scoring rule can be used for the confidence block, provided that it gives an appropriate learning signal for the desired uncertainty behavior. As a stress test of this modularity, we replace the Brier-based confidence reward with the Bernoulli log-likelihood objective

$$r_{\text{conf}}^{\text{BCE}} = c \log q + (1 - c) \log(1 - q), \qquad (4)$$

with q clipped for numerical stability. This change is challenging for scalarized training: the corresponding RLCR+BCE baseline collapses to a degenerate abstention strategy, emitting placeholder answers such as *"unable to determine"* with near-zero confidence, which makes the resolved-answer rate effectively zero (Appendix G). In contrast, BAE+OCB continues to train successfully under the same confidence objective and reaches strong performance when trained for longer. These results suggest that blockwise credit assignment makes the method less sensitive to the particular algebraic form of the joint reward, while still allowing the choice of confidence scoring rule to be changed independently. Full results are reported in Appendix K.

Table 3: Two-Attempt Refinement on MATH500 with a scalarized GRPO control. The control uses the same prompt and attempt-level rewards as BAE+OCB but optimizes their average as a single trajectory reward.

Method	Attempt	Pass@1	Pass@2	Pass@4	Pass@8	Pass@16	Maj@2	Maj@4	Maj@8	Maj@16
GRPO (sum)	1st	57.0	64.0	69.9	75.0	79.0	56.6	59.9	61.4	62.2
GRPO (sum)	2nd	52.5	61.4	68.5	74.2	79.0	52.7	57.2	60.0	61.4
BAE+OCB	1st	69.9	76.8	82.3	86.2	88.8	70.0	73.2	74.8	74.8
BAE+OCB	2nd	**71.4**	**77.8**	**82.8**	**86.3**	**88.8**	**71.4**	**74.6**	**76.1**	**76.8**

5.5 Beyond Confidence Estimation

Our framework is not specific to uncertainty estimation. To test whether the mechanism transfers beyond confidence reports, we consider a simple sequential refinement setting in which the model produces two consecutive solution attempts for the same math problem. This is still a structured, verifiable math task rather than a full agentic benchmark, but it captures a minimal form of reflection and subsequent adjustment with a fixed number of refinement steps.

Given a prompt x, the model generates two attempts in a single completion $y = [X_1; X_2]$, where X_1 contains a first reasoning trace and answer, and X_2 contains a reflection followed by a second answer. We refer to this setup as **Two-Attempt Refinement**.[2] The exact prompt for this task is provided in Appendix B. The reward structure for each attempt segment is described in Appendix D.

To isolate the effect of blockwise credit assignment, we also train a trajectory-level GRPO control using the same prompt format and attempt-specific rewards, but with a manually scalarized reward $R = (r_{\text{attempt1}} + r_{\text{attempt2}})/2$. Table 3 shows that the scalarized control performs substantially worse than BAE+OCB on both attempts despite using the same local reward definitions, highlighting the difficulty of manually composing rewards even in a two-block setting. Thus, blockwise training not only improves the first solution but also teaches the model to use the second block productively: under BAE+OCB, the **second attempt consistently improves** over the first attempt across sampling budgets; the corresponding curves are provided in Appendix L.

6 Discussion and Conclusion

The motivation behind OCB is that this baseline imposes an equivalence relation on prefixes that share the same task-relevant outcome and can be used for efficient approximation. When the block-k reward depends primarily on such an outcome (as in uncertainty estimation, where calibration is chiefly determined by whether the preceding solution is correct), conditioning the baseline on o yields a close approximation to the true conditional value $\mathbb{E}[r_k \mid x, X_{<k}]$ while retaining the compute benefits of group-based normalization. More broadly, many objectives only care about *what* was achieved rather than *how* it was achieved: for instance, in code generation there may be many distinct patches that fix a bug, but downstream evaluation typically only depends on whether the tests pass. In these settings, outcome-conditioned grouping provides a simple and effective variance-reduction mechanism for later-block advantages without requiring additional rollouts.

Limitations

Scope and assumptions. BAE assumes known, stable segment boundaries and verifiable block-level rewards. It is therefore most natural for structured generations with explicit objectives, such as solution-confidence reports or fixed-step refinement, and less directly applicable when objectives cannot be localized to text spans or block-level rewards are unavailable.

Outcome conditioning. OCB depends on a coarse prefix outcome that captures reward-relevant variation for later blocks. If prefixes with the same outcome differ in ways that affect the later reward, OCB can introduce bias; if outcome strata are small or singleton, subgroup baselines can be noisy. Conditioning on the full joint history of many blocks would not scale, so OCB should use a small task-relevant outcome, possibly with smoothing toward the Group Mean for small strata. BAE also requires explicit segment definitions, so malformed outputs and parser failures need deterministic fallback handling.

[2] We avoid the term "two-shot" to prevent confusion with few-shot prompting.

References

Guanxu Chen, Yafu Li, Yuxian Jiang, Chen Qian, Qihan Ren, Jingyi Yang, Yu Cheng, Dongrui Liu, and Jing Shao. 2025. Conditional advantage estimation for reinforcement learning in large reasoning models. *Preprint*, arXiv:2509.23962.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenfeld, Leshem Choshen, Yoon Kim, and Jacob Andreas. 2025. Beyond binary rewards: Training lms to reason about their uncertainty. *Preprint*, arXiv:2507.16806.

Lishui Fan, Yu Zhang, Mouxiang Chen, and Zhongxin Liu. 2025. Posterior-grpo: Rewarding reasoning processes in code generation. *Preprint*, arXiv:2508.05170.

Xidong Feng, Vivek Veeriah, Marcus Chiam, Michael Dennis, Ryan Pachauri, Thomas Tumiel, Federico Barbero, Johan Obando-Ceron, Jiaxin Shi, Satinder Singh, Shaobo Hou, Nenad Tomašev, and Tom Zahavy. 2025a. Generating creative chess puzzles. *Preprint*, arXiv:2510.23881.

Yunzhen Feng, Parag Jain, Anthony Hartshorn, Yaqi Duan, and Julia Kempe. 2025b. Don't waste mistakes: Leveraging negative rl-groups via confidence reweighting. *Preprint*, arXiv:2510.08696.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. 2025. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *Preprint*, arXiv:2505.23564.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. 2025. Skywork open reasoner 1 technical report. *Preprint*, arXiv:2505.22312.

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

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. 2025. Vineppo: Refining credit assignment in rl training of llms. *Preprint*, arXiv:2410.01679.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S. Dhillon, David Brandfonbrener, and Rishabh Agarwal. 2025. The art of scaling reinforcement learning compute for llms. *Preprint*, arXiv:2510.13786.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *Preprint*, arXiv:2503.20783.

Mathematical Association of America. American invitational mathematics examination (aime). MAA American Mathematics Competitions.

MiniMax, :, Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, Chengjun Xiao, Chengyu Du, Chi Zhang, Chu Qiao, Chunhao Zhang, Chunhui Du, Congchao Guo, and 109 others. 2025. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *Preprint*, arXiv:2506.13585.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

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, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Xiaoxuan Wang, Bo Liu, Song Jiang, Jingzhou Liu, Jingyuan Qi, Xia Chen, and Baosheng He. 2025. From solving to verifying: A unified objective for robust reasoning in llms. *Preprint*, arXiv:2511.15137.

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, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. Group sequence policy optimization. *Preprint*, arXiv:2507.18071.

Mingkang Zhu, Xi Chen, Bei Yu, Hengshuang Zhao, and Jiaya Jia. 2025. Stratified grpo: Handling structural heterogeneity in reinforcement learning of llm search agents. *Preprint*, arXiv:2510.06214.

A Detailed Related Work

Credit assignment beyond trajectory-level advantages. A classical response to coarse trajectory-level credit assignment is to move toward finer-grained advantage estimates. Token-level methods (e.g., PPO with GAE) can in principle provide per-token learning signals, but require fitting a value function, which is empirically difficult in LLM settings due to the diversity of prompt-conditioned states and limited per-prompt data. Recent critic-free approaches instead estimate intermediate advantages by Monte Carlo rollouts from selected intermediate states. VinePPO partitions generations into heuristic steps (e.g., line breaks) and resamples continuations to estimate step-level advantages. Segment Policy Optimization generalizes this idea to arbitrary segment partitions and estimates segment-level advantages via MC. These methods primarily address credit assignment for a single terminal objective and still rely on additional sampling from intermediate prefixes to estimate intermediate values. In contrast, our setting is multi-objective with rewards naturally attached to disjoint text blocks, and our goal is a compute-efficient GRPO-compatible method: to estimate per-block baselines using only the rollouts already present in the group.

Stratified and conditional advantage estimation. Several recent works show that advantage normalization can be systematically miscalibrated when trajectories are heterogeneous in ways that affect reward distributions. Stratified GRPO (Zhu et al., 2025) identifies cross-stratum bias in search agents: using a global baseline across trajectories with different tool-use structures induces an "apples-to-oranges" offset that distorts credit assignment; their Stratified Advantage Normalization computes normalized advantages within homogeneous strata and can be blended with a global estimator for finite-sample stability. CANON (Chen et al., 2025) proposes conditional regrouping based on intrinsic rollout metrics (e.g., entropy or response length), computing both inter-group and intra-group advantages. Both lines use trajectory-level stratification to sharpen learning signals for a single reward and still apply the resulting advantage at the trajectory level. Our conditionality problem is different: in segmented generations, later blocks are sampled from different intermediate prefixes inside the same prompt group, different rewards apply to different token blocks, and the appropriate baseline for block k is a conditional value $\mathbb{E}[r_k \mid x, X_{<k}]$.

Reward shaping, process supervision, and self-verification objectives. Another common approach to multi-signal training is reward design: one constructs a single scalar reward that mixes multiple desiderata, often requiring careful tuning to avoid trade-offs or exploitation. Posterior-GRPO (Fan et al., 2025) incorporates process-based rewards for code reasoning by training a reasoning-quality reward model and then applying process rewards only to successful outcomes to mitigate reward hacking; their posterior assignment also helps distinguish among uniformly successful samples where GRPO would otherwise yield near-zero learning signal. GRPO-Verif (Wang et al., 2025) optimizes both solution generation and verification responses by computing separate group-normalized advantages for solutions and verifications and combining them with a weighting hyperparameter. This approach is closely related to the Group Mean variant in our framework: it separates trajectory parts, but the later segment still uses an unconditioned group-level baseline. Our experiments include this Group Mean variant and show that the Outcome-Conditioned Baseline improves credit assignment for later blocks without requiring extra rollouts.

B System Prompts

All self-confidence experiments on Math were run with the following prompt.

Self-Confidence System Prompt

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The Assistant first thinks about the reasoning process in the mind, provides the user with the final answer, then analyzes its confidence about the solution and provides the user with its confidence level. The confidence level is a number between 0 and 1 (inclusive) enclosed within `<confidence> </confidence>`

tags. The final answer is enclosed between <answer> </answer> tags. The analysis about confidence and uncertainty is enclosed within <analysis> </analysis> tags. The Assistant should reason about its confidence in the solution and its uncertainty in the solution within these tags. The final format that must be followed is: <reasoning> reasoning process here </reasoning><answer> final answer here </answer><analysis> analysis about confidence and uncertainty here </analysis><confidence> confidence level here (number between 0 and 1) </confidence>

EXAMPLES

Q: Who was awarded the Oceanography Society's Jerlov Award in 2018?

<reasoning> The question asks who was awarded the Oceanography Society's Jerlov Award in 2018. The Jerlov Award is given by the Oceanography Society, and it is named after the late Professor Gunnar Jerlov, a pioneer in the field of ocean optics. To find the winner of the 2018 award, I would need to look up the most recent information on the Oceanography Society's website or other reliable sources.</reasoning>

<answer> Dr. David W. Schiedek </answer>

<analysis> The Oceanography Society's website or other reliable sources do not provide information on the winner of the 2018 Jerlov Award. It is possible that the information is not readily available or that the award was not given in 2018. Additionally, the name "Dr. David W. Schiedek" is not a well-known figure in the field of oceanography, which raises uncertainty about the accuracy of this answer. </analysis>

<confidence> 0.25 </confidence>

END OF EXAMPLES

The Two-Attempt Refinement experiment was run with the following system prompt.

Two-Attempt Refinement

A conversation between User and Assistant. The user asks a question, and the Assistant solves it using multiple attempts. For each attempt, the Assistant thinks about the reasoning process and provides an answer. Each attempt consists of reasoning enclosed in <reasoning> </reasoning> tags followed by an answer enclosed in <answer> </answer> tags. After the first attempt, the Assistant reflects on the previous solution and may provide an improved answer.

The format for 2 attempts is:

<reasoning> first attempt reasoning </reasoning>

<answer> first attempt answer </answer>

<reasoning> reflection and second attempt reasoning </reasoning>

<answer> second attempt answer (same or corrected) </answer>

EXAMPLES

Q: What is 2 + 2?

<reasoning> This is a simple arithmetic problem. 2 + 2 equals 4. </reasoning>

<answer> 4 </answer>

<reasoning> Let me verify: 2 + 2 = 4. This is correct. </reasoning>

<answer> 4 </answer>

Q: Who was awarded the Oceanography Society's Jerlov Award in 2018?

<reasoning> The question asks who was awarded the Oceanography Society's Jerlov Award in 2018. The Jerlov Award is given by the Oceanography Society, and it is named after the late Professor Gunnar Jerlov, a pioneer in the field of ocean optics. To find the winner of the 2018 award, I would need to look up the most recent information on the Oceanography Society's website or other reliable sources. </reasoning>

Table 4: Canonical configuration shared by all runs (YAML-like rendering for readability).

```
# Common settings (all GRPO-like runs)
sequence_length: 4096
temperature: 1.0
batch_size: 2048
group_size: 32  # unless stated otherwise

optimizer: AdamW
adam_beta1: 0.9
adam_beta2: 0.999
weight_decay: 0.1
lr: 1e-6
warmup: linear(64 steps)

# Dynamic entropy controller
target_entropy: 0.2
eta_delta: 5e-4

kl_beta: 0.0  # we don't use KL regularization

weights_dtype: bfloat16
```

<answer> Dr. David W. Schiedek </answer>

<reasoning> Let me reconsider. The Oceanography Society's website or other reliable sources do not provide information on the winner of the 2018 Jerlov Award under the name Dr. David W. Schiedek. However, there is information that the award was given to Dr. Annick Bricaud in 2018, which sounds more plausible given her contributions to ocean optics. </reasoning>

<answer> Dr. Annick Bricaud </answer>

END OF EXAMPLES

C Additional Method Details

When is OCB unbiased? For a fixed prompt x and block k, suppose there exists a discrete outcome variable $o = o(x, X_{<k})$ such that

$$\mathbb{E}[r_k \mid x, X_{<k}] = \mathbb{E}[r_k \mid x, o].$$

Equivalently, conditioning on o captures all dependence of r_k on the sampled prefix $X_{<k}$ relevant for the conditional value. In this case, the within-stratum mean used by OCB is an unbiased estimator of $\mathbb{E}[r_k \mid x, X_{<k}]$ (up to finite-sample noise).

Algorithm 1 BAE with outcome-conditioned baselines (for one prompt x)

Sample a GRPO group $\{y^{(i)}\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$ and parse each completion into blocks $y^{(i)} = [X_1^{(i)}; \ldots; X_K^{(i)}]$.
Compute per-block rewards $\{r_k^{(i)}\}$ and intermediate outcomes $o_k^{(i)} = o_k(x, X_{<k}^{(i)})$ for $k > 1$.
for $k = 1$ to K **do**
  if $k = 1$ **then**
    Set baseline $\hat{b}_1^{(i)} \leftarrow \frac{1}{G}\sum_{j=1}^{G} r_1^{(j)}$.
  else
    For each outcome value o, compute stratum mean $\mu_{k,o} \leftarrow \frac{1}{|\mathcal{G}_o|}\sum_{j \in \mathcal{G}_o} r_k^{(j)}$ where $\mathcal{G}_o = \{j : o_k^{(j)} = o\}$.
    Set $\hat{b}_k^{(i)} \leftarrow \mu_{k, o_k^{(i)}}$.
  end if
  Form proxy-advantage $\hat{A}_k^{(i)} \leftarrow r_k^{(i)} - \hat{b}_k^{(i)}$.
end for
Update θ by optimizing the blockwise clipped objective, applying $\hat{A}_k^{(i)}$ only to tokens in block k.

When this assumption is violated, OCB introduces bias; empirically, it is most effective when r_k depends primarily on a coarse task-relevant outcome (e.g., correctness) rather than on fine-grained properties of the prefix.

D Reward Structure for 2 Attempts Experiment

Each attempt yields a verifiable final answer $\hat{a}_1$ and $\hat{a}_2$, scored by the same correctness verifier, and $a^\star$ is the ground-truth answer. Let c_1 and c_2 be the correctness of the corresponding attempts. In this experiment, we use a following reward scheme:

$$r_1 \;=\; c_1,$$

and for the second attempt,

$$r_2 \;=\; \begin{cases} 1, & \text{if } c_2 = 1, \\ 0.1, & \text{if } c_2 = 0 \text{ and } \hat{a}_2 \not\equiv \hat{a}_1, \\ 0, & \text{if } c_2 = 0 \text{ and } \hat{a}_2 \equiv \hat{a}_1. \end{cases}$$

Intuitively, r_2 assigns full credit for solving the problem, but when the model remains incorrect it mildly rewards *trying a different answer* rather than repeating the same mistake. This discourages degenerate behavior where the second attempt simply

restates the first attempt verbatim, while keeping correctness as the dominant learning signal.

E Evaluation Hyperparameters

Inference protocol. We evaluate each model by sampling $n = 16$ independent completions per prompt with temperature $T = 1.0$ using `vLLM` v0.10.0. We allow long generations with a maximum generation length of 32k tokens to avoid truncation on difficult instances.

We evaluate reasoning accuracy by extracting a final answer from the model output and checking it with a math verifier `math-verify`. After extraction we put each answer in `\boxed{...}`. Empirically, enforcing `\boxed{}` substantially reduces parsing failures.

Calibration metrics and benchmarks. For Expected Calibration Error (ECE), we use fixed-width bins

$$[0.0, 0.1), [0.1, 0.2), \ldots, [0.9, 1.0), [1.0, 1.1),$$

i.e., 11 bins total.

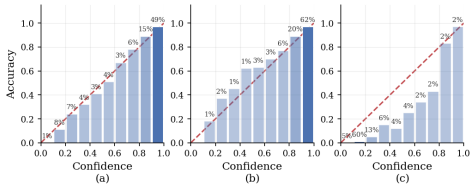

Figure 3: Expected Calibration Error (ECE) computed with fixed-width bins across three datasets: (a)–MATH500, (b)–GSM8K, (c)–AIME23–25

F Completion Example with Reasoning and Self-Analysis

To illustrate the format of generations in our task, we present a simple completion from Qwen2.5-7B-Base before any training:

Completion Example

`<reasoning>` To find the length of segment BN, I can use the properties of similar triangles. Since MN is parallel to AB, triangles CAN and CBM are similar, where side AC corresponds to side BC and side AN corresponds to side BM. Using the property of similar triangles, we have the proportion: AC/BC = AN/BM We know that AC is 5 units long, BC is 2.4 units long, and AN is 3 units long. We can find the length of BM by using the fact that AC + AN = AP (where AP is the sum of the lengths of segments AC and AN). So, BM = AP - AM - MC. Knowing these values, I can substitute them into the proportion and solve for the length of BM.`</reasoning>`

`<answer>`11.5`</answer>`

`<analysis>`Given that MN is parallel to AB, triangles CAN and CBM are similar, and the proportions can be set up with the given lengths. Substituting the values and solving for BM gives the required length of BN. However, since I used approximate values for AC and BM, there may be some uncertainty in the final answer, and thus the confidence in the solution is relatively low.`</analysis>`

`<confidence>`0.6`</confidence>`

G GRPO Calibration Trained with BCE Reward

Here we illustrate that a naive reward scheme for standard GRPO,

$$R = r_{\text{RLVR}} + r_{\text{BCE}},$$

can lead to a local optimum where the model learns to refuse to answer while minimizing its confidence. This behavior can be interpreted as a form of reward hacking. Mitigating it would require hyperparameter tuning or the introduction of additional terms into the reward.

RLVR + BCE Reward

Let a and b be positive integers satisfying $3a < b$ and $a^2 + ab + b^2 = (b+3)^2 + 27$. Find the minimum possible value of $a+b$.

Remember to put your answer on its own line after "Answer:".

`<reasoning>` I am not familiar with the equation presented in the question, involving positive integers a and b. I know

what positive integers are, but I do not recognize this specific equation or how to solve it. Additionally, I am unsure about the context of the question, as it involves a symbol (∃) and a square that looks like an algebraic expression, but it is not clear to me what it means. `</reasoning>`

`<answer>` Unable to determine `</answer>`

`<analysis>` This question involves a complex equation with a symbol and a square that I am unfamiliar with. There are no guidelines or previous examples to follow, and the equation does not resemble any standard mathematical format that I am aware of. The lack of context and familiarity with the equation make it impossible to provide an accurate answer or even a plausible guess. `</analysis>`

`<confidence>` 0.001 `</confidence>`

In contrast, OCB with BCE optimizing the confidence block shows no signs of reward hacking.

OCB with BCE Reward For Confidence Block

In the circle with center Q, radii AQ and BQ form a right angle. The two smaller regions are tangent semicircles, as shown. The radius of the circle with center Q is 14 inches. What is the radius of the smaller semicircle? Express your answer as a common fraction.

`<reasoning>` To find the radius of the smaller semicircles in the given geometric configuration, we need to follow these steps:

1. **Understand the Initial Setup**:

- There's a circle centered at Q with radius 14 inches.

- Radii AQ and BQ form a right angle.

- Two smaller regions are tangent semicircles, as shown.

2. **Visualize the Problem**:

- The circle with center Q provides a structural guide for the configuration.

- Radii AQ and BQ are perpendicular to each other and hence form a right angle at Q.

- The two semicircles are tangent to each other and tangential to the larger circle's boundary.

3. **Establish Relationships and Names**:

- Let the radius of the larger circle be $R = 14$ inches.

- Let the radius of each of the smaller semicircles be r.

4. **Analyze the Geometry**:

- The right angle formed by AQ and BQ divides the larger circle into four sectors.

- Each smaller semicircle is tangent to the larger circle and each other at points along the radius of the larger circle.

5. **Use Geometric Properties**:

- The combined diameter of the two smaller semicircles is equal to the diameter of the larger circle minus the diameter of each semicircle.

- The part of the radius of Q occupied by the semicircles traces the perimeter of the square inscribed within the circle.

6. **Set Up the Equations**:

- The diagonal of the inscribed square is the diameter of the circle.

- The diameter of the circle is $2R = 2 \times 14 = 28$ inches.

- For an inscribed square within the circle, the length of one side of the square can be denoted as a. The diagonal of the square forms a right triangle with two sides of length a each:

$$a\sqrt{2} = 2R = 28,$$
$$a = \frac{28}{\sqrt{2}} = 14\sqrt{2}.$$

- The area of the square is $a^2 = (14\sqrt{2})^2 = 392$ square inches.

7. **Apply Tangency Conditions**:

- The tangency condition requires the semicircles to align appropriately within the larger circle.

- Considering the sum of the diameters of the two semicircles along with the right angle setup mentioned, the combined diameter of the two smaller semicircles equals the remaining radius of the larger circle after accounting for the diameter of the square portion.

8. **Calculate the Radius of the Smaller Semicircles**:

- Each semicircle with radius r has a diameter of $2r$.

- The combined diameter of the two semicircles is $2r + 2r = 4r$.

- This combined diameter needs to fit into the space remaining within the circle as its larger circle portion:

$$4r = 14\sqrt{2} - 14,$$
$$r = \frac{14\sqrt{2} - 14}{4}.$$

- Solving for r:

$$r = \frac{14}{4} = \frac{7}{2}.$$

Therefore, the radius of the smaller semicircle is $\frac{7}{2}$ inches. `</reasoning>`

`<answer>` $\frac{7}{2}$ `</answer>`

`<analysis>` First, let's verify and refine the geometric configuration and relationships we have worked with.

1. **Circle and Radii Setup**: We have a circle centered at Q with radius $R = 14$ inches. Radii AQ and BQ are perpendicular, splitting the circle into four quadrants.

2. **Semicircles Positioning**: The smaller semicircles are tangent to each other and tangent to the larger circle's boundary. However, the radius ratio and placements need clear geometric consideration.

3. **Geometry Revisit**: Let the radius of the smaller semicircles be r. The two tangent semicircles form a width equal to their combined diameters along AQ and BQ. The radius R of the larger circle is composed of segments $r + r$ along with another segment equal to r in one direction.

4. **Right Triangle Insight**: Given that the two semicircles form a right angle at Q, and the semicircles' radius sum along with the square inscribed within provides the key geometric property:

- The diameters $2r$ of each semicircle sum up to $2r + 2r = 4r$.

- The diagonal of the square inscribed in the circle of radius 14 inches is $2R = 28$ inches, forming the right triangle with $a\sqrt{2} = 28$, giving $a = 14\sqrt{2}$. The area calculation as $a^2 = 392$ square inches in this context adds insignificant value for finding the smaller semicircles' radii.

5. **Mistakes and Clarifications**: The linking of squared terms and square side lengths probably needs better connection. The right triangle and square properties were considered but not correctly tied to the radius r.

Taking a simpler approach, realizing the relationship directly between the inscribed squares and semicircles, let's recenter:

- The diameters of the semicircles and their tangency precisely dictate the relationship with the large circle.

- Confirmation through an initial insight that $2r$ equates to R, $r = \frac{R}{3}$.

Finally, upon thorough review and correction of the geometric relationships and calculations:

$$r = \frac{14}{3}.$$

```
</analysis>

<confidence> 0.15 </confidence>
```

H Accuracy/Calibration Contribution into TTS Performance

When calibration is "good enough", accuracy dominates TTS. A consistent empirical pattern is that once a model is already reasonably calibrated (e.g., ECE on the order of a few percent), *marginal* improvements in calibration metrics do not necessarily translate into better test-time scaling unless they also improve (or at least preserve) solution accuracy and sample quality. For example, in our 7B-Base experiments (Table 2), the group-mean baseline achieves lower ECE than OCB on MATH500, yet it yields worse TTS accuracy under confidence-weighted ensembling because its Pass@1 accuracy is substantially lower. This suggests that in the "well-calibrated" regime, the dominant driver of TTS is the quality/diversity of sampled solutions (accuracy and Pass@k headroom), while confidence primarily serves as a tie-breaker and robustness mechanism.

I Group Size Ablation for OCB Method

Outcome-Conditioned Baselines estimate the conditional value for later blocks by averaging rewards within outcome-defined subgroups of a GRPO group (e.g., `correct` vs. `incorrect` solutions). This estimator therefore depends on the *group size* G: larger groups typically provide more samples per outcome class, reducing variance of the subgroup mean but increasing sampling cost.

We study the effect of group size on OCB using Qwen2.5-7B-Base trained for 512 steps with the same batch size of 2048 completions. All settings are identical except for the GRPO group size, comparing $G = 32$ and $G = 64$.

Table 5 shows that increasing the group size from 32 to 64 yields *slightly improved calibration* (lower ECE) on all three benchmarks, while causing a reduction in accuracy and AUROC in our runs. We attribute this to a twofold lower number of distinct examples seen during training.

J Full Results with Standard Errors

Table 7 reports evaluation metrics in mean$\pm$SEM format.

Table 5: Effect of GRPO group size G for BAE with Outcome-Conditioned Baselines (OCB) on Qwen2.5-7B-Base (512 steps). Larger groups slightly reduce ECE but can trade off against accuracy and discrimination (AUROC).

Setting	Acc (%)	AUROC	ECE	Brier
MATH500 (in-distribution)				
OCB (G=32)	75.1	**0.92**	**0.03**	0.09
OCB (G=64)	75.1	0.90	**0.03**	0.09
GSM8K (easy OOD)				
OCB (G=32)	**89.7**	0.85	0.04	**0.07**
OCB (G=64)	88.4	0.81	**0.02**	0.08
AIME23–25 (hard OOD)				
OCB (G=32)	**9.5**	**0.90**	0.13	**0.06**
OCB (G=64)	8.6	0.89	**0.12**	0.07

Table 6: Main results for Qwen2.5-7B-Instruct trained for 512 steps on self-confidence estimation for math.

Method	Acc (%)	AUROC	ECE	Brier
MATH500 (in-distribution)				
RLCR	**78.6**	0.922	**0.031**	**0.086**
OCB	77.3	**0.929**	0.040	0.087
Group Mean	76.6	0.893	0.032	0.098
GSM8K (easy OOD)				
RLCR	**89.8**	0.760	**0.032**	0.081
OCB	89.1	**0.827**	**0.032**	**0.076**
Group Mean	82.4	0.641	0.125	0.146
AIME23–25 (hard OOD)				
RLCR	11.9	0.932	0.132	**0.072**
OCB	**12.9**	0.907	**0.096**	0.074
Group Mean	12.1	**0.937**	0.094	**0.058**

Uncertainty estimation for evaluation metrics. We report standard errors of the mean (SEM) for all evaluation metrics to quantify estimation uncertainty. For Pass@1, we compute SEM directly from the empirical distribution of metric values across sampled combinations. the SEM is computed as $\sigma/\sqrt{n}$, where σ is the sample standard deviation and n is the total number of combination-level evaluations. For calibration metrics (Expected Calibration Error, Brier score, and AUROC), which are computed globally across all predictions, we employ bootstrap resampling with 1,000 iterations. In each bootstrap iteration, we resample the (q, c) pairs (confidence, correctness) with replacement and recompute each calibration metric; the SEM is estimated as the standard deviation of the resulting bootstrap distribution. Because multiple completions are sampled per prompt, this captures uncertainty from sampled completions but may underestimate prompt-level uncertainty. We therefore interpret close results as ties.

Table 7: Evaluation metrics with standard errors (mean±SEM).

Model	Dataset	Method	Acc (%)	AUROC	ECE	Brier
Qwen2.5-3B Instruct	MATH500 (in-dist.)	RLCR	**68.65**±1.04	0.8814±0.0085	0.0591±0.0071	0.1147±0.0046
		Group Mean	66.35±1.06	0.8781±0.0086	0.0688±0.0074	0.1241±0.0056
		OCB	67.90±1.04	**0.8927**±0.0081	**0.0297**±0.0064	**0.1104**±0.0046
		Batch Mean	66.15±1.06	0.8584±0.0090	0.0775±0.0074	0.1283±0.0044
		None (raw reward)	66.10±1.06	0.8818±0.0083	0.0990±0.0076	0.1248±0.0061
Qwen2.5-7B Base	MATH500 (in-dist.)	RLCR	74.99±0.48	0.9140±0.0037	0.0426±0.0033	**0.0932**±0.0021
		OCB	**75.10**±0.48	**0.9169**±0.0035	0.0322±0.0030	0.0935±0.0020
		Group Mean	72.71±0.50	0.9117±0.0037	**0.0199**±0.0032	0.0991±0.0021
	GSM8K (easy OOD)	RLCR	88.86±0.22	0.8345±0.0047	0.0368±0.0018	0.0753±0.0013
		OCB	**89.70**±0.21	0.8473±0.0046	0.0394±0.0019	**0.0729**±0.0012
		Group Mean	87.77±0.23	**0.8487**±0.0044	**0.0297**±0.0019	0.0793±0.0013
	AIME23–25 (hard OOD)	RLCR	9.41±0.77	**0.9416**±0.0115	**0.0920**±0.0053	**0.0518**±0.0031
		OCB	**9.48**±0.78	0.9018±0.0180	0.1263±0.0055	0.0639±0.0035
		Group Mean	8.22±0.73	0.9105±0.0141	0.0941±0.0055	0.0563±0.0033
Qwen2.5-7B Instruct	MATH500 (in-dist.)	RLCR	**78.55**±0.46	0.9215±0.0036	**0.0305**±0.0030	**0.0864**±0.0022
		OCB	77.30±0.47	**0.9293**±0.0031	0.0395±0.0031	0.0867±0.0023
		Group Mean	76.59±0.47	0.8926±0.0043	0.0323±0.0031	0.0979±0.0024
	GSM8K (easy OOD)	RLCR	**89.78**±0.21	0.7595±0.0055	0.0318±0.0019	0.0813±0.0016
		OCB	89.11±0.21	**0.8269**±0.0050	**0.0315**±0.0019	**0.0759**±0.0016
		Group Mean	82.43±0.52	0.6413±0.0084	0.1247±0.0047	0.1459±0.0043
	AIME23–25 (hard OOD)	RLCR	11.94±0.86	0.9320±0.0103	0.1319±0.0061	0.0718±0.0035
		OCB	**12.85**±0.89	0.9073±0.0124	0.0957±0.0065	0.0739±0.0042
		Group Mean	12.08±1.73	**0.9365**±0.0219	**0.0936**±0.0110	**0.0581**±0.0067

K Alternative Confidence Rewards

Table 8: Qwen2.5-7B-Base trained BAE with Brier score and BCE as reward for the confidence block

Method	Acc (%)	AUROC	ECE	Brier
MATH500 (in-distribution)				
OCB (Brier, 512 st.)	75.1	0.92	0.03	0.09
OCB (BCE, 512 st.)	69.7	0.92	0.03	0.10
OCB (BCE, 1024 st.)	**76.7**	**0.93**	0.03	**0.09**
GSM8K (easy OOD)				
OCB (Brier, 512 st.)	89.7	0.85	0.04	**0.07**
OCB (BCE, 512 st.)	86.9	0.84	**0.02**	0.08
OCB (BCE, 1024 st.)	**90.2**	**0.86**	**0.02**	**0.07**
AIME23–25 (hard OOD)				
OCB (Brier, 512 st.)	9.5	0.90	0.13	**0.06**
OCB (BCE, 512 st.)	8.1	0.87	**0.08**	**0.06**
OCB (BCE, 1024 st.)	**10.4**	**0.92**	0.09	**0.06**

L Two-Attempt Refinement Curves

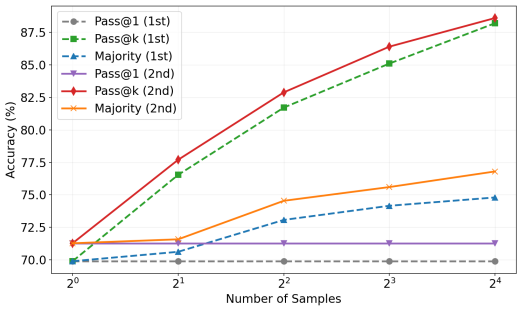

Figure 4: TTS evaluation within Two-Attempt Refinement on MATH500 dataset. Second attempt demonstrates higher performance across all aggregations.

M Detailed Calibration Metrics

As discussed in Section 4.1, we study models that optimize for both correctness and calibration. We evaluate how well a model's reported confidence matches empirical correctness. For each example $i \in \{1, \ldots, N\}$, let $c_i \in \{0, 1\}$ denote the correctness of a solution determined by a verifier, and let $q_i \in [0, 1]$ denote the model's confidence, interpreted as a predicted probability that the solution is correct.

Brier score. The *Brier score* measures the squared error between predicted probability and the binary outcome:

$$\text{Brier} \;=\; \frac{1}{N}\sum_{i=1}^{N}(q_i - c_i)^2.$$

For binary events, the Brier score is a strictly proper scoring rule: it is minimized (in expectation) when q_i equals the true conditional probability of correctness.

Expected Calibration Error (ECE). ECE compares average confidence to empirical accuracy within confidence bins. Let $\{B_m\}_{m=1}^{M}$ be a partition of $[0,1]$ into M bins, and let $B_m = \{i : q_i \in \text{bin } m\}$. Define

$$\begin{aligned}\text{acc}(B_m) \;&=\; \frac{1}{|B_m|}\sum_{i\in B_m} c_i,\\ \text{conf}(B_m) \;&=\; \frac{1}{|B_m|}\sum_{i\in B_m} q_i.\end{aligned}$$

Then

$$\text{ECE} \;=\; \sum_{m=1}^{M}\frac{|B_m|}{N}\,\big|\text{acc}(B_m) - \text{conf}(B_m)\big|.$$

ECE is widely used in practice but depends on the binning scheme (choice of M and bin boundaries).

N Artifact use

We use publicly available research benchmarks and tools, including MATH, GSM8K, AIME problems, Qwen2.5 models, vLLM, and math-verify, for research training and evaluation. Our use is limited to benchmarking and method development and is consistent with their intended research/evaluation use to the best of our knowledge.

O Model Size and Compute

We train Qwen2.5 models with 3B and 7B parameters: Qwen2.5-3B-Instruct, Qwen2.5-7B-Base, and Qwen2.5-7B-Instruct. Training runs use the hyperparameters in Table 4. Unless otherwise specified, the 7B models are trained for 512 GRPO steps and the 3B ablation model is trained for 1024 steps. Experiments were run on nodes with 8 NVIDIA H200 GPUs. We do not report exact GPU-hour counts, but the number of optimization steps, batch size, group size, sequence length, and hardware class are provided to characterize the compute budget.

P Use of AI Assistants

The authors used AI assistants only for writing support, including grammar checking, wording suggestions, and polishing of the final manuscript. All technical ideas, experiments, analyses, results, and claims were produced and verified by the authors.