GOLDILOCKS RL: TUNING TASK DIFFICULTY TO ESCAPE SPARSE REWARDS FOR REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning has emerged as a powerful paradigm for unlocking reasoning capabilities in language models. However, relying on sparse rewards makes this process highly sample-inefficient, as models must navigate vast search spaces with minimal feedback. While classic curriculum learning aims to mitigate this by ordering data based on complexity, prior works have primarily targeted small datasets and do not directly transfer to the large-scale settings typical of modern language model training. Furthermore, the right ordering for a specific model is often unclear. To address this, we propose Goldilocks, an adaptive data-selection strategy that uses a Selector network to predict the standard deviation of rewards across the model's rollouts for each candidate question. The Selector prioritizes questions with high predicted reward variability, corresponding to questions that are neither too easy nor too hard for the model's current capabilities (Goldilocks principle), while training the model with GRPO. By leveraging the model's performance on seen samples, the Selector continuously adapts to the model's evolving abilities. Across the OpenMathReasoning and Polaris datasets, Goldilocks consistently improves over standard GRPO, requiring up to 78% fewer optimization steps to reach the corresponding GRPO performance.

1 INTRODUCTION

The reasoning capabilities of Transformer-based models have been a focal point of recent research, with Reinforcement Learning (RL) emerging as a powerful paradigm for enhancing these abilities. Recent RL methods incentivize the generation of a detailed Chain of Thought (CoT) (Wei et al., 2022), enabling models to solve complex problems (Zelikman et al., 2022; Guo et al., 2025a; Hou et al., 2025). Additionally, increasing the computational budget at inference time, known as scaling test-time compute, has been shown to further enhance these reasoning capabilities (Snell et al., 2024). While RL has proven effective in unlocking complex reasoning behaviors, the computational efficiency of these methods remains a significant challenge. Recent findings by Khatri et al. (2026) underscore that scaling is a fundamental driver of performance. This observation suggests that while models have the potential for continued improvement, realizing this potential through brute-force scaling is resource-intensive, making training efficiency a paramount concern.

Reinforcement learning with verifiable rewards (RLVR) (Cobbe et al., 2021; Shao et al., 2024) provides a scalable way to improve reasoning by assigning rewards using automatically checkable outcomes. However, when the reward is determined primarily by final-answer correctness, the resulting feedback is sparse, since a generated trajectory is primarily judged by whether it reaches a correct final answer: a generated trajectory is rewarded if it reaches a correct answer. This creates a substantial efficiency challenge during training. Questions that are far beyond the model's current capabilities are likely to produce groups of uniformly incorrect rollouts, while questions that are already mastered tend to produce uniformly correct ones. In either case, substantial generation compute may be spent on rollout groups that provide little or no within-group reward variation, and therefore little useful relative signal for the GRPO update.

This inefficiency becomes increasingly important as language-model training moves toward larger data and compute regimes. Scaling studies have established the importance of jointly increasing model capacity, data, and compute (Hoffmann et al., 2022), while recent frontier models are trained on trillions of tokens (DeepSeek-AI, 2024). Reasoning post-training is following a similar trend, with

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

recent systems relying on increasingly large-scale reinforcement learning (Guo et al., 2025b; Yu et al., 2025; Khatri et al., 2026). Reasoning datasets are also growing to millions of questions (Moshkov et al., 2025). In such a large-pool regime, an effective data-selection strategy cannot assume that the model has already interacted with every candidate question. Instead, it should be able to estimate the learning value of previously unseen questions from limited model feedback, and ideally make this decision *before* incurring the cost of generating rollouts.

Curriculum Learning (CL) (Bengio et al., 2009) provides a natural framework for prioritizing training examples, but many existing approaches rely on information that becomes difficult to obtain in this regime. History-based methods estimate the difficulty or utility of a question from the model's previous performance on that same instance (Qu et al., 2026; Shen et al., 2026); such estimates become informative only after sufficient observations of the question have been collected. Other approaches schedule predefined difficulty categories or data partitions (Parashar et al., 2025; Chen et al., 2025), requiring structure that may not be available for a large heterogeneous question pool. A different line of work identifies uninformative examples only after generating their rollouts and then skips the corresponding policy update (Wang et al., 2025; Xiong et al., 2025); this avoids an unhelpful optimization step, but not the generation cost already incurred. This motivates a curriculum that can infer the utility of previously unseen questions from recent model feedback, make the selection before rollout generation, and adapt as the model improves.

In this work, we propose **Goldilocks**, a pre-rollout data-selection framework that uses a learned Selector. Alongside model RL training, a lightweight Selector network learns from recent model outcomes to predict the reward variability of candidate questions. It then prioritizes questions with high reward variability that are expected to produce informative rollout groups (neither those already mastered by the model nor those currently beyond its capabilities). Because the prediction is made from the question itself, the Selector can generalize to candidates the model has never encountered and continuously update its curriculum as the model improves. We show theoretically that this criterion is aligned with the learning signal of GRPO (Shao et al., 2024) under binary verifiable rewards, and empirically evaluate it across large- and small-pool training regimes against predictive and history-based curriculum baselines.

The paper is organized as follows. Section 2 discusses related work. Section 3 analyzes GRPO gradient dynamics under verifiable rewards. Section 4 presents our joint training framework. Sections 5 and 7 report empirical results and ablations and Section 8 concludes.

2 Related Work

Reasoning and Reinforcement Learning. Intermediate reasoning traces have been shown to substantially improve language-model performance on complex tasks, through supervised scratchpads (Nye et al., 2021), Chain-of-Thought prompting (Wei et al., 2022; Kojima et al., 2022), and reinforcement learning (Shao et al., 2024). Recent RL methods for reasoning optimize such trajectories using verifiable rewards, but their efficiency is limited by the large generation cost and by questions that provide little useful learning signal. This has motivated methods that improve which problems are selected for training or how rollout compute is allocated.

History-Based Curricula and Adaptive Sampling. A large class of curriculum methods estimates question utility from previous interactions with the same question. MoPPS (Qu et al., 2026) maintains prompt-specific estimates of model success, while BOTS (Shen et al., 2026) similarly uses accumulated evidence to guide online task selection. Such history-based methods are most effective when questions are revisited frequently enough for their estimates to become informative. We therefore include MoPPS as a baseline in our smaller-dataset setting, where repeated exposure provides a favorable regime for question-specific histories. Other approaches operate over predefined data categories (Chen et al., 2025), adapt rollout allocation (Li et al., 2025), or identify uninformative samples only after generation (Wang et al., 2025; Xiong et al., 2025; Foster et al., 2026); these improve training efficiency but do not directly address predicting the utility of previously unseen questions before rollout.

Predictive Pre-Rollout Selection. The closest prior work to Goldilocks is Prompt Curriculum Learning (PCL) (Gao et al., 2026), which, among the methods we consider, is the only approach that learns a prompt-conditioned predictor from model feedback and uses it to score unseen questions

before generating model rollouts. PCL predicts the model's probability of success and selects questions near a target difficulty, whereas Goldilocks directly predicts the reward variability of the rollout group, the quantity motivated by our GRPO analysis. PCL is evaluated on substantially smaller question pools than the large-scale regime considered in our main experiments; we therefore use it as our primary predictive baseline to test whether this form of pre-rollout generalization remains effective as the training pool grows.

3 Theoretical Motivation: The Goldilocks Zone

In the context of fine-tuning language models with RL, the objective is to maximize the expected reward on a dataset of verifiable answers. The standard policy gradient loss is defined as $\mathcal{L}_{PG} = -\mathbb{E}_{s_t,a_t}\left[\log \pi_\theta(a_t|s_t)\hat{A}_t\right]$, where π_θ denotes the policy and $\hat{A}_t$ represents the estimated advantage. In the context of updating language models, the state s_t corresponds to the prompt q combined with the history of generated tokens $o_{<t}$, and the action a_t is the next token o_t. The objective is defined over the generated sequence of length T:

$$\mathcal{L}_{\text{PG}}(\theta) = -\mathbb{E}_{q\sim\mathcal{D},o\sim\pi_\theta}\left[\sum_{t=1}^{T}\log \pi_\theta(o_t \mid q, o_{<t})\hat{A}_t\right],$$

where $\hat{A}_t$ represents the advantage estimated at token step t.

In GRPO, the advantage is computed by normalizing rewards across a group of G outputs sampled for a single prompt. In the RLVR setting, the reward is assigned at the sequence level rather than per token, meaning all tokens within an output o share the same advantage estimate. Thus $\hat{A}_t$ is constant across token steps t for a given question q and output o, and we denote it $\hat{A}_{q,o}$. We consider a binary verification reward $r_{\text{ver}} = 1$ for a correct final answer and $r_{\text{ver}} = 0$ otherwise.

Advantage computation. The verification reward follows a Bernoulli distribution: $r_{\text{ver}} \sim \text{Bernoulli}(p_q)$. The GRPO algorithm computes the advantage $\hat{A}_{q,o}$ by standardizing the reward:

$$\hat{A}_{q,o} = \frac{r_{\text{ver}} - \text{mean}(r_{\text{ver}})}{\text{std}(r_{\text{ver}})}.$$

Since $r_{\text{ver}} \sim \text{Bernoulli}(p_q)$, the moments are $\text{mean}(r_{\text{ver}}) = p_q$ and $\text{std}(r_{\text{ver}}) = \sqrt{p_q(1-p_q)}$. Substituting these values, the advantage $\hat{A}_{q,o}$ takes on two distinct values depending on the correctness of the output[1]:

- **Correct answer** (probability p_q): $\hat{A}_{q,o} = \dfrac{1-p_q}{\sqrt{p_q(1-p_q)}}$
- **Incorrect answer** (probability $1-p_q$): $\hat{A}_{q,o} = \dfrac{-p_q}{\sqrt{p_q(1-p_q)}}$

Zero-variance batches. When all G rollouts for a prompt q are simultaneously correct or simultaneously incorrect, every rollout receives the same reward, all advantages $\hat{A}_{q,o}$ are zero. The gradient contribution of the entire batch therefore vanishes and no parameter update occurs. The probability of this zero-variance event is

$$\mathbb{P}(\text{zero variance}) = p_q^G + (1-p_q)^G, \tag{1}$$

which is minimized at $p_q = 1/2$ and approaches 1 as $p_q \to 0$ or $p_q \to 1$. Therefore, prompts near the extremes waste an entire group of rollouts with high probability, regardless of model capacity or training budget.

[1]This analysis extends to any reward of the form $r_{\text{total}} = r_{\text{ver}} + f(t)$, where $f(t)$ converges to a constant during training. See Appendix B for a formal proof.

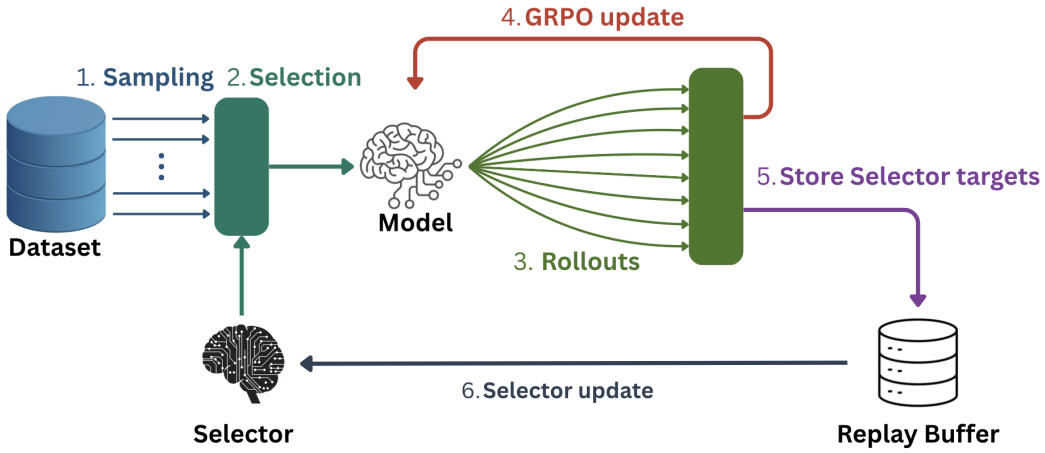


Figure 1: **Overview of the Goldilocks Framework.** The continuous training loop.

Gradient norm decomposition. We have shown that substituting these advantage values into the policy gradient objective and separating the contributions of correct and incorrect outcomes gives

$$\|\nabla_\theta L_{PG}\| = \sqrt{p_q(1-p_q)} \left\| \mathbb{E}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(o_t \mid q, o_{<t}) \,\middle|\, r_{\text{ver}} = 1 \right] - \mathbb{E}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(o_t \mid q, o_{<t}) \,\middle|\, r_{\text{ver}} = 0 \right] \right\|. \quad (2)$$

The proof is given in Appendix C.

Vanishing signal. Equation 2 isolates a $\sqrt{p_q(1-p_q)}$ prefactor in the gradient norm. Since the remaining conditional-expectation term is itself question- and model-dependent, this factor alone does not determine the gradient magnitude. We therefore make its role precise by deriving a uniform upper bound. Under the additional assumption that the per-token gradient $\|\nabla_\theta \log \pi_\theta(o_t \mid q, o_{<t})\|$ is uniformly bounded by some constant B and the context length T is finite, this scaling translates into the upper bound

$$\|\nabla_\theta L_{PG}\| \;\le\; 2BT\sqrt{p_q(1-p_q)}, \quad (3)$$

where $p_q = p_q(\theta)$ is the verification probability at the current parameters. Eq. 3 shows that the gradient norm is structurally bounded by $\sqrt{p_q(1-p_q)}$, with the constant $2BT$ uniform across all prompts at a fixed checkpoint. Therefore, as $p_q \to 0$ or $p_q \to 1$, the gradient norm $\|\nabla_\theta \mathcal{L}_{\text{PG}}\| \to 0$, leading to vanishing gradients. The formal proof is given in Appendix C.

Taken together, Eq. 1 and Eq. 3 give two independent reasons why prompts near $p_q \in \{0, 1\}$ are wasteful: they suppress the gradient magnitude and are overwhelmingly likely to produce zero-variance batches. Conversely, prompts near $p_q = 1/2$ minimize the probability of zero-variance batches (Eq. 1) and maximize the structural upper bound on the gradient norm (Eq. 3). This motivates the Goldilocks Selector: select prompts at the edge of the model's current solvability, where $\sqrt{p_q(1-p_q)}$ is largest. We emphasize that the analysis is not intended as a complete theoretical account of RL dynamics, but rather as a simplified intuition for why intermediate-difficulty questions can provide more informative GRPO updates.

4 Joint Training Procedure

We now detail the overall training loop, illustrated in Figure 1. The process operates in a cycle that alternates between data selection, model optimization via GRPO, and continuous Selector refinement.

4.1 Data Selection Policy (Steps 1-2)

During the data selection phase, the Selector samples a candidate pool, denoted as $\mathcal{C}$, consisting of $K_{\text{candidate}}$ examples drawn randomly from the dataset. For each candidate question $q \in \mathcal{C}$, the Selector computes a predicted utility score $\hat{v}_q = f_\phi(q)$. To balance exploration and exploitation, the final training sample q^* is selected using an ϵ-greedy strategy (Algorithm 1):

- With probability ϵ, a question is selected uniformly at random from $\mathcal{C}$ to ensure diverse data coverage.
- With probability $1 - \epsilon$, the question with the highest predicted utility is selected deterministically, i.e.,

$$q^* = \arg\max_{q \in \mathcal{C}} \hat{v}_q.$$

4.2 Model Rollouts and Optimization (Steps 3-4)

Once the prompt q^* is selected, the model generates a group of G rollouts, $\{o_1, \ldots, o_G\}$. The verification reward r_{ver} is computed for each output, and the advantages are estimated using the group-relative normalization described in Section 3. Based on these advantages, the gradient is computed. To ensure optimization stability, gradients are accumulated over a batch of selected prompts before a parameter update is applied to the model.

4.3 Selector Refinement (Steps 5-6)

The Selector is trained online to predict the learning potential of specific prompts based on the model's real-time feedback. Following the model's rollouts for a question q, we calculate the empirical success rate $\hat{p}_q$ as the fraction of correct solutions within the generated group. The regression target y_q is then set to the empirical standard deviation of the verification reward $y_q = \sqrt{\hat{p}_q(1 - \hat{p}_q)}$. This tuple (q, y_q) is pushed into a replay buffer $\mathcal{D}_{\text{replay}}$ of capacity N_{replay}, which operates as a sliding window to retain the most recent interaction data.

The Selector updates its parameters ϕ periodically as detailed in Algorithm 2. After every M_{update} samples processed by the model, the Selector undergoes a dedicated training phase lasting E_{Selector} epochs. During this phase, we iterate over the entire replay buffer $\mathcal{D}_{\text{replay}}$, processing the data in mini-batches $\mathcal{B}$ to minimize the Mean Squared Error:

$$\mathcal{L}_{\mathcal{B}}(\phi) = \frac{1}{|\mathcal{B}|} \sum_{(q, y_q) \in \mathcal{B}} \left(f_\phi(q) - y_q \right)^2 .$$

This objective ensures the Selector continuously aligns its predictions with the model's capabilities.

5 Experiments and Results

To evaluate Goldilocks, we primarily focus on the large-scale **OpenMathReasoning** dataset (Moshkov et al., 2025), which comprises over 3 million chain-of-thought problems that motivates our setting. We compare Goldilocks against standard GRPO and Prompt Curriculum Learning (PCL) (Gao et al., 2026), our closest predictive pre-rollout baseline. Given the substantial computational cost of long-run GRPO training on OpenMathReasoning, we prioritize extensive training on this benchmark rather than broad evaluation across many datasets. As a complementary smaller-scale setting, we additionally train and evaluate on **Polaris** (An et al., 2025), which contains approximately 50,000 questions, where questions are revisited more frequently. In this regime, we compare against both PCL and MoPPS (Qu et al., 2026), allowing us to contrast Goldilocks with a history-based curriculum under conditions where question-specific histories are more informative.

Across these settings, we evaluate five models spanning different model families and sizes: OLMo2-1B (OLMo et al., 2024), Qwen2.5-1.5B (Yang et al., 2024), Qwen3-4B-Base (Yang et al., 2025), Phi-4-mini-Instruct (4B) (Abdin et al., 2024), and Qwen2.5-7B (Yang et al., 2024). Following common practice in RL-based reasoning research (Shao et al., 2024; Amani et al., 2025; Chen et al., 2025), we focus on mathematical reasoning and train all models with GRPO. All experiments

are implemented using the TRL library (von Werra et al., 2020). Additional dataset, training, and compute details are provided in Appendices D and E.

5.1 Experimental Setup

For each model–dataset pair, we first tune the GRPO learning rate and global batch size, and use the best-performing configuration as the reference for all subsequent comparisons. We then evaluate Goldilocks and the corresponding baselines using the selected settings. All methods use a rollout group size of $G = 16$ in the main experiments, with additional ablations evaluating sensitivity to alternative group sizes.

For all experiments, we use Qwen3-0.6B (Yang et al., 2025) as the Selector. In all experiments, we utilized a fixed computational budget of 8 GPUs. Vanilla GRPO uses all eight GPUs for model training, whereas Goldilocks and PCL reserve one GPU for the Selector and use the remaining seven for the model. Accordingly, if the tuned GRPO global batch size is B, we use a model batch size of $\frac{7}{8}B$ for Goldilocks and PCL, while keeping the per-GPU model configuration unchanged. This provides a matched hardware budget and explicitly accounts for the GPU required by the learned curriculum predictor rather than giving Selector-based methods additional compute. MoPPS, which does not require a Selector network, follows the corresponding model's GRPO configuration. We justify this scheme in Appendix E.1.

6 Results

Our primary experiments on OpenMathReasoning use a long-run training regime to reflect the scale of the dataset and evaluate whether the benefits of adaptive data selection persist over extended training. This extended training regime is particularly important in the large-pool setting, as it allows us to test whether the benefit of Goldilocks persists as the model's capabilities evolve rather than being confined to the early stages of RL training. Our primary comparison focuses on training efficiency relative to tuned GRPO, while we also report validation accuracy at matched training steps. Because the models differ in size and are trained with different batch sizes, they are trained for different numbers of optimization steps under this regime; we therefore report the evaluation step in our tables. Complete hyperparameter searches, selected configurations, and compute accounting are reported in Appendix E.

6.1 OpenMathReasoning: Long-Run Performance and Efficiency

Table 1 reports both performance and training efficiency on OpenMathReasoning. For each model, all methods are evaluated at the same model-specific training step, enabling a direct comparison under a matched training horizon. We further measure efficiency by identifying the earliest checkpoint whose held-out accuracy matches or exceeds the final GRPO accuracy at that horizon, and report the corresponding percentage reduction in optimization steps. Together, these results capture both the accuracy achieved after extended training and the relative training-step savings required for Goldilocks to reach the corresponding GRPO performance.

Table 1: **OpenMathReasoning Results.** Validation accuracy (%) is averaged over the final five evaluation points at each model's respective training horizon. **Step Reduction** reports the relative reduction in optimization steps required for each method to reach GRPO's final accuracy, using five-point rolling averages. A $\times$ indicates that the method does not reach GRPO's final accuracy within the available training horizon.

		Accuracy (%)			Step Reduction	
Model	**Eval. Step**	**GRPO**	**PCL**	**Goldilocks**	**PCL**	**Goldilocks**
OLMo	40,000	16.02	14.47	**20.01**	$\times$	**64.0%**
Qwen2.5-1.5B	13,400	40.99	37.46	**44.56**	$\times$	**28.4%**
Qwen3-4B	21,000	54.00	53.86	**56.67**	$\times$	**41.9%**
Phi (4B)	16,000	40.12	38.39	**44.27**	$\times$	**48.8%**
Qwen2.5-7B	17,000	48.27	50.63	**52.61**	32.9%	**49.4%**

Figure 2 shows the validation accuracy throughout training for Qwen2.5-7B, comparing standard GRPO, Goldilocks, and PCL. Figures 5 and 6 in Appendix F report this comparison against wall-clock time and cumulative model-generated tokens, respectively. Goldilocks reaches the corresponding GRPO reference accuracy with less elapsed time and fewer generated tokens.

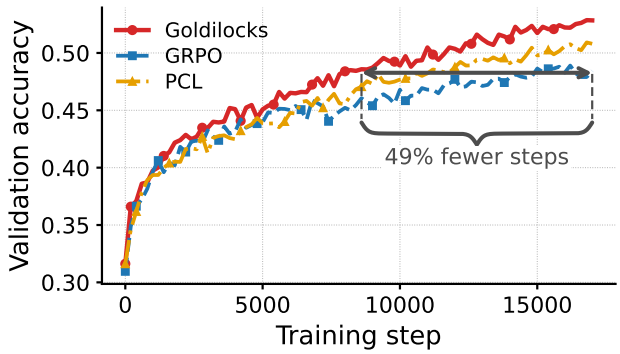


Figure 2: **Long-run reasoning training.** Held-out accuracy on OpenMathReasoning compares curriculum benefits across a large question pool.

6.2 Polaris: Retaining the Benefit When Questions Recur

The smaller Polaris dataset tests a different explanation for curriculum performance. With more opportunities to revisit each question, a history-based method can build and reuse estimates across a larger fraction of the training pool. This setting is therefore favorable to MoPPS, whose prompt-specific posteriors become more informative as observations accumulate. A benefit confined to sparse-history settings would be less compelling if it disappeared once these histories became readily available.

Goldilocks remains effective on Polaris despite the more frequent question revisitation that favors history-based methods. As shown in Table 2, it achieves higher final accuracy than GRPO and PCL at the common end step for each model. For Qwen2.5-1.5B, Goldilocks reaches 22.26% final accuracy compared with 21.52% for GRPO; it also reaches GRPO's final accuracy with 40.0% fewer optimization steps. Goldilocks surpasses MoPPS in most settings; the exception is Qwen2.5-7B, where MoPPS achieves the highest final accuracy. Goldilocks also reaches the final GRPO performance earlier across all models. These results complement the OpenMathReasoning experiments by showing that Goldilocks remains competitive even when repeated exposure makes historical estimates more informative.

Table 2: **Training on the smaller Polaris dataset.** Validation accuracy (%) averaged over the final five evaluation points at a common end step for each model. All methods are truncated at the shortest run's end step. The smaller training pool leads to more frequent question revisits, providing a favorable setting for history-based selection methods such as MoPPS. **Step Reduction** reports the relative reduction in optimization steps required for each method to reach GRPO's final accuracy, using five-point rolling averages. A $\times$ indicates that the method does not reach GRPO's final accuracy within the available training horizon.

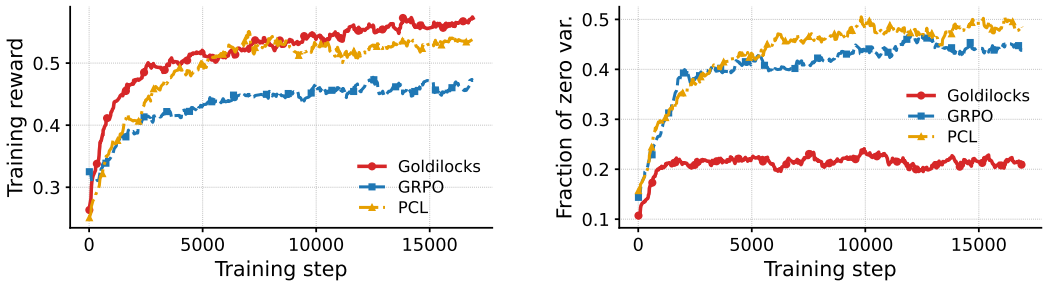
		Accuracy (%)				Step Reduction		
Model	**End Step**	**GRPO**	**PCL**	**MoPPS**	**Goldilocks**	**PCL**	**MoPPS**	**Goldilocks**
OLMo	29,000	21.15	18.65	20.04	**23.12**	$\times$	$\times$	**51.7%**
Qwen2.5-1.5B	6,000	21.52	19.85	21.56	**22.26**	$\times$	23.3%	**40.0%**
Qwen3-4B	10,000	31.32	30.08	30.95	**31.60**	$\times$	$\times$	**26.0%**
Phi (4B)	10,000	21.35	23.39	22.65	**25.87**	48.0%	38.0%	**78.0%**
Qwen2.5-7B	17,400	33.63	33.94	**38.89**	35.95	16.1%	**58.6%**	42.5%

7 Training Dynamics and Ablations

We first analyze how Goldilocks affects the training dynamics on OpenMathReasoning using Qwen2.5-7B as a representative model. We then use Qwen2.5-1.5B for controlled ablations.

7.1 Training Dynamics

We study the training behavior of Goldilocks on Qwen2.5-7B by tracking the average training reward and the fraction of zero-variance rollout groups throughout optimization. As shown in Figure 3a, PCL keeps the average reward closer to 0.5, whereas Goldilocks does not necessarily match this value as closely. However, Figure 3b shows that Goldilocks produces substantially fewer zero-variance rollout groups than both GRPO and PCL. This distinction is important: an average reward near 0.5 does not by itself imply that individual questions have intermediate success probability, since the same average can arise from a mixture of very easy and very difficult questions. Goldilocks instead directly targets $\sqrt{p_q(1-p_q)}$, which favors questions with high within-question reward variability and suppresses questions near $p_q \in \{0, 1\}$. We discuss this distinction further in Appendix K. We provide analysis of the Selector training dynamics in Appendix G.



(a) Average training reward.

(b) Zero-variance rollout fraction.

Figure 3: **Training dynamics on Qwen2.5-7B.**

7.2 Ablations

We evaluate the main design choices of Goldilocks using Qwen2.5-1.5B, with Qwen3-0.6B as the Selector throughout. Table 3 summarizes objective robustness, selection hyperparameters, and rollout-group size, while additional learning curves are provided in Appendix H.

Robustness across seeds. We repeat the standard GRPO–Goldilocks comparison across four model seeds. Figure 4 shows the corresponding learning curves, demonstrating that the improvement is consistent across different initializations.

Model Objective. Goldilocks continues to improve over the corresponding baseline when replacing the standard GRPO objective with a DAPO-style loss (Yu et al., 2025) and when adding entropy regularization. This suggests that the benefit of adaptive question selection is complementary to changes in the model optimization objective.

Selection hyperparameters. Goldilocks is relatively insensitive to moderate changes in the exploration probability ϵ and candidate-pool size K at the final evaluation horizon. Without exploration, the Selector only receives feedback on questions favored by its current predictions, which can reinforce selection errors and reduce coverage of underestimated questions. A small amount of random exploration mitigates this feedback loop. The full learning curves for ϵ and K are in Appendix H.

Generations per question. We additionally vary the rollout-group size G for both GRPO and Goldilocks. Across the tested settings, Goldilocks retains its advantage, indicating that its benefit is not specific to the default number of generations per question.

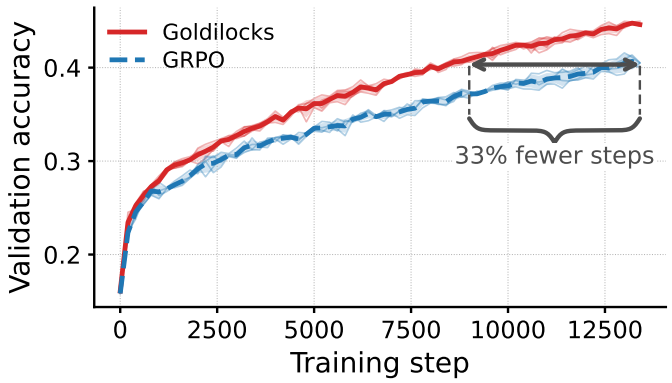


Figure 4: **Seed robustness on Qwen2.5-1.5B.** GRPO and Goldilocks across four model seeds. Solid lines represent the mean performance, while the shaded regions show one standard deviation of the raw validation accuracies across seeds at each evaluation step.

Table 3: **Ablations on OpenMathReasoning.** Validation accuracy (%) with Qwen2.5-1.5B, averaged over the final five evaluations at the listed training horizon, shared by all runs in each comparison. The reference configuration uses $G = 16$; Goldilocks additionally uses $K = 8$ and $\epsilon = 0.2$. Seed results are mean $\pm$ sample standard deviation across four seeds; their step reduction is computed from the mean curves. **Step Red.** reports the reduction in optimization steps required for Goldilocks to reach GRPO's final rolling-average accuracy at the same horizon.

			Accuracy (%)		Efficiency
Group	**Setting**	**Steps**	**GRPO**	**Goldilocks**	**Step Red.**
Main Results	Standard	13,400	40.99	**44.56**	28.4%
Objective Robustness	DAPO Loss	13,400	39.50	**43.66**	38.8%
	Entropy Reg. $\beta = 3 \times 10^{-4}$	13,400	40.28	**44.24**	29.9%
Seeds	Four seeds	13,400	40.33 ± 0.58	$\mathbf{44.46 \pm 0.30}$	32.8%
Selection Ablations	$\epsilon = 0$	13,400	40.99	**44.26**	31.3%
	$\epsilon = 0.2$	13,400	40.99	**44.56**	28.4%
	$\epsilon = 0.5$	13,400	40.99	**42.43**	11.9%
	$K = 8$	13,400	40.99	**44.56**	28.4%
	$K = 16$	13,400	40.99	**43.76**	29.9%
	$K = 32$	13,400	40.99	**44.02**	29.9%
Rollout Ablation	$G = 8$	13,400	40.87	**44.38**	32.8%
	$G = 16$	13,400	40.99	**44.56**	28.4%
	$G = 32$	13,400	38.36	**42.75**	41.8%

8 Limitations and Future Work

Larger models. Our experiments study models up to 7B parameters and use a 0.6B Selector throughout. They do not establish how the approach scales to substantially larger models, whose evolving capabilities may be harder for a small predictor to track. Future work should evaluate larger Selector and model combinations, measure how prediction quality changes with the capacity gap, and determine whether curriculum gains persist after all Selector costs are included.

Datasets beyond mathematics. OpenMathReasoning and Polaris differ in scale and question reuse, but both concern mathematical reasoning with verifiable answers. Generalization to other domains remains untested. Code generation, theorem proving, and multi-step tasks offer natural next settings, with different verifiers, reward distributions, and computational costs. Extending the selection target to these datasets, particularly when feedback is graded or noisy rather than binary, is an important direction for future work.

AI USE STATEMENT

We used generative AI tools to assist with language editing, improving clarity and organization, and refining the presentation of the manuscript. All scientific ideas, methodological choices, theoretical claims, experiments, results, and conclusions were developed and verified by the authors. We did not rely on generative AI to generate experimental results or to evaluate the validity of the reported findings. All AI-assisted text was reviewed and edited by the authors, who take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

We provide the details necessary to reproduce our experiments and theoretical results throughout the paper and appendices. The training procedure and Goldilocks selection mechanism are described in Section 4, while Selector architecture and experimental setup are given in Sections E.2 and 5.1. Dataset construction, evaluation protocols, hyperparameter selection, implementation details, reward specification, and baseline configurations are documented in the appendices. We additionally provide complete derivations for the theoretical results and report extended training curves and ablations to facilitate verification of the empirical findings.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Mohammad Hossein Amani, Aryo Lotfi, Nicolas Mario Baldwin, Samy Bengio, Mehrdad Farajtabar, Emmanuel Abbe, and Robert West. Rl for reasoning by adaptively revealing rationales. *arXiv preprint arXiv:2506.18110*, 2025.

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL https://hkunlp.github.io/blog/2025/Polaris.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for llm reasoning. *arXiv preprint arXiv:2505.14970*, 2025.

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

DeepSeek-AI. Deepseek-v3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

Thomas Foster, Anya Sims, Johannes Forkel, and Jakob Foerster. Lilo: Learning to reason at the frontier of learnability. *Advances in Neural Information Processing Systems*, 38:48941–48974, 2026.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient llm post-training. In *International Conference on Learning Representations*, volume 2026, pages 58614–58652, 2026.

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

Daya Guo, Dejian Yang, Haowei Zhang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633–638, 2025b. doi: 10.1038/s41586-025-09422-z.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.

Zhenyu Hou, Xin Lv, Rui Lu, Jiajie Zhang, Yujiang Li, Zijun Yao, Juanzi Li, Jie Tang, and Yuxiao Dong. T1: Advancing language model reasoning through reinforcement learning and inference scaling. *arXiv preprint arXiv:2501.11651*, 2025.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Venkata Sai Surya Subramanyam Duvvuri, Manzil Zaheer, Inderjit Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for llms. In *International Conference on Learning Representations*, volume 2026, pages 72438–72467, 2026.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

Ziniu Li, Congliang Chen, Tianyun Yang, Tian Ding, Ruoyu Sun, Ge Zhang, Wenhao Huang, and Zhi-Quan Luo. Knapsack rl: Unlocking exploration of llms via optimizing budget allocation. *arXiv preprint arXiv:2509.25849*, 2025.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

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

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, et al. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning. *arXiv preprint arXiv:2506.06632*, 2025.

Yun Qu, Qi Wang, Yixiu Mao, Vincent Tao Hu, Björn Ommer, and Xiangyang Ji. Can prompt difficulty be online predicted for accelerating rl finetuning of reasoning models? In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 1240–1250, 2026.

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

Qianli Shen, Daoyuan Chen, Yilun Huang, Zhenqing Ling, Yaliang Li, Bolin Ding, and Jingren Zhou. Bots: A unified framework for bayesian online task selection in llm reinforcement finetuning. In *International Conference on Learning Representations*, volume 2026, pages 99871–99911, 2026.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL `https://github.com/huggingface/trl`.

Zhenting Wang, Guofeng Cui, Yu-Jhe Li, Kun Wan, and Wentian Zhao. Dump: Automated distribution-level curriculum learning for rl-based llm post-training. *arXiv preprint arXiv:2504.09710*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Wei Xiong, Chenlu Ye, Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, Jiang Bian, Nan Jiang, and Tong Zhang. Reinforce-ada: An adaptive sampling framework under non-linear rl objectives. *arXiv preprint arXiv:2510.04996*, 2025.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runxin Xu, Tianhao Li, Tianyu Liu, Wang Fan, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xinyuan Li, Yang Fan, Yang Tan, Yu Wan, Yuze Gao, Yuqiang Xie, Zekun Wang, Zengmao Wang, Zhihao Xiang, Zhitang Li, Zhiyuan Gao, Zhiyuan Xu, and Zick Ye. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

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.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pages 15476–15488, 2022.

A ALGORITHMS

Algorithm 1 Goldilocks Selector: Data Selection Policy

Input: Dataset $\mathcal{D}$, Selector network f_ϕ, Candidate Size K, Exploration ϵ.
Output: Selected question q^*.
function SelectQuery(ϕ, $\mathcal{D}$)
  $\mathcal{C} \leftarrow$ Sample K random indices from $\mathcal{D}$
  $\mathcal{V} \leftarrow \emptyset$
  for each candidate $q \in \mathcal{C}$ **do**
    $\hat{v}_q \leftarrow f_\phi(q)$ {Predict learning utility}
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{\hat{v}_q\}$
  end for
  $r \sim \text{Uniform}(0, 1)$
  if $r < \epsilon$ **then**
    $q^* \leftarrow \text{Uniform}(\mathcal{C})$ {Ensure coverage (ϵ-greedy)}
  else
    $q^* \leftarrow \arg\max_{q \in \mathcal{C}} \hat{v}_q$ {Greedy maximization}
  end if
  return q^*
end function

Algorithm 2 Goldilocks Selector: Online Refinement

Input: Question q, Rollouts $\{o_1, ..., o_G\}$, Selector network f_ϕ, Replay Buffer $\mathcal{D}_{\text{replay}}$, Capacity N_{replay}.
function Update Selector(q, $\{o_i\}_{i=1}^G$)
  $\hat{p}_q \leftarrow \frac{1}{G} \sum_{i=1}^G \mathbb{I}(o_i \text{ is correct})$ {Empirical success rate}
  $y_q \leftarrow \sqrt{\hat{p}_q(1 - \hat{p}_q)}$ {Compute target utility}
  Push (q, y_q) to $\mathcal{D}_{\text{replay}}$
  if $|\mathcal{D}_{\text{replay}}| > N_{\text{replay}}$ **then**
    Remove oldest sample from $\mathcal{D}_{\text{replay}}$ {Sliding window}
  end if
  {Periodic Update Trigger (Step 6)}
  if Update Condition Met **then**
    Shuffle $\mathcal{D}_{\text{replay}}$ and create batches $\mathcal{B}$
    for epoch 1 to E_{Selector} **do**
      for batch $B \in \mathcal{B}$ **do**
        $\mathcal{L} \leftarrow \frac{1}{|B|} \sum_{(q,y) \in B} (f_\phi(q) - y)^2$
        $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}$ {Gradient descent step}
      end for
    end for
  end if
end function

B GENERALIZATION TO COMPOSITE REWARDS

Assumption B.1 (Convergence of auxiliary reward)**.** Let $r_{\text{total}} = r_{\text{ver}} + f(t)$, where $f(t)$ is any reward component depending on training step t. There exists an iteration threshold $N \in \mathbb{N}$ such that for all $t \geq N$, $f(t) = C$ for some constant $C \in \mathbb{R}$.

Proposition B.2. *Under Assumption B.1, for sufficiently large t, the GRPO advantage computed from r_{total} is identical to that computed from r_{ver} alone.*

Proof. For $t \geq N$, we have $r_{\text{total}} = r_{\text{ver}} + C$. Since $r_{\text{ver}} \sim \text{Bernoulli}(p_q)$:

$$\text{mean}(r_{\text{total}}) = p_q + C, \quad \text{std}(r_{\text{total}}) = \sqrt{p_q(1-p_q)}.$$

The advantage is:

$$\begin{aligned}\hat{A}_q &= \frac{r_{\text{total}} - \text{mean}(r_{\text{total}})}{\text{std}(r_{\text{total}})} \\ &= \frac{(r_{\text{ver}} + C) - (p_q + C)}{\sqrt{p_q(1-p_q)}} \\ &= \frac{r_{\text{ver}} - p_q}{\sqrt{p_q(1-p_q)}},\end{aligned}$$

which is identical to the advantage computed from r_{ver} alone. □

C PROOFS FOR GRPO WITH VERIFIABLE REWARD

In this appendix, we provide the formal derivations and proofs for the policy gradient properties under the GRPO framework with binary verifiable rewards. First, we derive the decomposition of the policy gradient norm. Then, we use this decomposition to prove the exact scaling identity and the structural upper bound of the gradient signal.

C.1 DERIVATION OF THE GRADIENT NORM DECOMPOSITION

Recall that the gradient of the standard policy objective for a fixed prompt q is the expected product of the trajectory score $S_\theta(o)$ and the sequence-level advantage $\hat{A}_{q,o}$:

$$\nabla_\theta \mathcal{L}_{\text{PG}} = -\mathbb{E}_{o\sim\pi_\theta}\left[S_\theta(o)\hat{A}_{q,o}\right], \tag{4}$$

where the trajectory score is $S_\theta(o) = \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(o_t \mid q, o_{<t})$.

As established in Section 3, the advantage is the standardized binary reward. Since $r_{\text{ver}} \sim \text{Bernoulli}(p_q)$, we substitute the advantage:

$$\hat{A}_{q,o} = \frac{r_{\text{ver}} - p_q}{\sqrt{p_q(1-p_q)}}$$

Plugging this into the gradient expectation yields:

$$\nabla_\theta \mathcal{L}_{\text{PG}} = -\mathbb{E}_{o\sim\pi_\theta}\left[S_\theta(o)\frac{r_{\text{ver}} - p_q}{\sqrt{p_q(1-p_q)}}\right]$$

Because $r_{\text{ver}} \in \{0, 1\}$, we apply the law of total expectation to split this into two conditional branches weighted by their respective probabilities, p_q and $(1 - p_q)$:

$$\nabla_\theta \mathcal{L}_{\text{PG}} = -\left(p_q\mathbb{E}\left[S_\theta(o)\frac{1-p_q}{\sqrt{p_q(1-p_q)}}\,\middle|\, r_{\text{ver}} = 1\right] + (1-p_q)\mathbb{E}\left[S_\theta(o)\frac{0-p_q}{\sqrt{p_q(1-p_q)}}\,\middle|\, r_{\text{ver}} = 0\right]\right)$$

Since p_q is constant with respect to the inner expectation, we can factor the scalars out. Notice that the coefficients for both terms evaluate to the exact same magnitude:

$$\frac{p_q(1-p_q)}{\sqrt{p_q(1-p_q)}} = \sqrt{p_q(1-p_q)}$$

Substituting this simplified coefficient back into the expression gives:

$$\nabla_\theta \mathcal{L}_{\text{PG}} = -\left(\sqrt{p_q(1-p_q)}\mathbb{E}\left[S_\theta(o) \,\middle|\, r_{\text{ver}} = 1\right] - \sqrt{p_q(1-p_q)}\mathbb{E}\left[S_\theta(o) \,\middle|\, r_{\text{ver}} = 0\right]\right)$$

Factoring out the $\sqrt{p_q(1-p_q)}$ scalar and taking the norm of both sides yields the gradient norm decomposition:

$$\|\nabla_\theta \mathcal{L}_{\text{PG}}\| = \sqrt{p_q(1-p_q)}\left\|\mathbb{E}\left[S_\theta(o)\,\middle|\, r_{\text{ver}} = 1\right] - \mathbb{E}\left[S_\theta(o)\,\middle|\, r_{\text{ver}} = 0\right]\right\| \tag{5}$$

C.2 DERIVATION OF THE GRADIENT UPPER BOUND

Starting from Eq. 5, the policy-gradient norm for a fixed question q can be written as

$$\|\nabla_\theta \mathcal{L}_{\text{PG}}\| = \sqrt{p_q(1-p_q)}\,\|\mathbb{E}[S_\theta(o) \mid r_{\text{ver}} = 1] - \mathbb{E}[S_\theta(o) \mid r_{\text{ver}} = 0]\|\,,$$

where

$$S_\theta(o) = \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(o_t \mid q, o_{<t})$$

is the trajectory score.

Assume that the per-token score is uniformly bounded by a constant B,

$$\|\nabla_\theta \log \pi_\theta(o_t \mid q, o_{<t})\| \le B.$$

By the triangle inequality, the norm of the trajectory score is then bounded by

$$\|S_\theta(o)\| \le \sum_{t=1}^{T} \|\nabla_\theta \log \pi_\theta(o_t \mid q, o_{<t})\| \le BT.$$

The same bound therefore holds for the conditional expectations,

$$\|\mathbb{E}[S_\theta(o) \mid r_{\text{ver}} = 1]\| \le BT, \qquad \|\mathbb{E}[S_\theta(o) \mid r_{\text{ver}} = 0]\| \le BT.$$

Applying the triangle inequality once more gives

$$\|\nabla_\theta \mathcal{L}_{\text{PG}}\| \le \sqrt{p_q(1-p_q)}\,(\|\mathbb{E}[S_\theta(o) \mid r_{\text{ver}} = 1]\| + \|\mathbb{E}[S_\theta(o) \mid r_{\text{ver}} = 0]\|) \tag{6}$$

$$\le 2BT\sqrt{p_q(1-p_q)}. \tag{7}$$

Thus, under a uniform bound on the per-token score, the policy-gradient norm is upper bounded by a term proportional to $\sqrt{p_q(1-p_q)}$. This bound vanishes as $p_q \to 0$ or $p_q \to 1$ and is largest at $p_q = 1/2$, providing the intuition behind prioritizing questions of intermediate difficulty.

D DATASET DETAILS

For evaluation, we construct fixed validation sets of 5,000 problems for both OpenMathReasoning and Polaris, sampled randomly from their respective datasets and kept strictly disjoint from the corresponding training sets. The same validation subset is used across all methods and runs for each dataset to ensure consistent and comparable evaluation.

E Hyperparameter Selection and Implementation

E.1 Model Configuration

For each Model–dataset pair, we tune GRPO over batch sizes $\{256, 512, 768, 1024, 2048\}$ and learning rates $\{10^{-5}, 10^{-6}, 3.33 \times 10^{-6}, 6.66 \times 10^{-6}\}$, and select the best-performing configuration using validation accuracy on same wall time. The selected settings depend on both the Model and the training dataset. We then use the selected Model learning rate, per-device batch size, and gradient-accumulation factor for Goldilocks and the corresponding curriculum baselines, with the effective batch adjusted for the allocation of GPUs between Selector network and model described below. Table 4 summarizes the search space, and Table 5 reports the selected GRPO settings for every Model–dataset pair.

Table 4: **GRPO hyperparameter search and shared Model configuration.** Batch size and learning rate are tuned separately for each Model–dataset pair. Batch sizes count generated completions per optimization step.

Setting	Value
Batch-size search space	256, 512, 768, 1,024, 2,048
Learning-rate search space	10^{-5}, 10^{-6}, 3.33×10^{-6}, 6.66×10^{-6}
Selection criterion	Validation accuracy
Generations per question G	16

Table 5: **Selected GRPO Model hyperparameters.** Batch size is the total number of completions per optimization step across all eight Model GPUs. All compared methods use the listed Model learning rate; Goldilocks and PCL use $7/8$ of the listed batch size when one GPU is reserved for the Selector.

	OpenMathReasoning		Polaris	
Model	**Batch size**	**Learning rate**	**Batch size**	**Learning rate**
OLMo2-1B	768	10^{-5}	1,024	10^{-5}
Qwen2.5-1.5B	1,024	10^{-5}	1,024	10^{-5}
Qwen3-4B-Base	256	3.33×10^{-6}	256	3.33×10^{-6}
Phi-4-mini-Instruct	256	10^{-5}	256	10^{-5}
Qwen2.5-7B	256	3.33×10^{-6}	256	3.33×10^{-6}

The effective completion batch is $B_{\text{comp}} = n_{\text{GPU}} b_{\text{device}} a_{\text{accum}}$, where n_{GPU} is the number of Model GPUs, b_{device} is the per-device completion batch, and a_{accum} is the gradient-accumulation factor. Each update therefore contains B_{comp}/G prompt groups. The selected GRPO configurations use eight Model GPUs and a per-device batch size of four. Under the fixed eight-GPU budget in Section 5.1, Goldilocks and PCL allocate one GPU to the Selector and seven to the Model. Keeping the per-device batch and accumulation factor fixed therefore gives these methods $7/8$ of the GRPO completion batch. This accounts for the Selector's hardware cost while preserving the selected Model learning rate and per-device configuration.

E.2 Selector Architecture

The Selector functions as a value predictor built upon a trainable language model backbone. For a given input question q, we extract the final layer embeddings and apply mean pooling to obtain a fixed-size vector representation. This vector is then passed through a linear projection layer. To strictly enforce the valid range of the standard deviation term (which lies between 0 and 0.5 for a Bernoulli variable), we apply a scaled sigmoid activation, i.e.,

$$f_\phi(q) = 0.5 \cdot \sigma(\mathbf{w}^T \cdot \text{MeanPool}(\text{Embed}(q)) + b),$$

where $f_\phi(q)$ represents the predicted utility of question q. The parameters ϕ encompass both the projection head and the pretrained language model, allowing the Selector to learn deep semantic

features relevant to difficulty estimation. For a detailed exploration of alternative Selector backbones and projection strategies, please refer to Appendix E.3.

E.3 Selector Configuration

Every learned curriculum predictor uses Qwen3-0.6B, independent of the Model architecture or training dataset. Goldilocks fine-tunes this backbone to predict the standard deviation of binary correctness rewards from question text (Section E.2). Its Selector configuration is shared across the main experiments (Table 6); the ablations vary only the specified component (Appendix H).

Table 6: Goldilocks Selector configuration.

Setting	Value
Backbone	Qwen3-0.6B
Representation	Masked mean of final hidden states
Output	Scaled sigmoid in $[0, 1/2]$
Adaptation	Full-parameter fine-tuning
Optimizer / learning rate	AdamW / 10^{-5}
Selector minibatch size	32
Passes over replay buffer per update	4
Replay capacity	128 questions
Random warm-up	64 question selections
Candidate count K	8
Exploration probability ϵ	0.2
Prediction target / loss	Reward standard deviation / squared error

Selector updates require at least one complete replay minibatch. The bounded buffer emphasizes recent Model behavior as the policy improves; all Selector labels come from Model rollouts already used for training. The number of completions used to form each Selector target matches the Model's rollout group size, including in the generation-count ablations (Appendix H).

E.4 Baseline Details

GRPO uses the standard sampler over the shuffled training dataset. PCL (Gao et al., 2026) learns a success-probability predictor with the same Qwen3-0.6B backbone as Goldilocks and prioritizes questions near a target success rate of 0.5. MoPPS (Qu et al., 2026) maintains a Beta posterior for each question and selects prompts by posterior sampling.

For PCL and MoPPS, we use the exact method-specific hyperparameters reported in their original papers, except that we additionally tune PCL's Selector learning rate to ensure a well-tuned baseline implementation. The selected Selector learning rate is 10^{-6} across all Model–dataset pairs; the remaining baseline-specific hyperparameters retain their published values. Model optimization follows the GRPO-selected learning rate and batch configuration in Table 5, including the GPU-allocation adjustment for learned Selectors.

F Compute and Training Efficiency

To complement the optimization-step comparisons in the main text, we evaluate training efficiency using two additional measures: wall-clock time and the number of Model-generated tokens. Wall-clock time captures the end-to-end training cost, including Selector scoring and optimization for learned curriculum methods, while generated tokens isolate the dominant cost of Model rollout generation. These views allow us to distinguish improvements in sampling efficiency from differences in system-level overhead. The following comparisons use Qwen2.5-7B on OpenMathReasoning. GRPO's last available point within that budget defines the reference accuracy.

Wall-clock efficiency. Figure 5 reports accuracy as a function of elapsed training time. All methods are evaluated under the same hardware allocation described in Section 5.1. For Goldilocks and PCL, the reported time includes Selector inference and Selector updates.

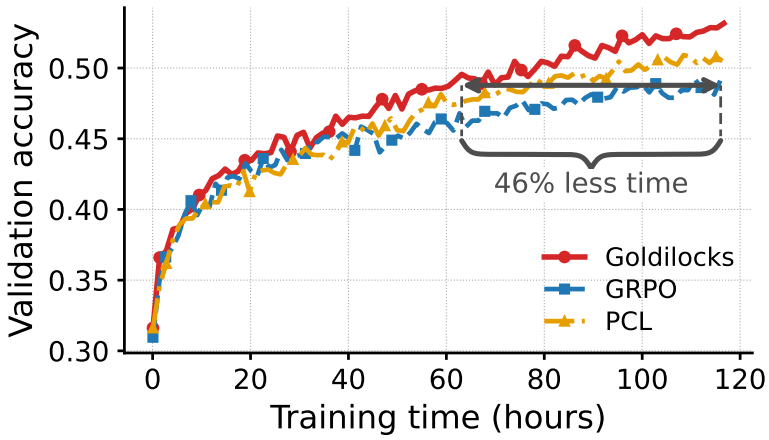


Figure 5: **Accuracy versus wall-clock time.** Held-out accuracy of Qwen2.5-7B on OpenMathReasoning throughout training for GRPO, PCL, and Goldilocks.

Generated-token efficiency. Figure 6 reports accuracy against the cumulative number of tokens generated by the Model during training. This comparison measures how effectively each method converts rollout generation into improvements in held-out performance, independent of Selector-side computational overhead.

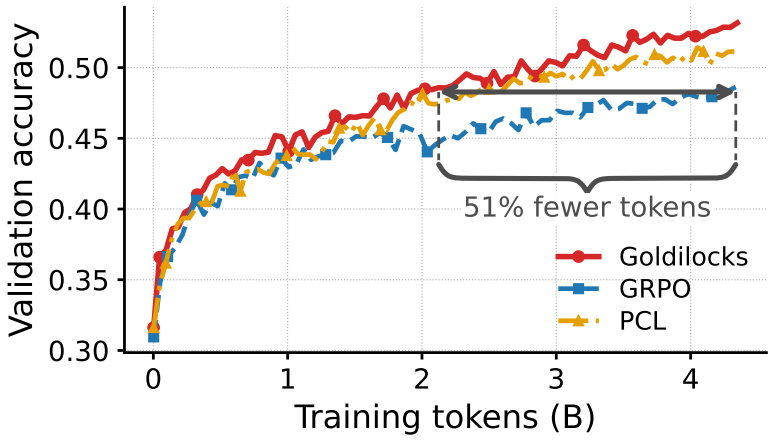


Figure 6: **Accuracy versus generated-token budget.** Held-out accuracy of Qwen2.5-7B on OpenMathReasoning as a function of cumulative Model-generated tokens for GRPO, PCL, and Goldilocks.

Together, the step-, wall-clock-, and token-based evaluations provide complementary measures of efficiency. Optimization steps measure learning progress at the algorithmic level, generated tokens capture rollout efficiency, and wall-clock time reflects the end-to-end cost of running each method.

G SELECTOR TRAINING DYNAMICS

We further analyze the behavior of the Goldilocks Selector during training. In these experiments, each Model optimization step corresponds to four Selector optimization steps. Consequently, the Selector-step axis in the following figures spans approximately four times as many steps as the Model-training axes reported elsewhere in the paper.

First, we measure how accurately the Selector tracks the reward variability observed from the Model's rollouts. For each question, the Selector predicts a utility score $f_\phi(q)$ before generation. After obtaining the rollout group, we compute the realized target

$$y_q = \sqrt{\hat{p}_q(1-\hat{p}_q)},$$

and evaluate the absolute prediction error $|f_\phi(q) - y_q|$. Figure 7 shows that this error decreases over training, indicating that the Selector progressively adapts to the Model's evolving capabilities. We additionally report the error on zero-variance groups, for which $y_q = 0$, to measure how well the Selector identifies questions expected to provide little relative learning signal.

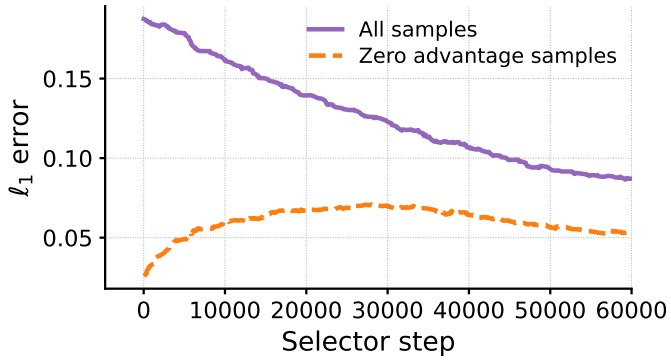


Figure 7: **Selector prediction error on OpenMathReasoning.** Absolute error between predicted and realized reward-variability targets throughout training. The x-axis reports Selector optimization steps; each Model optimization step corresponds to four Selector steps.

Diversity of Selector predictions. We also track the distribution of utility scores predicted by the Selector throughout training. Figure 8 reports the mean predicted score together with one standard deviation. The persistent spread in predicted utilities shows that the Selector does not collapse to a nearly constant scoring function as training progresses; instead, it continues to distinguish questions according to their predicted reward variability as the Model evolves.

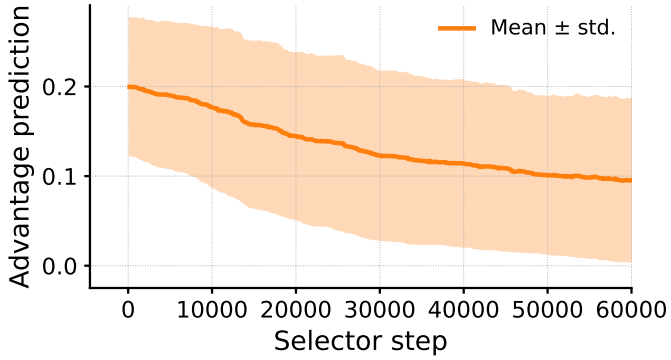
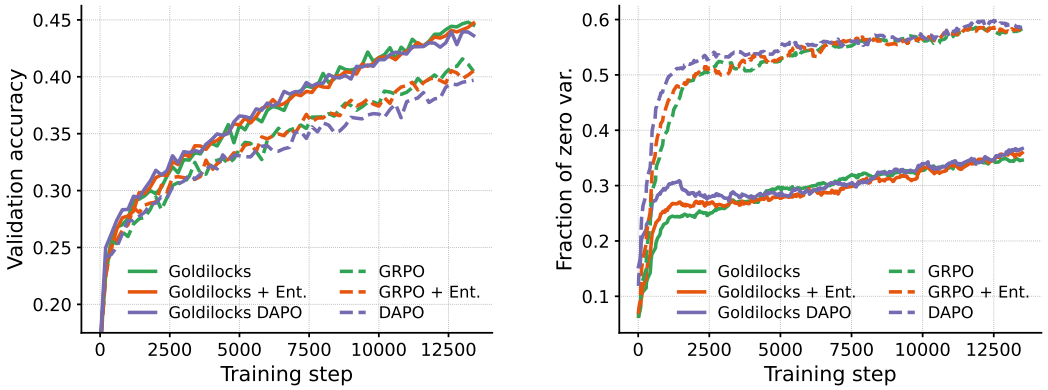


Figure 8: **Distribution of Goldilocks Selector scores.** Mean predicted utility with ± one standard deviation over Selector optimization steps.

H Detailed Ablations

All ablations use Qwen2.5-1.5B on OpenMathReasoning with Qwen3-0.6B as the Selector. Unless varied explicitly, we use the reference configuration $\epsilon = 0.2$, $K = 8$, and $G = 16$ (Table 6), together with the Model batch size and learning rate selected for GRPO (Table 4).

Optimization objective. We first test whether the benefit of Goldilocks depends on the particular model optimization objective. In addition to standard GRPO, we consider a DAPO-style loss (Yu et al., 2025) and GRPO with entropy regularization $\beta = 3 \times 10^{-4}$. In each case, GRPO and Goldilocks use the same objective, isolating the effect of adaptive question selection. Figure 9 shows the corresponding validation accuracy and zero-variance fraction.

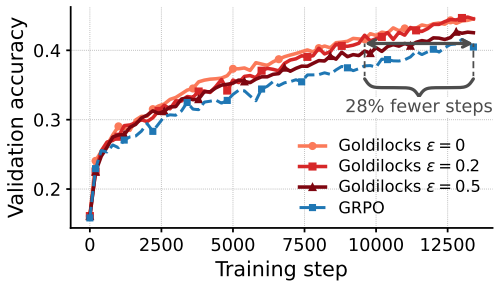


(a) Validation accuracy.

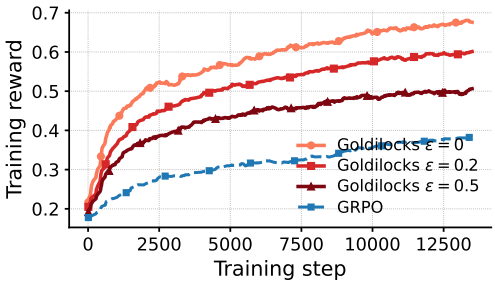
(b) Zero-variance fraction.

Figure 9: **Optimization-objective ablation.** GRPO and Goldilocks under standard GRPO, DAPO, and entropy-regularized objectives.

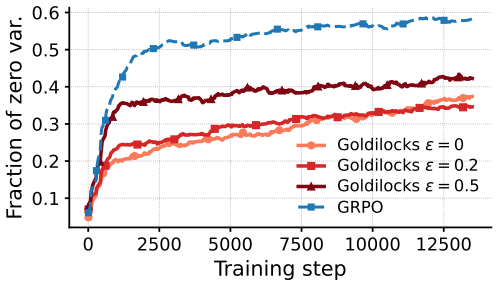
Exploration probability. We vary $\epsilon \in \{0, 0.2, 0.5\}$ to study the role of random exploration in the Selector. Final performance is relatively insensitive to moderate changes in ϵ: larger values increase randomness in the selected questions, whereas smaller values make the Selector rely more heavily on its own predictions and therefore on the subset of questions it already favors. Figure 10 reports validation accuracy together with the corresponding training-reward and zero-variance dynamics.



(a) Validation accuracy.



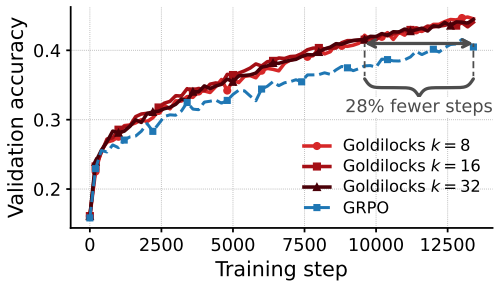
(b) Average training reward.



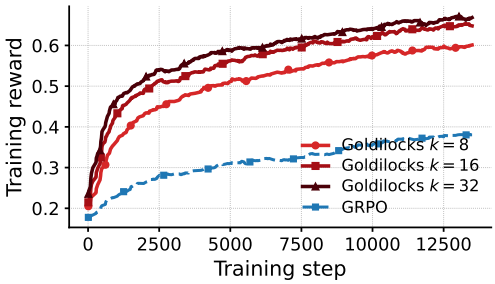
(c) Zero-variance fraction.

Figure 10: **Exploration-probability ablation.** Training dynamics for $\epsilon \in \{0, 0.2, 0.5\}$.

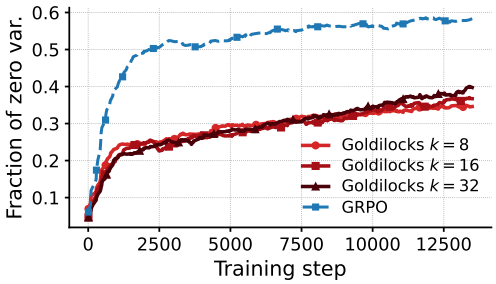
Candidate-pool size. We vary $K \in \{8, 16, 32\}$ to study sensitivity to the number of candidate questions scored before each selection. Increasing K gives the Selector more opportunities to identify informative questions, while also increasing Selector inference cost. Across the tested range, performance changes only moderately, indicating that Goldilocks does not rely on a narrowly tuned candidate-pool size. The corresponding learning curves are shown in Figure 11.



(a) Validation accuracy.



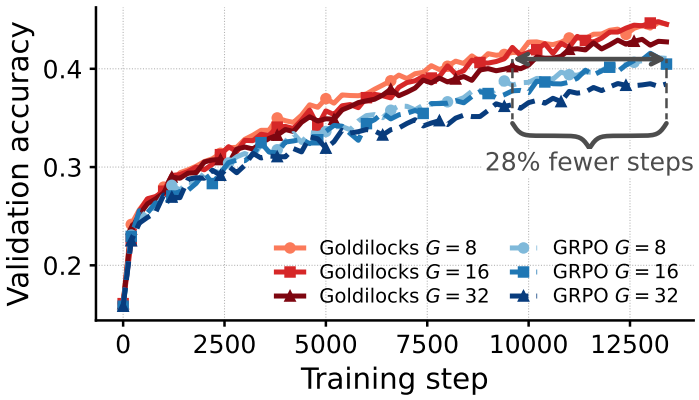
(b) Average training reward.



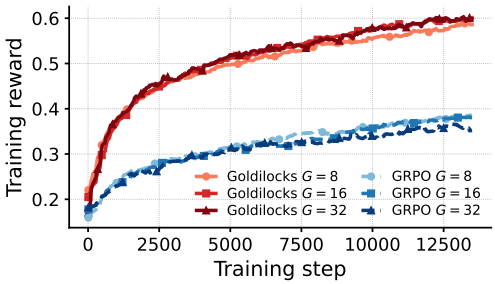
(c) Zero-variance fraction.

Figure 11: **Candidate-pool size ablation.** Training dynamics for $K \in \{8, 16, 32\}$.

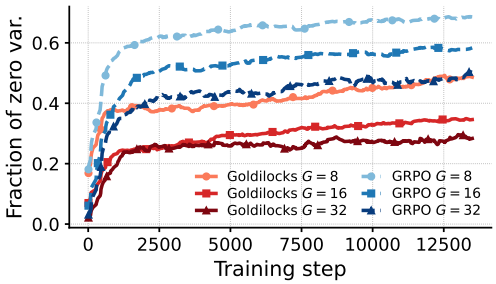
Generations per question. We compare GRPO and Goldilocks at $G \in \{8, 16, 32\}$ while keeping the global batch size fixed. Increasing G therefore allocates more generations to each question while reducing the number of distinct prompt groups in an update. It also changes both the noise in the Selector target and the probability of observing mixed rewards within a rollout group. Figure 12 shows that Goldilocks retains its advantage across the tested group sizes.



(a) Validation accuracy.



(b) Average training reward.



(c) Zero-variance fraction.

Figure 12: **Rollout-group-size ablation.** GRPO and Goldilocks with $G \in \{8, 16, 32\}$ at a fixed global batch size.

Robustness across seeds. Finally, we repeat the standard GRPO–Goldilocks comparison across four Model seeds. Figure 13 shows the mean learning trajectories and their variation across seeds.

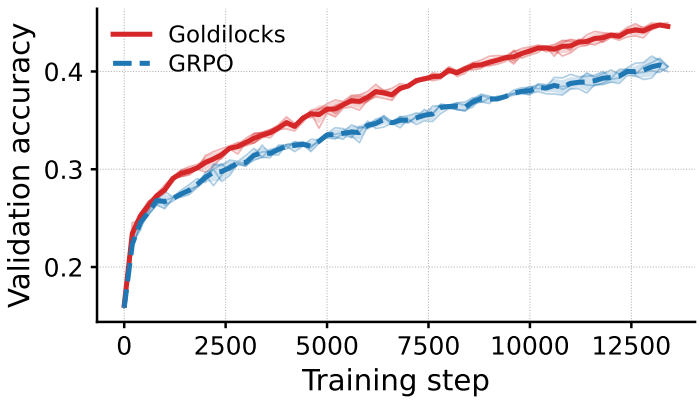


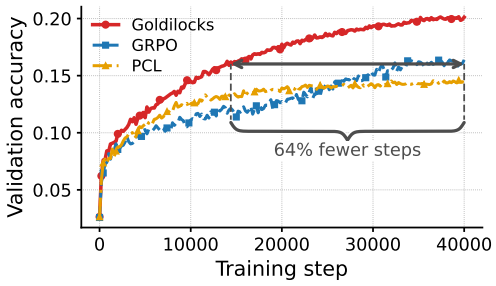
Figure 13: **Seed robustness.** Validation accuracy for GRPO and Goldilocks across four Model seeds.

I Additional Validation and Training Dynamics

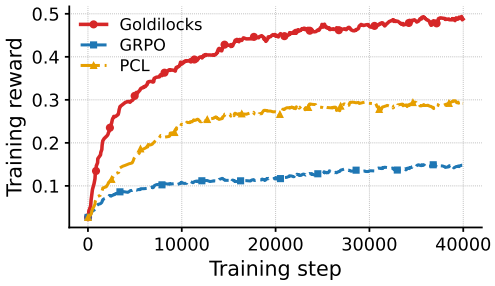
We provide additional validation and training-dynamics curves for all Models on OpenMathReasoning and Polaris. For each model, we report validation accuracy throughout training, the average training reward of the selected questions, and the fraction of rollout groups with zero reward variance. Validation accuracy characterizes learning progress, while the latter two metrics characterize the distribution and informativeness of the questions selected by each sampling strategy. In particular, the zero-variance fraction directly measures how often all generations for a selected question receive the same reward and therefore provide no relative reward signal.

I.1 OpenMathReasoning

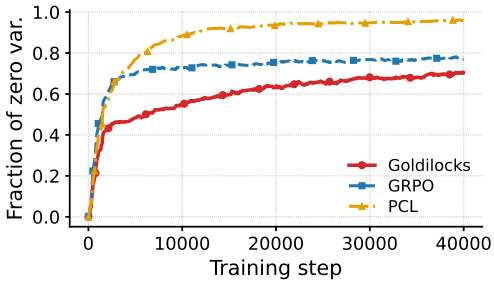
Figures 14–18 report the corresponding curves for all Models on OpenMathReasoning. These results complement Table 1 by showing how validation performance and the training distribution evolve over the full optimization horizon.



(a) Validation accuracy.

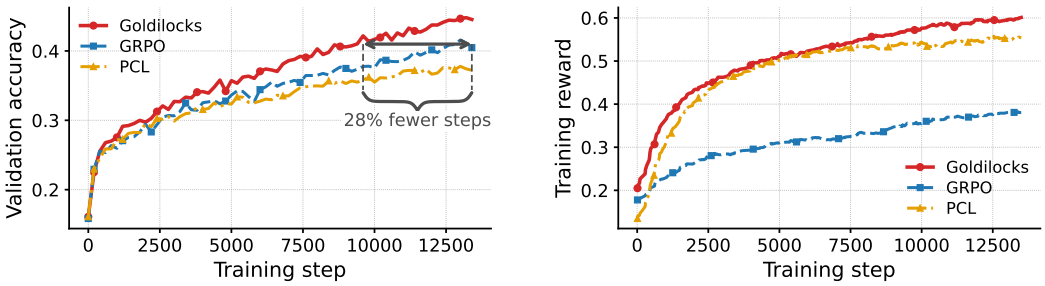


(b) Average training reward.



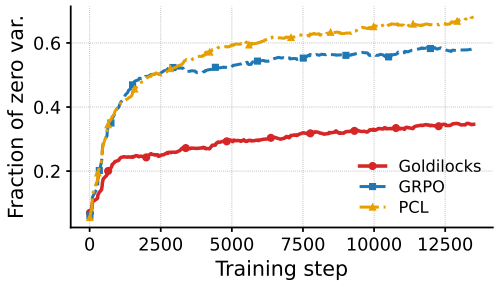
(c) Zero-variance fraction.

Figure 14: **OLMo2-1B on OpenMathReasoning.** Validation performance and training dynamics throughout optimization.



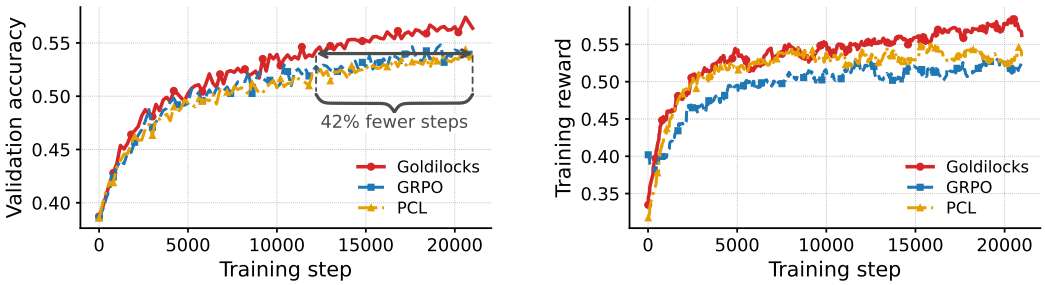
(a) Validation accuracy.

(b) Average training reward.



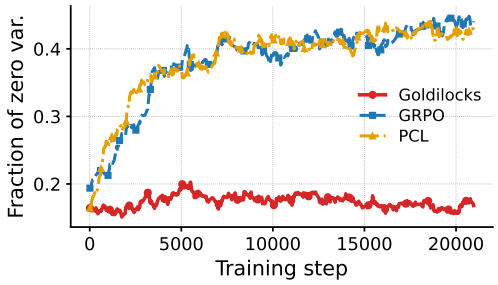
(c) Zero-variance fraction.

Figure 15: **Qwen2.5-1.5B on OpenMathReasoning.** Validation performance and training dynamics throughout optimization.



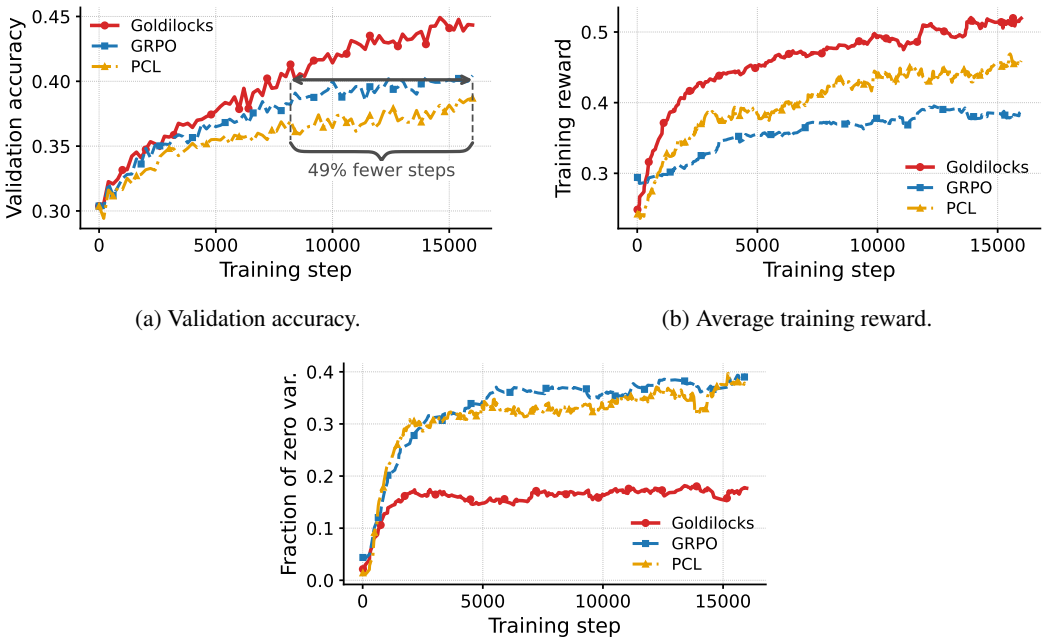
(a) Validation accuracy.

(b) Average training reward.



(c) Zero-variance fraction.

Figure 16: **Qwen3-4B on OpenMathReasoning.** Validation performance and training dynamics throughout optimization.

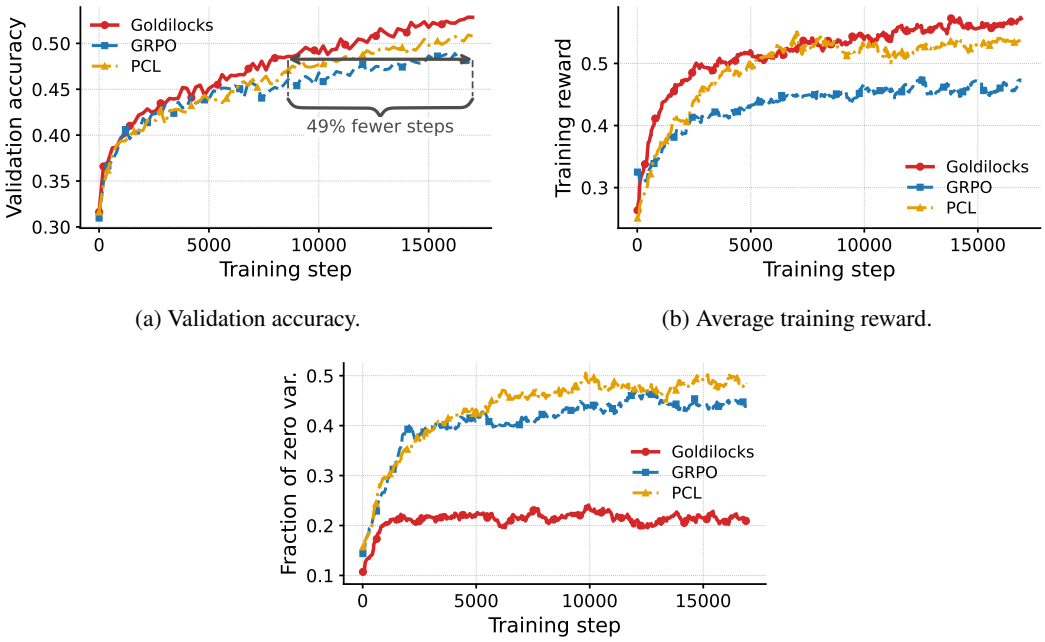


(a) Validation accuracy.

(b) Average training reward.

(c) Zero-variance fraction.

Figure 17: **Phi-4-mini-Instruct on OpenMathReasoning.** Validation performance and training dynamics throughout optimization.



(a) Validation accuracy.

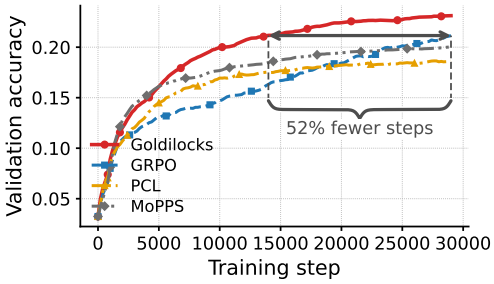
(b) Average training reward.

(c) Zero-variance fraction.

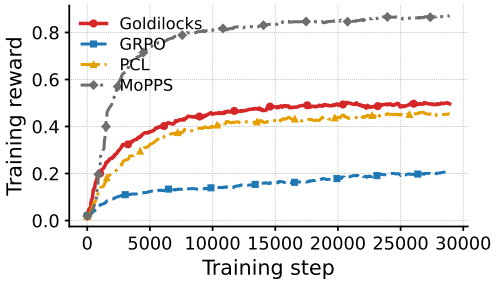
Figure 18: **Qwen2.5-7B on OpenMathReasoning.** Validation performance and training dynamics throughout optimization.

I.2 POLARIS

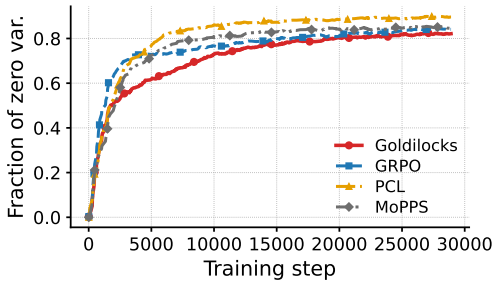
Figures 19–23 report the same diagnostics on Polaris. The smaller training pool produces more frequent question revisitation, allowing history-based methods such as MoPPS to accumulate increasingly informative question-specific estimates. These curves complement Table 2 by showing how the relative performance and selected-question distributions evolve throughout training.



(a) Validation accuracy.

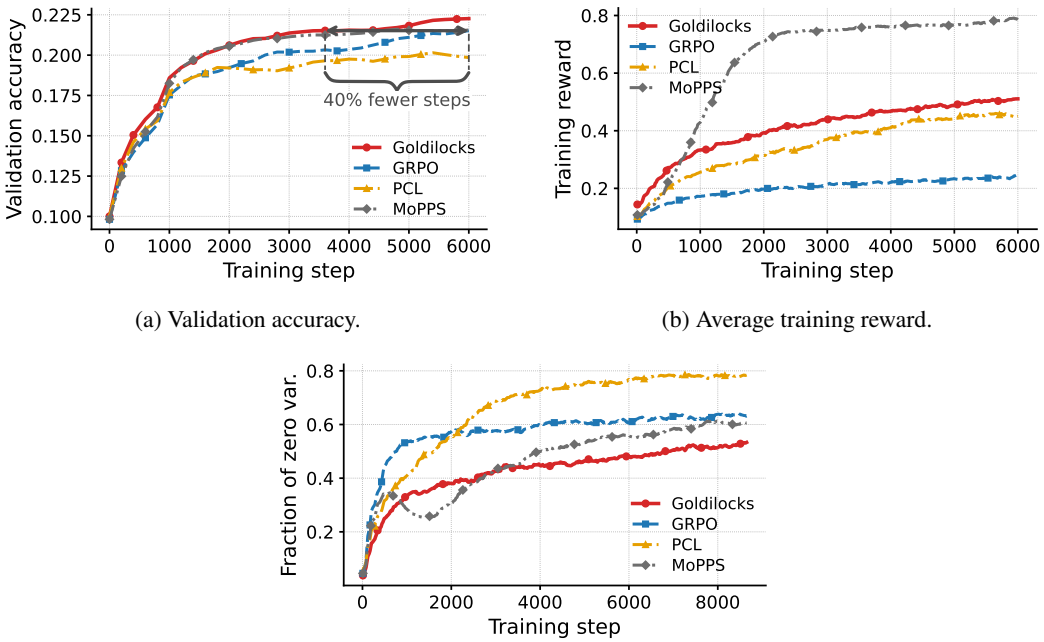


(b) Average training reward.



(c) Zero-variance fraction.

Figure 19: **OLMo2-1B on Polaris.** Validation performance and training dynamics throughout optimization.

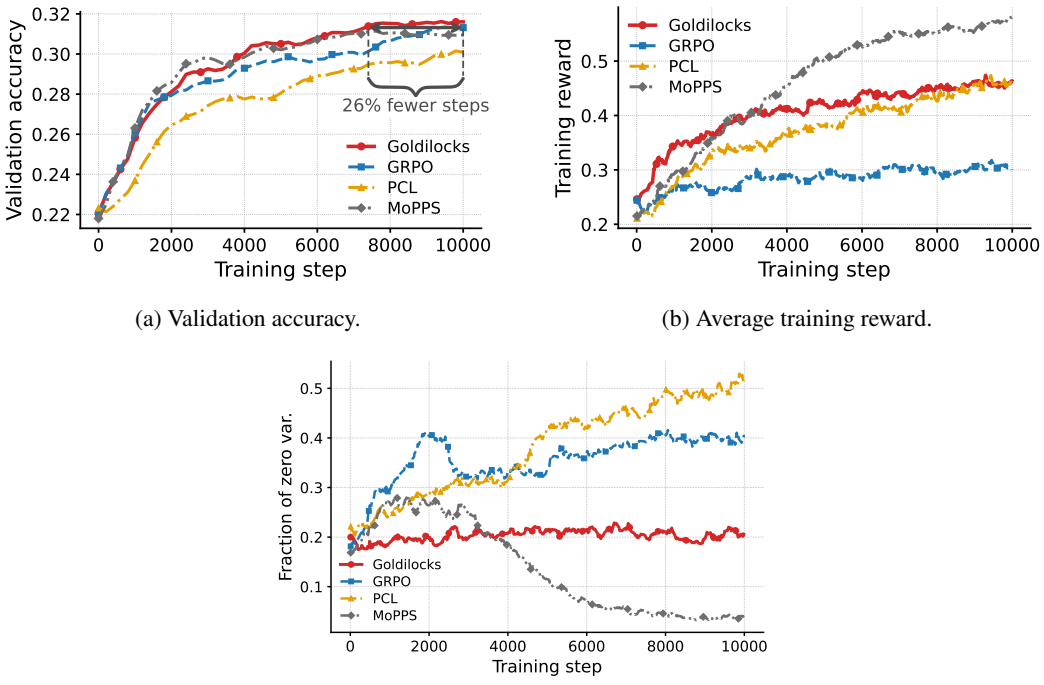


(a) Validation accuracy.

(b) Average training reward.

(c) Zero-variance fraction.

Figure 20: **Qwen2.5-1.5B on Polaris.** Validation performance and training dynamics throughout optimization.

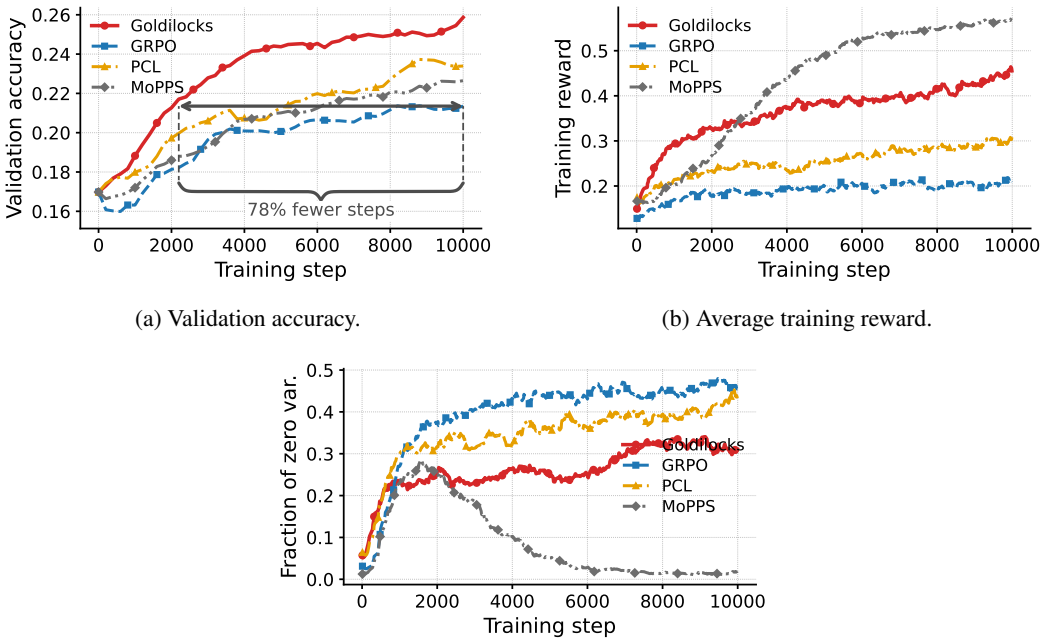


(a) Validation accuracy.

(b) Average training reward.

(c) Zero-variance fraction.

Figure 21: **Qwen3-4B on Polaris.** Validation performance and training dynamics throughout optimization.

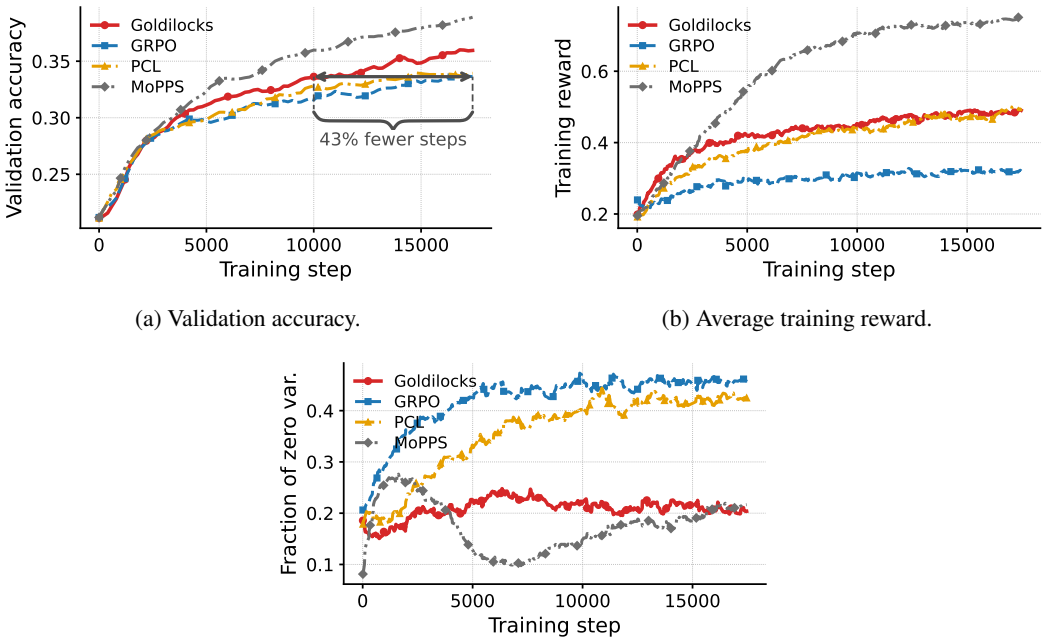


(a) Validation accuracy.

(b) Average training reward.

(c) Zero-variance fraction.

Figure 22: **Phi-4-mini-Instruct on Polaris.** Validation performance and training dynamics throughout optimization.



(a) Validation accuracy.

(b) Average training reward.

(c) Zero-variance fraction.

Figure 23: **Qwen2.5-7B on Polaris.** Validation performance and training dynamics throughout optimization.

J PROMPTS

The final input to the model is constructed by concatenating a fixed header, the specific math problem, and a fixed footer:

$$\text{Input} = \texttt{PROMPT_HEADER} + \text{Question} + \texttt{PROMPT_FOOTER}$$

The exact text used for the header and footer is provided below.

PROMPT HEADER

```
You are an expert problem solver. Your task is to solve the problem by
explicitly showing your detailed Chain of Thought.
You must write out your reasoning step-by-step, explaining your logic
for every calculation or deduction.
**Crucially, the final answer must be enclosed in a \boxed{...} command.**
```

PROMPT FOOTER

```
Strictly follow this output format:
### Step-by-Step Reasoning
<Write your detailed thought process here. Do not skip steps.>

### Final Answer
\boxed{<your final answer>}
**Important: The \boxed{...} command should only contain the final,
simplified numerical or symbolic answer, with no extra text or units.**
```

K SUCCESS PROBABILITY VERSUS REWARD-VARIABILITY TARGETS

If the success probability of every question were known exactly, a curriculum targeting $p_q \approx 1/2$ and one maximizing the ideal reward-variability score $\sqrt{p_q(1-p_q)}$ would favor the same intermediate-difficulty questions. The distinction becomes important in practice because p_q is not known and must instead be inferred from the question and from finite rollout feedback.

A predicted success probability near $1/2$ does not necessarily mean that the underlying question itself has success probability near $1/2$. For example, the same prediction can arise when the information available to the predictor cannot distinguish between questions that are very easy and questions that are very difficult. Goldilocks does not train the Selector only to predict mean success probability; instead, it directly predicts the reward variability realized by rollout groups.

K.1 SETUP

We analyze a fixed Model checkpoint with parameters θ; all quantities below should therefore be understood as checkpoint-specific. Let $Q \sim \mathcal{D}$ denote a question drawn from the candidate-question distribution, and let

$$H = h(Q)$$

denote the representation of that question available to the Selector.

For a particular question q, define

$$p_q(\theta) = \Pr_{o \sim \pi_\theta(\cdot|q)}\big(r_{\mathrm{ver}}(o) = 1\big),$$

the probability that the current Model produces a correct answer under the binary verification reward. We then define the random variable

$$P = p_Q(\theta),$$

which varies because the randomly drawn question Q varies.

Conditional on the information available to the Selector, we define

$$\mu(h) \coloneqq \mathbb{E}[P \mid H = h], \qquad \tau^2(h) \coloneqq \mathrm{Var}(P \mid H = h). \tag{8}$$

Here, $\mu(h)$ is the mean solvability of questions represented by h, while $\tau^2(h)$ measures how much their true success probabilities can differ even after conditioning on the representation available to the Selector. Thus, $\tau^2(h)$ captures representation-conditional ambiguity rather than finite-sample estimation error.

For a rollout group of size G, let $R_1, \ldots, R_G \in \{0, 1\}$ denote the verification rewards of independently sampled Model outputs. Conditional on a question with success probability P,

$$R_i \mid P \sim \mathrm{Bernoulli}(P).$$

Let

$$S = \sum_{i=1}^{G} R_i, \qquad S \mid P \sim \mathrm{Binomial}(G, P), \qquad \hat{p} = \frac{S}{G}.$$

We use

$$\sigma(p) = \sqrt{p(1-p)}$$

for the standard deviation of a Bernoulli variable with success probability p, and define the empirical Goldilocks target as

$$\hat{\sigma} = \sigma(\hat{p}) = \sqrt{\hat{p}(1-\hat{p})}.$$

The Selector is trained to predict $\hat{\sigma}$. Notice that $\hat{\sigma} = 0$ exactly when $S = 0$ or $S = G$, i.e., when every rollout in the group receives the same verification reward. Such groups contain no within-group reward variation and therefore provide no relative reward signal to GRPO.

We call a rollout group *informative* when it contains both correct and incorrect responses, i.e., when $0 < S < G$. Define

$$\rho(h) \coloneqq \Pr(0 < S < G \mid H = h). \tag{9}$$

Conditioning first on P, the probability of a zero-variance group is $P^G + (1-P)^G$. Averaging over questions represented by h therefore gives

$$\rho(h) = 1 - \mathbb{E}\left[P^G + (1-P)^G \mid H = h\right]. \tag{10}$$

Thus, $\rho(h)$ averages over both variation among questions consistent with the representation h and the randomness of Model rollouts.

K.2 What the two prediction targets learn

We first consider a predictor trained with squared loss on the empirical success rate $\hat{p}$. The population minimizer of squared-error regression is the conditional expectation of its target. Since

$$\mathbb{E}[\hat{p} \mid P] = P,$$

the tower property gives

$$m(h) = \mathbb{E}[\hat{p} \mid H = h] = \mathbb{E}[P \mid H = h] = \mu(h). \tag{11}$$

Thus, in the population limit, a success-probability predictor such as that used by PCL estimates the conditional mean solvability $\mu(h)$. A curriculum targeting success probability $1/2$ therefore favors representations for which $\mu(h)$ is close to $1/2$.

Goldilocks instead trains the Selector on $\hat{\sigma}$. Under the same population squared-loss argument, its prediction target is

$$v(h) = \mathbb{E}[\hat{\sigma} \mid H = h]. \tag{12}$$

Hence, $v(h)$ is the expected empirical reward variability of the rollout group that would be generated after selecting a question represented by h.

K.3 Why the predicted mean can be misleading

The conditional mean $\mu(h)$ does not fully characterize how informative the underlying questions are. To see this, consider the expected Bernoulli reward variance:

$$\begin{aligned}
\mathbb{E}[P(1-P) \mid H = h] &= \mathbb{E}[P \mid H = h] - \mathbb{E}[P^2 \mid H = h] && (13)\\
&= \mu(h) - \left(\tau^2(h) + \mu(h)^2\right) && (14)\\
&= \mu(h)(1-\mu(h)) - \tau^2(h). && (15)
\end{aligned}$$

Therefore, the variance obtained by simply applying the Bernoulli formula to the predicted mean, $\mu(h)(1-\mu(h))$, is larger than the expected within-question reward variance by exactly $\tau^2(h)$.

A related consequence follows for the reward standard deviation. Because the square-root function is concave, Jensen's inequality gives

$$\mathbb{E}[\sigma(P) \mid H = h] \le \sqrt{\mathbb{E}[P(1-P) \mid H = h]} = \sqrt{\mu(h)(1-\mu(h)) - \tau^2(h)} \le \sigma(\mu(h)). \tag{16}$$

Thus, computing variability from the conditional mean can systematically overstate the reward variability of the underlying questions whenever $\tau^2(h) > 0$.

The same issue appears directly in the probability of obtaining a zero-variance rollout group. For $G \ge 2$, the function

$$z(p) = p^G + (1-p)^G$$

is convex on $[0,1]$. Jensen's inequality therefore yields

$$\Pr(\text{zero variance} \mid H = h) = \mathbb{E}[z(P) \mid H = h] \ge z(\mu(h)) = \mu(h)^G + (1-\mu(h))^G. \tag{17}$$

A mean-based estimate therefore gives a lower bound on the actual zero-variance probability conditional on h.

Consequently, a predicted mean near $1/2$ is not sufficient to guarantee that the selected questions are themselves near the Model's solvability frontier. As an extreme illustration, suppose that conditional on some representation h, half of the questions have $P = 0$ and half have $P = 1$. Then

$$\mu(h) = \frac{1}{2},$$

so a mean-based curriculum targeting $1/2$ gives this representation its highest possible score. Nevertheless, every individual question has deterministic outcomes, and every rollout group is zero-variance.

This also explains why an average training reward near $1/2$ is not, by itself, evidence that the selected questions individually have success probability near $1/2$. The same aggregate mean can arise from a mixture of questions near the two extremes.

K.4 The Goldilocks score and informative rollout groups

The Goldilocks target has a direct relationship to the probability of obtaining an informative rollout group. Define

$$c_G \coloneqq \sigma\left(\frac{1}{G}\right) = \frac{\sqrt{G-1}}{G}.$$

For a zero-variance group, $\hat{\sigma} = 0$. For an informative group, $S \in \{1, \dots, G-1\}$, so

$$\hat{p} \in \left[\frac{1}{G}, 1 - \frac{1}{G}\right].$$

Since $\sigma(p) = \sqrt{p(1-p)}$ is symmetric around $1/2$, is increasing on $[0, 1/2]$, and has maximum $1/2$, every informative group satisfies

$$c_G \le \hat{\sigma} \le \frac{1}{2}.$$

Equivalently,

$$c_G \, \mathbf{1}\{0 < S < G\} \le \hat{\sigma} \le \frac{1}{2} \, \mathbf{1}\{0 < S < G\}.$$

Taking conditional expectations given $H = h$ gives

$$c_G \, \rho(h) \le v(h) \le \frac{1}{2} \, \rho(h). \tag{18}$$

Rearranging,

$$2v(h) \le \rho(h) \le \frac{v(h)}{c_G}. \tag{19}$$

Thus, the quantity predicted by the Goldilocks Selector is directly tied to the probability that a selected question produces a mixed-reward rollout group. In particular, a high value of $v(h)$ implies a nontrivial lower bound on the probability of obtaining an informative group. The conditional mean $\mu(h)$ alone does not provide such a guarantee.

The connection to representation-conditional heterogeneity is especially transparent if we examine the *squared* empirical variability target. Since

$$\mathrm{Var}(\hat{p} \mid P) = \frac{P(1-P)}{G}$$

and $\mathbb{E}[\hat{p} \mid P] = P$, we have

$$\mathbb{E}[\hat{p}(1-\hat{p}) \mid P] = \mathbb{E}[\hat{p} \mid P] - \mathbb{E}[\hat{p}^2 \mid P] \tag{20}$$

$$= \left(1 - \frac{1}{G}\right) P(1-P). \tag{21}$$

Taking the conditional expectation over P given $H = h$ and applying Eq. 15 gives

$$\mathbb{E}[\hat{p}(1-\hat{p}) \mid H = h] = \left(1 - \frac{1}{G}\right) \left[\mu(h)(1-\mu(h)) - \tau^2(h)\right]. \tag{22}$$

Thus, for the squared variability target, the contribution suggested by the mean success probability is reduced exactly by the conditional heterogeneity $\tau^2(h)$, up to the finite-group factor $1 - 1/G$.

Goldilocks uses the standard-deviation target $\hat{\sigma} = \sqrt{\hat{p}(1-\hat{p})}$ rather than its square, so Eq. 22 is not itself the exact population target learned by Goldilocks. The direct relationship for the actual Goldilocks target is instead given by the informative-group bounds in Eq. 18.

Overall, this analysis predicts that a success-probability curriculum can maintain an average selected reward near $1/2$ while still selecting a heterogeneous mixture of very easy and very difficult questions. Goldilocks instead trains the Selector directly on realized within-group reward variability, which is tied to the probability of obtaining mixed-reward groups. We therefore expect Goldilocks to produce fewer zero-variance rollout groups even when another curriculum maintains an average reward closer to $1/2$. This prediction is consistent with the training dynamics in Figure 3.