
CoDaPO: Confidence and Difficulty-Adaptive Policy Optimization for Post-Training Language Models

Zhanke Zhou^{1 2 *} Xiangyu Lu^{1 *} Chentao Cao¹ Brando Miranda² Tongliang Liu³ Bo Han¹ Sanmi Koyejo²

Abstract

Large language models (LLMs) increasingly rely on reinforcement learning (RL) post-training to improve step-by-step reasoning. Therein, Group Relative Policy Optimization (GRPO) emerges as a prevailing approach that avoids the need for fully supervised traces. However, GRPO can struggle with high-difficulty tasks, overfit to easy problems, and suffer from sensitivity to reward design. To diagnose these weaknesses, we introduce a general analysis framework that maps training trajectories onto an advantage-confidence plane, revealing three critical phenomena: (1) *advantage contraction*: reward-normalized advantages collapse as accuracy improves; (2) *confidence saturation*: policies become overconfident even on incorrect outputs; and (3) *hierarchical convergence*: easy problems are quickly mastered while harder ones lag. Based on these insights, we propose CoDaPO (Confidence- and Difficulty-Adaptive Policy Optimization), an RL algorithm that adopts correctness-based reward and advantage reweighting *w.r.t.* confidence and difficulty. Experiments on several benchmarks demonstrate that CoDaPO achieves higher reasoning accuracy and better generalization than existing RL approaches.

1. Introduction

Large language models (LLMs) have transformed problem-solving across diverse real-world applications, including tool use (Schick et al., 2023), retrieval-augmented generation (Lewis et al., 2020), and autonomous agents (Yao et al., 2023b). Central to this progress is their capacity for explicit, step-by-step reasoning. Early studies such

as Chain-of-Thought prompting (Wei et al., 2022) and Tree-of-Thoughts (Yao et al., 2023a) showed that carefully designed prompts can elicit multi-step reasoning without changing the underlying model weights. Recent work extends beyond prompt engineering by *post-training*, which continues to train models after the pre-training phase, to create even more capable and general-purpose machine reasoners (Kumar et al., 2025; Li et al., 2025). Post-trained models like OpenAI o1 (Jaech et al., 2024) and DeepSeek R1 (Guo et al., 2025) achieve higher levels of intelligence, which can be further amplified through test-time scaling that allocates more compute at inference to extend the exploration (Snell et al., 2024; Muennighoff et al., 2025).

Notably, reinforcement-learning (RL) post-training methods, most prominently Proximal Policy Optimization (PPO) (Schulman et al., 2017), Group Relative Policy Optimization (GRPO) (Shao et al., 2024), and their variants, have gained momentum because they markedly enhance a model’s reasoning ability without requiring to curating “gold” step-by-step solutions. Instead of *teaching* the model every intermediate step, these RL algorithms *incentivize* the autonomous discovery of high-quality reasoning paths by supplying feedback signals derived from rule-based (Shao et al., 2024) or model-based (Liu et al., 2025c) reward estimators. This incentive-driven paradigm departs sharply from traditional supervised approaches that simply mimic human demonstrations and has already shown empirical gains in mathematics (Shao et al., 2024), code generation (Luo et al., 2025), and formal logic (Xie et al., 2025).

Despite encouraging progress, RL-based reasoning is still in its infancy, and several weaknesses of the dominant GRPO framework have become apparent. Sensitivity to reward design is the first concern: the popular format reward can actually degrade performance when mis-specified (Stiennon et al., 2020; Zeng et al., 2025a;b). On difficult tasks, such as MATH problems at levels 3-5, the policy seldom produces trajectories that earn positive feedback. Unless the difficulty of the training tasks is tightly matched to the model’s current exploration capacity, learning often stalls or even gradually collapses over time. Crafting robust reward curricula or adaptive scaling schemes that keep the signal informative across diverse problem regimes remains an open challenge.

*Equal contribution ¹TMLR Group, Department of Computer Science, Hong Kong Baptist University ²Stanford University ³Sydney AI Center, The University of Sydney. Correspondence to: Bo Han <bhanml@comp.hkbu.edu.hk>.

The second AI for MATH Workshop at the 42nd International Conference on Machine Learning, Vancouver, Canada. Copyright 2025 by the author(s).

A second issue is training stability and cost. GRPO optimizes group-based relative advantages, a formulation that introduces extra variance into the gradient estimate. Combined with the large batch sizes and long roll-outs needed to stabilize updates, this makes large-scale training computationally expensive (Schulman et al., 2017). Improving the robustness of the optimization objective and reducing its computational cost are thus key priorities for the next generation of RL post-training methods for LLM reasoning.

To pinpoint the root causes of GRPO’s shortcomings, we carry out a systematic analysis of RL post-training through the lens of three intertwined distributions: input questions, model outputs, and rewards (advantages). We closely monitor GRPO’s training dynamics along three axes—problem difficulty, model confidence, and output advantage—using only quantities that can be directly computed from model logits and reward signals, so our methodology generalizes robustly across different algorithms, datasets, and models.

Concretely, for each difficulty level in the data, we map the responses generated by the model to an advantage-confidence (AC) plane. Here, advantage is the reward-normalized advantage returned by the RL update rule, while confidence is the probability mass of the policy assigned to the chosen response. This projection yields an intuitive, two-dimensional snapshot of how learning unfolds over the training process. We apply the above framework to post-training Qwen2.5-Math-1.5B (Yang et al., 2024b) with GRPO (Shao et al., 2024) on MATH (Hendrycks et al., 2021) and evaluate on MATH500 (Lightman et al., 2024). The resulting AC planes reveal several insightful observations:

- **Advantage Contraction:** As GRPO training improves the model’s accuracy on the training set, the advantage distributions gradually contract around zero, which makes it harder to differentiate outputs within the response group.
- **Confidence Saturation:** As GRPO training progresses, the model’s confidence increasingly clusters at a high value, improving accuracy but also leading to overconfidence in wrong outputs, especially on difficult problems.
- **Hierarchical Convergence:** During GRPO training, easy questions move towards higher confidence and positive advantage, while harder ones shift positively and slowly but continue to retain a significant proportion of errors.

Building on these insights, we propose CoDaPO, a confidence- and difficulty-adaptive policy optimization method for post-training LLMs on reasoning tasks. Unlike prior approaches, CoDaPO focuses purely on factual correctness when rewarding outputs, eliminating the noise introduced by format and length rewards. It dynamically reweights advantages using a combination of output confidence and problem difficulty, ensuring that high-confidence,

complex outputs receive proportionately greater reinforcement, while simpler, overconfident outputs are appropriately down-weighted. To address the potential length bias, CoDaPO normalizes advantages globally at the sequence level, avoiding the token-wise scaling that can overly penalize longer responses. Additionally, it completely discards KL regularization, which often restricts exploratory learning by enforcing close alignment with a frozen reference model. This design frees the policy to discover more diverse reasoning trajectories, ultimately improving the model’s ability to handle deliberate reasoning across a wide range of tasks.

Finally, we empirically validate the proposed algorithm on seven widely used reasoning-task benchmarks. The experimental results indicate that CoDaPO achieves higher reasoning accuracy and better generalization across various benchmarks compared to existing algorithms. This improvement is particularly evident for base models that have not undergone instruction-tuned training. For instance, CoDaPO boosts the accuracy of the Qwen2.5-Math-1.5B model from 31.20% to 71.68% on the in-domain MATH500 benchmark, and from 18.25% to 32.80% on the out-of-domain Olympiad Bench. These findings underscore CoDaPO’s significant potential to enhance the reasoning capabilities of LLMs, resulting in substantial gains in accuracy across both in-domain and out-of-domain evaluations. In summary, the main contributions of the work are three-fold as follows:

- **In-depth analysis of RL post-training.** We introduce a general diagnostic framework with AC planes that tracks the joint distributions of task difficulty, model confidence, and advantage, yielding actionable insights into why existing methods like GRPO often stall or regress (Sec. 3).
- **CoDaPO: a confidence- and difficulty-adaptive RL algorithm.** Building on the insights in Sec. 3, we design CoDaPO, a new post-training algorithm that tailors rewards, exploration, and normalization to reasoning tasks, significantly improving stability and performance (Sec. 4).
- **Comprehensive empirical verification.** Extensive experiments on MATH, AIME 2024, and related benchmarks demonstrate that CoDaPO consistently outperforms other post-training approaches, confirming the practical value of our analysis and algorithmic contributions (Sec. 5).

2. Preliminaries: Post-training Algorithms for Language Models

Notations. A training dataset \mathcal{D} contains question-answer pairs (q, a) . Given a question q , a language model $f_{\theta}(\cdot)$ parameterized by θ generates an output o as $o \sim f_{\theta}(\cdot|q)$. This output includes the model’s step-by-step reasoning process and the predicted answer to the question. Sorted by the frequency of updating in the optimization process, the

model can be classified into three types: 1) policy model f_θ (latest), 2) old policy model f_{old} (older), and 3) reference model f_{ref} (oldest). Next, we introduce representative post-training algorithms, including SFT, PPO, and GRPO.

Supervised Fine-tuning (SFT) finetunes the policy model to predict the next token on data that is more relevant to the downstream task. The objective of SFT is to maximize the token-wise log probability of dataset-collected outputs \mathcal{O} , which are treated as the ground truth for training. Namely,

$$\mathcal{J}_{\text{SFT}}(f_\theta) \triangleq \mathbb{E}_{(q,a) \sim \mathcal{D}, o \sim \mathcal{O}(q)} \left(\frac{1}{|o|} \sum_{t=1}^{|o|} \log f_\theta(o_t | q, o_{<t}) \right). \quad (1)$$

Although simple in optimization, SFT has several drawbacks. SFT focuses on exploiting (memorizing, to some extent) the dataset-collected outputs, resulting in limited generalization power, especially in the out-of-distribution scenarios (Chu et al., 2025a). Besides, collecting and annotating the output data can be expensive and often requires domain-specific knowledge in solving particular questions.

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is an actor-critic RL algorithm that is widely used in the RL fine-tuning stage of LLMs. Simplifying the TRPO (Schulman et al., 2015a), PPO maximizes the advantage A_t of the model-generated output o without the need to collect ground truth outputs. Here, the advantage A_t is computed by the Generalized Advantage Estimation (GAE) (Schulman et al., 2015b), taking 1) the output value estimated by a trainable value model and 2) the KL penalty between f_θ and f_{ref} . PPO maximizes the objective:

$$\mathcal{J}_{\text{PPO}}(f_\theta) \triangleq \mathbb{E}_{(q,a) \sim \mathcal{D}, o \sim f_{\text{old}}(\cdot|q)} \left[\frac{1}{|o|} \sum_{t=1}^{|o|} \min \left(\frac{f_\theta(o_t | q, o_{<t})}{f_{\text{old}}(o_t | q, o_{<t})} A_t, \right. \right. \\ \left. \left. \text{clip} \left(\frac{f_\theta(o_t | q, o_{<t})}{f_{\text{old}}(o_t | q, o_{<t})}, 1 - \epsilon, 1 + \epsilon \right) A_t \right) \right]. \quad (2)$$

Although widely used in alignment tasks, PPO has several significant limitations. Its learning process is unstable, computationally expensive, and requires extensive hyperparameter tuning. The clipped objective can slow convergence and yield suboptimal policies (Engstrom et al., 2020). Notably, training the value model is challenging due to inherently high variance and poor generalization (Andrychowicz et al., 2020). Besides, PPO is also prone to reward hacking, struggles with long-term credit assignment, and suffers from the issue of sample inefficiency (Henderson et al., 2018).

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) simplifies PPO via removing the learnable value model. Instead, GRPO uses the average reward of mul-

tiple sampled outputs for the same question as the baseline. Specifically, given a question q , GRPO requires to sample G outputs from the old policy as $\{o_i\}_{i=1}^G \sim f_{\text{old}}(\cdot|q)$. Then, it computes the reward r_i for each output o_i (through deterministic reward functions) and obtains a group of rewards $\{r_i\}_{i=1}^G$. The advantage \hat{A}_i of GRPO is estimated as:

$$\hat{A}_i = \tilde{r}_i = \frac{r_i - \text{mean}(\{r_i\}_{i=1}^G)}{\text{std}(\{r_i\}_{i=1}^G)}. \quad (3)$$

The objective of GRPO is to maximize the advantage (the first term) while ensuring that the policy model remains close to the reference policy (the KL divergence term):

$$\mathcal{J}_{\text{GRPO}}(f_\theta) \triangleq \mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim f_{\text{old}}(\cdot|q)} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left[\min \left(\frac{f_\theta(o_{i,t} | q, o_{i,<t})}{f_{\text{old}}(o_{i,t} | q, o_{i,<t})} \hat{A}_{i,t}, \right. \right. \right. \\ \left. \left. \text{clip} \left(\frac{f_\theta(o_{i,t} | q, o_{i,<t})}{f_{\text{old}}(o_{i,t} | q, o_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right) \right. \\ \left. \left. - \beta \mathbb{D}_{\text{KL}}[f_\theta || f_{\text{ref}}] \right] \right]. \quad (4)$$

Here, the clip $(\cdot, 1 - \epsilon, 1 + \epsilon)$ mechanism ensures that the update of every step do not deviate excessively from the old policy by bounding the policy ratio between $1 - \epsilon$ and $1 + \epsilon$. Besides, the KL divergence is estimated as follows:

$$\mathbb{D}_{\text{KL}}[f_\theta || f_{\text{ref}}] = \frac{f_{\text{ref}}(o_{i,t} | q, o_{i,<t})}{f_\theta(o_{i,t} | q, o_{i,<t})} - \log \frac{f_{\text{ref}}(o_{i,t} | q, o_{i,<t})}{f_\theta(o_{i,t} | q, o_{i,<t})} - 1. \quad (5)$$

Nonetheless, GRPO can be challenging to implement because it sometimes produces outputs with unintended token distributions or incoherent language patterns (Guo et al., 2025). It also demands careful reward function design to balance fairness constraints and meaningful group-based advantage estimation (Stiennon et al., 2020). The RL training process in GRPO can be unstable due to its reliance on group-based relative advantages, and it remains computationally expensive, especially for large-scale implementations (Schulman et al., 2017; Ouyang et al., 2022). Furthermore, while GRPO introduces optimizations to post-training, it does not consistently outperform other simpler methods like SFT, particularly in small-scale training.

In addition, several RL algorithms have been developed primarily for alignment tasks. Therein, DPO (Rafailov et al., 2023), CPO (Xu et al., 2024), and their variants (Li et al., 2024; Guo et al., 2024; Munos et al., 2024; Hong et al., 2024; Xie et al., 2024) rely on pairs of outputs labeled by human preference. In contrast, KTO (Ethayarajh et al., 2024) and

BCO (Jung et al., 2024) require only a single binary label (like or dislike) for each output. Besides, the PRM (Uesato et al., 2022; Lightman et al., 2024) and Step-KTO (Lin et al., 2025a) offer stepwise guidance by incorporating feedback at each reasoning step rather than focusing solely on the final outputs. Recently, the follow-up work of GRPO improves the optimization objective, *e.g.*, DAPO (Yu et al., 2025), Dr. GRPO (Liu et al., 2025a), REINFORCE++ (Hu, 2025), CPPO (Lin et al., 2025b), and GPG (Chu et al., 2025b). Another line of research generalizes GRPO to broader applications such as multimodal reasoning (Zhou et al., 2025; Huang et al., 2025; Chu et al., 2025b; Liu et al., 2025b; Zhang et al., 2025) and logical reasoning (Xie et al., 2025).

3. Tracking the Training Dynamics of RL Post-training

Post-training fundamentally relies on three key components: the algorithm, the dataset, and the base model. Specifically, the base model is post-trained using a dataset and optimized with an algorithm, resulting in a refined post-trained model. To uncover the underlying behaviors that contribute to the observed limitations of GRPO, we conduct a systematic analysis of RL post-training on reasoning tasks. In this study, we track the training dynamics of the GRPO algorithm across three critical metrics: *problem difficulty*, *model confidence*, and *output advantages*. These metrics are defined in a general manner, ensuring their applicability to a wide range of algorithms, datasets, and models.

Specifically, given a language model f_θ and a question-answer pair (q, a) , we sample G outputs from the model as $\{o_i\}_{i=1}^G \sim f_\theta(\cdot|q)$. We only compute the accuracy reward r_i for each output by matching the ground-truth answer a and the predicted answer in o_i , where $r_i = 1$ if correct and 0 otherwise. With outputs $\{o_i\}_{i=1}^G$ and rewards $\{r_i\}_{i=1}^G$, we compute the metrics of confidence and difficulty as:

$$\begin{aligned} \text{Confidence}(f_\theta, q, o_i) &= \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} f_\theta(o_{i,t}|q, o_{i,<t}), \\ \text{Difficulty}(f_\theta, q, a) &= 1 - \frac{1}{G} \sum_{i=1}^G r_i. \end{aligned} \quad (6)$$

Then, we split the questions into five difficulty levels using the difficulty metric:¹ 0-0.2 (level 1), 0.2-0.4 (level 2), 0.4-0.6 (level 3), 0.6-0.8 (level 4), and 0.8-1 (level 5). For questions in each level, we map each generated output to an Advantage-Confidence (AC) plane (Fig. 1). We plot its confidence (exponentiated average token log-probability) on the y-axis against its advantage (the reward relative to the group) on the x-axis. This visualization reveals patterns

¹In Appendix B, we verify that the model-estimated difficulty well aligns with the ground truth difficulty.

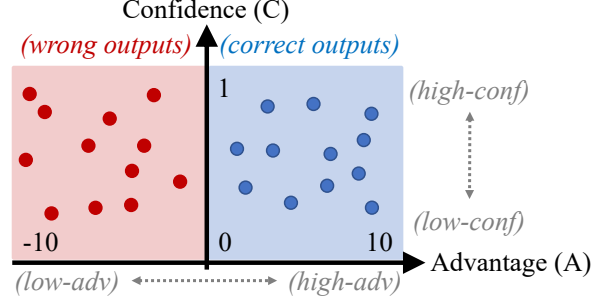


Figure 1: The illustration of AC plane. Higher confidence indicates greater decoding probability (Eqn. 6). Advantage indicates the relative output quality within a group (Eqn. 3).

that are not apparent from accuracy metrics alone.

Next, we post-train the Qwen2.5-Math-1.5B (Yang et al., 2024b) model with GRPO. The AC planes of the training samples of the MATH dataset (Hendrycks et al., 2021) and evaluation samples of the MATH500 dataset (Lightman et al., 2024) are shown in Figs. 2 and 3, respectively. Based on these, we summarize the following three observations.

Observation 1 (Advantage Contraction). *As GRPO training improves accuracy, the advantage distributions contract around zero, making it harder to differentiate outputs within response groups.* Due to the inherent design of the GRPO advantage mechanism, the magnitude of advantage is not directly linearly correlated with model accuracy. Instead, it captures the internal variance within response groups. As GRPO training progresses, the advantage distributions across all difficulty levels tend to contract around zero. This convergence is driven by increasing model accuracy, which results in a larger fraction of correct outputs within each group. Consequently, while the advantage of these correct outputs diminishes, it typically remains non-negative. However, this also implies that the signal for differentiating between highly effective and slightly less effective outputs within a group can weaken, potentially reducing the granularity of the advantage feedback in optimization iterations.

Observation 2 (Confidence Saturation). *As GRPO training progresses, the model’s confidence increasingly concentrates near 100%, improving accuracy but also leading to overconfidence in incorrect outputs.* Before training, the model exhibits a similar confidence distribution across difficulty levels: it tends to assign higher confidence to correct outputs, while the confidence for incorrect outputs remains highly variable. As GRPO training progresses, the model’s overall confidence grows towards 100%. This trend aligns with improving model accuracy but also reveals a downside. Namely, the model becomes increasingly overconfident in its incorrect outputs, indicating a potential overfitting or miscalibration in the training process. This aligns with the entropy collapse observed in Yu et al. (2025). Besides, we observe similar trends in the validation set (see Fig. 3).

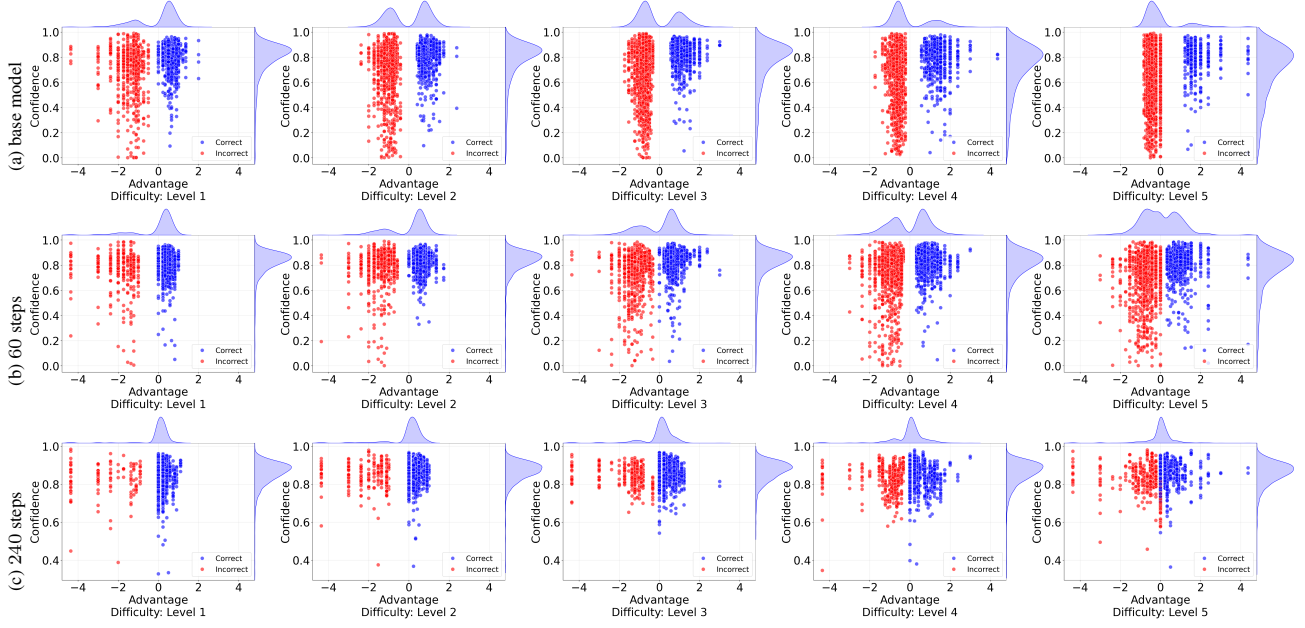


Figure 2: The AC planes of the Qwen2.5-Math-1.5B model, post-training on the MATH dataset with the GRPO algorithm. We present the AC plane of model checkpoints captured at training steps 0 (base model), 60, and 240 on the training set. Columns 1 through 5 correspond to difficulty levels 1 through 5. Additionally, the marginal distributions (*i.e.*, the density distributions) of advantage and confidence across all samples are shown above and to the right of each AC plane.

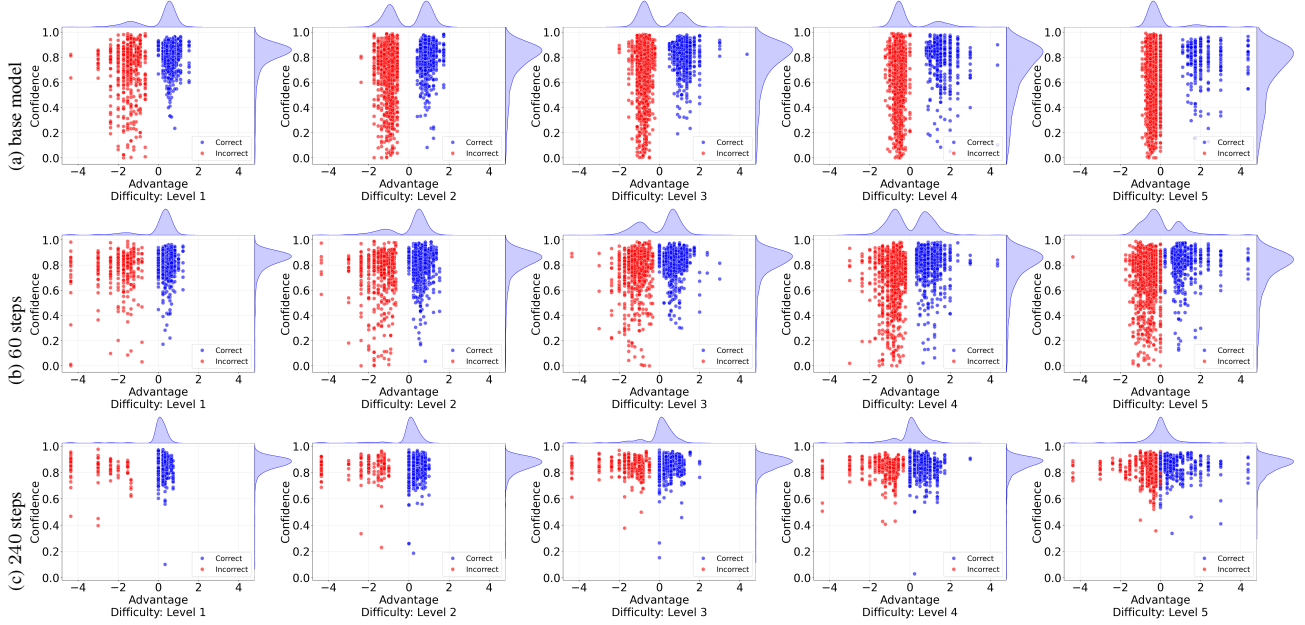


Figure 3: The AC planes of the Qwen2.5-Math-1.5B model, evaluated on the MATH validation set with the checkpoints captured at training steps 0 (base model), 60, and 240 (consistent with Fig. 2).

Observation 3 (Hierarchical Convergence). *Easy questions move towards higher confidence and positive advantage, while harder ones shift positively but retain a significant proportion of errors.* For easier questions, the model’s initial advantage is concentrated in the non-negative range with high confidence, even before GRPO training. As training progresses, the advantage shifts closer to zero, accom-

panied by a further increase in confidence. In contrast, for more challenging questions, the initial advantage is skewed towards the non-positive range, reflecting a higher frequency of incorrect, low-confidence responses. With the progression of GRPO training, this distribution gradually shifts toward the positive range as the model accuracy improves, though a notable portion of incorrect responses persists.

4. CoDaPO: Confidence and Difficulty-Adaptive Policy Optimization

This section introduces CoDaPO, a novel RL-based post-training algorithm inspired by the insights from Sec. 3. CoDaPO dynamically adjusts the weighting of advantages based on both model confidence and problem difficulty, prioritizing uncertain outputs in challenging scenarios where the potential for improvement is greatest. The learning objective provides the strongest positive reinforcement for correct and high-confidence responses on difficult problems.

$$\mathcal{J}_{\text{CoDaPO}}(f_{\theta}) \triangleq \mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim f_{\text{old}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \left[\frac{f_{\theta}(o_{i,t}|q, o_{i,<t})}{f_{\text{old}}(o_{i,t}|q, o_{i,<t})} \cdot \hat{A}_i \cdot \hat{c}_i \cdot \hat{d}_q \right] \right], \quad (7)$$

$$\begin{aligned} \text{where } \hat{A}_i &= \frac{r_i - \text{mean}(\{r_i\}_{i=1}^G)}{\text{std}(\{r_i\}_{i=1}^G)}, \\ \hat{c}_i &= \sigma \left(\frac{\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \log f_{\theta}(o_{i,t}|q, o_{i,<t})}{\frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \log f_{\theta}(o_{i,t}|q, o_{i,<t})} \right), \\ \hat{d}_q &= 1 + \delta \left(\frac{1}{G} \sum_{i=1}^G r_i - \frac{1}{2} \right)^2. \end{aligned}$$

Here, the $\sigma(\cdot)$ denotes the sigmoid function, and δ is a hyperparameter to scale the average value of rewards within a group. In the following, we elaborate on the underlying design rationales of each component in CoDaPO.

Computing rewards and advantages. Following the understanding part in Sec. 3, we only adopt the accuracy reward r_i (instead of integrating the format or length reward) in evaluating the quality of each output o_i . Next, we estimate the advantage \hat{A}_i of the output o_i as $\hat{A}_i = r_i - \text{mean}(\{r_i\}_{i=1}^G) / \text{std}(\{r_i\}_{i=1}^G)$. Building upon this estimate, we propose a confidence- and difficulty-adaptive reweighting mechanism to refine the use of advantages. Specifically, we multiply the advantage \hat{A}_i with the output-wise confidence weight \hat{c}_i and the question-wise difficulty weight \hat{d}_q , and obtain the final product $\hat{A}_i \cdot \hat{c}_i \cdot \hat{d}_q$. To optimize the scalar \hat{A}_i , we multiply it with the token-wise probability $f_{\theta}(o_{i,t}|q, o_{i,<t}) / f_{\text{old}}(o_{i,t}|q, o_{i,<t})$ to ensure correct gradient flow.

Advantage reweighting w.r.t. confidence. The confidence-aware scaling factor \hat{c}_i compares the confidence of a specific output against the average group confidence. Outputs with higher confidence receive greater weight when correct (positive advantage) and stronger penalties when incorrect (negative advantage), thereby naturally addressing the confidence saturation problem. We stabilize the factor smoothly using the sigmoid function to avoid extreme values that may arise due to the unbounded nature of logarithms.

Advantage reweighting w.r.t. difficulty. Instead of directly

applying the linear difficulty estimation proposed in Eqn. 6, we adopt a quadratic reweighting strategy that emphasizes both the easiest and most difficult questions, creating a U-shaped importance curve. This encourages the model to maintain strong performance on easy problems while accelerating improvement on difficult ones. The training effect of this reweighting is: (1) For hard questions, correct answers tend to receive higher advantages, while incorrect ones yield small-magnitude negative advantages, and (2) for easy questions, incorrect answers are penalized more severely, and correct answers do not receive excessive advantages.

Sample-level normalization. We remove the normalization terms $1/G$ and $1/|o_i|$ in the original GRPO objective (Eqn. 4) to address the potential length bias within a group (Yu et al., 2025; Liu et al., 2025a). Specifically, we sum up the advantage terms of all tokens and all outputs and then normalize the accumulated value by multiplying $1/\sum_{i=1}^G |o_i|$.

Discarding the KL regularization. While KL regularization (e.g., Eqn. 5) is valuable for alignment tasks to prevent divergence from human preferences, we find it unnecessarily constrains exploration in reasoning tasks where discovering novel solution strategies is beneficial. Besides, the KL term increases the overall computational cost, attributed to the forward inference of the reference model. Therefore, we remove this term to streamline the optimization as well as allow the policy to explore more diverse reasoning paths while still maintaining coherence through the PPO objective.

5. Experiments

In this section, we evaluate our proposed RL algorithm for post-training LLMs on mathematical reasoning tasks. We first describe the experimental setup, followed by the main experimental results and ablation studies, to demonstrate the effectiveness of our approach against established baselines.

Baselines. We compare our proposed algorithm with several strong and representative baselines: GRPO (Shao et al., 2024), a widely adopted RL method for LLM post-training, and its variants. Specifically, we evaluate against recently proposed RL algorithms, including DAPO (Yu et al., 2025), Dr. GRPO (Liu et al., 2025a), and GPG (Chu et al., 2025b).

Models. We evaluate the above algorithms on Qwen2.5-1.5B-Instruct (Yang et al., 2024a), a general-purpose model with 1.5B parameters that has been instruction-tuned on a diverse corpus, and Qwen2.5-Math-1.5B (Yang et al., 2024b), a specialized model pre-trained on high-quality synthetic mathematical content with the same parameter count.

Training setup. We use the MATH (Lightman et al., 2024; Hendrycks et al., 2021) dataset for post-training, which includes 7,500 problems from mathematics competitions. Due to computing constraints, all RL algorithms are implemented using 6 NVIDIA A100 GPUs (80 GB each), with 4

| Normalization | Advantage | Regularization | Note |
|---|---|---|---|
| GRPO (Shao et al., 2024) | | | |
| $\mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim f_{\text{old}}(\cdot q)} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{ o_i } \sum_{t=1}^{ o_i } \left[\min \left(\frac{f_{\theta}(o_{i,t} q, o_{i,<t})}{f_{\text{old}}(o_{i,t} q, o_{i,<t})} \hat{A}_i, \right. \right. \right.$ | $\left. \left. \text{clip} \left(\frac{f_{\theta}(o_{i,t} q, o_{i,<t})}{f_{\text{old}}(o_{i,t} q, o_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_i \right) - \log \frac{f_{\text{ref}}(o_{i,t} q, o_{i,<t})}{f_{\theta}(o_{i,t} q, o_{i,<t})} - 1 \right] \right]$ | $-\beta \left(\frac{f_{\text{ref}}(o_{i,t} q, o_{i,<t})}{f_{\theta}(o_{i,t} q, o_{i,<t})} \right)$ | $\hat{A}_i = \frac{r_i - \text{mean}(\{r_i\}_{i=1}^G)}{\text{std}(\{r_i\}_{i=1}^G)}$ |
| Dr. GRPO (Liu et al., 2025a) | | | |
| $\mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim f_{\text{old}}(\cdot q)} \left[\frac{1}{G \cdot c} \sum_{i=1}^G \sum_{t=1}^{ o_i } \left[\min \left(\frac{f_{\theta}(o_{i,t} q, o_{i,<t})}{f_{\text{old}}(o_{i,t} q, o_{i,<t})} \hat{A}_i, \right. \right. \right.$ | $\left. \left. \text{clip} \left(\frac{f_{\theta}(o_{i,t} q, o_{i,<t})}{f_{\text{old}}(o_{i,t} q, o_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_i \right) - \log \frac{f_{\text{ref}}(o_{i,t} q, o_{i,<t})}{f_{\theta}(o_{i,t} q, o_{i,<t})} - 1 \right] \right]$ | $-\beta \left(\frac{f_{\text{ref}}(o_{i,t} q, o_{i,<t})}{f_{\theta}(o_{i,t} q, o_{i,<t})} \right)$ | $\hat{A}_i = r_i - \text{mean}(\{r_i\}_{i=1}^G)$ |
| DAPO (Yu et al., 2025) | | | |
| $\mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim f_{\text{old}}(\cdot q)} \left[\frac{1}{\sum_{i=1}^G \frac{1}{ o_i } \sum_{i=1}^G \sum_{t=1}^{ o_i } \left[\min \left(\frac{f_{\theta}(o_{i,t} q, o_{i,<t})}{f_{\text{old}}(o_{i,t} q, o_{i,<t})} \hat{A}_i, \right. \right. \right.$ | $\left. \left. \text{clip} \left(\frac{f_{\theta}(o_{i,t} q, o_{i,<t})}{f_{\text{old}}(o_{i,t} q, o_{i,<t})}, 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}} \right) \hat{A}_i \right) \right] \right]$ | None | $\hat{A}_i = \frac{r_i - \text{mean}(\{r_i\}_{i=1}^G)}{\text{std}(\{r_i\}_{i=1}^G)}$ |
| GPG (Chu et al., 2025b) | | | |
| $\mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim f_{\text{old}}(\cdot q)} \left[\frac{1}{\sum_{i=1}^G \frac{1}{ o_i } \sum_{i=1}^G \sum_{t=1}^{ o_i } \left[\log f_{\theta}(o_{i,t} q, o_{i,<t}) \hat{A}_i \right] \right]$ | | None | $\hat{A}_i = \alpha \cdot (r_i - \text{mean}(\{r_i\}_{i=1}^G))$ |
| CoDaPO (ours) | | | |
| $\mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim f_{\text{old}}(\cdot q)} \left[\frac{1}{\sum_{i=1}^G \frac{1}{ o_i } \sum_{i=1}^G \sum_{t=1}^{ o_i } \left[\frac{f_{\theta}(o_{i,t} q, o_{i,<t})}{f_{\text{old}}(o_{i,t} q, o_{i,<t})} \cdot \hat{A}_i \cdot \hat{c}_i \cdot \hat{d}_q \right] \right]$ | | None | $\hat{A}_i = \frac{r_i - \text{mean}(\{r_i\}_{i=1}^G)}{\text{std}(\{r_i\}_{i=1}^G)}, \hat{c}_i = \sigma \left(\frac{\frac{1}{G} \sum_{i=1}^G \frac{1}{ o_i } \sum_{t=1}^{ o_i } \log f_{\theta}(o_{i,t} q, o_{i,<t})}{\frac{1}{ o_i } \sum_{t=1}^{ o_i } \log f_{\theta}(o_{i,t} q, o_{i,<t})} \right),$ $\hat{d}_q = 1 + \delta \left(\frac{1}{G} \sum_{i=1}^G r_i - \frac{1}{2} \right)^2$ |

Table 1: Comparing the components in different RL algorithms for reasoning. Note that the **highlighted** contents indicate the differences with the original implementation of the GRPO algorithm. Specifically, (1) the advantage term evaluates the quality of model-generated outputs, (2) the regularization term measures the divergence between the policy model and reference model, and (3) the normalization term scales the final optimization signal across multiple problems and outputs.

GPUs allocated for inference during training and the remaining two for parameter optimization. We use Transformer Reinforcement Learning (von Werra et al., 2020) as the training backbone and use the same settings in all method training to ensure fair comparison. For each question, we sample 12 outputs and set the accumulation step to 12 to reduce the impact of batch variance. We set the learning rate to 3.0e-06 for 3 epochs and perform early stopping when the accuracy reward converges. In CoDaPO, we set the δ in the difficulty-adaptive weighting function to 0.4.

Evaluation setup. Our framework adapts Qwen2.5-Math’s evaluation codebase², ensuring consistent and reliable measurement across all experiments. For conducting a rigorous evaluation, we sample 5 responses per question with temperature 0.6, averaging the accuracy across these samples. This approach reduces variance from sampling randomness while maintaining a reasonable diversity of reasoning paths. We evaluate on 7 benchmarks of reasoning tasks, including MATH500 (Lightman et al., 2024; Hendrycks et al., 2021), AIME 2024, AIME 2025, AMC 2023, Olympiad Bench (Huang et al., 2024), Minerva (Lewkowycz et al., 2022), and GSM8K (Cobbe et al., 2021).

²<https://github.com/QwenLM/Qwen2.5-Math/tree/main/evaluation>

Main results. We summarize the following observations with respect to the experimental results in Tab. 2:

- **CoDaPO can significantly improve LLMs’ reasoning abilities.** Applying CoDaPO to various models trained on the MATH training set consistently yields significant performance improvements across different benchmarks. For instance, with the Qwen2.5-Math-1.5B model, CoDaPO elevates the accuracy on MATH500 from 31.20% to 71.68%, and the average accuracy across seven distinct benchmarks improves from 18.37% to 43.41%. Notably, this enhancement in accuracy is more pronounced for base models that have not undergone instruction tuning.
- **CoDaPO exhibits remarkable generalization capabilities.** Despite being post-trained solely on the MATH training set, CoDaPO demonstrates surprisingly strong generalization to out-of-domain datasets. For instance, on the out-of-domain Olympiad Bench benchmark, the model’s accuracy improves from 18.25% to 32.80%, and on Minerva benchmark, the accuracy increases from 9.63% to 31.69%. This indicates the potential of CoDaPO to significantly enhance reasoning abilities across a wide range of tasks, even those not encountered during training.
- **CoDaPO’s performance significantly surpasses other baselines.** The comparison of CoDaPO with other RL

| Base Model | Algorithm | Datasets | | | | | | | Average |
|-----------------------|----------------------|--------------|--------------|--------------|--------------|-------------------|--------------|--------------|--------------|
| | | MATH 500 | AIME 2024 | AIME 2025 | AMC 2023 | Olympiad Bench | Minerva | GSM8K | |
| Qwen2.5-1.5B-Instruct | Base | 49.36 | 3.00 | 0.00 | 23.50 | 17.01 | 16.84 | 59.56 | 24.18 |
| | GRPO | 55.08 | 3.00 | 1.33 | 26.00 | 19.08 | 22.35 | 73.92 | 28.68 |
| | DAPO | 56.04 | 2.67 | 0.00 | <u>27.50</u> | 18.90 | 21.91 | 73.04 | 28.58 |
| | Dr. GRPO | 56.24 | <u>3.33</u> | 0.00 | 25.50 | 19.11 | 21.18 | 73.80 | 28.45 |
| | GPG | 54.36 | 4.00 | 1.33 | 23.00 | <u>19.29</u> | 21.54 | <u>74.10</u> | 28.34 |
| | CoDaPO (ours) | 55.60 | <u>3.33</u> | 1.33 | 31.00 | 19.44 | <u>22.21</u> | 74.30 | 29.60 |
| | | (6.24↑) | (0.33↑) | (1.33↑) | (7.50↑) | (2.43↑) | (5.37↑) | (14.74↑) | (5.42↑) |
| Qwen2.5-Math-1.5B | Base | 31.20 | 6.00 | 0.00 | 29.50 | 18.25 | 9.63 | 34.01 | 18.37 |
| | GRPO | 68.48 | 12.67 | 8.00 | <u>51.00</u> | 30.55 | 28.53 | 79.56 | 39.82 |
| | DAPO | <u>71.12</u> | <u>14.00</u> | 8.00 | 49.00 | <u>31.97</u> | 32.13 | <u>82.38</u> | <u>41.23</u> |
| | Dr. GRPO | 70.68 | 10.67 | <u>10.67</u> | 49.50 | 31.67 | 30.51 | 82.17 | 40.84 |
| | GPG | 70.60 | 11.33 | 5.33 | 47.00 | 31.97 | 29.49 | 82.38 | 39.73 |
| | CoDaPO (ours) | 71.68 | 14.67 | 16.00 | 53.50 | 32.80 | <u>31.69</u> | 83.50 | 43.41 |
| | | (40.48↑) | (8.67↑) | (16.00↑) | (24.00↑) | (14.55↑) | (22.06↑) | (49.49↑) | (25.04↑) |

Table 2: The main results of the post-training experiments (in accuracy). Note that the **boldface** numbers mean the best results, while the underlined numbers indicate the second-best results. We also show the absolute improvement of CoDaPO over the base model for each benchmark.

algorithms reveals that CoDaPO outperforms these base-lines on most benchmarks. Specifically, in terms of average accuracy, CoDaPO demonstrates a 3.59% improvement over GRPO and a 2.18% improvement over DAPO, the best-performing baseline method with the Qwen2.5-Math-1.5B model. This indicates that CoDaPO’s unique confidence and difficulty-adaptive mechanism is capable of robustly improving models’ reasoning capabilities.

The AC planes of CoDaPO. Fig. 4 and Fig. 5 present the AC planes of the Qwen2.5-Math-1.5B model on the MATH training set and validation set after training with CoDaPO. The complete AC planes are presented in Appendix B. We summarize our observations as follows:

- **CoDaPO significantly enhances the model’s accuracy on difficult problems.** On high-difficulty subsets (such as Level 4 and Level 5), the advantage distribution under CoDaPO is more concentrated on the positive side of zero compared to GRPO in Fig. 2 and Fig. 3, indicating improved accuracy on these questions. This likely stems from CoDaPO’s difficulty-adaptive mechanism introduced in Sec. 4, which assigns higher advantage weights to difficult questions, thereby guiding the model to pay more attention to problems with sparse rewards.
- **CoDaPO mitigates the model’s overconfidence issue.** Compared to GRPO (Fig. 2 and Fig. 3), CoDaPO exhibits a greater disparity between the confidence scores of correct and incorrect answers, with correct answers demonstrating higher confidence and incorrect answers showing a larger variance in confidence. This is attributed to CoDaPO’s confidence-adaptive mechanism introduced in

Sec. 4, which assigns higher advantage to correct answers with high confidence and imposes a greater penalty on incorrect answers with high confidence. This encourages the model to reinforce familiar, high-quality solutions while reducing overconfidence in erroneous responses.

Case Studies. We present several representative case studies in Appendix C. Compared to GRPO, CoDaPO demonstrates more coherent and logically structured reasoning trajectories. Its reasoning steps are typically well-ordered, mathematically sound, and more interpretable. In contrast, GRPO often exhibits fragmented or inconsistent reasoning, sometimes relying on brittle patterns such as incomplete algebraic transformations or unverified code snippets. This can lead to logically invalid outputs or incorrect final answers. These observations highlight CoDaPO’s advantage in producing not only more accurate answers, but also more robust and verifiable intermediate reasoning trajectories.

Ablation Studies. To further validate the effectiveness of our designed confidence and difficulty reweighting mechanism, we conducted a series of ablation experiments. We systematically disassembled the components of CoDaPO and performed individual training runs for each. As shown in Tab. 3, both the confidence and difficulty reweighting techniques, when implemented in isolation, yield a noticeable improvement over the vanilla GRPO baseline. Moreover, the combination of both mechanisms results in the most substantial performance gains. This suggests that the synergistic integration of confidence and difficulty reweighting within CoDaPO plays a crucial role in optimizing the learning process and achieving superior performance.

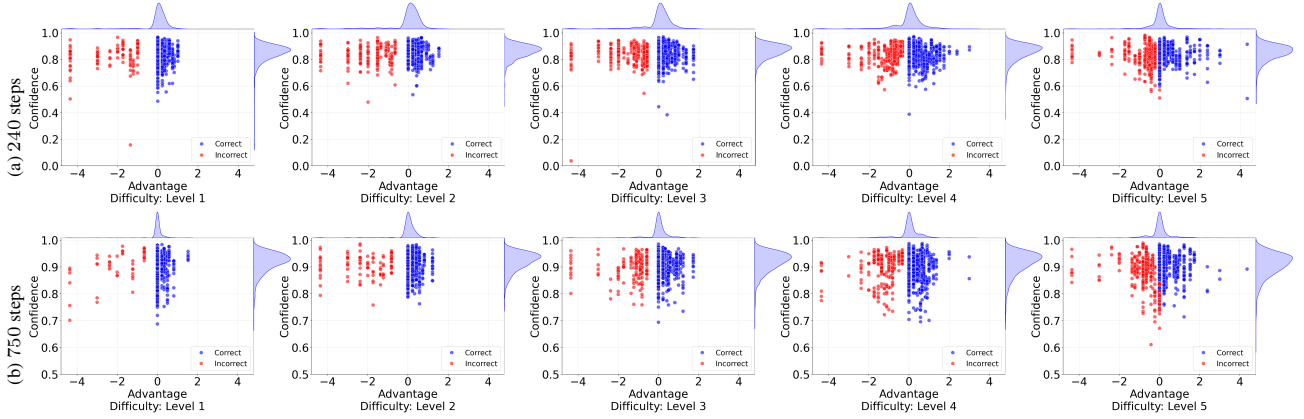


Figure 4: The AC planes of the Qwen2.5-Math-1.5B model, post-training on the MATH dataset with the CoDaPO algorithm. We present the AC plane of model checkpoints captured at the midpoint of training (240 steps) and at the end of training (750 steps). Columns 1 through 5 correspond to difficulty levels 1 through 5.

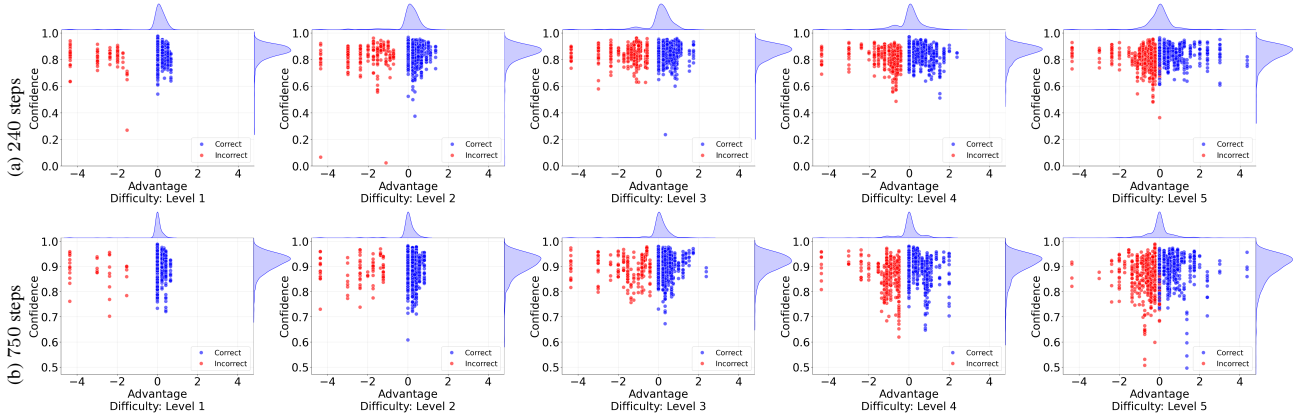


Figure 5: The AC planes of the Qwen2.5-Math-1.5B model during CoDaPO training, evaluated on the MATH validation set with the checkpoints captured at training steps 240, and 750 (consistent with Fig. 4).

| Model | MATH500 |
|--------------------------|--------------|
| Qwen2.5-Math-1.5B (base) | 31.20 |
| + GRPO | 68.48 |
| + confidence reweighting | 71.52 |
| + difficulty reweighting | 71.32 |
| + both (CoDaPO) | 71.68 |

Table 3: Ablation study on MATH500 accuracy.

6. Limitation and Conclusion

Limitation. While our results are promising, several important limitations should be noted. First, our evaluation is limited to 1.5B parameter models, and the scaling behavior of CoDaPO to models with tens or hundreds of billions of parameters remains unknown. This is particularly important as larger models may exhibit different advantage-confidence dynamics. Second, although we benchmarked on seven reasoning tasks, post-training was confined to the MATH dataset, so the robustness of CoDaPO to other domains (*e.g.*, commonsense, scientific, coding) remains unclear. Third, our experiments were conducted under moderate compute,

typical for small-scale models, and we have yet to quantify resource usage or optimize for large-scale or production-scale efficiency in RL-based post-training.

Conclusion. In this work, we identify key weaknesses of GRPO through an advantage-confidence perspective, highlighting how reward contraction, confidence saturation, and hierarchy convergence hinder RL post-training. To address these challenges, we propose CoDaPO, an RL algorithm that reweights advantages based on model confidence and problem difficulty. Across seven reasoning benchmarks, CoDaPO consistently improves reasoning performance, establishing a practical path toward more capable reasoning models and laying the foundation for future work on larger-scale models and broader post-training scenarios. Moreover, the insights from our AC plane analysis may extend beyond mathematical reasoning to other domains requiring deliberate step-by-step thinking, such as logical reasoning, planning, and scientific problem-solving. CoDaPO demonstrates that reward reshaping based on confidence and difficulty signals can help improve learning dynamics.

References

- Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 2020.
- Chu, T., Zhai, Y., Yang, J., Tong, S., Xie, S., Schuurmans, D., Le, Q. V., Levine, S., and Ma, Y. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025a.
- Chu, X., Huang, H., Zhang, X., Wei, F., and Wang, Y. Gpg: A simple and strong reinforcement learning baseline for model reasoning. *arXiv preprint arXiv:2504.02546*, 2025b.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.
- Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.
- Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Guo, S., Zhang, B., Liu, T., Liu, T., Khalman, M., Llinares, F., Rame, A., Mesnard, T., Zhao, Y., Piot, B., et al. Direct language model alignment from online ai feedback. *arXiv preprint arXiv:2402.04792*, 2024.
- Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. In *AAAI*, 2018.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. In *NeurIPS*, 2021.
- Hong, J., Lee, N., and Thorne, J. Orpo: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*, 2024.
- Hu, J. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025.
- Huang, W., Jia, B., Zhai, Z., Cao, S., Ye, Z., Zhao, F., Xu, Z., Hu, Y., and Lin, S. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.
- Huang, Z., Wang, Z., Xia, S., Li, X., Zou, H., Xu, R., Fan, R.-Z., Ye, L., Chern, E., Ye, Y., et al. Olympicarena: Benchmarking multi-discipline cognitive reasoning for superintelligent ai. *Advances in Neural Information Processing Systems*, 37:19209–19253, 2024.
- Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- Jung, S., Han, G., Nam, D. W., and On, K.-W. Binary classifier optimization for large language model alignment. *arXiv preprint arXiv:2404.04656*, 2024.
- Kumar, K., Ashraf, T., Thawakar, O., Anwer, R. M., Cholakkal, H., Shah, M., Yang, M.-H., Torr, P. H., Khan, F. S., and Khan, S. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*, 2025.
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*, 2020.
- Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.
- Li, Z., Xu, T., Zhang, Y., Lin, Z., Yu, Y., Sun, R., and Luo, Z.-Q. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. In *ICML*, 2024.
- Li, Z.-Z., Zhang, D., Zhang, M.-L., Zhang, J., Liu, Z., Yao, Y., Xu, H., Zheng, J., Wang, P.-J., Chen, X., et al. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025.
- Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let’s verify step by step. In *ICLR*, 2024.
- Lin, Y.-T., Jin, D., Xu, T., Wu, T., Sukhbaatar, S., Zhu, C., He, Y., Chen, Y.-N., Weston, J., Tian, Y., et al. Step-kto: Optimizing mathematical reasoning through stepwise binary feedback. *arXiv preprint arXiv:2501.10799*, 2025a.

- Lin, Z., Lin, M., Xie, Y., and Ji, R. Cppo: Accelerating the training of group relative policy optimization-based reasoning models. [arXiv preprint arXiv:2503.22342](#), 2025b.
- Liu, Z., Chen, C., Li, W., Qi, P., Pang, T., Du, C., Lee, W. S., and Lin, M. Understanding rl-zero-like training: A critical perspective. [arXiv preprint arXiv:2503.20783](#), 2025a.
- Liu, Z., Sun, Z., Zang, Y., Dong, X., Cao, Y., Duan, H., Lin, D., and Wang, J. Visual-rft: Visual reinforcement fine-tuning. [arXiv preprint arXiv:2503.01785](#), 2025b.
- Liu, Z., Wang, P., Xu, R., Ma, S., Ruan, C., Li, P., Liu, Y., and Wu, Y. Inference-time scaling for generalist reward modeling. [arXiv preprint arXiv:2504.02495](#), 2025c.
- Luo, M., Tan, S., Huang, R., Patel, A., Ariyak, A., Wu, Q., Shi, X., Xin, R., Cai, C., Weber, M., Zhang, C., Li, L. E., Popa, R. A., and Stoica, I. Deepcoder: A fully open-source 14b coder at o3-mini level, 2025.
- Muennighoff, N., Yang, Z., Shi, W., Li, X. L., Fei-Fei, L., Hajishirzi, H., Zettlemoyer, L., Liang, P., Candès, E., and Hashimoto, T. s1: Simple test-time scaling. [arXiv preprint arXiv:2501.19393](#), 2025.
- Munos, R., Valko, M., Calandriello, D., Azar, M. G., Rowland, M., Guo, Z. D., Tang, Y., Geist, M., Mesnard, T., Michi, A., et al. Nash learning from human feedback. In [ICML](#), 2024.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. In [NeurIPS](#), 2022.
- Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In [NeurIPS](#), 2023.
- Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. In [NeurIPS](#), 2023.
- Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In [ICML](#), 2015a.
- Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. [arXiv preprint arXiv:1506.02438](#), 2015b.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. [arXiv preprint arXiv:1707.06347](#), 2017.
- Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. [arXiv preprint arXiv:2402.03300](#), 2024.
- Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters. [arXiv preprint arXiv:2408.03314](#), 2024.
- Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. In [NeurIPS](#), 2020.
- Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., Creswell, A., Irving, G., and Higgins, I. Solving math word problems with process-and outcome-based feedback. [arXiv preprint arXiv:2211.14275](#), 2022.
- von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., and Galouédec, Q. Trl: Transformer reinforcement learning, 2020.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., brian ichter, Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. In [NeurIPS](#), 2022.
- Xie, T., Foster, D. J., Krishnamurthy, A., Rosset, C., Awadallah, A., and Rakhlin, A. Exploratory preference optimization: Harnessing implicit q^* -approximation for sample-efficient rlhf. [arXiv preprint arXiv:2405.21046](#), 2024.
- Xie, T., Gao, Z., Ren, Q., Luo, H., Hong, Y., Dai, B., Zhou, J., Qiu, K., Wu, Z., and Luo, C. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. [arXiv preprint arXiv:2502.14768](#), 2025.
- Xu, H., Sharaf, A., Chen, Y., Tan, W., Shen, L., Van Durme, B., Murray, K., and Kim, Y. J. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. [arXiv preprint arXiv:2401.08417](#), 2024.
- Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2. 5 technical report. [arXiv preprint arXiv:2412.15115](#), 2024a.
- Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. [arXiv preprint arXiv:2409.12122](#), 2024b.
- Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. R. Tree of thoughts: Deliberate problem solving with large language models. In [NeurIPS](#), 2023a.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In ICLR, 2023b.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., et al. Dapo: An open-source llm reinforcement learning system at scale. arXiv preprint arXiv:2503.14476, 2025.

Zeng, W., Huang, Y., Liu, Q., Liu, W., He, K., Ma, Z., and He, J. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. arXiv preprint arXiv:2503.18892, 2025a.

Zeng, W., Huang, Y., Liu, W., He, K., Liu, Q., Ma, Z., and He, J. 7b model and 8k examples: Emerging reasoning with reinforcement learning is both effective and efficient, 2025b.

Zhang, J., Huang, J., Yao, H., Liu, S., Zhang, X., Lu, S., and Tao, D. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization. arXiv preprint arXiv:2503.12937, 2025.

Zhou, H., Li, X., Wang, R., Cheng, M., Zhou, T., and Hsieh, C.-J. R1-zero's "aha moment" in visual reasoning on a 2b non-sft model. arXiv preprint arXiv:2503.05132, 2025.

Appendix

| | | |
|----------|-------------------------------|-----------|
| A | Implementation Details | 14 |
| B | Full Experiments | 15 |
| C | Case Studies | 25 |

A. Implementation Details

Experiment framework. In this work, we utilize Transformer Reinforcement Learning (TRL) as the training backbone, specifically version 0.16.1. To ensure consistency with the vanilla GRPO algorithm, we revise the original GRPO trainer in TRL accordingly since it has some discrepancies in loss computation from Shao et al. (2024). Building upon this implementation, we develop several baseline methods evaluated in this work, including DAPO, Dr.GRPO, GPG, and our proposed approach, CoDaPO.

General hyperparameters. To ensure fair comparisons, we train all algorithms using the same set of hyperparameters.

- **Sampling setting.** For each question, we sample 12 responses to form a response group. The sampling temperature is set to 0.9, and we use a top-p value of 1.0 to consider the full token distribution.
- **Learning rate.** We use a learning rate of $3.0e - 6$ and apply a warm-up over the first 10% of global training steps. A cosine learning rate scheduler is employed to gradually reduce the learning rate to zero throughout training.
- **Batch size.** We set the batch size to 6 to evenly distribute the generated responses across training devices. To mitigate the variance introduced by different question samples, we use a gradient accumulation step of 12.
- **Randomness control.** To ensure reproducibility, we set the random seed to 42 and enable full determinism.

Prompt template. We use the prompt template shown in Fig. 6 for both training and evaluation. Furthermore, we apply chat template shown in Fig. 7 when processing the raw data.

Prompt Template

Please reason step by step, and put your final answer within \boxed{ }.

Figure 6: Prompt template used during training and evaluation.

Chat Template

```
<|im_start|>system
Please reason step by step, and put your final answer within \boxed{ }.
<|im_end|>
<|im_start|>user
Find the sum of all integer bases  $b > 9$  for which  $17_b$  is a divisor of  $97_b$ .
<|im_end|>
<|im_start|>assistant
```

Figure 7: Example of a prompt used after applying the chat template.

Reward setting. We observe that although using a format reward can improve the readability of LLM outputs to some extent, it may lead to reward hacking in the later stages of training, potentially resulting in irreversible training collapse. This observation is consistent with the findings of (Zeng et al., 2025a). Therefore, in all experiments conducted in this work, we rely solely on the accuracy reward, defined as follows:

$$r_i(y_i, \hat{y}) = \begin{cases} 1, & \text{is_equivalent}(y_i, \hat{y}) \\ 0, & \text{otherwise} \end{cases}.$$

Here, y_i denotes the answer produced for the i -th output and \hat{y} represents the corresponding ground truth. To accurately determine whether y_i and \hat{y} are equivalent, we employ Math-Verify³ as our reward evaluation system, where all LaTeX expressions are parsed and compared for mathematical equivalence.

³<https://github.com/huggingface/Math-Verify>

Completion length constraint. Following the work of (Zeng et al., 2025a), which argues that overly lenient length constraints—or any heuristics that encourage the model to produce excessively long chains of thought—can, in many cases, lead to overthinking, we impose a strict maximum completion length of 1024 tokens during both training and evaluation. While this constraint may appear intuitively unreasonable, our experimental results demonstrate that reinforcement learning post-training under such a short-CoT constraint can still yield strong performance (see Fig. 10).

B. Full Experiments

Difficulty estimation. Accurate estimation of question difficulty plays a crucial role in our proposed algorithm, CoDaPO, as it directly influences the computation of difficulty-adaptive weights in the optimization objective. However, relying solely on pre-existing difficulty annotations presents significant limitations. First, not all datasets contain ground-truth difficulty labels. Second, since the model’s performance evolves during training, the perceived difficulty of a question may vary over time. Consequently, fixed difficulty labels may fail to reflect the dynamic nature of the model’s learning process.

To explore a more adaptive and robust difficulty estimation approach, we conduct preliminary experiments on the MATH dataset, which includes human-annotated difficulty levels. Using GRPO on Qwen2.5-Math-1.5B, we analyze the accuracy trajectories for different difficulty levels throughout training. As shown in Fig. 8, model accuracy aligns well with the ground-truth difficulty labels: easier questions correspond to higher accuracy, and harder questions to lower accuracy. Moreover, the accuracy gap between different difficulty levels increases and stabilizes as training progresses, indicating a consistent difficulty signal.

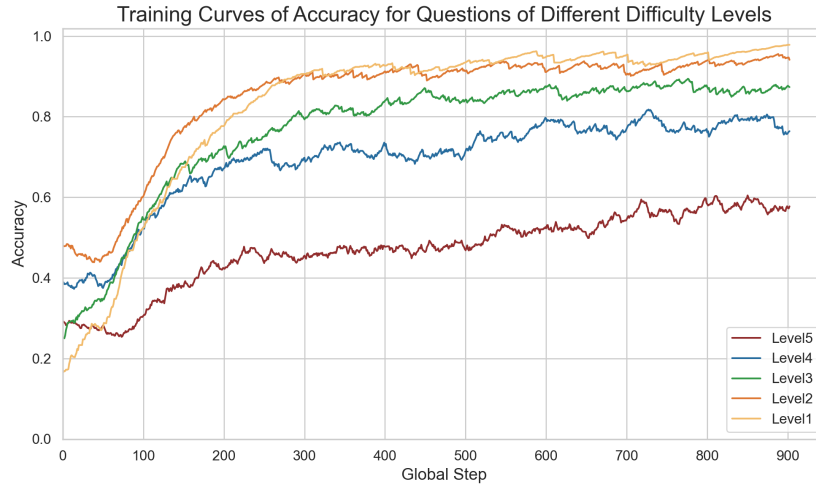


Figure 8: The GRPO training accuracy curves for questions of different difficulty levels on Qwen2.5-Math-1.5B.

These observations motivate us to estimate question difficulty based on the model’s own accuracy. This approach is naturally integrated into the online RL process without requiring any additional evaluation model, as multiple samples per question are generated during training. It generalizes well to datasets without difficulty annotations and provides a dynamic estimation mechanism that adapts to the model’s evolving capabilities over time, overcoming the limitations of static difficulty labels.

Completion length. In Figs. 9 and 10, we illustrate how the length of model responses evolves throughout the training process. Contrary to prior studies that advocate for encouraging longer responses as a means of enhancing model performance, our observations reveal a different perspective—namely, that even under stringent response length constraints (e.g., 1024 tokens), models can still achieve substantial performance gains. In fact, although we explicitly impose a hard stop when a model’s response exceeds 1024 tokens, the model never actually reaches this limit during training.

Interestingly, performance improvements are often accompanied by a decrease in output length. We observe a similar trend in vanilla GRPO as well. We hypothesize that this phenomenon stems from the nature of the math questions, which typically do not require excessively long reasoning chains. Instead, longer responses from the base model often reflect overthinking, including the generation of unnecessary reasoning steps or placeholder verification logic (e.g., writing pseudo Python code

to "verify" results without external execution support).

RL mitigates these tendencies by discouraging redundant generation and encouraging the model to produce concise, solution-oriented answers. As a result, we observe a consistent downward trend in average completion length, which aligns with improved performance.

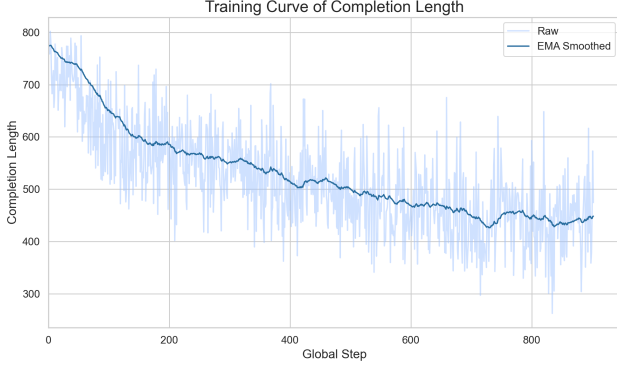


Figure 9: The GRPO training curves for completion length on Qwen2.5-Math-1.5B.

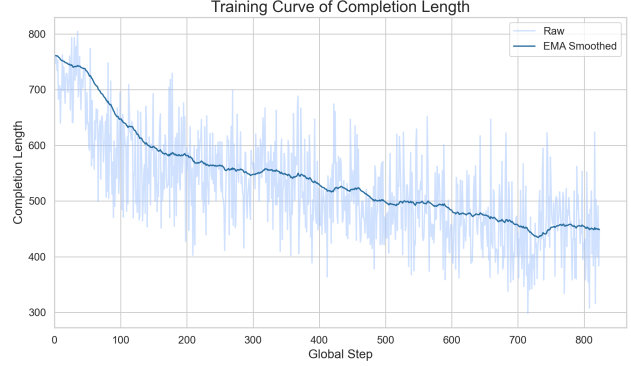


Figure 10: The CoDaPO training curves for completion length on Qwen2.5-Math-1.5B.

Reward curves. We present the reward curve and its standard deviation throughout the training process in Figs. 11 and 12, where Qwen2.5-Math-1.5B is trained on the MATH dataset using CoDaPO. Note that since we adopt accuracy as the sole reward, the reward values are equivalent to the model’s average accuracy. Overall, CoDaPO consistently improves model performance, exhibiting a stable upward trend in accuracy across training steps.

Despite removing the KL regularization term, the training process remains stable, and we do not observe any catastrophic degradation in accuracy—an issue sometimes encountered in RL-based training. Notably, the most significant performance gain occurs within the first 200 steps, which corresponds to our designated warm-up phase, highlighting the early-stage effectiveness of CoDaPO in guiding the model toward better reasoning behaviors.

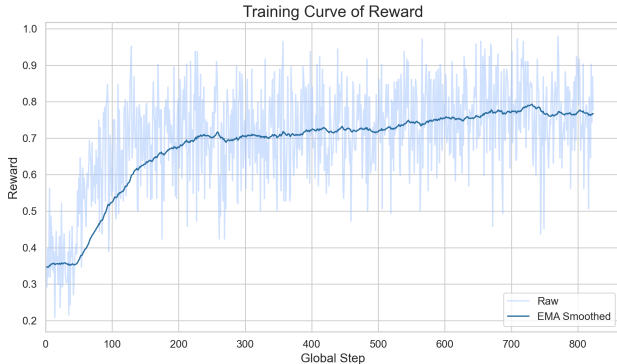


Figure 11: The CoDaPO training curves for reward on Qwen2.5-Math-1.5B.

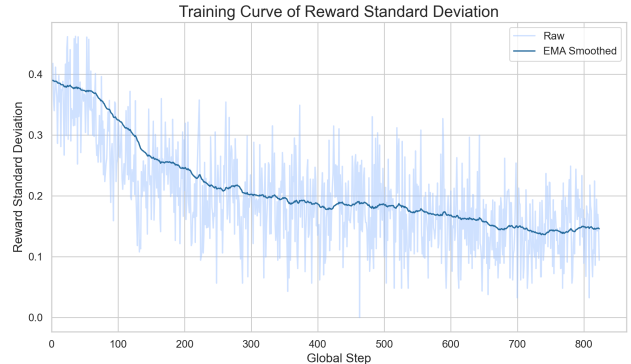


Figure 12: The CoDaPO training curve for reward standard deviation on Qwen2.5-Math-1.5B.

Confidence curves. We present the average confidence curve of model responses, along with separate confidence curves for correct and incorrect responses, in Fig. 13. To compute the confidence for each output, we first calculate the log probability of each token in the response and take the average at the token level. Given the unbounded nature of log probabilities, we exponentiate the average log probability to obtain a normalized confidence score for each response.

From the confidence curves, we observe a clear and consistent gap between correct and incorrect responses throughout the CoDaPO training process: the model assigns significantly higher confidence to correct outputs compared to incorrect ones. This demonstrates that CoDaPO not only improves accuracy, but also enhances the model’s ability to calibrate its confidence

in a more meaningful and discriminative manner.

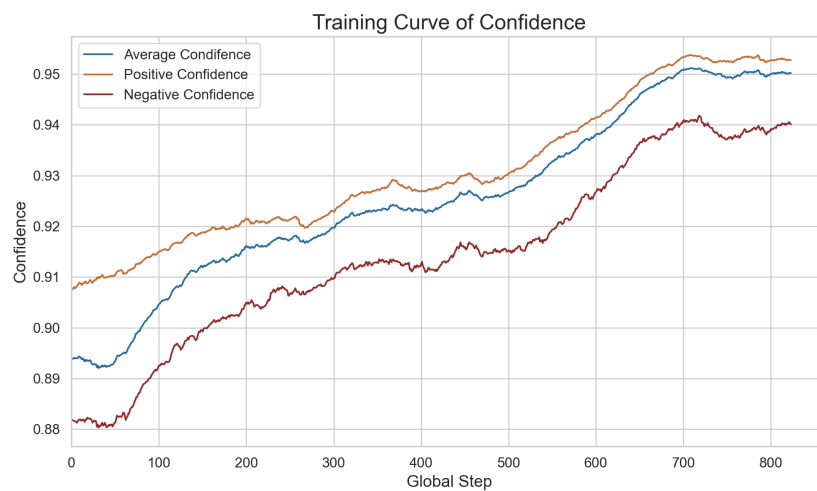
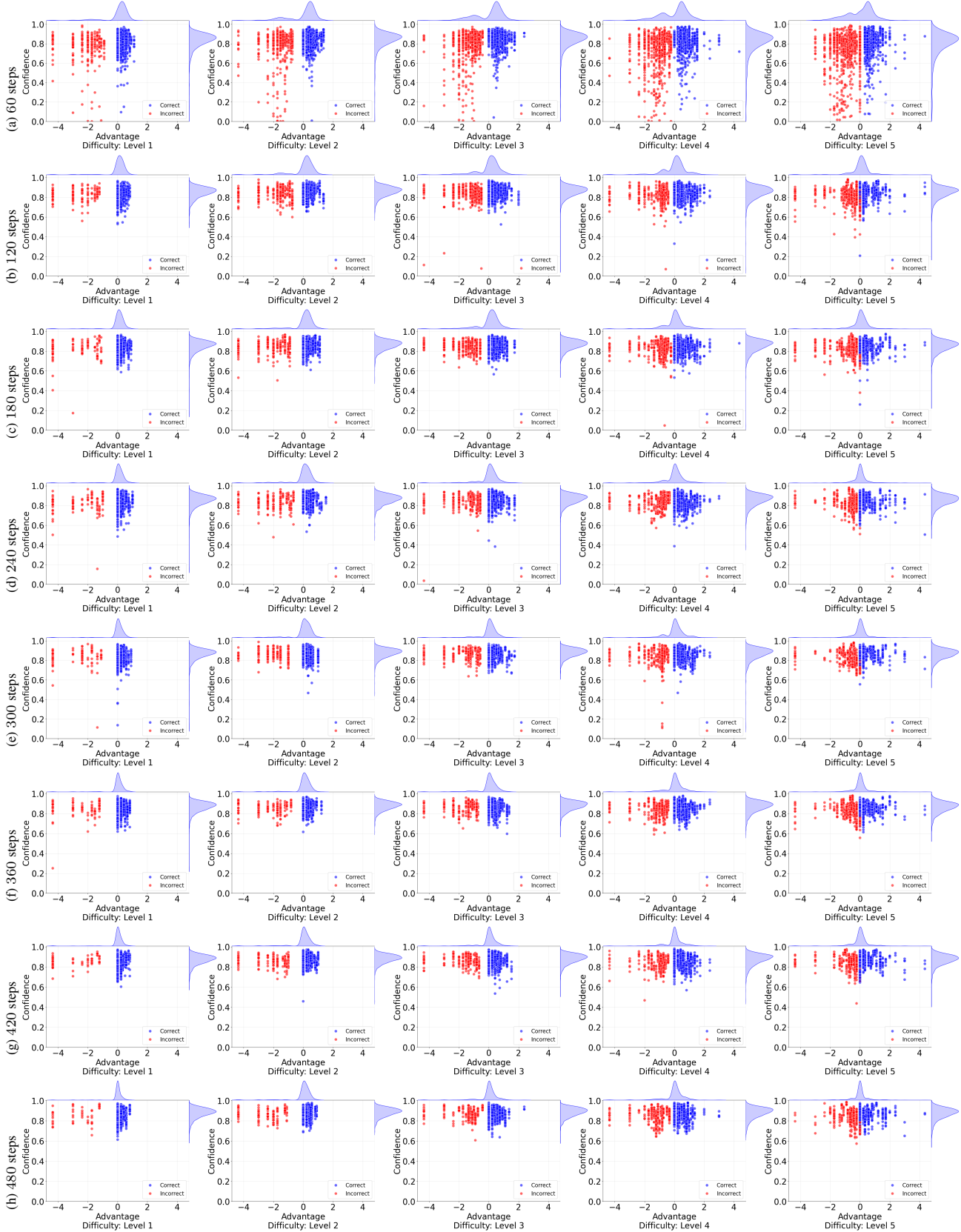


Figure 13: The CoDaPO training curves for response confidence on Qwen2.5-Math-1.5B.

Complete AC planes. We show the full AC planes of CoDaPO and GRPO checkpoints on MATH training and validation sets in Figs. 14–17. For fair comparison, all the experiments are conducted under the same training setup.



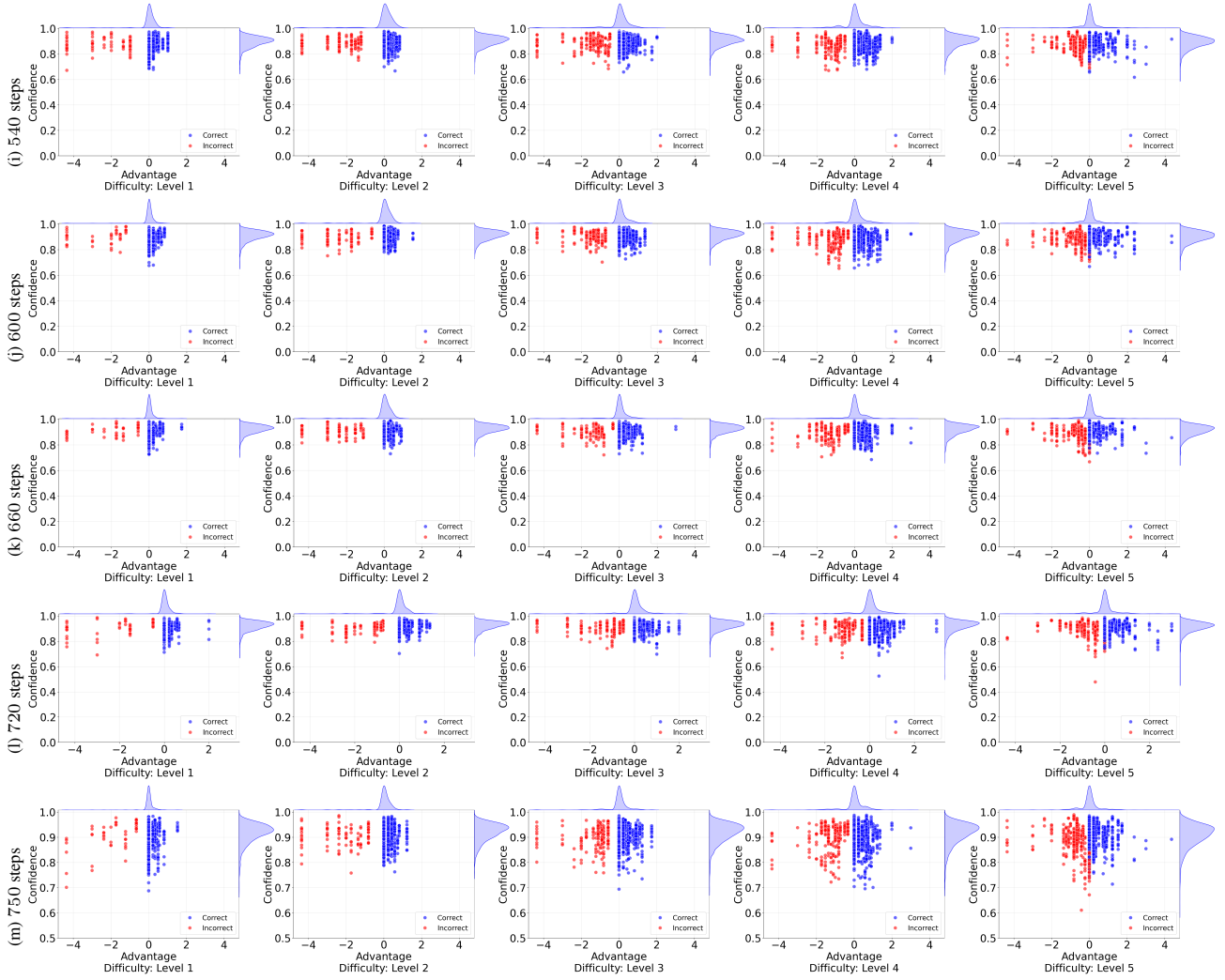
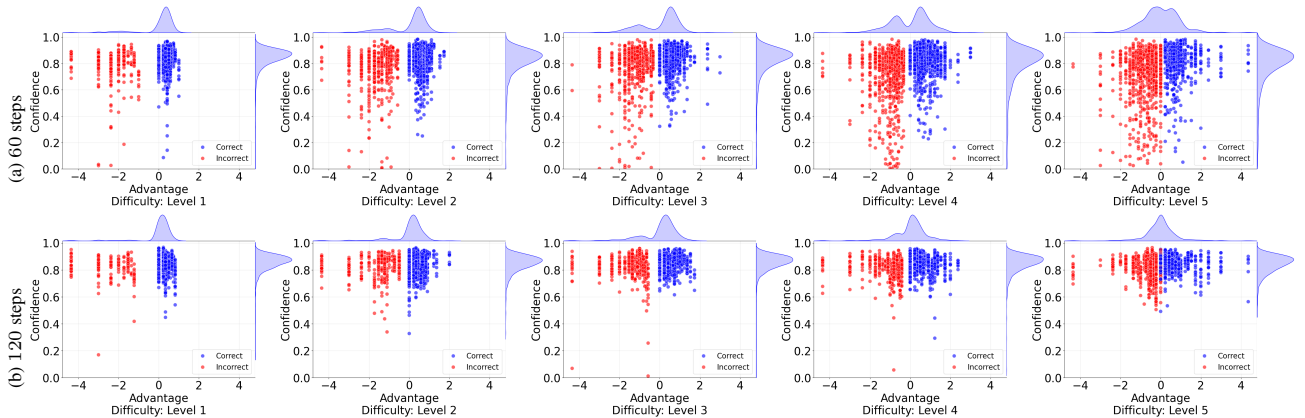
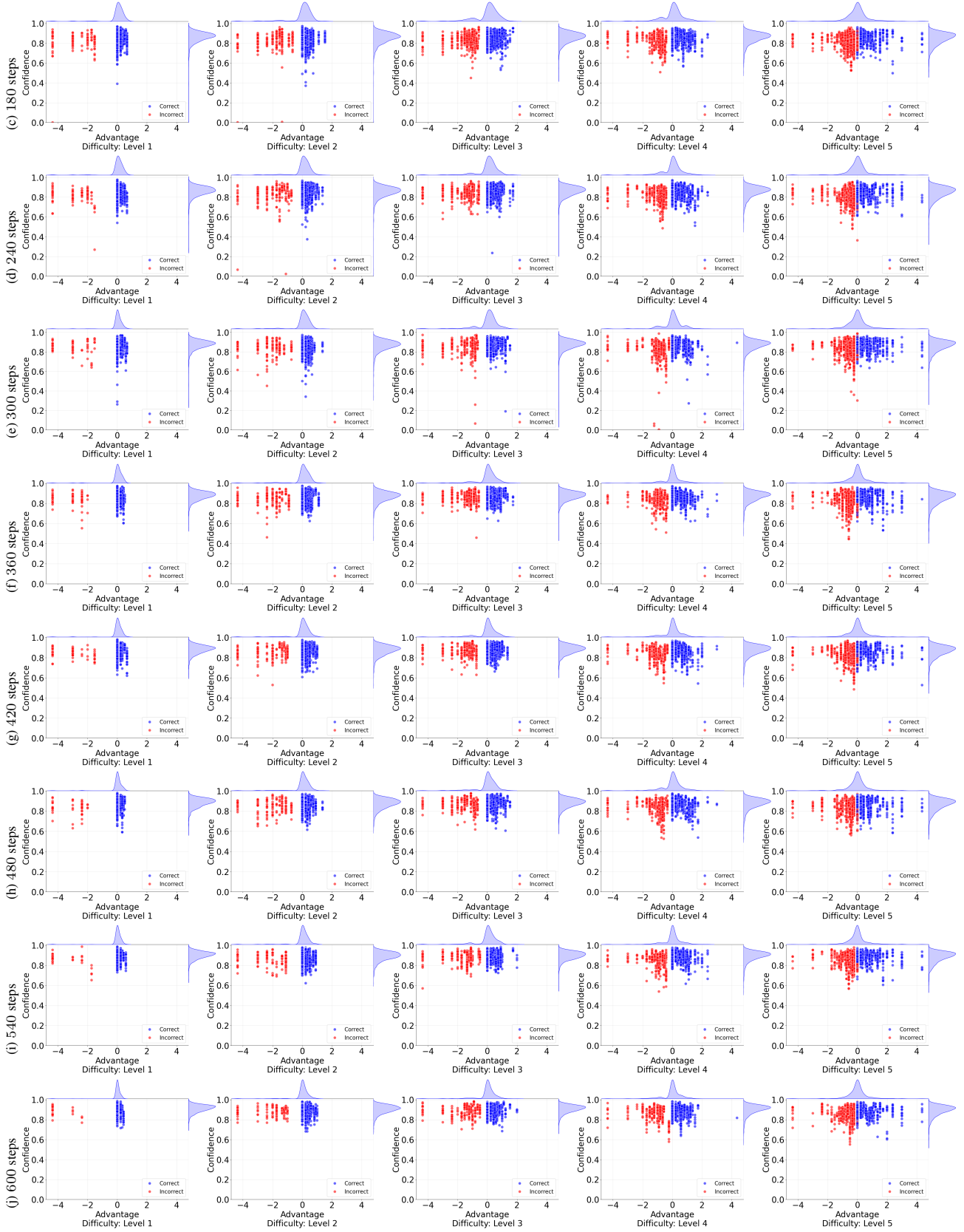


Figure 14: The AC planes of the Qwen2.5-Math-1.5B model, post-training on the MATH dataset with the CoDaPO algorithm. We present the AC planes of model checkpoints captured at every 60 training steps. Columns 1 through 5 correspond to difficulty levels 1 through 5. Additionally, the marginal distributions (*i.e.*, the density distributions) of advantage and confidence across all samples are shown above and to the right of each AC plane.





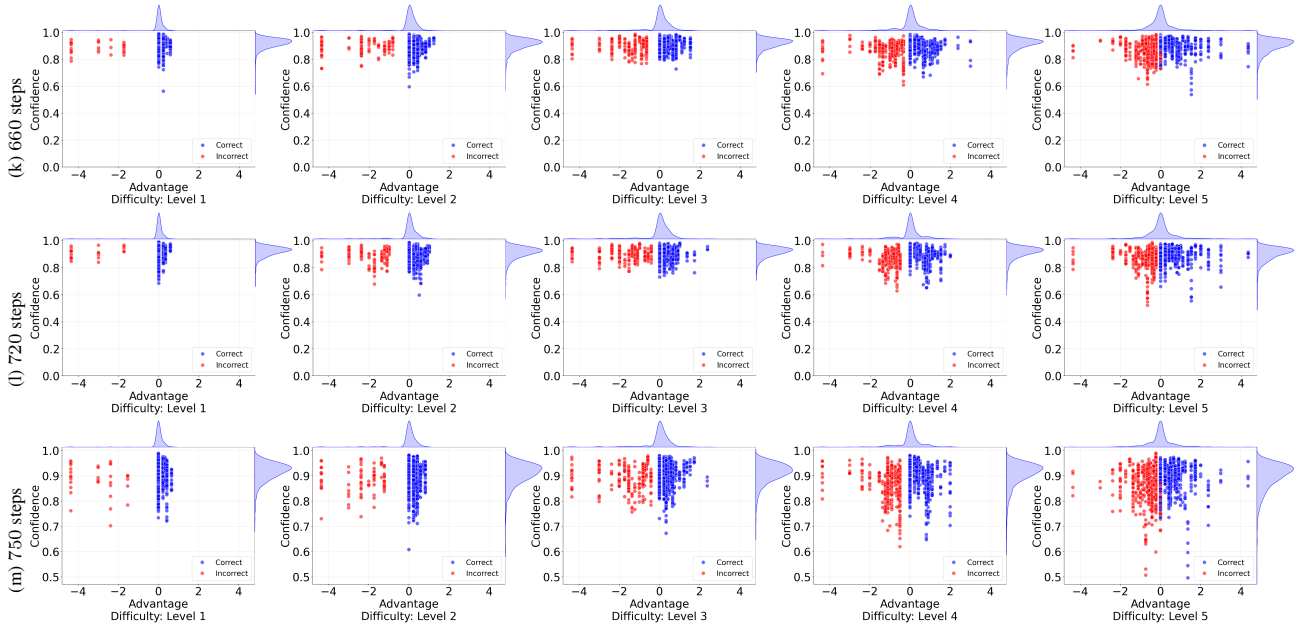
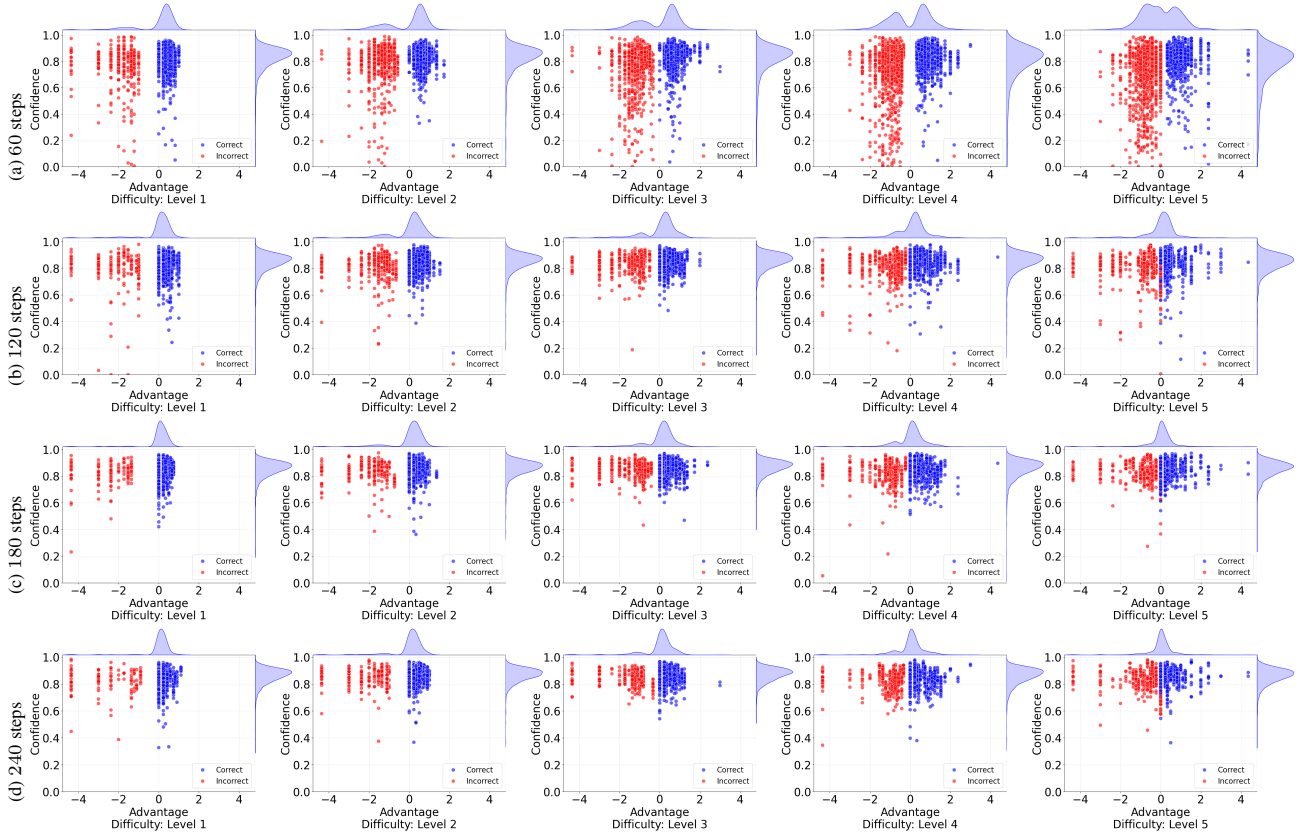


Figure 15: The AC planes of the Qwen2.5-Math-1.5B model with CoDaPO, evaluated on the MATH validation set with the checkpoints captured at every 60 steps (consistent with Fig. 14).



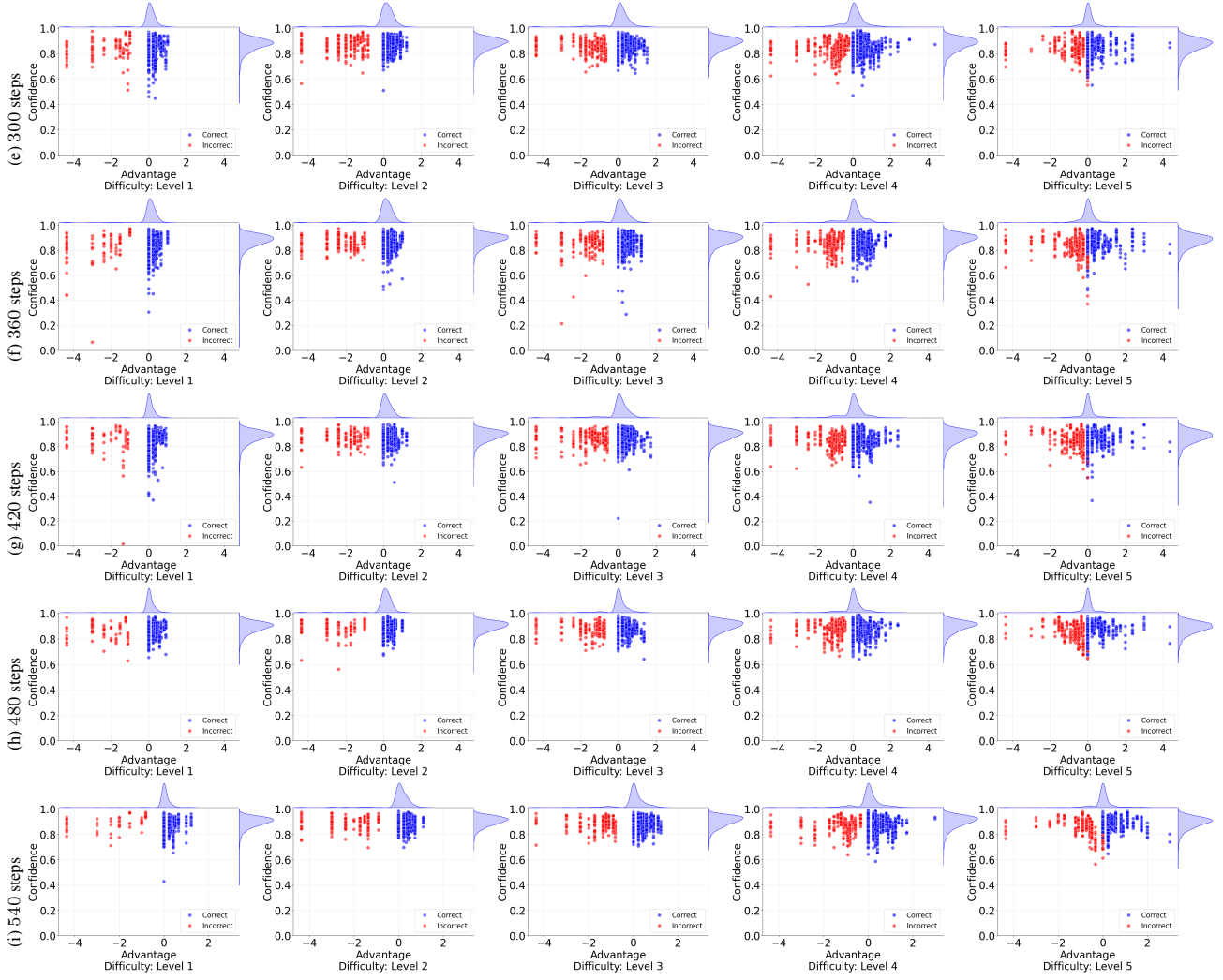
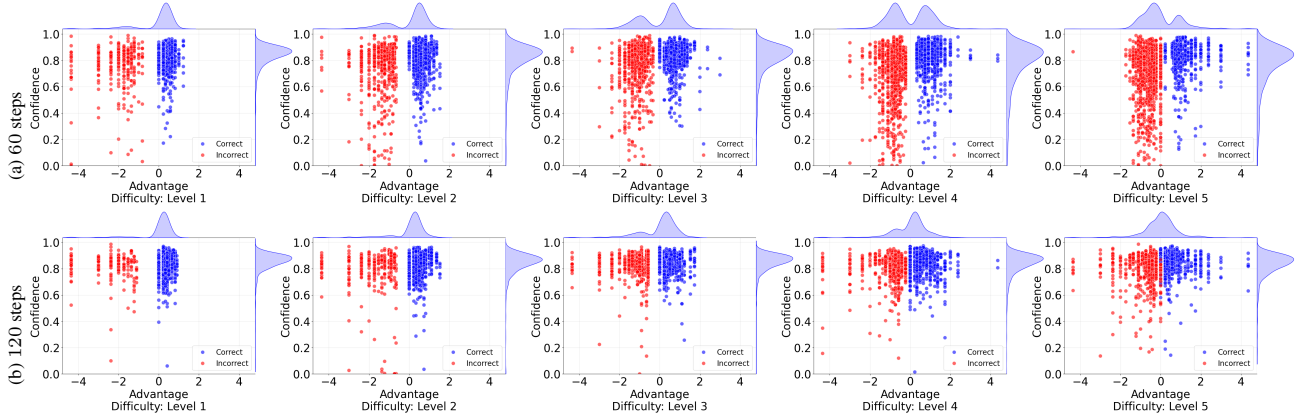


Figure 16: The AC planes of the Qwen2.5-Math-1.5B model, post-training on the MATH dataset with the GRPO algorithm. We present the AC planes of model checkpoints captured at every 60 training steps. Columns 1 through 5 correspond to difficulty levels 1 through 5. Additionally, the marginal distributions (*i.e.*, the density distributions) of advantage and confidence across all samples are shown above and to the right of each AC plane.



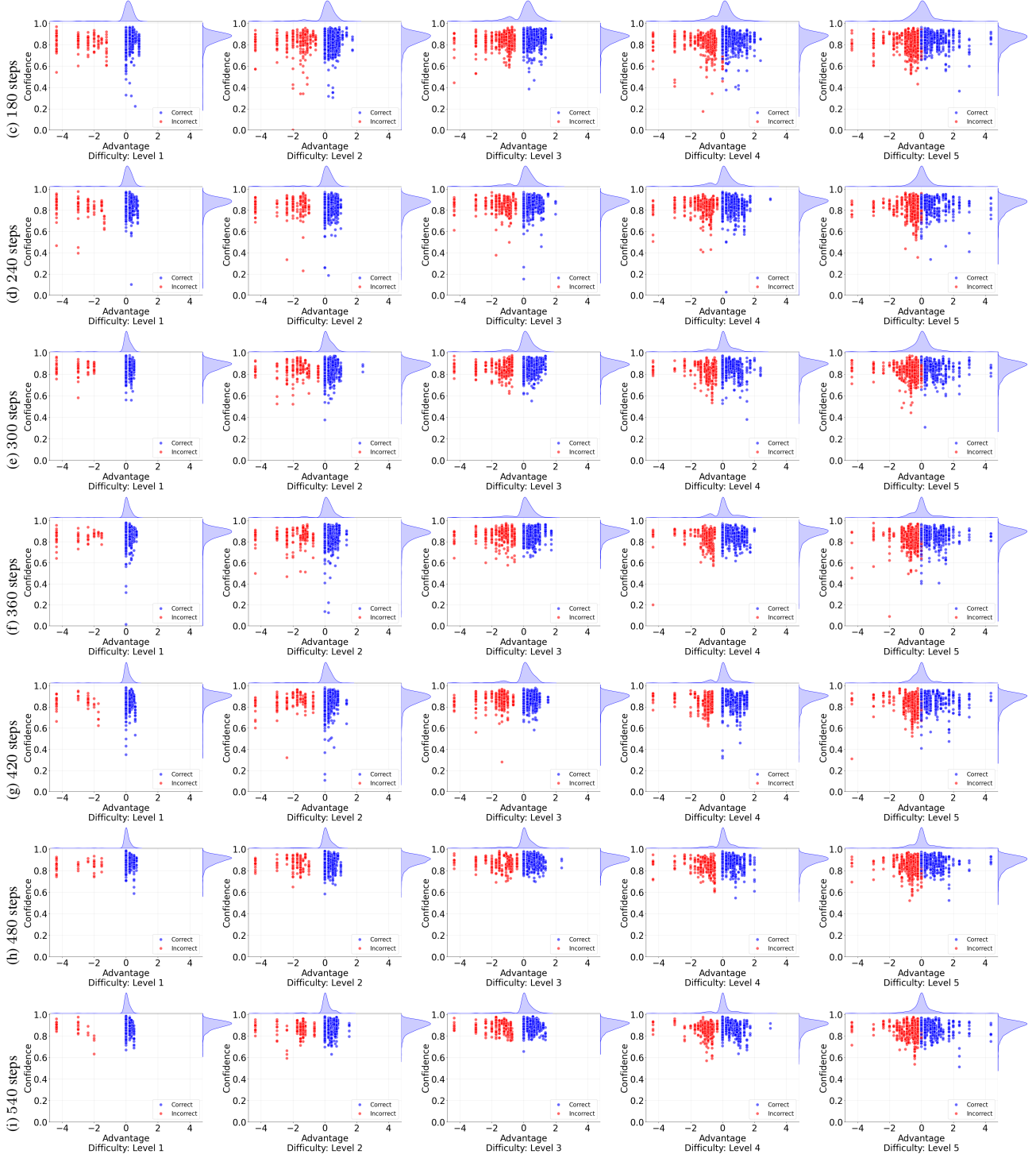


Figure 17: The AC planes of the Qwen2.5-Math-1.5B model with GRPO, evaluated on the MATH validation set with the checkpoints captured at every 60 steps (consistent with Fig. 16).

Benchmarks. We evaluate our proposed method and other baselines on the following diverse benchmarks.

1. **MATH 500 (Hendrycks et al., 2021).** A curated set of 500 challenging problems from the MATH dataset, focusing on high school-level mathematics across algebra, geometry, number theory, and combinatorics.
2. **AIME 2024 ⁴.** A benchmark based on the 2024 American Invitational Mathematics Examination, testing advanced problem-solving skills with 30 short-answer math problems designed for top high school students.
3. **AIME 2025 ⁵.** The 2025 version of the AIME benchmark is similarly structured, providing a fresh set of high-difficulty pre-Olympiad level math problems.
4. **AMC 2023 ⁶.** Based on the 2023 American Mathematics Competitions (AMC 10/12), this benchmark assesses middle-to-advanced high school math across a range of topics in a multiple-choice format.
5. **Olympiad Benchmark (Huang et al., 2024).** A collection of problems from various math olympiads (*e.g.*, USAMO, IMO), aimed at evaluating models on deep mathematical reasoning and multi-step proofs.
6. **Minerva (Lewkowycz et al., 2022).** A benchmark and model suite by Google DeepMind that tackles math and science questions (from grade school to graduate level) using CoT reasoning and LLMs.
7. **GSM8K (Cobbe et al., 2021).** A dataset of 8500 grade-school level math word problems designed to test models’ ability to perform multi-step numerical reasoning in natural language.

Test-time scaling. We present the test-time scaling results in Fig. 18 and 19. Specifically, we report the pass@ k accuracy on AIME25 before and after CoDaPO training, where $k \in \{1, 2, 4, 8, 16, 32, 64, 128\}$. Across all values of k , CoDaPO consistently improves the model’s performance. Notably, for Qwen2.5-Math-1.5B, the pass@8 accuracy after CoDaPO training surpasses the pass@32 accuracy of the model before training, effectively reducing the number of required responses by 24 while achieving better results. This result demonstrates that CoDaPO effectively improves the quality and reliability of model outputs under constrained sampling budgets. The fact that fewer samples are needed to achieve a given level of accuracy indicates that the model becomes more confident and consistent in producing correct answers.

Furthermore, for the instruction-tuned Qwen2.5-1.5B-Instruct model, CoDaPO also leads to substantial improvements in pass@ k accuracy. The performance gain becomes more pronounced as k increases. This indicates that CoDaPO also improves the diversity and calibration of the output distribution. The model is not only better at generating the correct answer in the top few samples but also more likely to include it somewhere in a wider sampling set, which further confirms the effectiveness of CoDaPO in enhancing both single-shot reliability and multi-sample robustness.

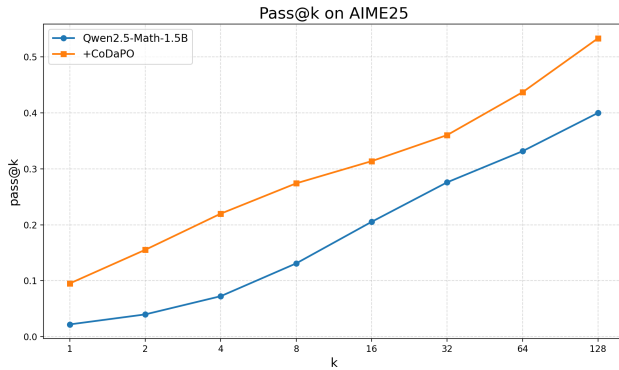


Figure 18: Pass@ k result of Qwen2.5-Math-1.5B with CoDaPO on AIME 2025.

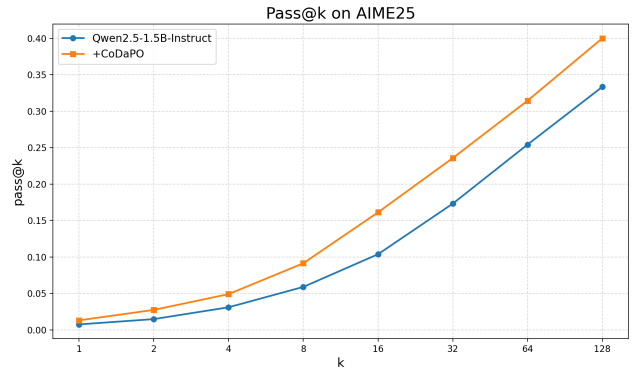


Figure 19: Pass@ k result of Qwen2.5-1.5B-Instruct with CoDaPO on AIME 2025.

⁴https://huggingface.co/datasets/HuggingFaceH4/aime_2024

⁵<https://huggingface.co/datasets/opencompass/AIME2025>

⁶<https://huggingface.co/datasets/math-ai/amc23>

C. Case Studies

In-domain case. We analyze the performance gap between CoDaPO and GRPO through a symbolic reasoning task that requires algebraic manipulation, base conversion, and number-theoretic reasoning in Fig. 20. While both methods adopt similar initial steps, only CoDaPO successfully arrives at the correct and complete solution. This discrepancy reveals a broader insight: CoDaPO demonstrates a stronger capacity for maintaining symbolic consistency and handling algebraic constraints, whereas GRPO is more susceptible to local errors and brittle logic execution.

A key distinction lies in how the two models approach intermediate decision points. CoDaPO tends to preserve the symbolic structure of the problem throughout the reasoning process, producing interpretable and logically coherent derivations. In contrast, GRPO is more prone to heuristic or trial-based reasoning patterns, which may yield superficially plausible but ultimately incorrect results—especially in cases requiring discrete enumeration or careful constraint satisfaction.

This case exemplifies a common challenge in reinforcement learning for language models: small reasoning errors in early steps often cascade into incorrect final answers, and methods lacking robust symbolic understanding struggle to recover. CoDaPO mitigates this through more structured reasoning and better alignment between confidence and difficulty signals, which enhances its robustness in solving multi-step, discrete, and mathematically grounded problems.

These findings suggest that CoDaPO is not only effective in improving accuracy but also in enhancing reasoning fidelity and interpretability, especially in domains like mathematics, programming, and logic that require precise, symbolic manipulation.

Out-of-domain case. In tasks that require translating natural language descriptions into symbolic formulations grounded in domain knowledge—such as those involving physical systems or scientific laws (Fig. 21)—we observe a consistent performance gap between the two methods. CoDaPO demonstrates stronger semantic alignment between the problem description and the constructed solution model. Its responses tend to follow a coherent chain of physical and mathematical reasoning, with clear transitions from qualitative understanding to quantitative execution.

In contrast, GRPO is more prone to semantic drift: it often introduces irrelevant concepts, misapplies formulas, or incorrectly anchors symbolic steps to the problem’s actual objective. This can lead to answers that are numerically plausible but logically invalid, due to disconnects between intermediate computations and the final question. Such behavior suggests difficulties in maintaining global consistency across multi-step derivations—particularly when domain-specific reasoning is required.

More generally, these observations highlight the importance of model capabilities in selective abstraction and structured model-building. An effective llm solver must not only recall relevant formulas or operations, but also identify the appropriate scope of reasoning, exclude irrelevant detail, and maintain logical consistency through multi-step symbolic transformations. CoDaPO exhibits greater stability along these dimensions, making it more reliable on tasks that blend symbolic reasoning with physical interpretation.

AIME25 Question

Find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b .

Ground Truth: 70

CoDaPO Response

To solve this problem, we need to convert the numbers 17_b and 97_b from base b to base 10 and then find the values of b for which 17_b divides 97_b .

1. Convert 17_b to base 10:

$$17_b = 1 \cdot b + 7 = b + 7.$$

2. Convert 97_b to base 10:

$$97_b = 9 \cdot b + 7 = 9b + 7.$$

We need 17_b to be a divisor of 97_b , so:

$$9b + 7 \equiv 0 \pmod{b + 7}.$$

This means:

$$9b + 7 = k(b + 7)$$

for some integer k . Simplifying this, we get:

$$\begin{aligned} 9b + 7 &= kb + 7k \\ 9b - kb &= 7k - 7 \\ b(9 - k) &= 7(k - 1) \end{aligned}$$

So:

$$b = \frac{7(k - 1)}{9 - k}.$$

Since $b > 9$, k must be chosen such that b is an integer greater than 9. Let's find the valid values of k and corresponding b . We check the values of k for which the right-hand side is an integer and $b > 9$:

- For $k = 2$: $b = \frac{7(2-1)}{9-2} = \frac{7}{7} = 1$, not greater than 9.
- For $k = 3$: $b = \frac{7(3-1)}{9-3} = \frac{14}{6} = \frac{7}{3}$, not an integer.
- For $k = 4$: $b = \frac{7(4-1)}{9-4} = \frac{21}{5}$, not an integer.
- For $k = 5$: $b = \frac{7(5-1)}{9-5} = \frac{28}{4} = 7$, not greater than 9.
- For $k = 6$: $b = \frac{7(6-1)}{9-6} = \frac{35}{3}$, not an integer.
- For $k = 7$: $b = \frac{7(7-1)}{9-7} = \frac{42}{2} = 21$, valid.
- For $k = 8$: $b = \frac{7(8-1)}{9-8} = \frac{49}{1} = 49$, valid.
- For $k = 9$: $b = \frac{7(9-1)}{9-9}$ is undefined.
- For $k = 10$: $b = \frac{7(10-1)}{9-10} = \frac{63}{-1} = -63$, not greater than 9.

So the valid values for b are 21 and 49. The sum of these values is:

$$21 + 49 = 70.$$

Thus, the final answer is:

70

GRPO Response

To solve the problem, we need to translate the numbers 17_b and 97_b from base b to base 10. The number 17_b in base 10 is:

$$1 \cdot b + 7 = b + 7$$

The number 97_b in base 10 is:

$$9 \cdot b + 7 = 9b + 7$$

We need to find the values of b such that 17_b (which is $b + 7$) is a divisor of 97_b (which is $9b + 7$). This means that:

$$b + 7 \mid 9b + 7$$

or equivalently:

$$9b + 7 \equiv 0 \pmod{b + 7}$$

We can express this congruence as an equation:

$$9b + 7 = k(b + 7)$$

for some integer k . Rearranging the equation:

$$\begin{aligned} 9b + 7 &= kb + 7k \\ 9b - kb &= 7k - 7 \\ b(9 - k) &= 7(k - 1) \\ b &= \frac{7(k - 1)}{9 - k} \end{aligned}$$

For b to be an integer greater than 9, the right-hand side must be an integer. This means $7(k - 1)$ must be divisible by $9 - k$. We can iterate through possible integer values of k to find valid solutions for b :

```
# Python code to find valid bases
valid_bases = []
for k in range(10): # Try k from 0 to 9
    if (9 - k) != 0:
        b = 7 * (k - 1) / (9 - k)
        if b.is_integer() and b > 9:
            valid_bases.append(int(b))
```

```
sum_of_bases = sum(valid_bases)
print((valid_bases, sum_of_bases))
```

Running this code yields:

$$\text{valid bases: } [16, 21] \Rightarrow \text{sum} = 16 + 21 = \boxed{37}$$

Therefore, the final answer is:

37

Figure 20: Case study of CoDaPO and GRPO Responses to an AIME 2025 Question.

Minerva Question

An eclipsing binary consists of two stars of different radii and effective temperatures. Star 1 has radius R_1 and T_1 , and Star 2 has $R_2 = 0.5R_1$ and $T_2 = 2T_1$. Find the change in bolometric magnitude of the binary, Δm_{bol} , when the smaller star is behind the larger star. (Consider only bolometric magnitudes so you don't have to worry about color differences.)

Ground Truth: 1.75

CoDaPO Response

To solve this problem, we need to calculate the change in bolometric magnitude of the binary system when the smaller star (Star 2) is behind the larger star (Star 1). The bolometric magnitude of a star is given by:

$$m_{\text{bol}} = -2.5 \log_{10}(L_{\text{bol}})$$

where L_{bol} is the bolometric luminosity of the star. The bolometric luminosity of a star is proportional to its surface area and its effective temperature:

$$L_{\text{bol}} \propto R^2 T^4$$

So, the bolometric luminosities of Star 1 and Star 2 are:

$$\begin{aligned} L_{\text{bol},1} &\propto R_1^2 T_1^4 \\ L_{\text{bol},2} &\propto R_2^2 T_2^4 = (0.5R_1)^2 (2T_1)^4 \\ &= 0.25R_1^2 \cdot 16T_1^4 = 4R_1^2 T_1^4 \end{aligned}$$

The total bolometric luminosity of the binary system when both stars are visible is:

$$\begin{aligned} L_{\text{bol,total}} &= L_{\text{bol},1} + L_{\text{bol},2} = kR_1^2 T_1^4 + k \cdot 4R_1^2 T_1^4 \\ &= 5kR_1^2 T_1^4 \end{aligned}$$

When Star 2 is behind Star 1, the total bolometric luminosity is just the luminosity of Star 1:

$$L_{\text{bol,total}} = L_{\text{bol},1} = kR_1^2 T_1^4$$

The change in bolometric magnitude is given by:

$$\begin{aligned} \Delta m_{\text{bol}} &= m_{\text{bol,after}} - m_{\text{bol,before}} \\ &= -2.5 \log_{10} \left(\frac{L_{\text{bol,total,after}}}{L_{\text{bol,total,before}}} \right) \\ &= -2.5 \log_{10} \left(\frac{kR_1^2 T_1^4}{5kR_1^2 T_1^4} \right) \\ &= -2.5 \log_{10} \left(\frac{1}{5} \right) \\ &= -2.5 \log_{10}(5^{-1}) \\ &= -2.5(-\log_{10}(5)) \\ &= 2.5 \log_{10}(5) \end{aligned}$$

Using the value $\log_{10}(5) \approx 0.6990$, we get:

$$\Delta m_{\text{bol}} = 2.5 \times 0.6990 = 1.7475 \approx 1.75$$

Thus, the change in bolometric magnitude is:

$$\boxed{1.75}$$

GRPO Response

To find the change in bolometric magnitude of the binary when the smaller star (Star 2) is behind the larger star (Star 1), we need to consider the effect of the limb darkening and the phase angle of the eclipsing binary.

The bolometric magnitude of a star can be approximated using the Stefan-Boltzmann law, which states that the luminosity of a star is proportional to the fourth power of its temperature and the square of its radius. The bolometric magnitude m_B of a star is related to its luminosity by:

$$m_B = -2.5 \log_{10}(L) + C$$

where L is the luminosity of the star and C is a constant. For a binary system, the change in bolometric magnitude when one star passes in front of the other can be approximated by considering the change in the effective area of the star that is visible during the eclipse.

The effective area of a star that is visible during an eclipse is given by:

$$A = R^2 \left(1 - \left(\frac{R}{d} \right)^2 \sin^2(\theta) \right)$$

where R is the radius of the star, d is the distance to the star, and θ is the phase angle. The change in the effective area when Star 2 is behind Star 1 is given by the difference in the areas of the two stars:

$$\Delta A = \pi R_1^2 - \pi R_2^2 \left(1 - \left(\frac{R_2}{d} \right)^2 \sin^2(\theta) \right)$$

Since we are considering only the bolometric magnitude, we can simplify the problem by considering the ratio of the luminosities of the two stars. The change in bolometric magnitude is given by:

$$\Delta m_{\text{bol}} = -2.5 \log_{10} \left(\frac{L_2}{L_1} \right)$$

where L_1 and L_2 are the luminosities of Star 1 and Star 2, respectively. Using the Stefan-Boltzmann law, we have:

$$\begin{aligned} \frac{L_2}{L_1} &= \left(\frac{T_2}{T_1} \right)^4 \left(\frac{R_2}{R_1} \right)^2 = \left(\frac{2T_1}{T_1} \right)^4 \left(\frac{0.5R_1}{R_1} \right)^2 \\ &= 2^4 \cdot 0.5^2 = 16 \cdot 0.25 = 4 \end{aligned}$$

Thus, the change in bolometric magnitude is:

$$\begin{aligned} \Delta m_{\text{bol}} &= -2.5 \log_{10}(4) = -2.5 \cdot 0.60206 \\ &= -1.50515 \approx -1.51 \end{aligned}$$

Therefore, the change in bolometric magnitude of the binary when the smaller star is behind the larger star is:

$$\boxed{-1.51}$$

Figure 21: Case study of CoDaPO and GRPO Responses to a Minerva Question.