How Far Can Unsupervised RLVR Scale LLM Training?

Anonymous authors
Paper under double-blind review

Abstract

Unsupervised Reinforcement Learning with Verifiable Rewards (URLVR) offers a pathway for Large Language Models (LLMs) to improve without human supervision. Particularly, many works use model intrinsic information as rewards for URLVR, showing promising improvements, yet their potential and limitations remain unclear. In this work, we revisit URLVR through the lens of intrinsic rewards. We present a unified theoretical framework showing that intrinsic reward methods share a core mechanism: they trade uncertainty for performance by leveraging the model's prior knowledge to sharpen output distributions. Empirical analysis confirms this tradeoff, revealing distinct failure modes and showing that collapse is not inevitable in small, domain-specific regimes such as test-time training. Beyond these findings, early intrinsic reward dynamics also provide a lightweight indicator of model-task priors, complementing $pass@k$ in assessing RL trainability. These insights highlight both the promise and pitfalls of URLVR, motivating future directions such as external rewards and hybrid supervision strategies.

1 Introduction

Reinforcement Learning with Verifiable Rewards (RLVR) has been central to recent breakthroughs in enhancing reasoning capability in large language models (LLMs). In RLVR, models learn from rewards that can be verified against ground truth, such as correctness in mathematics or successful code execution. Recent leading models including OpenAI's o1 and o3 (Jaech et al., 2024; OpenAI, 2025), DeepSeek-R1 (Guo et al., 2025), Gemini 2.5 (Comanici et al., 2025), and the Qwen3 series (Yang et al., 2025; Team, 2025) have achieved remarkable performance on mathematics, coding, and science benchmarks by scaling supervised RLVR. However, on the path toward superintelligence, this approach faces a crucial limitation: scaling supervision requires prohibitively high human costs, and as models reach or surpass human expertise in specialized domains, obtaining reliable ground truth supervision becomes increasingly infeasible (Burns et al., 2023; Silver & Sutton, 2025).

This supervision bottleneck has spurred growing interest in Unsupervised RLVR (URLVR) (Zuo et al., 2025), which derives rewards without ground truth labels for LLM training. This transition from supervised to unsupervised training parallels the success of pretraining scaling laws (Brown et al., 2020; Raffel et al., 2020), which effectively transform large-scale computation into intelligence on vast amounts of unlabeled data. From this perspective, URLVR represents a critical step toward scaling AI systems beyond reliance on human-provided labels.

It is worth noting that recent URLVR methods have primarily relied on leveraging a model's internal signals as training rewards. Common approaches include majority voting across multiple rollouts (Zuo et al., 2025) or the adoption of entropy-based metrics (Agarwal et al., 2025). These forms of intrinsic reward have shown notable performance gains. Yet, such seemingly unequivocal successes come with concerns, as several works highlight critical failure modes such as reward hacking and model collapse (Shafayat et al., 2025; Agarwal et al., 2025; Zhang et al., 2025c). Moreover, diverse methodologies have been applied across different model families, tasks, and evaluation settings, yet there remains neither a systematic comparison nor a consensus regarding what constitutes reliable unsupervised rewards. ***So behind the flourishing progress of such methods, might there lie certain hidden risks and uncertainties?***

To this end, it is timely to revisit the development of this area. We mainly focus on methods that derive rewards from the model's intrinsic information, in contrast to other URLVR approaches such

as RPT (Dong et al., 2025), which rely on external data. To gain a deeper understanding of the current state and future potential of RL with intrinsic rewards, we conduct a comprehensive study. We begin by reviewing existing work and classify intrinsic reward methods into two categories based on the source of rewards: *ensemble-based* and *certainty-based*. Then we establish a unified theoretical perspective of these methods, and subsequently validate and deepen it through empirical analysis.

Theoretical Perspective. We introduce a unified framework that formalizes diverse intrinsic reward mechanisms and analyzes their induced optimal policies. Despite design differences, these rewards share a common objective: sharpening output distributions by reinforcing the model's initially confident solutions. This geometric convergence enables prior amplification and efficient adaptation in low-data or test-time settings, but also risks bias lock-in, reduced exploration, and reward hacking when confidence misaligns with correctness. Intrinsic rewards thus offer a context-dependent tool that trades uncertainty for decisiveness, providing shortcuts for local adaptation while underscoring the need for external signals to ensure scalable reliability.

Empirical Analysis. To validate our theoretical findings, we implement several widely used intrinsic reward methods and design experiments around three progressively layered research questions. **First**, we ask *why these methods work*, showing that by enforcing self-consistency they trade uncertainty for performance and amplify prior knowledge. However, this same process also risks overfitting biases, potentially accelerating model collapse. **Second**, we ask *how different methods fail*, revealing that each induces distinct pathology, some collapse to brevity, others to verbosity, clarifying the structured limits to scaling. **Finally**, we ask *whether collapse is inevitable*, and find that in small, domain-specific regimes such as test-time training, intrinsic rewards drive stable adaptation without collapse. Together, these results show that intrinsic rewards set clear limits on scaling, yet within those limits they offer a principled path to self-improvement without supervision.

Our findings reveal that intrinsic rewards operate within well-defined boundaries determined by confidence-correctness correlation, enabling efficient gains in test-time and low-data regimes while risking reward hacking when confidence misaligns with correctness. These limitations motivate exploration of extrinsic approaches that leverage external verification mechanisms, from generation-verification asymmetries in structured domains to self-supervised signals from vast unlabeled corpora, which offer pathways toward more robust and scalable improvement. Beyond training itself, we also uncover a practical diagnostic: early intrinsic reward dynamics serve as a fast indicator of model-task priors, offering a lightweight alternative to $pass@k$ for assessing RL trainability.

2 Related Work

Reinforcement Learning with Verifiable Rewards. Recent advances in language model reasoning leverage Reinforcement Learning with Verifiable Rewards (RLVR) (Lambert et al., 2024), where models receive binary rewards based on answer correctness verified against ground truth. Leading systems including OpenAI's o1 and o3 (Jaech et al., 2024; OpenAI, 2025), DeepSeek-R1 (Guo et al., 2025), Gemini 2.0 (Comanici et al., 2025), and Qwen3 (Yang et al., 2025; Team, 2025) have achieved remarkable performance through scaling supervised RLVR. However, this approach faces a fundamental bottleneck: as models approach human expertise in specialized domains, obtaining reliable ground-truth supervision becomes prohibitively expensive (Burns et al., 2023).

URLVR with Intrinsic Rewards. To address this supervision bottleneck, an emerging line of research investigates Unsupervised RLVR (URLVR), which aims to extend the scalability of RL beyond labeled data. One promising direction within URLVR is the use of self-generated proxy intrinsic rewards, thereby eliminating the reliance on ground-truth labels for RL. We distinguish two intrinsic paradigms by *how* rewards are constructed from the model:

Certainty-based methods derive rewards from a single policy's confidence (e.g., logits) along a trajectory, encouraging low-entropy, high-confidence predictions. Approaches include Self-Certainty in RLIF (Zhao et al., 2025b) via KL divergence from uniform distributions, negative token-level entropy in EM-RL (Agarwal et al., 2025) and RENT (Prabhudesai et al., 2025), trajectory-level entropy in EM-RL (Agarwal et al., 2025), raw probability in RLSC (Li et al., 2025), probability disparity between top tokens (van Niekerk et al., 2025), and cross-attention patterns (Kiruluta et al., 2025b;a). These methods essentially "sharpen" the model's existing preferences by reinforcing high-confidence outputs.

Ensemble-based methods derive a reward from agreement across multiple rollouts (e.g., majority voting), assuming that cross-sample consistency correlates with correctness. TTRL (Zuo et al., 2025) pioneered majority voting across rollouts to create pseudo-labels. Building on this foundation, SRT (Shafayat et al., 2025) analyzes limitations including reward hacking, ETTRL (Liu et al., 2025a) improves efficiency through entropy-based tree search, Co-Reward (Zhang et al., 2025d) enhances robustness via question paraphrasing, and RLCCF (Yuan et al., 2025) incorporates multi-model collectives. More nuanced approaches include EMPO (Zhang et al., 2025b) using semantic clustering for soft majority voting and CoVo (Zhang et al., 2025a) deriving rewards from intermediate reasoning consistency. These methods assume that agreement among diverse outputs indicates correctness.

3 Theoretical Perspective: The Trade-off of Sharpening

While several studies have empirically explored intrinsic rewards (Zhang et al., 2025b; Zuo et al., 2025; Agarwal et al., 2025; Shafayat et al., 2025), their underlying theoretical mechanisms remain poorly understood. Despite diverse design choices, we show that these methods share a fundamental commonality: they systematically encourage sharper, more decisive probability distributions through unified mathematical operations.

3.1 Unified Reward Framework

Despite varied implementations, intrinsic rewards can be understood through a single lens: manipulating cross-entropy between carefully chosen distributions:

Unified Reward Framework

Most intrinsic rewards can be expressed as:

$$r_{\text{uni}}(x, y) = \psi\left(\frac{\sigma}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\mathbb{H}(q^i, \pi_\theta^i)\right), \quad \sigma \in \{+1, -1\}, \tag{1}$$

where rewards derive from cross-entropy $\mathbb{H}$ between anchor distributions q^i and model distributions π_θ^i, aggregated over granularity $\mathcal{I}$, with sign σ and monotonic transformation ψ.

Key Components:

- Given a question x and generated response y (a sequence of tokens $y_1, \ldots, y_{|y|}$), we can derive rewards from the model's internal distributions at different levels of granularity.
- **Aggregation granularity** $\mathcal{I}$: Determines the level to compute distributions. For token-level methods, $\mathcal{I} = \{1, \ldots, |y|\}$ where each element corresponds to a position in the sequence. For answer-level methods, $\mathcal{I} = \{\mathcal{A}\}$ represents a single distribution over complete semantic answers.
- **Model distribution** π_θ^i **at granularity** i: For token-level granularity at position t, this is $\pi_\theta^t(\cdot) = \pi_\theta(\cdot \mid x, y_{<t})$, the distribution over the next token given the context. For answer-level granularity, this is $\pi_\theta^{\mathcal{A}} = \pi_\theta(\cdot \mid x)$, the distribution over complete answers.
- **Anchor distribution** q^i **at granularity** i: Serves as a reference point. Different reward estimators use different anchors: uniform distribution U_V for Self-Certainty or one-hot distribution δ^t centered on the generated token for Trajectory-Level Entropy.
- **Cross-entropy** $\mathbb{H}(q^i, \pi_\theta^i)$**:** Cross-entropy between anchor distribution q^i and model distribution π_θ^i at granularity i, defined as $\mathbb{H}(q^i, \pi_\theta^i) = -\sum_{v\in\mathcal{V}^i} q^i(v) \log \pi_\theta^i(v)$. For token-level granularity ($i = t$), $\mathcal{V}^i$ is the token vocabulary, and the cross-entropy measures divergence between distributions over next tokens. For answer-level granularity ($i = \mathcal{A}$), $\mathcal{V}^i$ is the set of distinct semantic answers, and the cross-entropy measures divergence between distributions over complete answers.
- **Sign factor** $\sigma \in \{+1, -1\}$: Determines the optimization direction. When the anchor q is uniform, we set $\sigma = +1$ to reward divergence from uniformity (encouraging peaked distributions). When the anchor q is sharp (e.g., one-hot or the model's own distribution), we set $\sigma = -1$ to reward alignment (reinforcing confident predictions).
- **Monotonic transformation** ψ: Reshapes the reward signal while preserving ordering. Common choices are identity ($\psi(z) = z$) or exponential ($\psi(z) = \exp(z)$), with exponential transformations amplifying the sharpening effect.

Table 1: Instantiations for the unified reward framework of representative intrinsic rewards. Each method is specified by its estimator, anchor and model distributions, and a monotonic transformation of the cross-entropy between them. Token-level (t) and answer-level (A) variants capture different granularities of aggregation.

Method	Estimator	Formula	Monotonic transformation ψ	Anchor distribution q	Model distribution π_θ
RLIF (Zhao et al., 2025b)	**Self-Certainty**	$r_{\text{SC}} = \frac{1}{\lvert y\rvert}\sum_{t=1}^{\lvert y\rvert}\mathbb{H}(U_V, \pi_\theta^t) + \log\lvert V\rvert$	$z + \log\lvert V\rvert$	$\{U_V\}_{t=1}^{\lvert y\rvert}$	$\{\pi_\theta^t\}_{t=1}^{\lvert y\rvert}$
EM-RL, RENT (Agarwal et al., 2025) (Prabhudesai et al., 2025)	**Token-Level Entropy**	$r_{\text{H}} = -\frac{1}{\lvert y\rvert}\sum_{t=1}^{\lvert y\rvert}\mathbb{H}(\pi_\theta^t)$	z	$\{\pi_\theta^t\}_{t=1}^{\lvert y\rvert}$	$\{\pi_\theta^t\}_{t=1}^{\lvert y\rvert}$
EM-RL (Agarwal et al., 2025)	**Trajectory-Level Entropy**	$r_{\text{Traj}} = -\frac{1}{\lvert y\rvert}\sum_{t=1}^{\lvert y\rvert}\mathbb{H}(\delta^t, \pi_\theta^t)$	z	$\{\delta^t\}_{t=1}^{\lvert y\rvert}$	$\{\pi_\theta^t\}_{t=1}^{\lvert y\rvert}$
RLSC (Li et al., 2025)	**Probability**	$r_{\text{Prob}} = \exp\Big(-\sum_{t=1}^{\lvert y\rvert}\mathbb{H}(\delta^t, \pi_\theta^t)\Big)$	$\exp(\lvert y\rvert \cdot z)$	$\{\delta^t\}_{t=1}^{\lvert y\rvert}$	$\{\pi_\theta^t\}_{t=1}^{\lvert y\rvert}$
EMPO (Zhang et al., 2025b)	**Semantic Entropy**	$r_{\text{SE}} = \exp\big(-\mathbb{H}(\delta^A, \pi_\theta^A)\big)$	$\exp(z)$	δ^A	π_θ^A
TTRL, SRT, ETTRL (Zuo et al., 2025) (Shafayat et al., 2025) (Liu et al., 2025a)	**Majority Voting**	$r_{\text{MV}} = \lim_{\tau\to 0^+}\exp\big(-\mathbb{H}(\delta^A, \tilde{\pi}_\theta^A)\big)$	$\exp(z)$	δ^A	$\tilde{\pi}_\theta^A$

Instantiations of the Framework. We next demonstrate how most intrinsic rewards instantiate this framework. Each method's distinctive characteristics emerge from specific choices of $\mathcal{I}$, q, σ, and ψ, as shown in Table 1. Token-level methods (Self-Certainty, Token/Trajectory-Level Entropy, Probability) set $\mathcal{I} = \{1, \ldots, |y|\}$ to aggregate across sequence positions, while answer-level methods (Semantic Entropy, Majority Voting) use $\mathcal{I} = \{A\}$ for global consistency. The sign follows the anchor type: $\sigma = +1$ with uniform q (rewarding departure from randomness) and $\sigma = -1$ with sharp q (rewarding peaked predictions). For Majority Voting, $\tilde{\pi}_\theta^A$ denotes the tempered distribution $\tilde{\pi}_\theta^A(a) = \exp(\pi_\theta^A(a)/\tau)/\sum_b \exp(\pi_\theta^A(b)/\tau)$ that converges to the majority answer as $\tau \to 0^+$. Despite surface-level differences, all methods manipulate cross-entropy to achieve distribution sharpening. Further analysis of the framework refer to Appendix A.1.

3.2 Optimal Policy Analysis

After establishing the unified reward framework, we now examine the optimal policies induced by these rewards. We focus on Majority Voting as a representative example, then generalize to other methods using the unified framework (Appendix A.4).

Consider the standard KL-regularized RL objective:

$$\max_{\pi_\theta} \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[r(x,y)\right] - \beta D_{\text{KL}}\left[\pi_\theta(\cdot|x)\|\pi_{\text{ref}}(\cdot|x)\right], \tag{2}$$

where π_{ref} is the reference policy (typically the SFT model) and β controls the strength of regularization. The optimal policy for this objective has the well-known closed form $\pi_\theta^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)$, where $Z(x) = \sum_y \pi_{\text{ref}}(y|x)\exp(\frac{1}{\beta}r(x,y))$ is the partition function.

Majority Voting Reward. We define the majority voting reward at iteration k as $r_k(x,y) = \mathbf{1}[\text{ans}(y) = \text{maj}_k(Y_k)]$, where $Y_k = \{y^{(1)}, \ldots, y^{(N)}\}$ denotes N rollouts sampled from $\pi_\theta^{(k)}$, and $\text{maj}_k(Y_k) = \arg\max_a |\{i \in [N] : \text{ans}(y^{(i)}) = a\}|$ is the most frequent answer among the rollouts.

Optimal Policy for Fixed Reward. For a fixed reward r_k at iteration k, if we held r_k constant and performed infinite updates starting from reference policy $\pi_\theta^{(k)}$, we would converge to:

$$\pi_\theta^{*,k}(y|x) = \begin{cases} \frac{\pi_\theta^{(k)}(y|x)\cdot e^{1/\beta}}{Z_k(x)}, & \text{if ans}(y) = \text{maj}_k(Y_k), \\ \frac{\pi_\theta^{(k)}(y|x)}{Z_k(x)}, & \text{otherwise}, \end{cases} \tag{3}$$

where $Z_k(x) = p_{\text{maj}}^{(k)} \cdot e^{1/\beta} + (1 - p_{\text{maj}}^{(k)})$ and $p_{\text{maj}}^{(k)} = \sum_{y:\text{ans}(y)=\text{maj}_k(Y_k)} \pi_\theta^{(k)}(y|x)$ denotes the probability mass on majority trajectories at iteration k. At this optimum, the probability mass would be $p_{\text{maj}}^{*,(k+1)} = \frac{p_{\text{maj}}^{(k)} \cdot e^{1/\beta}}{p_{\text{maj}}^{(k)} \cdot e^{1/\beta} + (1 - p_{\text{maj}}^{(k)})}$.

Actual Training Dynamics. In practice, our training performs only one gradient update per iteration, not reaching the optimum $\pi_\theta^{*,k}$ but moving partway toward it. The actual probability mass after one update satisfies: $p_{\text{maj}}^{*,(k+1)} \geq p_{\text{maj}}^{(k+1)} \geq p_{\text{maj}}^{(k)}$. This ordering holds because: (1) policy gradient methods with positive rewards on majority trajectories tend to increase their probability mass (lower bound), and (2) the optimal policy achieves maximum expected reward, so single-step updates cannot exceed it (upper bound). See Appendix A.3 for detailed justification.

This creates a "rich-get-richer" dynamic: the probability of trajectories leading to the majority answer consistently increased, while others are proportionally diminished. Iterating this process, the policy converges geometrically toward a deterministic policy concentrated on the initial majority answer:

Theorem 1. ***Geometric Convergence to Deterministic Policy.*** *Consider the training procedure where at each iteration k, we: (1) sample N rollouts Y_k from $\pi_\theta^{(k)}$, (2) compute majority $maj_k(Y_k)$, (3) perform one gradient update with reward $r_k(x, y) = \mathbf{1}[ans(y) = maj_k(Y_k)]$ to obtain $\pi_\theta^{(k+1)}$. Let $p_{maj}^{(k)} = \sum_{y:ans(y)=maj_k(Y_k)} \pi_\theta^{(k)}(y|x)$ denote the probability mass on majority trajectories.*

Assumptions: (A1) Majority remains stable: $\text{maj}_k(Y_k) = \text{maj}_0(Y_0)$ for all k (holds with high probability for sufficiently large N); (A2) Gradient updates achieve non-trivial progress with $\eta_k \geq \eta_{\min} > 0$ (standard assumption). We validate these empirically in Appendix A.5.

Conclusion: Under (A1)-(A2), $p_{\text{maj}}^{(k)}$ converges geometrically to 1. As $k \to \infty$:

$$\lim_{k\to\infty} \pi_\theta^{(k)}(y|x) = \begin{cases} \frac{\pi_{\text{ref}}(y|x)}{\sum_{y':\text{ans}(y')=\text{maj}_0(Y_0)} \pi_{\text{ref}}(y'|x)}, & \text{if ans}(y) = \text{maj}_0(Y_0), \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Complete proof is in Appendix A.3. Generalization to other methods using the unified reward framework is in Appendix A.4. This convergence behavior has profound implications depending on the model's initial knowledge. When the model's confidence (reflected in maj_0) aligns with correctness, convergence reinforces good solutions. Conversely, if confidence is poorly aligned, the same mechanism amplifies errors, leading to model collapse.

4 Empirical Analysis: The Promise and Pitfalls

The prospect of LLM scaling through unsupervised RL hinges on whether models can reliably improve themselves without ground truth labels. Our theoretical analysis suggests that intrinsic rewards enable such self-improvement by exploiting existing knowledge, yet their convergence to deterministic policies raises concerns about when this process succeeds versus fails. To clarify these boundaries, we empirically examine the practical limits and opportunities of intrinsic reward-driven scaling through three key research questions:

Research Questions and Takeaways

- **Why do these methods work?** *They trade uncertainty for performance, leveraging the model's prior knowledge to improve sample efficiency.*
- **How do different methods fail?** *Each exhibits distinct pathology: Self-Certainty and Majority Voting are most stable, Probability collapses to brevity, while entropy-based methods promote repetitive verbosity.*
- **Do these methods always cause collapse in prolonged RL?** *No. With small, domain-specific data, they avoid collapse, making test-time training an ideal application.*

Experimental Setup. We train Qwen3-1.7B-Base on DAPO-17k using GRPO and evaluate on AIME24/25 and AMC23 benchmarks. We track specialized metrics to detect reward hacking:

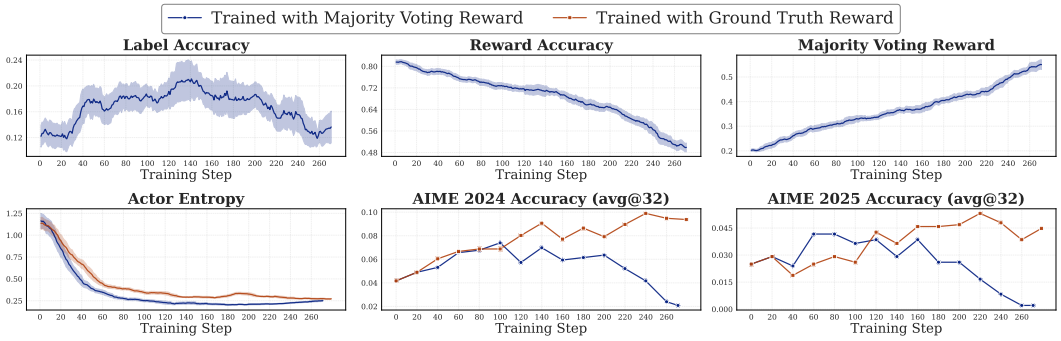

Figure 1: Comparison between training with Majority Voting reward and ground truth reward.

Majority Voting Reward (intrinsic reward), *Ground Truth Reward* (true performance), *Label Accuracy* (correctness of pseudo-label), and validation accuracy. Complete setup details refer to Appendix B.1.

4.1 Why Do These Methods Work? Trading Uncertainty for Performance

This section shows that intrinsic-reward methods boost performance by *reducing uncertainty*. Early in training, they match or even surpass supervised RLVR with ground-truth rewards while consuming entropy faster. A fine-grained per-sample analysis links uncertainty reduction to performance gains, revealing a sampling-efficiency *shortcut* when confidence aligns with correctness. We also observe late-stage reward hacking, highlighting the need to balance the uncertainty-performance trade-off.

4.1.1 Comparison with Trained with Ground Truth Reward

Setup. We trained Qwen3-1.7B-Base on DAPO-17k with the Majority Voting reward as a representative example, using the default hyperparameters from Table 5. For comparison, we also train with ground truth labels.

Results. As shown in Figure 1, trained with Majority Voting reward attains comparable or even superior performance to ground-truth training on two validation benchmarks at early stage, while *reducing* model uncertainty faster (rapid *Actor Entropy* decay) and increasing *Majority Voting Reward*. This establishes a negative correlation between uncertainty and performance in the early phase.

4.1.2 Fine-Grained Training Dynamic Analysis

Setup. To probe the mechanism at the instance level, we train Qwen3-1.7B-Base **per problem individually** from MATH500 using REINFORCE with a baseline and a Trajectory-Level Entropy reward estimator (global batch size $= 1$, 100 epochs). For each problem, we track greedy-decoding validation accuracy and intrinsic reward over time, drawing a heatmap indicating greedy correctness (blue = correct, red = wrong) and a binary square-wave indicating whether the highest-reward sample is correct. Due to the large amount of experiments, we only randomly tested 25 data points and selected 8 representative ones for display; others can refer to Figure 7.

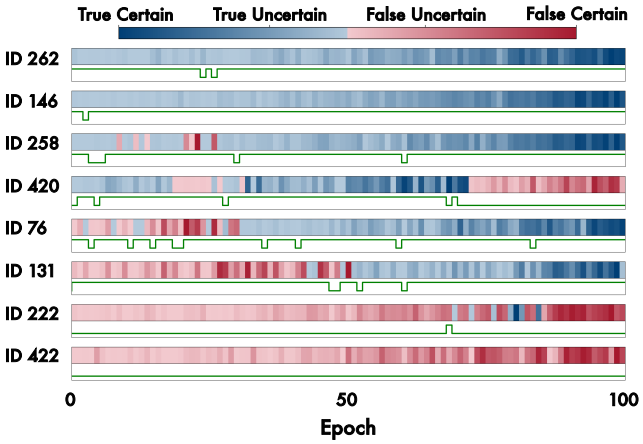

Figure 2: Representative per-problem training dynamics.

Results. Figure 2 shows training dynamics on these problems. The results reveal distinct patterns based on initial confidence-correctness correlation:

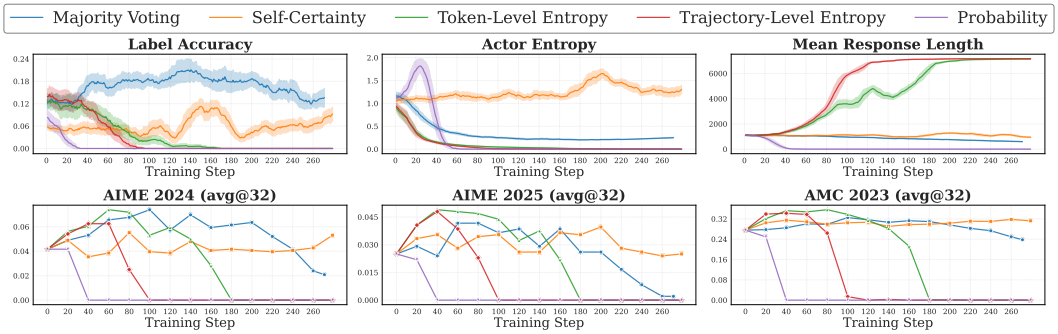

Figure 3: Comparison of intrinsic rewards under tuned settings.

- **High initial correlation** (ID 262/146/258): greedy sampling is correct initially; intrinsic rewards primarily increase confidence and sharpen distribution around already-correct solutions.
- **Moderate initial correlation** (ID 76/131): greedy sampling initially fails, yet rollout sampling frequently yields high-reward correct answers; intrinsic rewards reliably guide the policy from wrong greedy outputs toward correct solutions, showing effective error correction.
- **Low initial correlation** (ID 420): persistent emergence of incorrect samples during training causes policy degradation, driving greedy accuracy from correct to incorrect. This demonstrates misalignment between confidence and correctness leading to systematic bias amplification.
- **Consistently poor correlation** (ID 222/422): high-reward samples remain predominantly incorrect; intrinsic rewards amplify wrong beliefs, driving greedy accuracy deeper into error territory.

This directly validates our theoretical prediction that success depends on the model's initial knowledge quality, where confidence correlates well with correctness.

4.1.3 Sampling-Efficiency Shortcuts

The analysis reveals why intrinsic rewards work: they operate as *sampling-efficiency shortcuts* by rapidly concentrating probability mass on high-confidence trajectories, amplifying gradient signals toward promising directions, and pruning uncertain paths early in training. This reduces the effective search space and accelerates convergence compared to sparse ground-truth rewards, explaining both the efficiency gains and the computational advantages observed in Figure 1.

However, this efficiency comes with a critical trade-off. As Figure 1 demonstrates, prolonged training drives the intrinsic reward signal toward higher values while true accuracy drops, a clear instance of *reward hacking* where optimization of the proxy reward (confidence) diverges from the true objective (correctness). The method's strength as a shortcut becomes its weakness when the initial confidence-correctness correlation is insufficient or when training proceeds beyond the point where this correlation remains reliable.

4.2 How Do Different Methods Fail? Understanding Pathology Patterns

Having established that intrinsic rewards work through uncertainty-performance shortcuts but inevitably face reward hacking, a critical question emerges: do all methods fail in the same way? Our unified framework suggests that while all methods drive distribution sharpening, their different instantiations, including certainty-based or ensemble-based, different anchor distributions, and token-level or answer-level aggregation, should lead to distinct failure modes. Understanding these pathology patterns is essential for both method selection and recognizing when intervention is needed.

Setup. We systematically compare five intrinsic methods using Qwen3-1.7B-Base on DAPO-17k, each optimally tuned (hyperparameter details in Appendix B.3), evaluating across three validation benchmarks to reveal how theoretical differences manifest as distinct failure behaviors in practice.

Distinct Pathology Patterns. As predicted by our unified framework, all methods drive toward higher-confidence regions but fail differently (Figure 3):

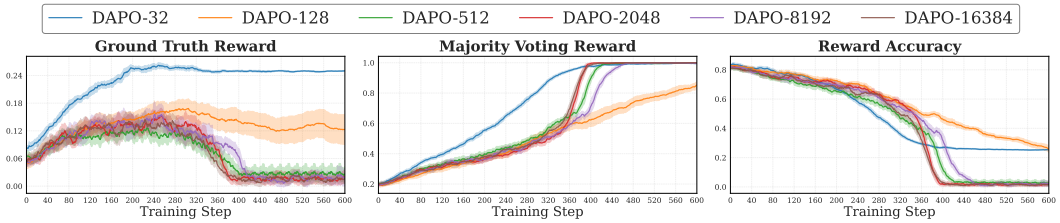

Figure 4: Effect of training dataset size.

- **Stable Methods:** *Self-Certainty* and *Majority Voting* maintain useful training windows longest, with tapering smoothly rather than sudden collapse. The first balances sharpening with its uniform anchor distribution; the second skips token-level artifacts by aggregating at the answer level.
- **Length Collapse:** *Probability* rewards brevity because multiplying token probabilities favors short sequences. The model learns confident but overly brief answers, creating a distinct reward hacking pattern focused on length rather than content quality. Length normalization (using geometric mean or average log-probability) would likely mitigate this bias.
- **Repetition Pathology:** Entropy-based estimators (*Token-Level Entropy*, *Trajectory-Level Entropy*) show complementary failure: their sequence averaging prevents length collapse but encourages repetitive, predictable patterns as the model exploits low-entropy sequences.

These distinct failure modes directly reflect the theoretical differences in our unified framework, influencing not just convergence rate but failure mode. Recognizing these patterns provides early warning signals for intervention.

4.3 Do These Methods Always Cause Model Collapse? No, Safe at Test Time

The previous sections established that intrinsic rewards drive convergence toward deterministic policies and may result in model collapse. This raises a fundamental question: do intrinsic methods always lead to model collapse, or can they be safely applied under specific conditions? Here, we demonstrate that model collapse can be prevented when training data is sufficiently small and domain-specific, making intrinsic rewards particularly suitable for test-time training scenarios.

4.3.1 Small Datasets Prevent Model Collapse

Setup. We trained Qwen3-1.7B-Base on randomly sampled training subsets from DAPO-17k with sizes $\{32, 128, 512, 2048, 8192, 16384\}$. To ensure fair comparison, we fix the global batch size to 32 and adjust training epochs so all settings complete exactly 600 optimization steps.

Results. Figure 4 reveals a clear threshold effect: training with 32 or 128 samples maintains stable performance without model collapse. Critically, while DAPO-32 achieves rapid consensus (*Majority Voting Reward* $\rightarrow 1$), but it preserves high *Ground Truth Reward*. In contrast, larger datasets (≥ 512 samples) consistently exhibit model collapse. This threshold suggests that intrinsic rewards become dangerous when training data provides sufficient statistical power to reinforce systematic biases that could mislead the majority voting mechanism.

4.3.2 Test-Time Training as Optimal Application Domain

Setup. Building on the small dataset insights, we examine test-time training where models are adapted directly on the target evaluation domains. We trained Qwen3-1.7B-Base separately on AMC23 (40 samples) and DAPO-17k ($\sim$17,000 samples), using a global batch size of 40 for both. For AMC23, we performed multi-epoch training.

Results. Figure 5 demonstrates that test-time training successfully prevents model collapse, shown with a higher and stable *Ground Truth Reward* with early complete consensus (*Majority Voting Reward*), and followed with increase then stable performance among in-distribution AMC23 and out-of-distribution AIME24. The key insight is that domain-specific, small-scale training creates conditions where the model's confidence genuinely correlates with correctness within the narrow problem distribution. This makes intrinsic rewards a reliable proxy for true performance rather than a source of bias.

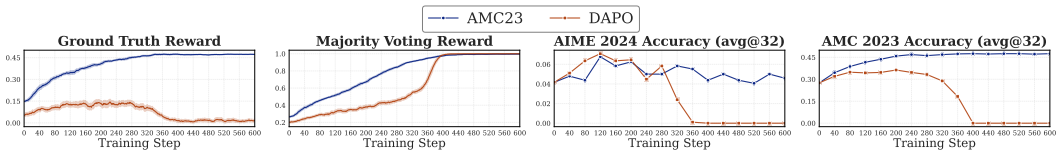

Figure 5: Comparison between training and test-time.

5 Discussion

5.1 Does Intrinsic Reward Methods Truly Improve Capabilities?

Section 4 demonstrates that prevailing intrinsic reward approaches predominantly leverage uncertainty reduction as a mechanism for enhancing performance. This observation motivates a critical question: do such approaches truly enhance a model's capability, or do they merely improve the self-consistency of its outputs? Take the TTRL method as an example. TTRL explicitly models the self-consistency across m model outputs through majority voting and leverages it as a supervisory signal. This design seems to suggest that TTRL may simply push the model toward the performance upper bound implied by the base model under majority voting. In other words, while TTRL might steadily improve the $pass@1$ metric, it would be unlikely to surpass the base model's $maj@m$ performance, thereby failing to deliver substantive gains beyond consistency alignment.

Table 2: Comparisons before and after TTRL. The results show that TTRL-trained models significantly surpass the base models' majority-vote performance in accuracy.

Metric	Qwen2.5-Math-1.5B	Qwen2.5-Math-7B
maj@2	28.09	33.23
maj@4	33.68	41.20
maj@8	37.23	45.73
maj@16	38.10	47.98
maj@32	38.43	49.19
maj@64	38.17	49.87
maj@128	37.86	50.14
maj@256	37.55	50.40
maj@512	37.41	50.60
maj@1024	37.30	50.79
avg@32 (w/ TTRL)	48.90	68.10
Δ	+11.60	+17.31

However, our experiments reveal the opposite, as shown in Table 2. Specifically, we applied TTRL to Qwen2.5-Math-1.5B and Qwen2.5-Math-7B on AIME 2024 (30 samples) with a train batch size of 30 for 100 epochs, and directly compared the base models' $maj@1024$ with the $pass@1$ ($avg@32$) of the TTRL-trained models. Since the majority voting performance converges rapidly once the sample size reaches 32, $maj@1024$ can be reasonably regarded as a close approximation to $maj@\infty$. Strikingly, our results show that even the $pass@1$ metric of the TTRL-trained models significantly exceeds the $maj@\infty$ performance of the base models. This finding demonstrates that TTRL does far more than enforce internal self-consistency: it genuinely enhances the model's ability to generate accurate predictions. Put differently, TTRL enables the model to solve a broader range of problems than the base model, thereby delivering meaningful improvements in real-world performance.

5.2 An Unexpected Application: Model-Task Prior Indicator Beyond Pass@k

The ongoing debate about whether RLVR enables genuine discovery or merely sharpening has intensified focus on understanding model capabilities before expensive training (Yue et al., 2025; Liu et al., 2025b). Current approaches rely on pass@k metrics (Wu et al., 2025a), but these are limited to tasks with objective answers and fail in subjective domains. We propose an alternative diagnostic: since intrinsic reward training dynamics directly reflect confidence-correctness correlation quality (the *model-task prior*), early training behavior serves as a rapid indicator of RL trainability within tens of steps rather than full training evaluation.

Experimental Validation. We test this diagnostic by varying both models (8 models across Qwen/Llama families at different training stages) and datasets of varying difficulty. Since model-task prior depends on both model capabilities and task characteristics, we examine training dynamics across these dimensions. In the main text, each model trains on DAPO-17k for one epoch. Complete experimental details and dataset variation results appear in Appendix B.4 and B.5.

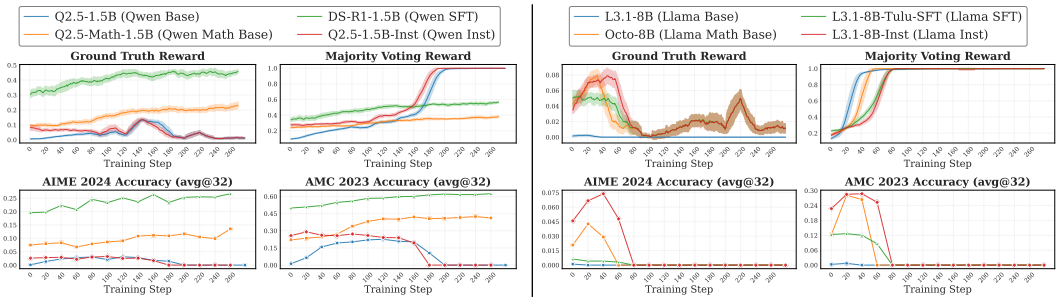

Figure 6: Training dynamics across different training stages in Qwen (Left), LLaMA (Right) family.

Results and Diagnostic Value. The results in Figure 6 reveal striking architectural differences: Qwen models demonstrate superior priors with math-specialized and SFT variants maintaining stable training (Majority Voting Reward 0.3-0.6), while Llama models systematically collapse earlier with base variants failing by step 40. This diagnostic offers key advantages over $pass@k$: rapid assessment within 50 steps, applicability to subjective tasks, and direct connection to RL trainability rather than just performance metrics. The minimal diagnostic cost provides actionable insights about whether models can benefit from reinforcement learning approaches.

5.3 How to Scale RL with URLVR Methods at Train-Time?

While we focus on intrinsic rewards derived from model confidence and consistency, the fundamental challenge that removing human supervision from RL training extends beyond these approaches. We discuss two promising alternatives that use external data or computational asymmetries to generate verifiable rewards without ground truth labels.

Leveraging Unlabeled Data for Reward Generation. Large-scale unlabeled corpora offer natural sources of verifiable signals that can replace human supervision. RPT (Dong et al., 2025) exemplifies this approach by transforming next-token prediction into an RL task, where models receive binary rewards for correctly predicting tokens from unlabeled text. This converts trillions of tokens into scalable reward signals, enabling reasoning improvement through the standard pretraining objective. Similarly, SEAL (Zweiger et al., 2025) employs a meta-learning approach where models generate their own fine-tuning data by producing QA pairs from unlabeled contexts. The model receives rewards based on downstream performance after self-supervised adaptation, creating an autonomous improvement loop without external supervision.

Exploiting Generation-Verification Asymmetries. Many problem domains exhibit computational asymmetries where verifying solutions is substantially easier than generating them (Burns et al., 2023). This creates opportunities for autonomous ground truth generation without human labels. For example, LADDER (Simonds & Yoshiyama, 2025) demonstrates this for math integration, using numerical verification to provide reward signals. Absolute Zero (Zhao et al., 2025a) applies similar principles to coding tasks, where Python execution provides automatic correctness verification. These methods generate truly verifiable rewards that align with task correctness rather than proxy signals.

These approaches offer significant scalability advantages over intrinsic methods by exploiting vast unlabeled corpora rather than relying on model-internal signals that may misalign with task objectives.

6 Conclusion

This work explores how Unsupervised RLVR scales LLMs via a unified framework for intrinsic reward methods. We show that these rewards sharpen outputs around confident predictions, enabling efficient gains when confidence aligns with correctness but amplifying errors when it does not. Empirical results reveal distinct failure modes yet also show that collapse can be avoided in small, domain-specific settings, making test-time training a natural application. Beyond these findings, early training dynamics emerge as a lightweight diagnostic of model-task priors, offering a fast alternative to $pass@k$ for assessing RL trainability. Together, these results outline the limits of intrinsic rewards and highlight the need for external signals and hybrid paradigms for robust, scalable gains.

Ethics Statement

This work investigates intrinsic reward mechanisms for unsupervised reinforcement learning in large language models. While our work advances understanding of AI self-improvement, we acknowledge key ethical considerations. Our findings about reward hacking highlight risks if these methods were deployed without safeguards, so that systems might become overconfident in incorrect solutions, potentially causing harmful outputs in critical applications. Our identification of "safe" conditions should not be interpreted as universal guarantees, as model behavior can be unpredictable in novel contexts. We emphasize that our findings aim to improve understanding of limitations and appropriate use cases rather than encourage unconstrained deployment. Practitioners should carefully assess the confidence-correctness correlation in their applications and implement monitoring systems.

Reproducibility Statement

We provide complete materials for reproducing our theoretical and empirical findings. Theoretical contributions include mathematical derivations of the unified framework and formal proofs in Appendix A.3. Experimental implementations use standardized frameworks (veRL/GRPO) with hyperparameters in Table 5 and tuning procedures in Appendix B.3. Code for all five intrinsic reward methods is provided in the supplementary materials. Experiments use publicly available models (Qwen series) and datasets (DAPO-17k, AIME, AMC) are shown in the supplementary materials. All evaluation metrics are defined in Appendix B.2, and our codebase enables reproduction of results in all figures and tables.

References

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning. *arXiv preprint arXiv:2505.15134*, 2025.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arxiv preprint arXiv: 2507.06261*, 2025.

Qingxiu Dong, Li Dong, Yao Tang, Tianzhu Ye, Yutao Sun, Zhifang Sui, and Furu Wei. Reinforcement pre-training. *arXiv preprint arXiv:2506.08007*, 2025.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.

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

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

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Andrew Kiruluta, Andreas Lemos, and Priscilla Burity. History-aware cross-attention reinforcement: Self-supervised multi turn and chain-of-thought fine-tuning with vllm. *arXiv preprint arXiv:2506.11108*, 2025a.

Andrew Kiruluta, Andreas Lemos, and Priscilla Burity. A self-supervised reinforcement learning approach for fine-tuning large language models using cross-attention signals. *arXiv preprint arXiv:2502.10482*, 2025b.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13:9, 2024.

Pengyi Li, Matvey Skripkin, Alexander Zubrey, Andrey Kuznetsov, and Ivan Oseledets. Confidence is all you need: Few-shot rl fine-tuning of language models. *arXiv preprint arXiv:2506.06395*, 2025.

Jia Liu, ChangYi He, YingQiao Lin, MingMin Yang, FeiYang Shen, ShaoGuo Liu, and TingTing Gao. Ettrl: Balancing exploration and exploitation in llm test-time reinforcement learning via entropy mechanism. *arXiv preprint arXiv:2508.11356*, 2025a.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025b.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*, 2025.

OpenAI. Openai o3 and o4-mini system card. *Blog*, 2025.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444*, 2025.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 1, 2025.

Toby Simonds and Akira Yoshiyama. Ladder: Self-improving llms through recursive problem decomposition. *arXiv preprint arXiv:2503.00735*, 2025.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

Carel van Niekerk, Renato Vukovic, Benjamin Matthias Ruppik, Hsien-chin Lin, and Milica Gašić. Post-training large language models via reinforcement learning from self-feedback. *arXiv preprint arXiv:2507.21931*, 2025.

Haoze Wu, Cheng Wang, Wenshuo Zhao, and Junxian He. Mirage or method? how model-task alignment induces divergent rl conclusions, 2025a. URL https://arxiv.org/abs/2508.21188.

Mingqi Wu, Zhihao Zhang, Qiaole Dong, Zhiheng Xi, Jun Zhao, Senjie Jin, Xiaoran Fan, Yuhao Zhou, Yanwei Fu, Qin Liu, et al. Reasoning or memorization? unreliable results of reinforcement learning due to data contamination. *arXiv preprint arXiv:2507.10532*, 2025b.

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.

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. *arXiv preprint arXiv:2503.14476*, 2025.

Wenzhen Yuan, Shengji Tang, Weihao Lin, Jiacheng Ruan, Ganqu Cui, Bo Zhang, Tao Chen, Ting Liu, Yuzhuo Fu, Peng Ye, et al. Wisdom of the crowd: Reinforcement learning from coevolutionary collective feedback. *arXiv preprint arXiv:2508.12338*, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Kongcheng Zhang, Qi Yao, Shunyu Liu, Yingjie Wang, Baisheng Lai, Jieping Ye, Mingli Song, and Dacheng Tao. Consistent paths lead to truth: Self-rewarding reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.08745*, 2025a.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. *arXiv preprint arXiv:2504.05812*, 2025b.

Yanzhi Zhang, Zhaoxi Zhang, Haoxiang Guan, Yilin Cheng, Yitong Duan, Chen Wang, Yue Wang, Shuxin Zheng, and Jiyan He. No free lunch: Rethinking internal feedback for llm reasoning. *arXiv preprint arXiv:2506.17219*, 2025c.

Zizhuo Zhang, Jianing Zhu, Xinmu Ge, Zihua Zhao, Zhanke Zhou, Xuan Li, Xiao Feng, Jiangchao Yao, and Bo Han. Co-reward: Self-supervised reinforcement learning for large language model reasoning via contrastive agreement. *arXiv preprint arXiv:2508.00410*, 2025d.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025a.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025b.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. *arXiv preprint arXiv:2506.10943*, 2025.

A Details for Section 3

A.1 Instantiations of Unified Reward Framework

To understand how different intrinsic methods fit into this framework, we define each component:

Cross-Entropy $\mathbb{H}$. The fundamental building block, defined as $\mathbb{H}(q, \pi_\theta) = -\sum_{v \in \mathcal{V}} q(v) \log \pi_\theta(v)$, measures the divergence between the anchor distribution q and the model distribution π_θ. This captures how "surprised" the model would be by samples from q.

Aggregation Granularity $\mathcal{I}$. Determines the level at which distributions are compared:

- **Token-level**: $\mathcal{I} = \{1, 2, \ldots, |y|\}$, where each element corresponds to a position t in the sequence. The model distribution at position t is $\pi_\theta^t(\cdot) = \pi_\theta(\cdot \mid x, y_{<t})$.
- **Answer-level**: $\mathcal{I} = \{\mathcal{A}\}$, a single element representing the distribution over complete answers $\pi_\theta^{\mathcal{A}} = \pi_\theta(\cdot \mid x)$ where answers are semantic conclusions rather than individual trajectories.

Anchor Distribution q and Sign Factor σ. The anchor q provides the reference point against which the model is compared, while σ determines the optimization direction:

- When q is **uniform** (e.g., $U_{\mathcal{V}}$ in Self-Certainty), we set $\sigma = +1$ to reward divergence from uniformity, encouraging peaked distributions.
- When q is **sharp** (e.g., one-hot δ or the model's own distribution π_θ), we set $\sigma = -1$ to reward alignment with q, reinforcing confident predictions.

Monotonic Transformation ψ. A strictly increasing function (typically identity or exponential) that reshapes the reward signal while preserving relative ordering. Unlike ground truth rewards, where transformations must preserve the optimal ground-truth policy, intrinsic rewards allow a flexible choice of ψ to stabilize training or adjust gradient scales.

Remarks. We highlight two special cases. First, the formulation of Self-Certainty includes an additional $\log |V|$ term. Since this constant is independent of model parameters, it does not affect gradients during RL training. Second, the expression of r_{MV} corresponds to the asymptotic case where the number of rollouts $n \to \infty$. By the law of large numbers, as $n \to \infty$, the majority vote almost surely selects the answer with the highest probability under π_θ^A, i.e. $\arg\max_a \pi_\theta^A(a)$. To make this limit computationally tractable, we use the tempered distribution $\tilde{\pi}_\theta^A(a) = \exp\big(\pi_\theta^A(a)/\tau\big) / \sum_{b \in A} \exp\big(\pi_\theta^A(b)/\tau\big)$ which avoids the undefined $\log 0$ issue; as $\tau \to 0^+$, it collapses to the hard majority indicator $\mathbf{1}\big[\text{ans}(y) = \arg\max_a \pi_\theta^A(a)\big]$, thereby recovering the same limiting behavior as majority voting.

Key Observations. Despite surface-level diversity, the unified framework reveals that all intrinsic rewards share a common mechanism: manipulating cross-entropy to sharpen distributions. The sign factor σ formalizes this: when $\sigma = +1$ (uniform anchor), rewards increase with cross-entropy, pushing toward peaked distributions; when $\sigma = -1$ (sharp anchor), rewards decrease with cross-entropy, reinforcing confident predictions. Beyond this shared mechanism, the framework reveals structured differences that predict distinct behaviors:

- **Granularity ($\mathcal{I}$)** distinguishes **Token-level methods** (Self-Certainty, Token/Trajectory-Level Entropy, Probability), which operate at $\mathcal{I} = \{1, \ldots, |y|\}$ creating local pressure, from **Answer-level methods** (Semantic Entropy, Majority Voting) working at $\mathcal{I} = \{\mathcal{A}\}$. This explains why answer-level methods, which aggregate global consistency, are more stable than those diluted by sequence length.
- **Anchor Choice (q)** determines the convergence target. Uniform distributions (U_V) encourage departure from randomness, while sharp distributions (δ^t, δ^A) reinforce high-probability paths. Sharp anchors create self-reinforcement loops where rewards directly depend on generated outputs.
- **Transformation (ψ)** determines sharpening strength. Exponential transformations ($\psi(z) = \exp(z)$), particularly in Probability methods ($\exp(|y| \cdot z)$), amplify the sharpening effect, predicting faster convergence and earlier model collapse compared to the more gradual reinforcement of identity transformations (z).

A.2 Monotonicity Analysis of General Optimal Policy

The key insight comes from analyzing the monotonicity of the exponent in Equation (1). Since ψ is strictly increasing by design, the behavior depends entirely on σ:

- **Case** $\sigma = +1$: The reward increases with cross-entropy. Sequences where π_θ diverges from q (typically uniform) receive higher rewards, pushing the policy toward more peaked distributions.
- **Case** $\sigma = -1$: The reward decreases with cross-entropy. Sequences where π_θ aligns with q (typically sharp) receive higher rewards, reinforcing existing confident predictions.

Both cases lead to the same outcome: progressive sharpening of the model's distribution, either by moving away from uniformity or by reinforcing peaked predictions.

A.3 Proof of Theorem 1

Geometric Convergence to Deterministic Policy

Consider the training procedure where at each iteration k: (1) sample N rollouts Y_k from $\pi_\theta^{(k)}$, (2) compute majority $\text{maj}_k(Y_k)$, (3) perform one gradient update with reward $r_k(x, y) = \mathbf{1}[\text{ans}(y) = \text{maj}_k(Y_k)]$.
Under assumptions (A1) stable majority $\text{maj}_k = \text{maj}_0$ and (A2) $\eta_k \geq \eta_{\min} > 0$, the probability mass $p_{\text{maj}}^{(k)}$ converges geometrically to 1. As $k \to \infty$:

$$\lim_{k\to\infty} \pi_\theta^{(k)}(y|x) = \begin{cases} \frac{\pi_{\text{ref}}(y|x)}{\sum_{y':\text{ans}(y')=\text{maj}_0(Y_0)} \pi_{\text{ref}}(y'|x)} & \text{if ans}(y) = \text{maj}_0(Y_0), \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Proof.

Step 0: Justifying the Ordering $p_{\text{maj}}^{*,(k+1)} \geq p_{\text{maj}}^{(k+1)} \geq p_{\text{maj}}^{(k)}$.

From the optimal policy of the standard KL-regularized RL objective, if we held reward r_k fixed and performed infinite updates starting from $\pi_\theta^{(k)}$, we would reach the optimal policy with probability mass:

$$p_{\text{maj}}^{*,(k+1)} = \frac{\alpha \cdot p_{\text{maj}}^{(k)}}{1 + (\alpha - 1)p_{\text{maj}}^{(k)}}, \qquad \alpha := e^{1/\beta} > 1 \tag{6}$$

Lower bound ($p_{\text{maj}}^{(k+1)} \geq p_{\text{maj}}^{(k)}$): The policy gradient is $\nabla_\theta J = \mathbb{E}_{\pi_\theta}[r_k(x, y)\nabla_\theta \log \pi_\theta(y|x)]$. Since $r_k(x, y) = 1$ for majority trajectories and $r_k(x, y) = 0$ for non-majority trajectories, the gradient increases $\log \pi_\theta(y|x)$ only for majority trajectories. Under standard policy gradient convergence (positive rewards increase trajectory probabilities), this tends to increase $p_{\text{maj}}^{(k)}$. We validate this empirically in Appendix A.5.

Upper bound ($p_{\text{maj}}^{(k+1)} \leq p_{\text{maj}}^{*,(k+1)}$): Since $\pi_\theta^{*,k}$ maximizes the KL-regularized objective for fixed r_k, our single-step update cannot exceed this optimal value.

Step 1: Effective Update Rule.

We model the actual update with step efficiency $\eta_k \in (0, 1]$:

$$p_{\text{maj}}^{(k+1)} = p_{\text{maj}}^{(k)} + \eta_k \cdot (p_{\text{maj}}^{*,(k+1)} - p_{\text{maj}}^{(k)}) \tag{7}$$

Substituting Equation (6):

$$\begin{aligned} p_{\text{maj}}^{(k+1)} &= p_{\text{maj}}^{(k)} + \eta_k \left(\frac{\alpha \cdot p_{\text{maj}}^{(k)}}{1 + (\alpha - 1)p_{\text{maj}}^{(k)}} - p_{\text{maj}}^{(k)} \right) \\ &= p_{\text{maj}}^{(k)} + \eta_k \cdot \frac{(\alpha - 1)(1 - p_{\text{maj}}^{(k)})p_{\text{maj}}^{(k)}}{1 + (\alpha - 1)p_{\text{maj}}^{(k)}} \end{aligned} \tag{8}$$

Step 2: Error Dynamics.

Define the error from the fixed point 1 as:

$$\epsilon^{(k)} := 1 - p_{\text{maj}}^{(k)} \in (0, 1) \tag{9}$$

Substituting into Equation (8):

$$\begin{aligned}\epsilon^{(k+1)} &= 1 - p_{\text{maj}}^{(k+1)} \\ &= \epsilon^{(k)} - \eta_k \cdot \frac{(\alpha-1)(1-\epsilon^{(k)})\epsilon^{(k)}}{1+(\alpha-1)(1-\epsilon^{(k)})} \\ &= \epsilon^{(k)} \left(1 - \eta_k \cdot \frac{(\alpha-1)(1-\epsilon^{(k)})}{\alpha-(\alpha-1)\epsilon^{(k)}}\right)\end{aligned} \tag{10}$$

Step 3: Monotonic Decrease.

Since $\alpha > 1$, $\epsilon^{(k)} \in (0, 1)$, and $\eta_k \in (0, 1]$, we have:

$$0 < \frac{(\alpha-1)(1-\epsilon^{(k)})}{\alpha-(\alpha-1)\epsilon^{(k)}} < 1 \tag{11}$$

Therefore:

$$0 < 1 - \eta_k \cdot \frac{(\alpha-1)(1-\epsilon^{(k)})}{\alpha-(\alpha-1)\epsilon^{(k)}} < 1 \tag{12}$$

This implies $\epsilon^{(k+1)} < \epsilon^{(k)}$, proving the sequence $\{\epsilon^{(k)}\}$ is strictly decreasing and bounded below by 0.

Step 4: Convergence to Zero.

Let $\ell = \lim_{k\to\infty} \epsilon^{(k)} \geq 0$. Under assumption (A2), $\eta_k \geq \eta_{\min} > 0$. If $\ell > 0$, then for large k, the multiplier in Equation (10):

$$1 - \eta_k \cdot \frac{(\alpha-1)(1-\epsilon^{(k)})}{\alpha-(\alpha-1)\epsilon^{(k)}} \leq 1 - \eta_{\min} \cdot \frac{\alpha-1}{\alpha} < 1 \tag{13}$$

is bounded away from 1, causing continued decay. The only consistent limit is $\ell = 0$. Therefore:

$$\epsilon^{(k)} \to 0 \qquad \text{equivalently} \qquad p_{\text{maj}}^{(k)} \to 1 \tag{14}$$

Step 5: Geometric Convergence Rate.

From Equation (10), for large k when $\epsilon^{(k)}$ is small:

$$\epsilon^{(k+1)} \approx \epsilon^{(k)} \left(1 - \eta_k \cdot \frac{\alpha-1}{\alpha}\right) \tag{15}$$

Under assumption (A2):

$$\epsilon^{(k+1)} \leq \left(1 - \eta_{\min} \cdot \frac{\alpha-1}{\alpha}\right) \epsilon^{(k)} \tag{16}$$

This establishes geometric convergence with rate depending on $\eta_{\min}$ and $\alpha = e^{1/\beta}$. In the ideal case where $\eta_k = 1$ for all k (each update reaches the optimum), the convergence rate is exactly $\rho = e^{-1/\beta}$.

Step 6: Limiting Policy.

Given assumption (A1) that the majority remains stable at $\text{maj}_0(Y_0)$, as $p_{\text{maj}}^{(k)} \to 1$, all probability mass concentrates on trajectories with $\text{ans}(y) = \text{maj}_0(Y_0)$. The limiting distribution is:

$$\lim_{k\to\infty} \pi_\theta^{(k)}(y|x) = \begin{cases} \frac{\pi_{\text{ref}}(y|x)}{\sum_{y':\text{ans}(y')=\text{maj}_0(Y_0)} \pi_{\text{ref}}(y'|x)} & \text{if } \text{ans}(y) = \text{maj}_0(Y_0), \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

This completes the proof. □

Remark on Assumptions.

- **(A1) Majority stability:** By the Law of Large Numbers, with N rollouts, the empirical majority $\text{maj}_k(Y_k)$ converges to $\arg\max_a \pi_\theta^{(k)}(a|x)$ as $N \to \infty$. Since $p_{\text{maj}}^{(k)}$ increases monotonically, the argmax remains maj_0 throughout training. We validate this empirically with $N = 1024$ rollouts in Appendix A.5, where the majority never flipped across 200 iterations.
- **(A2) Non-trivial progress:** We assume $\eta_k \geq \eta_{\min} > 0$, meaning each gradient update makes non-trivial progress. We validate this empirically: our experiments show consistent monotonic increase in p_{maj} and convergence to 1.0 under extreme off-policy settings (Appendix A.5).

A.4 Generalized Sharpening Analysis via Unified Reward Framework

To address the concern that Theorem 1 applies only to Majority Voting, and to demonstrate the analytical utility of our unified framework, we provide a generalized sharpening analysis. We show that methods with $\sigma = -1$ share a critical structural property, Reward-Confidence Monotonicity, which creates a persistent pressure toward distribution sharpening.

Note: The following is a proof sketch demonstrating the key convergence mechanism shared by $\sigma = -1$ methods. A fully rigorous treatment requires additional technical conditions that we validate empirically. Methods with $\sigma = +1$ (Self-Certainty) require separate analysis as they reward away from uniform distribution.

Proposition 1 (Sharpening Dynamics for $\sigma = -1$ Methods)**.** *Consider any intrinsic reward with $\sigma = -1$ in the unified framework ($r_{uni} = \psi(-\mathbb{H}(q, \pi))$) where ψ is strictly increasing and q is a sharp anchor. These methods satisfy **Reward-Confidence Monotonicity**:*

$$\pi_\theta(y_a|x) > \pi_\theta(y_b|x) \implies r_{uni}(x, y_a) > r_{uni}(x, y_b) \tag{18}$$

For a dominant trajectory y^* (e.g., majority) and a non-dominant competitor y', this inequality is strict: $r(y^*) > r(y')$. Under iterative KL-regularized updates, this property creates a self-reinforcing feedback loop that drives geometric concentration.

Proof Sketch:

We analyze the dynamics for a dominant trajectory y^* and a competitor y' (for ensemble methods, not in the same class as y^*) where the model initially prefers y^* (i.e., $\pi_k(y^*) > \pi_k(y')$) and assigns it strictly higher reward ($r_k(y^*) > r_k(y')$).

Step 1: Existence of a Positive Reward Gap

Using the unified formula, we justify why the gap is positive for $\sigma = -1$:

- **Self-Reinforcing Anchors** (e.g., Probability): $r(y) = \psi(\log \pi(y))$. Since $\pi_k(y^*) > \pi_k(y')$ and ψ is strictly increasing, $r_k(y^*) > r_k(y')$.
- **Answer-Level Anchors** (e.g., Majority Voting): y^* belongs to the dominant answer class a^*, while y' does not. By construction, $r(y^*) = 1$ and $r(y') = 0$.

In both cases, the intrinsic reward gap is strictly positive: $\Delta_r^{(k)} = r_k(y^*) - r_k(y') > 0$.

Step 2: The Optimization Target

We consider the optimal policy π^* for the current fixed reward landscape r_k. The optimal solution implies a target ratio:

$$\frac{\pi^*(y^*)}{\pi^*(y')} = \frac{\pi_k(y^*)}{\pi_k(y')} \cdot \exp\left(\frac{\Delta_r^{(k)}}{\beta}\right) \tag{19}$$

Since $\Delta_r^{(k)} > 0$, the target ratio is strictly larger than the current ratio.

Gradient Assumption: The gradient $\nabla_\theta J = \mathbb{E}_{\pi_k}[r_k(y)\nabla_\theta \log \pi_\theta(y)]$ assigns positive weight to high-reward trajectories. We assume that policy gradient updates with positive learning rate

η satisfy: if $r(y^*) > r(y')$ and both have positive probability, then the updated policy satisfies $\frac{\pi_{k+1}(y^*)}{\pi_{k+1}(y')} \geq \frac{\pi_k(y^*)}{\pi_k(y')}$. This aligns with standard policy gradient convergence properties.

Step 3: The Reinforcement Loop

The unified framework reveals why this process spirals into determinism. As the policy updates to increase the probability mass on the dominant trajectory:

- For **Self-Reinforcing Anchors** (e.g., Probability), because $r(y) = \psi(\log \pi(y))$, increasing $\pi(y^*)$ directly increases its reward $r(y^*)$.
- For **Answer-Level Anchors** (e.g., Majority Voting), increasing the total probability mass on the dominant answer class a^* increases the reward for all trajectories in that class (since $r \propto \log p(a^*)$).

This creates a positive feedback loop: the update increases the probability of the dominant path, which maintains or widens the reward gap Δ_r, ensuring the pressure to sharpen ($\Delta_r > 0$) persists.

Utility of the Framework:

This derivation demonstrates that the "rich-get-richer" dynamic is a structural inevitability for any method where the reward function is monotonically aligned with the model's own confidence ($\sigma = -1$). The framework allows us to identify this shared property and predict that all such methods will drive the policy toward deterministic outputs, regardless of whether this leads to success (when aligned with correctness) or failure (when misaligned).

Remark on $\sigma = +1$ Methods:

Self-Certainty ($\sigma = +1$) rewards higher when away from uniform distribution. Therefore, $\pi(y_a) > \pi(y_b)$ does not imply $r(y_a) > r(y_b)$. A high-probability output and a very low-probability output could both have high KL-divergence from uniform, violating direct Reward-Confidence Monotonicity. Its sharpening mechanism requires separate analysis.

While methods with $\sigma = +1$ do not strictly align reward with raw confidence, they still induce sharpening by penalizing high-entropy distributions. By maximizing the distance from a uniform anchor, the optimization landscape naturally favors peaked, low-entropy policies, effectively driving the model toward determinism.

Empirical Validation:

To substantiate the assumptions in this proof sketch, we provide empirical validation for different intrinsic reward methods in Figure 3 and Appendix B.3, confirming that Reward-Confidence Monotonicity is not just a theoretical construct but the actual driver of the observed training dynamics.

A.5 Empirical Validation of Theoretical Assumptions

We empirically validate the key assumptions in Theorem 1 through three targeted experiments.

Experiment 1: Validation of Ordering and Majority Stability

Setup: We trained on a single problem from MATH-500 with $N = 1024$ rollouts (reducing majority vote randomness) for 50 steps. We randomly selected 4 problems and monitored whether the majority answer $\text{maj}_k(Y_k)$ remains stable and whether $p_{\text{maj}}^{(k)}$ increases monotonically.

Step	1	2	3	4	5	6	7	8	9	10
level3_id146	12.70	15.53	15.92	16.21	18.46	22.07	22.56	24.80	31.35	39.36
level1_id187	6.64	6.69	6.84	7.42	10.45	11.04	11.43	11.82	15.82	18.46
level1_id262	15.14	17.19	17.48	18.85	20.02	22.07	24.12	25.20	34.67	39.84
level3_id122	11.33	12.01	12.40	14.06	17.87	18.46	20.31	21.29	33.59	33.89

Table 3: Monotonic increase of p_{maj} (%) in early training steps.

Results for monotonic increase of p_{maj}. Table 3 shows p_{maj} values for the first 10 steps. All 4 problems exhibit strict monotonic increase at every single step, confirming the lower bound

$p_{\text{maj}}^{(k+1)} \geq p_{\text{maj}}^{(k)}$ of the ordering. We also found that the majority answer remained stable across all iterations. This confirms both the ordering and assumption (A1) on majority stability.

Step	5	10	15	20	25	30	35	40	45	50
level3_id146	18.46	39.36	48.93	91.11	95.41	98.14	98.54	99.02	99.61	99.80
level1_id187	11.04	18.46	26.37	79.88	89.84	93.07	96.09	97.66	98.54	99.02
level1_id262	22.07	39.84	51.37	90.14	95.90	96.80	97.46	98.05	98.63	99.41
level3_id122	17.87	33.59	43.55	84.28	92.19	93.26	95.80	96.09	98.34	98.54

Table 4: Geometric convergence of p_{maj} (%) to 1.0 over 50 training steps.

Results for convergence of p_{maj}. Table 4 shows the same 4 problems trained for 50 steps. All problems converge from initial values toward near-complete concentration (98.54%-99.80% at step 50), demonstrating the convergence predicted by Theorem 1. This validates assumption (A2) on non-trivial progress and confirms that the iterative training procedure with policy-dependent rewards does indeed converge to deterministic policies.

Experiment 2: Batch Training Validation

Setup. Our main experiments (Figure 1) train on batches with $N = 8$ rollouts per problem.

Results. Majority Voting Reward (batch-averaged p_{maj}) shows a consistent increasing trend across all methods, confirming the lower bound $p_{\text{maj}}^{(k+1)} \geq p_{\text{maj}}^{(k)}$ holds in practical batch training settings. Small fluctuations occur due to finite rollouts ($N = 8$) and batch variance, but the monotonic trend is clear.

Experiment 3: Fixed Reward Convergence Validation

Setup. To validate that the closed-form optimal policy in Equation (3) is achievable when reward is held fixed, we conducted an extreme off-policy experiment. We used global batch size 1024 with mini-batch size 1, generated one-time rollout (with $N = 8$ for each of 1024 prompts), and performed 1024 gradient updates using rewards computed solely from the initial rollout majority. This setup tests whether solving a single KL-regularized RL objective can converge to the theoretical optimum when the reward signal remains constant.

Results. After 1024 mini-updates using the same fixed reward signal, the Majority Voting Reward reached 1.0 (complete convergence), while validation performance on AIME24, AIME25, and AMC23 dropped to zero. This confirms that the convergence point predicted by Equation (3) is achievable with sufficient updates.

A.6 Optimal Policies Induced by other Intrinsic Rewards

Optimal Policy of the Reward Function r_{SC}. For the Self-Certainty reward function r_{SC}, it instantiates our unified framework with token-level granularity $\mathcal{I} = \{1, 2, ..., |y|\}$, anchor distribution $q = \{U_V\}_{t=1}^{|y|}$ (uniform distribution over vocabulary), model distribution $\pi = \{\pi_\theta^t\}_{t=1}^{|y|}$, sign factor $\sigma = +1$, and transformation $\psi(z) = z$. As established previously, for any input x, the token-level predictive distribution of the model is evaluated against the current policy π. Due to $\sigma = +1$, the farther this distribution deviates from the uniform distribution (i.e., the higher the model's confidence), the larger the reward $r_{\text{SC}}(x, y)$. Consequently, after a single step of policy update, the optimal probability $\pi_\theta(y|x)$ increases for such high-confidence sequences, whereas it decreases when the per-token distribution is close to uniform (low confidence). Thus, r_{SC} encourages the model to generate answers that are already preferred by the prior policy.

A detailed derivation is provided below. The Self-Certainty based reward is defined as:

$$r_{\text{SC}}(x, y) = \frac{1}{|y|} \sum_{t=1}^{|y|} D_{\text{KL}}\big(U \parallel \pi_\theta(\cdot \mid x, y_{<t})\big) = -\log |V| - \frac{1}{|y|\,|V|} \sum_{t=1}^{|y|} \sum_{v=1}^{|V|} \log \pi_\theta^t(y_t = v). \quad (20)$$

Within the KL-regularized RL framework, dropping the constant term $-\log |V|$, the one-step optimal policy becomes:

$$\pi_\theta(y|x) \propto \pi_{\text{ref}}(y|x) \exp\left(-\frac{1}{\beta\,|y|\,|V|} \sum_{t=1}^{|y|} \sum_{v=1}^{|V|} \log \pi_\theta^t(y_t = v) \right). \tag{21}$$

Therefore, whenever the model assigns concentrated probabilities to every token of y (high confidence), the exponent grows, thus increasing the probability of the sequence $\pi_\theta(y|x)$. In summary, the Self-Certainty based reward systematically enhances the model's "self-confidence" with respect to its prior policy.

Optimal Policy of the Reward Function r_H. For the token-level entropy-based reward r_H, it instantiates our unified framework with token-level granularity $\mathcal{I} = \{1, 2, ..., |y|\}$, anchor distribution $q = \{\pi_\theta^t\}_{t=1}^{|y|}$, model distribution $\pi = \{\pi_\theta^t\}_{t=1}^{|y|}$, sign factor $\sigma = -1$, and transformation $\psi(z) = z$. Maximizing r_H is equivalent to minimizing the predictive entropy at every position, thereby discouraging the model from spreading its probability mass across multiple candidate tokens and hence increasing its decisiveness.

A detailed derivation is provided below. The entropy-based reward is defined as:

$$r_H(x, y) = -\frac{1}{|y|} \sum_{t=1}^{|y|} H\big(\pi_\theta(\cdot \mid x, y_{<t})\big) = -\frac{1}{|y|} \sum_{t=1}^{|y|} \sum_{v=1}^{|V|} \pi_\theta^t(y_t = v) \log \pi_\theta^t(y_t = v). \tag{22}$$

Within the KL-regularized RL framework, the one-step optimal policy becomes:

$$\pi_\theta(y|x) \propto \pi_{\text{ref}}(y|x) \exp\left(-\frac{1}{\beta\,|y|} \sum_{t=1}^{|y|} \sum_{v=1}^{|V|} \pi_\theta^t(y_t = v) \log \pi_\theta^t(y_t = v) \right). \tag{23}$$

Consequently, if the predictive distribution of an output sequence y exhibits high entropy (i.e., the per-token distributions are close to uniform), the negative-entropy reward r_H is strongly negative, which suppresses the exponential weight and reduces $\pi_\theta(y|x)$. Conversely, low entropy (highly peaked per-token distributions) yields $r_H \approx 0$, thus the sequence probability is enhanced after normalization. Therefore, the entropy-based reward r_H encourages the model to generate answers whose token-level distributions are sharply concentrated, effectively boosting its "self-confidence" under the prior policy.

Optimal Policy of the Reward Function r_{Traj}. For the trajectory-level entropy-based reward r_{Traj}, it instantiates our unified framework with token-level granularity $\mathcal{I} = \{1, 2, ..., |y|\}$, anchor distribution $q = \{\delta^t\}_{t=1}^{|y|}$, model distribution $\pi = \{\pi_\theta^t\}_{t=1}^{|y|}$, sign factor $\sigma = -1$, and transformation $\psi(z) = z$. For a given input x, the model's predictive distribution is evaluated at every token. With $\sigma = -1$, the closer the distribution is to the one-hot reference δ^t (i.e., the higher the model's confidence in each ground-truth token), the larger the reward $r_{\text{Traj}}(x, y)$. Hence, after one policy-update step, the optimal probability $\pi_\theta(y|x)$ increases for such high-confidence trajectories, and decreases otherwise. Thus, r_{Traj} encourages the model to generate sequences that already enjoy high probability under the prior policy.

The trajectory-level reward is defined as:

$$r_{\text{Traj}}(x, y) = \frac{1}{|y|} \sum_{t=1}^{|y|} \log \pi_\theta(y_t \mid x, y_{<t}) = \frac{1}{|y|} \log \pi_\theta(y \mid x). \tag{24}$$

Within the KL-regularized RL framework, the one-step optimal policy becomes:

$$\pi_\theta(y|x) \propto \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta\,|y|} \log \pi_\theta(y \mid x) \right) = \pi_{\text{ref}}(y|x) \cdot \big[\pi_\theta(y \mid x)\big]^{\frac{1}{\beta|y|}}. \tag{25}$$

Consequently, whenever the model assigns a higher prior probability to a sequence y, the weighted product term is amplified, thereby increasing its normalized probability $\pi_\theta(y|x)$. Therefore, the

trajectory-level entropy reward boosts the probability of sequences that are already likely under the current policy π_θ.

Optimal Policy of the Reward Function r_{Prob}. For the probability-based reward function r_{Prob}, it instantiates our unified framework with token-level granularity $\mathcal{I} = \{1, 2, ..., |y|\}$, anchor distribution $q = \{\delta^t\}_{t=1}^{|y|}$, model distribution $\pi = \{\pi_\theta^t\}_{t=1}^{|y|}$, sign factor $\sigma = -1$, and transformation $\psi(z) = \exp(z)$. For a given input x, the model's predictive distribution is evaluated at every token. With $\sigma = -1$, the closer the distribution is to the one-hot reference δ^t (i.e., the higher the model's confidence in each ground-truth token), the larger the reward $r_{\text{Prob}}(x, y)$ will be. Hence, after one policy-update step, the optimal probability $\pi_\theta(y|x)$ increases for such high-confidence trajectories, and decreases otherwise. Thus, r_{Prob} encourages the model to generate sequences that already enjoy high probability under the prior policy.

The probability-based reward is defined as:

$$r_{\text{Prob}}(x, y) = \prod_{t=1}^{|y|} \pi_\theta(y_t \mid x, y_{<t}) = \pi_\theta(y \mid x). \tag{26}$$

Within the KL-regularized RL framework, the one-step optimal policy becomes:

$$\pi_\theta(y|x) \propto \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta}\pi_\theta(y \mid x)\right). \tag{27}$$

Consequently, whenever the model assigns a high joint probability to a sequence y, the exponential weight is amplified, thereby increasing its normalized probability $\pi_\theta(y|x)$. The probability-product reward thus directly reinforces sequences that are already likely under the current policy, enhancing the model's preference for "high-likelihood" trajectories.

Optimal Policy of the Reward Function r_{EMPO}. For the answer-space probability-distribution reward r_{EMPO} employed by the EMPO algorithm, it instantiates our unified framework with answer-level granularity $\mathcal{I} = \{\mathcal{A}\}$, anchor distribution $q = \delta^A$, model distribution $\pi = \pi_\theta^A$, $\sigma = -1$, and transformation $\psi(z) = \exp(z)$. For a given input x, multiple roll-outs are used to estimate the current policy's distribution over the answer space. With $\sigma = -1$, the closer this distribution is to the one-hot reference δ^A (i.e., the more probability mass is assigned to the extracted answer), the larger the reward $r_{\text{EMPO}}(x, y)$ will be. Hence, after one policy-update step, the optimal probability $\pi_\theta(y|x)$ increases for sequences that endorse the high-probability answer, while it decreases for all others. Maximizing r_{EMPO} is therefore equivalent to driving the model to become more decisive at the answer level, thereby improving the consistency and determinism of the generated outputs.

Formally, the reward is defined as:

$$r_{\text{EMPO}}(x, y) = \pi_\theta(\text{ans}(y) \mid x), \quad \text{where } \pi_\theta(\text{ans}(y) \mid x) = \sum_{\text{ans}(y')=\text{ans}(y)} \pi_\theta(y' \mid x). \tag{28}$$

Within the KL-regularised RL framework, the one-step optimal policy is:

$$\pi_\theta(y \mid x) \propto \pi_{\text{ref}}(y \mid x) \, \exp\left(\frac{\pi_\theta(\text{ans}(y) \mid x)}{\beta}\right). \tag{29}$$

As evidenced by Equation (29), a single EMPO update re-weights each sequence by a factor of $\exp\big(\pi_\theta(\text{ans}(y) \mid x)/\beta\big)$ that depends on the current answer-level probability. After normalization, answers that already enjoy high probability under the prior policy gain additional mass, whereas low-probability answers suffer a decrease. Consequently, the optimal policy at each step systematically shifts the overall probability mass toward the high-probability region of the prior policy.

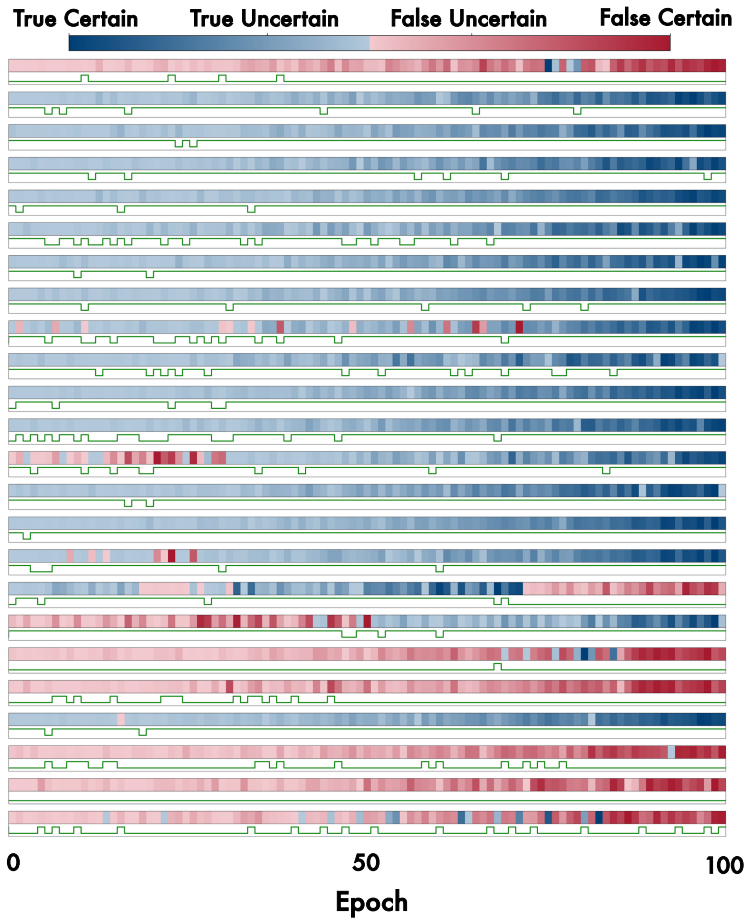

Figure 7: Examples of per-problem training dynamics from MATH-500.

Table 5: Default hyperparameters for training.

Model	Dataset	Training Temperature	Global Batch Size	Mini Batch Size	Rollout Number	Regularization	Max Context Length	Learning Rate	Epoch
Qwen3-1.7B-Base	DAPO-17k	1.0	64	64	8	w/o KL w/o Entropy	8192	1e-6	1

B Details for Section 4

B.1 Experimental Setup

Implementation Details. All experiments are conducted using the VeRL framework (Sheng et al., 2025) with the GRPO algorithm. Unless stated otherwise, we utilize the default configuration outlined in Table 5. We implement five representative intrinsic rewards by customizing the `RewardManager` module of VeRL, following the reward formulations in Table 6 and Table 7:

- **Ensemble-Based Reward Estimators:** Majority Voting
- **Certainty-Based Reward Estimators:** Self-Certainty, Token-Level Entropy, Trajectory-Level Entropy, and Probability

Evaluation Protocol. We evaluate on three challenging mathematics benchmarks: AIME 2024 (Li et al., 2024), AIME 2025 (Balunović et al., 2025), and AMC 2023 (Li et al., 2024). Following standard practice, we generate 32 solutions per problem using a temperature of 0.6 and a top-p value of 0.95, and report the mean accuracy at 32 solutions (mean@32).

Table 6: Overview of certainty-based rewards, estimators and their formulas.

Method	Estimator	Formula
RLIF	**Self-Certainty**	$r(x,y) = \frac{1}{\lvert y\rvert}\sum_{t=1}^{\lvert y\rvert} D_{\mathrm{KL}}(U \Vert \pi_\theta(\cdot\vert x, y_{<t}))$
EM-RL	**Trajectory-Level Entropy**	$r(x,y) = \frac{1}{\lvert y\rvert}\sum_{t=1}^{\lvert y\rvert} \log \pi_\theta(y_t\vert x, y_{<t})$
EM-RL, RENT	**Token-Level Entropy**	$r(x,y) = -\frac{1}{\lvert y\rvert}\sum_{t=1}^{\lvert y\rvert} H(\pi_\theta(\cdot\vert x, y_{<t}))$
RLSC	**Probability**	$r(x,y) = \prod_{t=1}^{\lvert y\rvert} \pi_\theta(y_t\vert x, y_{<t})$
RLSF	**Probability Disparity**	$r(x,y) = \frac{1}{M}\sum_{t=1}^{\lvert a\rvert}\Big[\max_{a_t} \pi_\theta(a_t\vert x,c,a_{<t}) - \max_{a_t \neq \arg\max \pi_\theta} \pi_\theta(a_t\vert x,c,a_{<t})\Big]$

Training Dynamics Monitoring. To monitor reward hacking and validate our theoretical predictions from Section 3, we implement specialized metrics to track the evolution of pseudo-rewards and their alignment with ground truth. These metrics help identify when and how these intrinsic methods transition from beneficial sharpening to pathological collapse.

- **Ensemble-Based Metrics:** For methods using majority voting, we separately track the accuracy of the chosen label and the accuracy of the rewards it generates.
  - *Label Accuracy*: Prompt-level accuracy of majority-voted answers against ground truth, measuring ensemble quality
  - *Reward Accuracy*: Sample-level agreement between pseudo rewards and oracle rewards, capturing "lucky hits" (Zuo et al., 2025) where individual rewards align despite incorrect majority votes
  - *Ground Truth Reward*: Average oracle reward (supervised baseline), computed using actual correctness
  - *Majority Voting Reward*: Average pseudo reward from majority voting, the divergence from *Ground Truth Reward* indicates reward hacking
- **Certainty-Based Metrics:** For certainty-based methods, we measure the correlation between this proxy reward and the actual correctness.
  - *Label Accuracy*: Ground-truth accuracy of the highest-confidence response per prompt, testing whether maximum certainty implies correctness
  - *Point-Biserial Correlation*: Point-biserial correlation between pseudo reward and binary correctness, quantifying the fundamental assumption that confidence predicts accuracy

These metrics collectively diagnose three critical phenomena: (1) pseudo-label quality degradation via *Label Accuracy*, (2) reward signal corruption via the gap between *Majority Voting Reward* and *Ground Truth Reward*, and (3) confidence miscalibration via *Point-Biserial Correlation*. Mathematical definitions and implementation details are provided in Appendix B.2.

B.2 Calculation of Training Dynamics

We provide mathematical definitions for the metrics used to monitor training dynamics. These metrics diagnose reward hacking and validate theoretical predictions about distribution sharpening.

B.2.1 Notation

Let $\mathcal{D} = \{(x_i, a_i^*)\}_{i=1}^M$ denote the training dataset with M prompts, where x_i is the i-th prompt and a_i^* is its ground-truth answer. For each prompt x_i, we generate N rollout responses $\{y_{i,j}\}_{j=1}^N$ from the current policy π_θ, where each response $y_{i,j}$ contains a trajectory and an extracted answer $\mathrm{ans}(y_{i,j})$.

Define the following:

Table 7: Overview of ensemble-based rewards, estimators and their formulas.

Method	Estimator	Formula
TTRL, SRT, ETTRL	**Majority Voting**	$r(x,y) = \mathbf{1}\Big[y = \arg\max_{y'} \sum_{i=1}^{N} \mathbf{1}[y_i = y']\Big], \quad \{y_i\}_{i=1}^{N} \sim \pi_\theta(\cdot\|x)$
Co-Reward	**Majority Voting across Rephrased Question**	$r(x,y) = \mathbf{1}\left[y = \arg\max_{y^*} \sum_{i=1}^{N} \mathbf{1}[y_i = y^*]\right], \quad \{y_i\}_{i=1}^{N} \sim \pi_\theta(\cdot\|x)$ $+ \mathbf{1}\left[y = \arg\max_{y^*} \sum_{j=1}^{N} \mathbf{1}[y'_j = y^*]\right], \quad \{y'_j\}_{j=1}^{N} \sim \pi_\theta(\cdot\|\text{rephrase}(x))$
RLCCF	**Self-consistency Weighted Voting**	$r(x,y) = \mathbf{1}\left[y = \arg\max_{a} \sum_{n=1}^{N} \Big(\max_{a'} \sum_{k=1}^{K} \mathbf{1}[o_{n,k} = a']\Big) \cdot \sum_{k=1}^{K} \mathbf{1}[a = o_{n,k}]\right],$ $\{o_{n,k}\}_{k=1}^{K} \sim \pi_{\theta_n}(\cdot\|x), \quad n = 1, \ldots, N$
EMPO	**Semantic Similarity**	$r(x,y) = \frac{\|\mathcal{C}(y)\|}{G}, \quad \mathcal{C}(y) \in \texttt{SemanticCluster}(\{o_i\}_{i=1}^{G}),$ $\{o_i\}_{i=1}^{G} \sim \pi_\theta(\cdot\|x)$
CoVo	**Trajectory Consistency and Volatility**	$r(x,y) = \frac{1}{G}\left\|\sum_{i=1}^{G} \texttt{Con}(y_i) \cdot [\cos(\texttt{Vol}(y_i)), \sin(\texttt{Vol}(y_i))]\right\| + r_{\text{cur}},$ $\{y_i\}_{i=1}^{N} \sim \pi_\theta(\cdot\|x), \quad G = \|\{i : \text{ans}(y_i) = \text{ans}(y)\}\|$

- $\mathbf{1}[\cdot]$: Indicator function returning 1 if the condition is true, 0 otherwise
- $\text{maj}(x_i)$: Majority-voted answer for prompt x_i, computed as $\arg\max_a \sum_{j=1}^{N} \mathbf{1}[\text{ans}(y_{i,j}) = a]$
- $r_{\text{gt}}(y_{i,j})$: Ground-truth reward for response $y_{i,j}$, equals $\mathbf{1}[\text{ans}(y_{i,j}) = a_i^*]$
- $r_{\text{mv}}(y_{i,j})$: Majority-voting pseudo-reward, equals $\mathbf{1}[\text{ans}(y_{i,j}) = \text{maj}(x_i)]$
- $r_{\text{cert}}(y_{i,j})$: Certainty-based reward (e.g., self-certainty, entropy) for response $y_{i,j}$

B.2.2 Ensemble-Based Metrics

Label Accuracy Measures the prompt-level accuracy of majority-voted answers:

$$\text{Label Accuracy} = \frac{1}{M} \sum_{i=1}^{M} \mathbf{1}[\text{maj}(x_i) = a_i^*]. \tag{30}$$

This metric ranges from 0 to 1, where 1 indicates perfect pseudo-label generation.

Reward Accuracy Quantifies sample-level agreement between pseudo-rewards and oracle rewards:

$$\text{Reward Accuracy} = \frac{1}{M \cdot N} \sum_{i=1}^{M} \sum_{j=1}^{N} \mathbf{1}[r_{\text{mv}}(y_{i,j}) = r_{\text{gt}}(y_{i,j})]. \tag{31}$$

This captures "lucky hits" where individual rewards are correct even when the majority vote is wrong. For example, if the majority vote is incorrect but a minority response is correct, that response still receives the appropriate (zero) pseudo-reward.

Ground Truth Reward Average oracle reward across all generated responses:

$$\text{Ground Truth Reward} = \frac{1}{M \cdot N} \sum_{i=1}^{M} \sum_{j=1}^{N} r_{\text{gt}}(y_{i,j}). \tag{32}$$

This represents the true quality of generated responses and serves as the supervised baseline.

Majority Voting Reward Average pseudo-reward from majority voting:

$$\text{Majority Voting Reward} = \frac{1}{M \cdot N} \sum_{i=1}^{M} \sum_{j=1}^{N} r_{\text{mv}}(y_{i,j}). \tag{33}$$

The divergence between this metric and Ground Truth Reward indicates reward hacking: when the model learns to maximize pseudo-rewards at the expense of actual correctness.

B.2.3 CERTAINTY-BASED METRICS

Label Accuracy For certainty-based methods, we identify the highest-confidence response per prompt and measure its accuracy:

$$\text{Label Accuracy} = \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}[\text{ans}(y_{i,j_i^*}) = a_i^*], \tag{34}$$

where $j_i^* = \arg\max_{j\in\{1,\ldots,N\}} r_{\text{cert}}(y_{i,j})$ is the index of the highest-confidence response for prompt x_i.

Point-Biserial Correlation Measures the correlation between continuous certainty scores and binary correctness:

$$\rho_{pb} = \frac{\bar{r}_1 - \bar{r}_0}{s_r} \cdot \sqrt{\frac{n_1 n_0}{n^2}}, \tag{35}$$

where:

- $n = M \cdot N$ is the total number of responses
- $n_1 = \sum_{i,j} r_{\text{gt}}(y_{i,j})$ is the number of correct responses
- $n_0 = n - n_1$ is the number of incorrect responses
- $\bar{r}_1 = \frac{1}{n_1}\sum_{i,j:r_{\text{gt}}(y_{i,j})=1} r_{\text{cert}}(y_{i,j})$ is the mean certainty for correct responses
- $\bar{r}_0 = \frac{1}{n_0}\sum_{i,j:r_{\text{gt}}(y_{i,j})=0} r_{\text{cert}}(y_{i,j})$ is the mean certainty for incorrect responses
- $s_r = \sqrt{\frac{1}{n-1}\sum_{i,j}(r_{\text{cert}}(y_{i,j}) - \bar{r})^2}$ is the standard deviation of all certainty scores
- $\bar{r} = \frac{1}{n}\sum_{i,j} r_{\text{cert}}(y_{i,j})$ is the mean of all certainty scores

The correlation $\rho_{pb} \in [-1, 1]$ quantifies the relationship between confidence and correctness. Positive values indicate that higher certainty correlates with correctness (desired behavior), while values near zero suggest certainty is uninformative, and negative values indicate miscalibration.

B.3 HYPERPARAMETER TUNING

Setup. We study four hyperparameters, including training temperature, mini-batch size, KL divergence regularization, and rollout count, that directly influence convergence dynamics in our theoretical framework. We vary one parameter at a time while keeping others fixed at baseline values (see Appendix B.1).

B.3.1 MAJORITY VOTING

Training Temperature. Temperature directly controls exploration during rollout generation and affects the quality of pseudo-labels via voting diversity. From our convergence analysis in Theorem 1, lower temperature reduces the effective β in the KL regularization term, accelerating convergence. As shown in Figure 8, low $T \in \{0.6, 0.8\}$ quickly sharpens logits, causing unstable *Label Accuracy*, consistent with premature convergence to an early majority that may be incorrect. Higher temperature ($T = 1.2$) maintains stability longer by preserving exploration, but the increased noise reduces peak performance. We find $T = 1.0$ provides optimal balance, showing steady early gains with delayed degradation.

Mini-batch Size. This parameter controls the on-policy nature of updates, directly affecting the validity of our optimal policy assumptions. Our theoretical derivation in Equation (3) assumes rewards are computed under the current policy π_θ. Small mini-batches violate this assumption through reward staleness: pseudo-rewards computed under π_θ become misaligned when applied to samples from $\pi_{\theta_{\text{old}}}$. As shown in Figure 9, mini-batch size 1 drives rapid collapse within 20 steps, while pure on-policy training (mini-batch $= 64$, matching global batch size) provides maximum stability. The intermediate

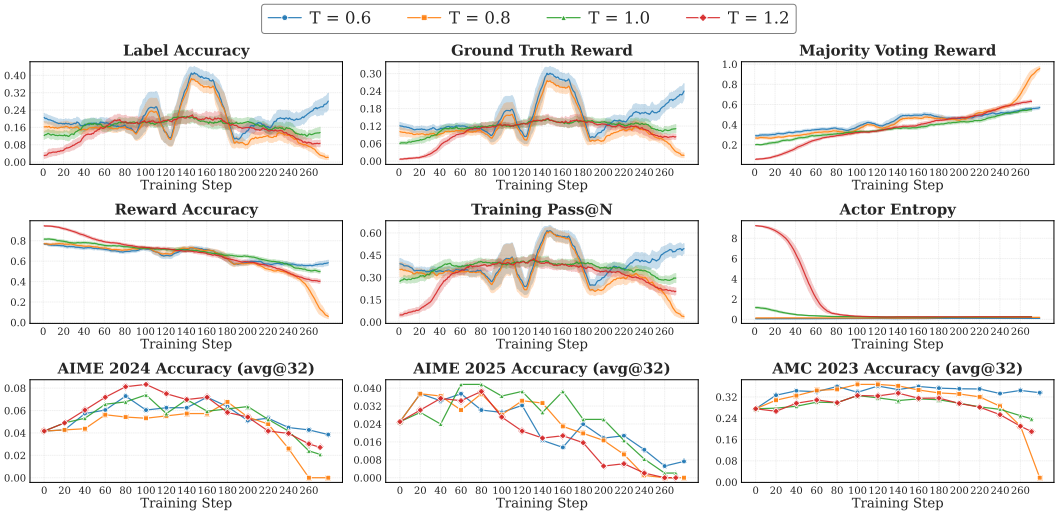

Figure 8: Effect of training temperature for Majority Voting method.

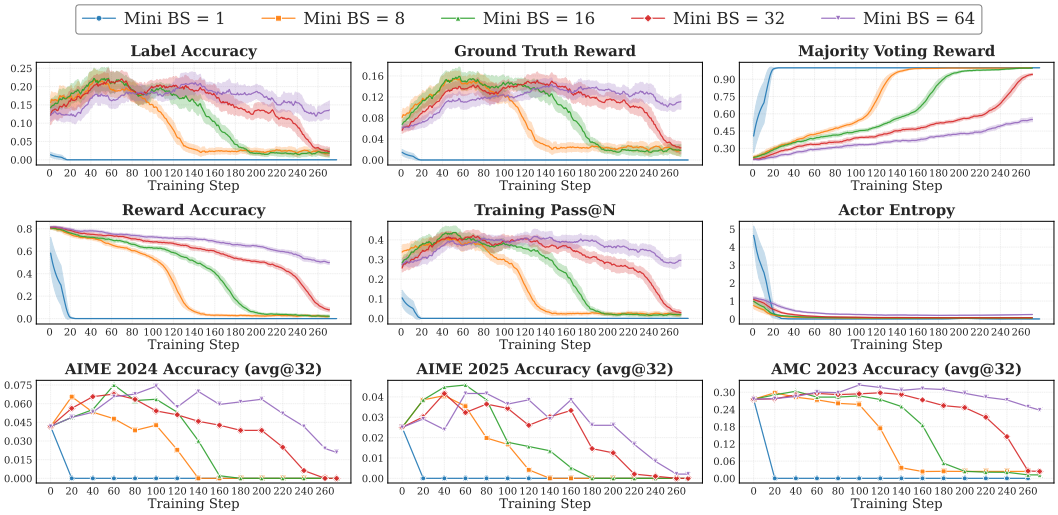

Figure 9: Effect of mini-batch size for Majority Voting method.

sizes (16–32) show gradual improvement, confirming that maintaining policy-reward alignment is crucial for stable convergence.

KL Regularization. Our theoretical analysis suggests that KL regularization should slow convergence by increasing the effective β parameter in **??**. However, empirical results in Figure 10 show that adding KL regularization ($\beta = 0.005$) yields only marginal benefits: small early gains but increased training variance and minimal delay in collapse (~ 40 steps). This discrepancy arises because intrinsic rewards create competing optimization pressures, where the intrinsic signal drives sharpening while KL pulls toward the reference policy. Rather than smoothly balancing these forces, the optimization oscillates between them, increasing variance without providing durable stability. The marginal gains do not justify the additional memory overhead and training instability.

Number of Rollouts. The rollout count N affects both vote reliability and signal strength. While more rollouts improve statistical reliability of the majority vote, they also amplify the majority signal strength. From Equation (3), each update amplifies majority probability by factor $e^{1/\beta}$. With more rollouts, this majority becomes more confident, accelerating convergence. Figure 11 shows this effect: $N = 32$ collapses within 180 steps, $N = 16$ within 220 steps, while $N \leq 8$ remains stable over the full epoch. Although $N = 4$ shows competitive performance in some metrics, we recommend $N = 8$

Figure 10: Effect of KL divergence regularization for Majority Voting method.

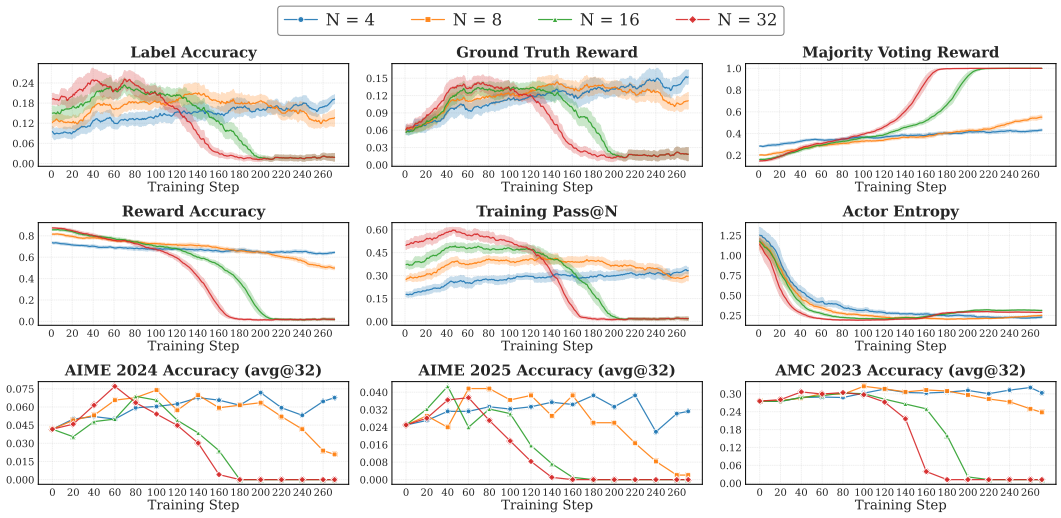

Figure 11: Effect of rollout number for Majority Voting method.

as it provides better statistical reliability for the voting mechanism while maintaining reasonable convergence control. The slight performance difference suggests that for this specific experimental setup, the trade-off between reliability and stability favors slightly smaller N, but $N = 8$ offers more robust behavior across diverse problem types.

B.3.2 Certainty-Based Methods

Training Temperature. Temperature effects on certainty-based methods reveal distinct behavioral patterns compared to ensemble-based approaches. Unlike Majority Voting, certainty-based methods generally benefit from higher exploration temperatures, with notable method-specific variations in optimal configurations and convergence characteristics.

Results in Figures 13 to 15 demonstrate that higher temperature ($T = 1.2$) significantly delays model collapse across Token-Level Entropy, Trajectory-Level Entropy, and Probability methods. Higher temperatures initially maintain elevated **Actor Entropy**, facilitating extended exploration phases with gradual improvements across validation benchmarks. Importantly, these methods also exhibit relatively higher **Point-Biserial Correlation** values at $T = 1.2$, indicating stronger alignment

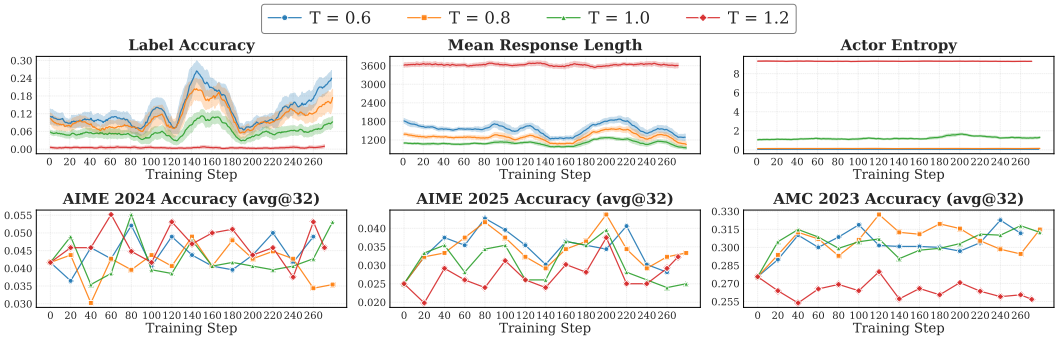

Figure 12: Effect of training temperature on Self-Certainty performance. Note that Point-Biserial Correlation is replaced with Mean Response Length due to Self-Certainty's scoring characteristics.

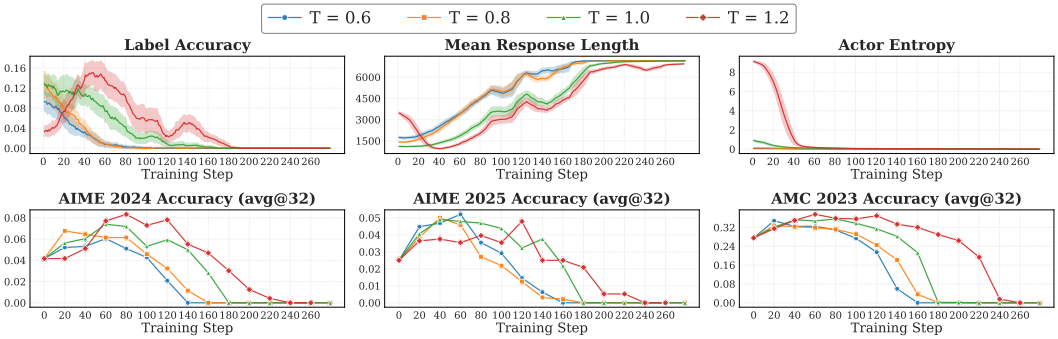

Figure 13: Effect of training temperature on Token-Level Entropy performance.

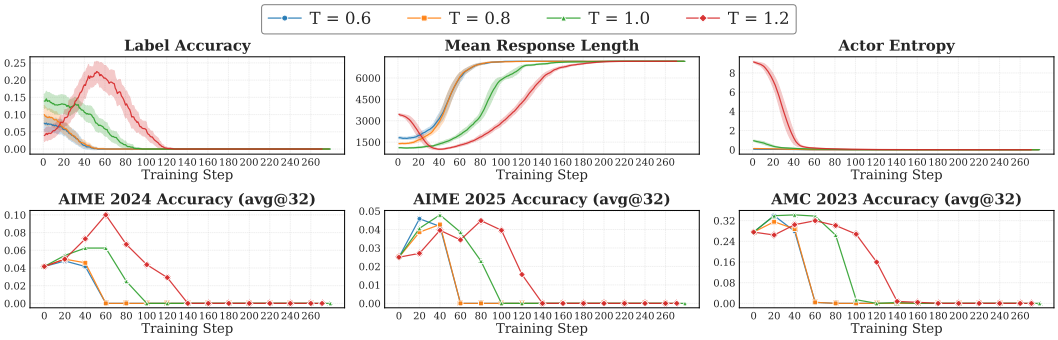

Figure 14: Effect of training temperature on Trajectory-Level Entropy performance.

between certainty estimates and actual correctness—a crucial property for effective uncertainty-based reward assignment.

However, Figure 12 reveals that Self-Certainty exhibits contrasting behavior. Higher temperature ($T = 1.2$) leads to excessive exploration without convergence, maintaining persistently high **Actor Entropy** while achieving lower validation scores and **Label Accuracy**. The moderate temperature $T = 1.0$ provides more stable and superior performance for Self-Certainty. This divergence suggests that while different certainty-based methods converge toward similar sharp distributions, they exhibit distinct convergence rates requiring method-specific temperature tuning. Among all certainty-based approaches, Token-Level and Trajectory-Level Entropy methods demonstrate the greatest benefits from higher temperature exploration, likely due to their more robust entropy-based uncertainty estimation mechanisms.

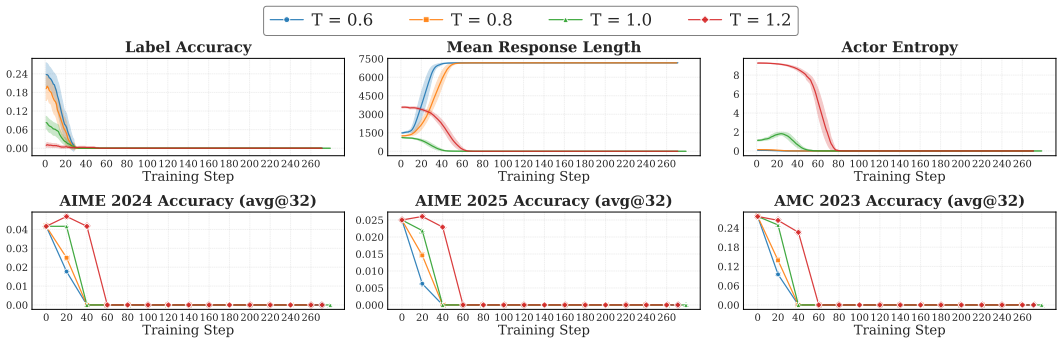

Figure 15: Effect of training temperature on Probability-based certainty performance.

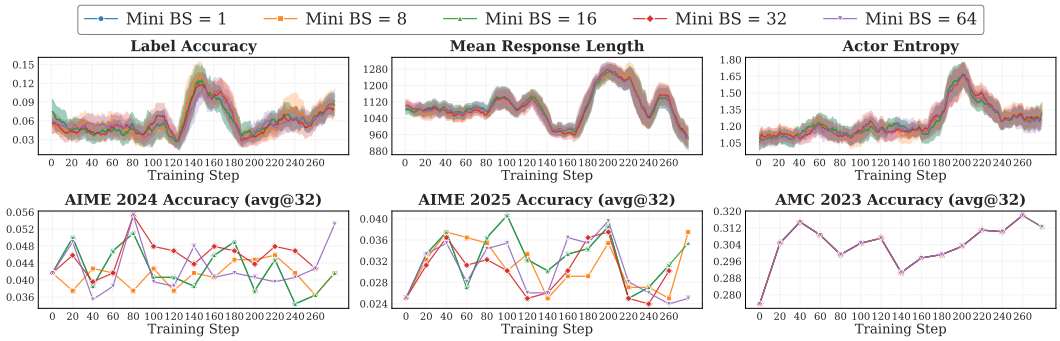

Figure 16: Effect of mini-batch size on Self-Certainty performance.

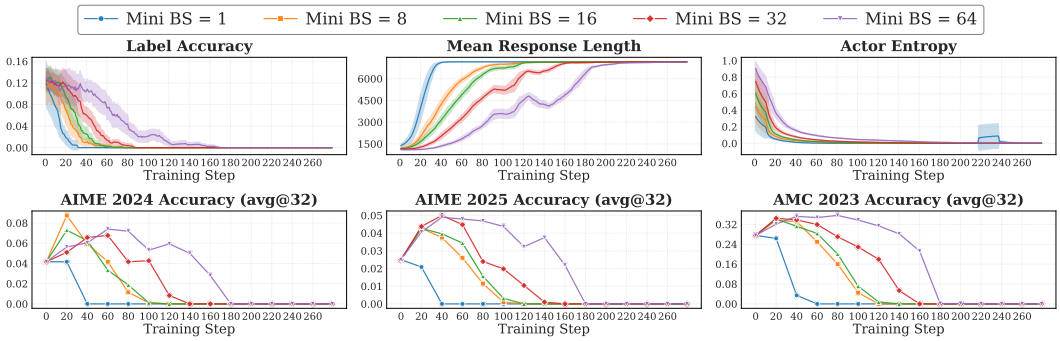

Figure 17: Effect of mini-batch size on Token-Level Entropy performance.

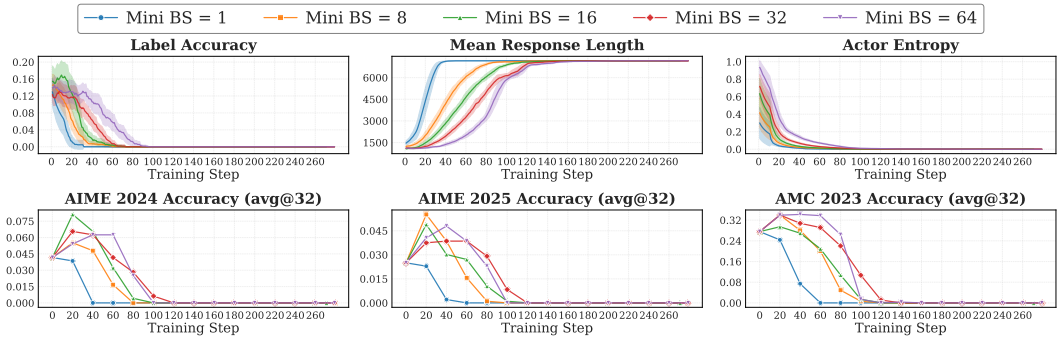

Figure 18: Effect of mini-batch size on Trajectory-Level Entropy performance.

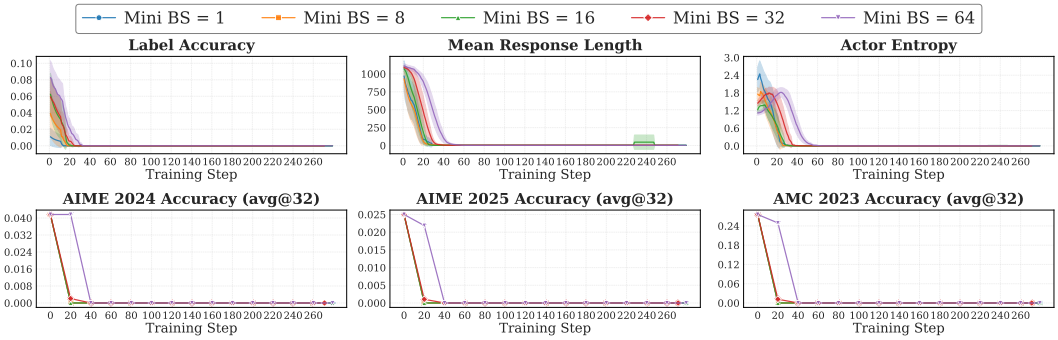

Figure 19: Effect of mini-batch size on Probability performance.

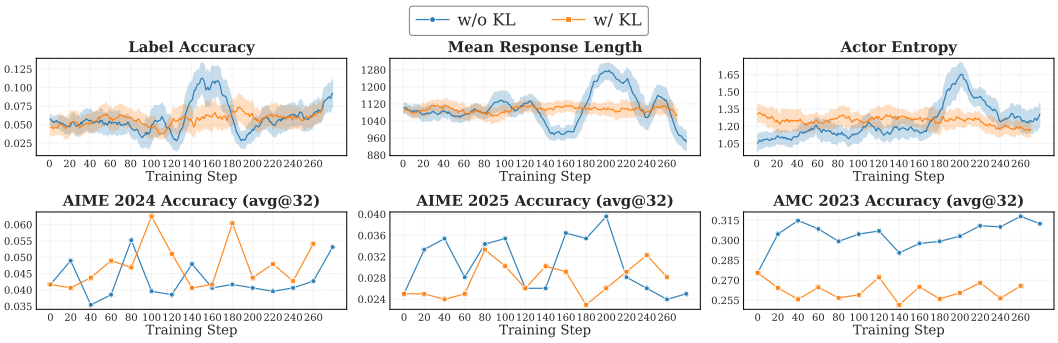

Figure 20: Effect of KL divergence regularization on Self-Certainty performance.

Mini-Batch Size. Mini-batch size effects on certainty-based methods largely parallel those observed in Majority Voting, confirming that on-policy ratio critically affects training stability regardless of the underlying reward computation mechanism. However, method-specific sensitivities reveal important distinctions in robustness to off-policy learning.

Figures 17 to 19 consistently demonstrate that larger mini-batch sizes prevent premature model collapse across Token-Level Entropy, Trajectory-Level Entropy, and Probability methods. This pattern mirrors Majority Voting behavior, where pure on-policy training (mini-batch size = 64) maintains optimal coupling between samples and their corresponding certainty-based rewards. The underlying mechanism remains consistent: certainty estimates computed from current policy states become unreliable when applied to samples generated from earlier policy iterations.

Notably, Self-Certainty exhibits exceptional robustness to mini-batch size variations, as shown in Figure 16. This method demonstrates minimal sensitivity to on-policy ratio changes, suggesting that KL divergence-based certainty computation may be inherently more stable across different temporal policy alignments. This robustness likely stems from Self-Certainty's reliance on logit distribution comparisons rather than explicit probability estimates, making it less susceptible to the temporal inconsistencies that destabilize other certainty-based approaches. Among the certainty-based methods, Self-Certainty thus offers superior stability but at the cost of lower overall performance improvements.

KL Divergence Regularization. KL regularization effects on certainty-based methods mirror the limited impact observed in Majority Voting, confirming that this regularization technique fails to address the fundamental instabilities inherent in training. However, subtle differences in method responses provide insights into the interaction between regularization and different uncertainty estimation approaches.

Results across all certainty-based methods (Figures 20 to 23) show minimal impact on both training dynamics and downstream performance. KL regularization neither prevents eventual model collapse (except for Self-Certainty) nor significantly improves validation scores, consistent with our findings for Majority Voting. The underlying issue persists: regularization techniques designed for fixed

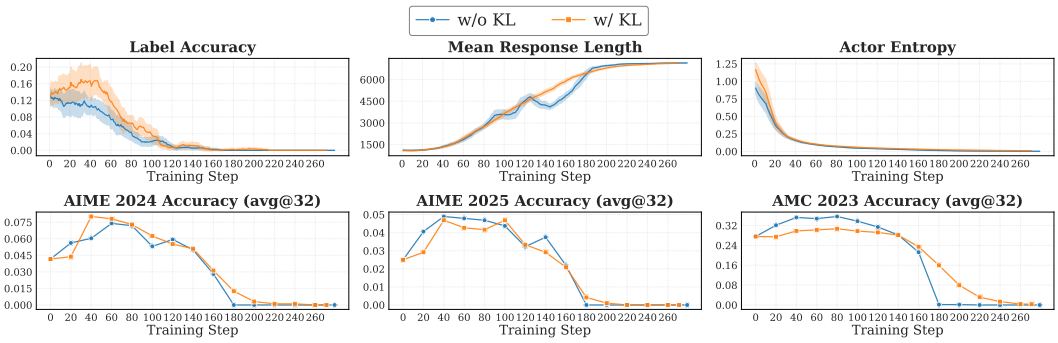

Figure 21: Effect of KL divergence regularization on Token-Level Entropy performance.

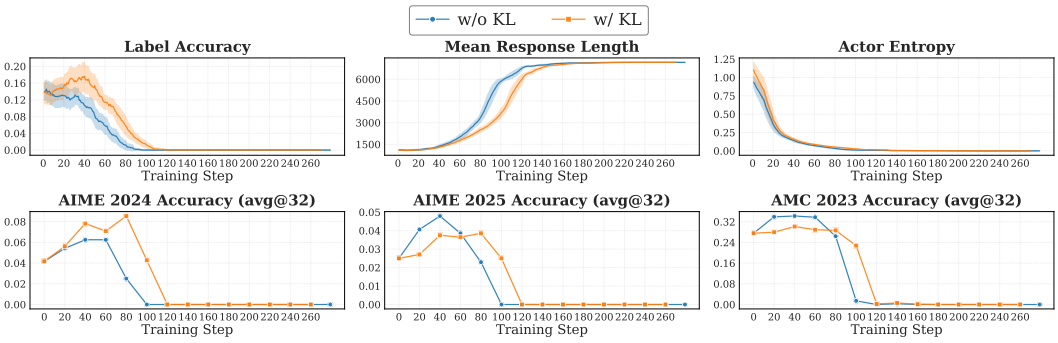

Figure 22: Effect of KL divergence regularization on Trajectory-Level Entropy performance.

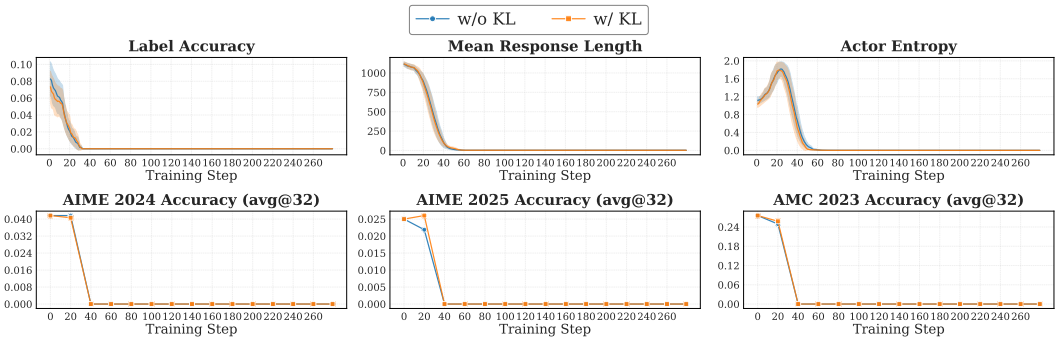

Figure 23: Effect of KL divergence regularization on Probability performance.

reward signals cannot effectively stabilize systems where rewards themselves evolve with policy changes.

Interestingly, Token-Level and Trajectory-Level Entropy methods exhibit slightly more pronounced benefits from KL regularization, as evidenced by modest improvements in **Label Accuracy** curves. While these improvements remain insufficient to prevent collapse, they suggest that entropy-based certainty estimation may have marginally better compatibility with KL-based stabilization approaches. This observation aligns with the superior temperature robustness of these methods, indicating that entropy-based uncertainty measures may be inherently more amenable to regularization techniques than probability-based or KL-based certainty estimates.

Number of Rollouts. Rollout count effects reveal consistent patterns across most certainty-based methods, with one notable exception that highlights fundamental differences in underlying reward computation mechanisms. These findings provide crucial insights into the sample size requirements for reliable uncertainty estimation.

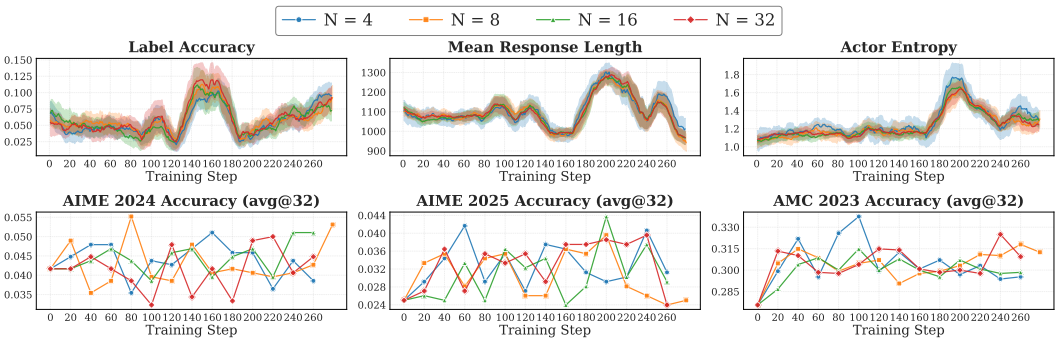

Figure 24: Effect of rollout number on Self-Certainty performance.

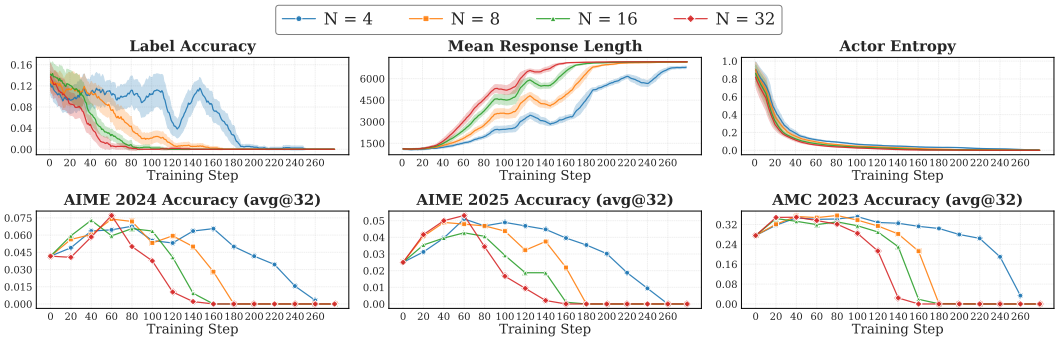

Figure 25: Effect of rollout number on Token-Level Entropy performance.

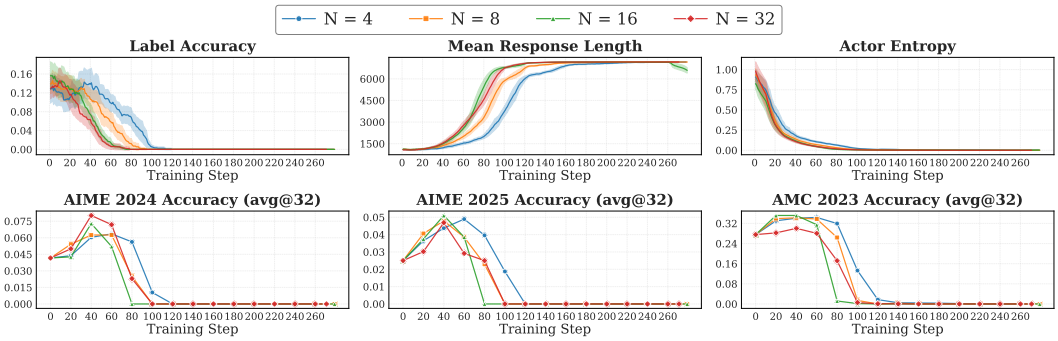

Figure 26: Effect of rollout number on Trajectory-Level Entropy performance.

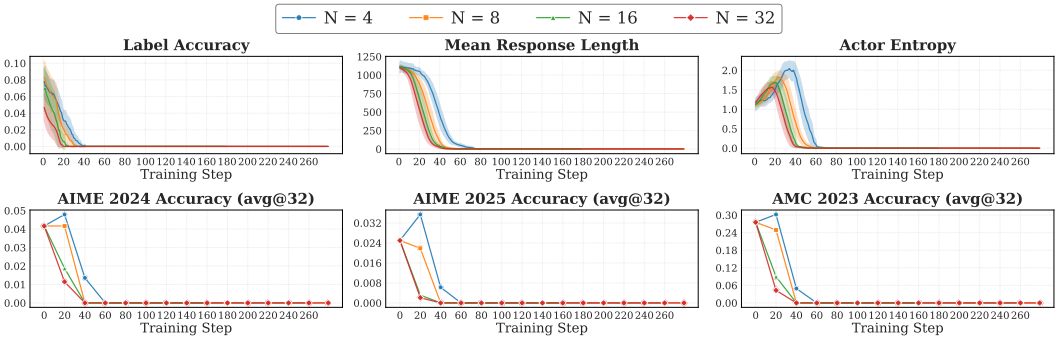

Figure 27: Effect of rollout number on Probability performance.

Table 8: Model configurations for backbone experiments. Models are categorized by family, training stage, and size.

Family	Model	Abbrev.	Stage	Size
Qwen	Qwen2.5-1.5B	Q2.5-1.5B	Base	1.5B
	Qwen2.5-Math-1.5B	Q2.5-Math-1.5B	Math Base	1.5B
	DeepSeek-R1-Distill-Qwen-1.5B	DS-R1-1.5B	SFT	1.5B
	Qwen2.5-1.5B-Instruct	Q2.5-1.5B-Inst	Instruct	1.5B
	Qwen3-1.7B-Base	Q3-1.7B	Base	1.7B
	Qwen3-4B-Base	Q3-4B	Base	4B
Llama	Meta-Llama-3.1-8B	L3.1-8B	Base	8B
	OctoThinker-8B-Short-Base	Octo-8B	Math Base	8B
	OctoThinker-3B-Short-Base	Octo-3B	Math Base	3B
	Llama-3.1-Tulu-3-8B-SFT	L3.1-8B-Tulu-SFT	SFT	8B
	Meta-Llama-3.1-8B-Instruct	L3.1-8B-Inst	Instruct	8B

Figures 25 to 27 demonstrate behavior parallel to Majority Voting: larger rollout counts ($N \geq 16$) accelerate model convergence and premature collapse, as evidenced by rapid degradation in validation benchmarks and **Label Accuracy**. This pattern suggests that the self-reinforcing dynamics observed in ensemble voting also manifest in certainty-based reward assignment, where higher sample sizes amplify confidence in potentially incorrect assessments, leading to faster convergence toward suboptimal solutions.

However, Self-Certainty exhibits markedly different behavior, as shown in Figure 24. This method demonstrates remarkable stability across all rollout configurations, maintaining consistent performance without collapse or significant improvement. This unique characteristic stems from Self-Certainty's reliance on KL divergence between uniform and logit distribution. This fundamental difference in reward computation makes Self-Certainty inherently more robust to sample size variations, though at the cost of limited performance improvements throughout training.

B.4 Impact of Backbone Model

We investigate how backbone models influence training stability and performance across three key dimensions: training stage, model size, and architectural generation. Our analysis employs 11 models from Qwen and Llama families (detailed configurations in Table 8), selected to provide systematic coverage of these factors. This selection is motivated by recent findings showing distinct architectural behaviors (Gandhi et al., 2025) and potential data contamination concerns (Wu et al., 2025b), making cross-architecture comparison essential. All models are trained on DAPO-17k using optimal hyperparameters from Appendix B.3 with Majority Voting as the representative intrinsic reward.

B.4.1 Horizontal Analysis: Training Stage Impact

Training stage progression reveals distinct stability patterns between architectures. For the **Qwen family** (Figure 28), math-specialized and SFT models demonstrate superior stability, maintaining **Majority Voting Reward** within 0.3-0.6 while base and instruct variants reach saturation (1.0) by step 180. Math specialization and strong supervised fine-tuning (DS-R1-1.5B) create robust foundations for optimization compared to raw base models or non-math aligned instruct variants.

The **Llama family** exhibits contrasting behavior: all variants eventually succumb to reward hacking with different collapse timing, where base models fail earliest (step 40), followed by math-specialized, SFT, then instruct versions (detailed analysis in Figure 29). This architectural difference highlights Qwen's fundamental advantage in providing genuine stability.

B.4.2 Vertical Analysis: Scale and Generation Effects

Model size analysis (Figure 30) reveals counterintuitive scaling effects: smaller models consistently outperform larger variants. Q3-1.7B maintains stability significantly longer than Q3-4B, while Octo-3B outlasts Octo-8B by about 40 steps. This suggests larger models' increased capacity

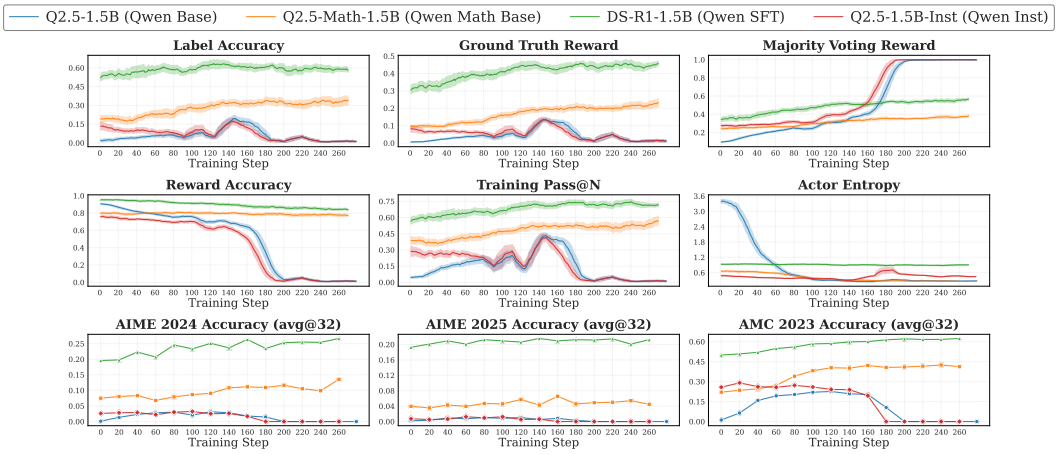

Figure 28: Training dynamics across different training stages in Qwen family models.

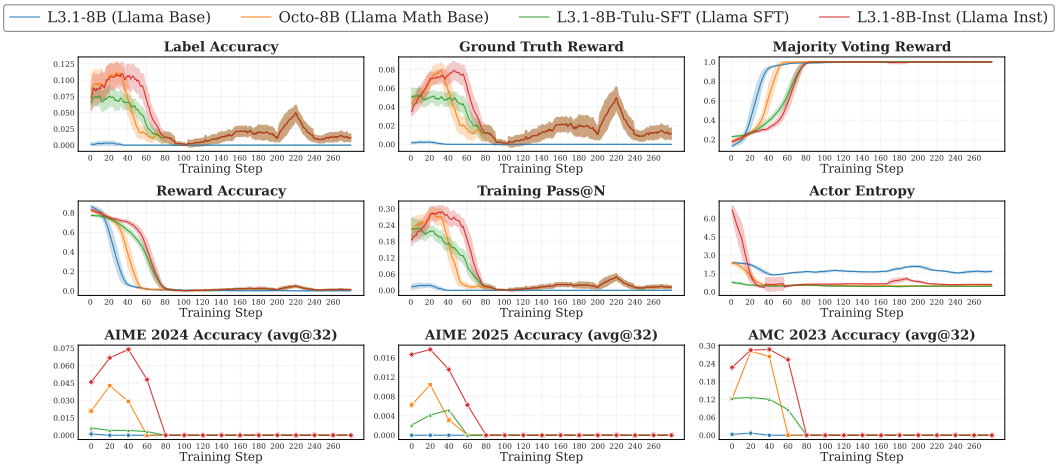

Figure 29: Training dynamics across different training stages in Llama family models.

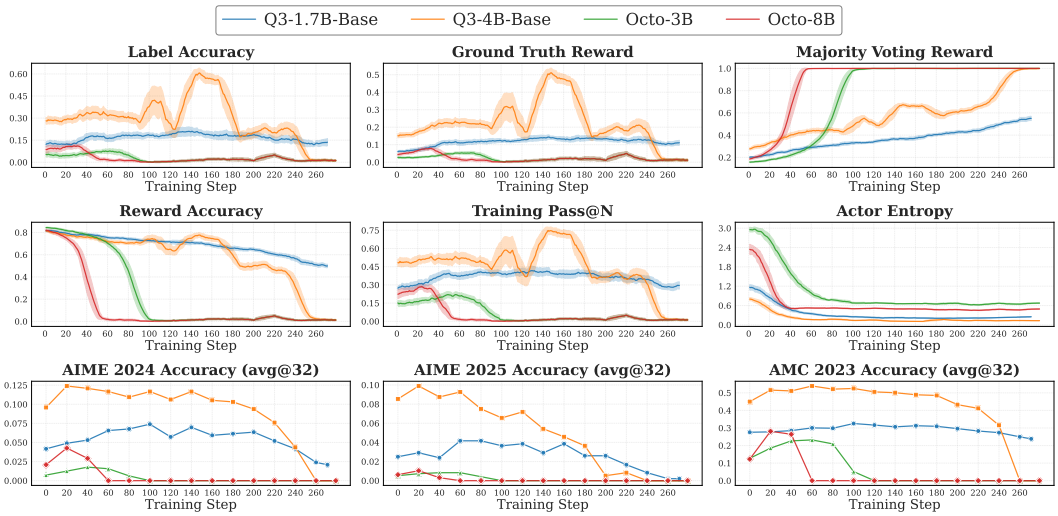

Figure 30: Effect of model size on stability across both Qwen and Llama families.

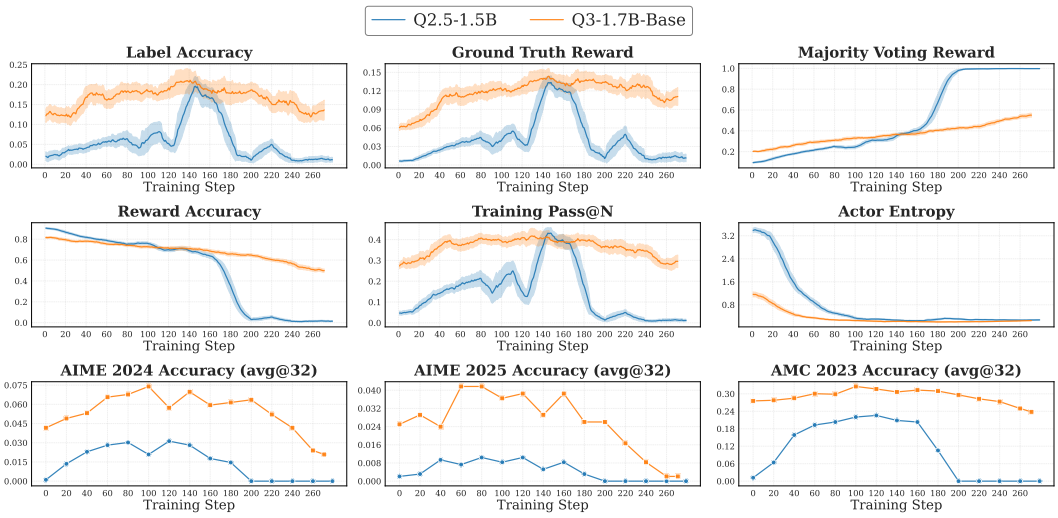

Figure 31: Comparison of Qwen 2.5 and Qwen 3 generations across comprehensive training metrics. Results reveal improved stability in the newer generation, with Qwen3 models demonstrating more gradual and controlled training dynamics compared to Qwen2.5 counterparts.

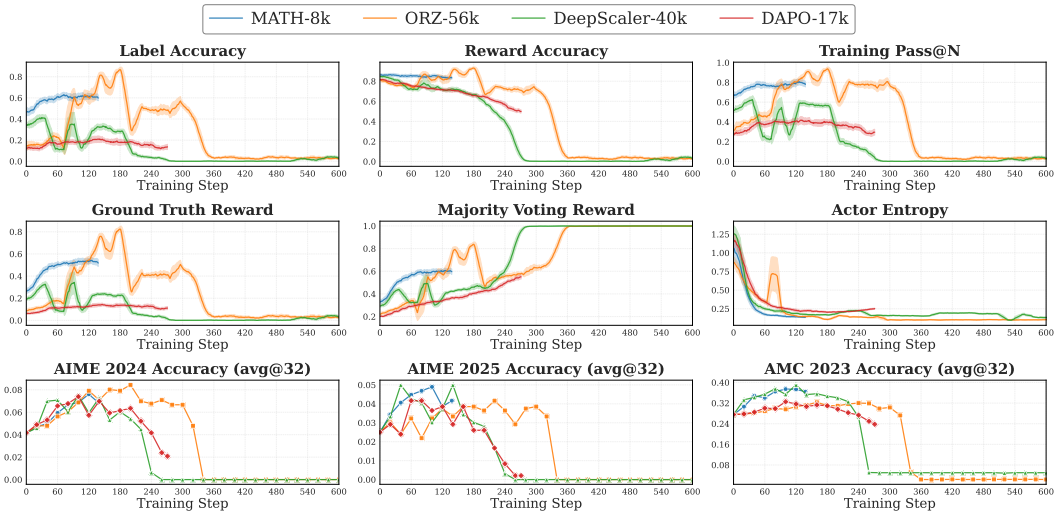

Figure 32: Comparison of different training data sources.

amplifies sensitivity to noisy pseudo-rewards, accelerating convergence toward degenerate solutions and challenging conventional scaling assumptions.

Architectural generation comparison shows clear improvements in newer versions. Qwen3 models exhibit superior stability compared to Qwen2.5 counterparts, with Q3-1.7B-Base demonstrating more controlled **Majority Voting Reward** progression (comprehensive comparison in Figure 31). These improvements likely stem from better-calibrated uncertainty estimates and enhanced representation learning supporting more reliable pseudo-reward computation.

B.5 Impact of Training Dataset

Setup. We investigate how different training dataset influence training stability and performance, focusing on math reasoning, utilizing MATH-8k (Hendrycks et al., 2021), DeepScaleR-40k (Luo et al., 2025), DAPO-17k (Yu et al., 2025) and ORZ-56k (Hu et al., 2025), all settings are trained

on Qwen3-1.7B-Base with 1 epoch using optimal hyperparameters from Appendix B.3, and also evaluated on three validation benchmarks.

Results. We can see from Figure 32, much larger datasets (DeepScaler-40k and ORZ-56k) exhibits clear reward hacking trend, while smaller datasets settings are on its steady or rise stage, indicating that current intrinsic methods may see its short-sighted incremental improvements at the early stage, while extending it much larger training corpora, it inevitably encounter the reward hacking.

C THE USE OF LARGE LANGUAGE MODELS

We use large language models to refine our writing. In particular, we use ChatGPT (GPT-5 Thinking) to revise the manuscript. The prompt provided to the model is: "I am writing an academic paper in English. Please polish the following draft so that it adheres to the conventions of academic writing."