INTERNALIZING CURRICULUM JUDGMENT FOR LLM REINFORCEMENT FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In LLM Reinforcement Fine-Tuning (RFT), curriculum learning drives both efficiency and performance. Yet, current methods externalize curriculum judgment via handcrafted heuristics or auxiliary models, risking misalignment with the policy's training dynamics. In this paper, we introduce METIS (METacognitive Internalized Self-judgment), a novel framework that internalizes curriculum judgment as a native capability. Leveraging a critical observation that within-prompt reward variance effectively gauges prompt informativeness, METIS predicts this metric based on recent training outcomes as lightweight in-context learning examples. This intrinsic self-judgment then dynamically dictates the training allocation. Moreover, METIS closes the loop between judgment and optimization by jointly optimizing the standard RFT rewards and a self-judgment reward. This allows the policy to learn what to learn next, as a form of metacognition. Across mathematical reasoning, code generation, and agentic function-calling benchmarks, METIS delivers superior performance while achieving up to a $2.1\times$ training speedup, with controlled ablations and in-depth analysis further validating the benefits of internalized curriculum judgment. By bypassing handcrafted heuristics and auxiliary models, our work establishes a simple, closed-loop, and highly efficient curriculum internalization paradigm for LLM reinforcement fine-tuning.

1 INTRODUCTION

Effective learning depends not only on how a learner updates from feedback, but also on what it chooses to practice. Problems already mastered offer little information, while problems far beyond current ability often provide little actionable signal. The most useful practice lies near the boundary of current competence, echoing *zone of proximal development* in education (Vygotsky, 1978). Because this frontier shifts as learning progresses, a capable learner should not only learn to solve problems, but also monitor their own learning state and decide which problems to learn next. This suggests a form of metacognition: using self-assessment to guide future learning effectively (Flavell, 1979).

This principle has become increasingly relevant in reinforcement fine-tuning (RFT) for large language models (LLMs) (Ouyang et al., 2022; Shao et al., 2024; Guo et al., 2025), which is operationalized as *curriculum learning*. As rollouts are expensive in LLM RFT, the curriculum is not merely a data reordering: it determines how training compute is allocated and therefore directly affects both efficiency and performance. A good curriculum should prioritize prompts that are *informative* for the current policy, avoiding those that are already solved or consistently beyond its capability.

A key signal of such *prompt informativeness* arises naturally in group-relative RFT (Shao et al., 2024; Yu et al., 2025; Zheng et al., 2025a), which evaluates multiple rollouts per prompt. When all rollouts receive the same reward, the prompt provides little relative signal since it is consistently solved or failed. In contrast, diverse rollout outcomes indicate that the prompt lies near the policy's competence frontier and can induce stronger updates. Therefore, *within-prompt reward variance* provides a simple proxy for prompt informativeness (Jiang et al., 2025; Sun et al., 2026; Gao et al., 2025). In this paper, we formalize and extend such a proxy to handle both discrete and continuous rewards in LLM RFT.

Moreover, prompt informativeness is inherently dynamic. As the policy improves, previously insurmountable prompts may become informative, while once-useful prompts eventually become saturated. Existing curricula address this through various forms of adaptation (see Fig. 1): some rely on predefined coarse prompt groups based on domains or data sources (Wang et al., 2025a; Chen et al.,

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.

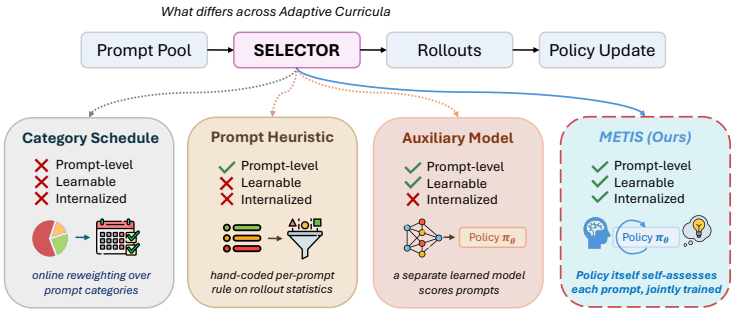


Figure 1: Conceptual comparison. While existing curricula rely on external schedules, heuristics, or auxiliary models, METIS internalizes curriculum judgment via in-context self-assessment, closing the loop by jointly optimizing standard RFT and self-judgment rewards.

2025b), others apply hand-coded heuristics over rollout statistics or offline difficulty labels (Zhang et al., 2025; Parashar et al., 2026; Pang et al., 2025; Jiang et al., 2025; Zheng et al., 2025b), and the rest train a separate model to predict (Sun et al., 2026; Gao et al., 2025). Crucially, curriculum judgment remains *external* to the policy across all these paradigms. But because the competence frontier moves with the policy as it learns, accurate curriculum judgment requires tracking the policy's current state. Thus, we argue that the policy itself is the entity most directly aware of that state. As modern LLMs have strong reasoning capabilities, it could be elicited to read its own rollout outcomes and predict which prompts will be informative. However, current methods neglect such metacognitive potential, treating the policy as a passive recipient rather than an active architect of its own curricula.

This motivates *curriculum internalization*: making curriculum judgment a capability of the policy itself, rather than relying on another auxiliary head. During training, rollout outcomes provide direct evidence about the policy's current learning state. Internalizing curriculum judgment exposes the policy to this evidence directly and trains it to map recent prompt-outcome patterns to future prompt informativeness. The goal is therefore not only to select better prompts for the model, but to make the ability to judge useful prompts part of the model's learned competence. This reframes curriculum learning from selecting data to training the model to decide what it should learn from next.

To this end, we introduce METIS (**MET**acognitive **I**nternalized **S**elf-judgment), a novel framework that internalizes curriculum judgment as a native policy capability. METIS trains the policy to predict the expected within-prompt reward variance of each prompt candidate as a measure of informativeness, using lightweight in-context examples constructed from recent training outcomes. The predicted scores then determine how the rollout budget is allocated. After rollout, the obtained data can be utilized for two coupled learning signals. The original task rewards improve the policy's problem-solving behavior, while we introduce a judgment reward that calibrates the policy's understanding of which prompts are informative. Together, METIS enables the policy to learn not only how to solve tasks, but also what to learn next, closing the loop between judgment and optimization.

Our main contributions are threefold: 1) Conceptually, we formulate curriculum learning for LLM RFT as policy-dependent competence-frontier tracking and identify within-prompt reward variance as a general informativeness signal across discrete and continuous rewards; 2) Technically, we propose METIS, the first LLM RFT framework to internalize curriculum judgment, while jointly optimizing task rewards and a self-judgment reward, transforming curriculum into a learned capability of the policy rather than an external selection mechanism; 3) Empirically, METIS improves performance across mathematical reasoning, code generation, and agentic function-calling benchmarks, yielding up to 2.1 $\times$ wall-clock speedups with only around 3.9% per-step overhead; controlled ablations and analyses further show that internalized judgment drives these gains while improving prompt coverage and training stability. METIS presents a metacognitive framework for efficient LLM RFT.

2 Related Work

Reinforcement fine-tuning (RFT). Reinforcement fine-tuning methods vary in reward type and optimization objective. RLHF typically uses preference-based rewards with PPO-style updates (Ouyang et al., 2022; Bai et al., 2022; Zheng et al., 2023), while critic-free alternatives such as RLOO, ReMax,

and REINFORCE++ simplify optimization (Ahmadian et al., 2024; Li et al., 2024; Hu et al., 2025). For reasoning tasks, recent work increasingly uses verifiable rewards (Shao et al., 2024; Guo et al., 2025), with group-relative methods such as GRPO, DAPO, and GSPO estimating advantages from multiple rollouts per prompt (Shao et al., 2024; Yu et al., 2025; Zheng et al., 2025a). Orthogonal to the update algorithm, METIS focuses on curriculum learning, i.e., which prompts the policy trains on in each RFT update.

Curriculum learning for RFT. We group existing methods by how curriculum judgment is produced (Fig. 1). Some methods operate at the category or data-source level, such as DUMP (Wang et al., 2025a) and SEC (Chen et al., 2025b), while others make prompt-level decisions using difficulty estimates, reward dynamics, or rollout statistics, including ADCL (Zhang et al., 2025), SPEED-RL (Zhang et al., 2026), Dynamic Sampling (Yu et al., 2025), VCRL (Jiang et al., 2025), and GRESO (Zheng et al., 2025b). More recent methods such as PCL (Gao et al., 2025) and attention-based predictors (Sun et al., 2026) learn the selector itself, but keep it separate from the task policy. In contrast, METIS *internalizes* curriculum judgment by letting the policy make prompt-level curriculum judgments directly and updates this capability online from realized rollout outcomes.

In-context learning and model self-assessment. LLMs can adapt to new tasks through in-context learning (ICL) (Brown et al., 2020; Dong et al., 2024; Wu et al., 2025; Liu et al., 2021; Min et al., 2022; Rubin et al., 2022) and estimate uncertainty in their own predictions (Kadavath et al., 2022; Lin et al., 2022; Kumaran et al., 2026; Ling et al., 2024; Binder et al., 2024). METIS unites and extends these two capabilities for curriculum self-judgment in RFT.

3 Problem Formulation

We consider reinforcement fine-tuning (RFT) of a language-model policy π_θ over a prompt dataset $\mathcal{P}$. For each prompt $x \in \mathcal{P}$, the policy samples a rollout $y \sim \pi_\theta(\cdot \mid x)$ and receives a bounded scalar reward $r(x, y) \in [0, 1]$. This formulation covers both binary verifiable rewards ($r \in \{0, 1\}$), common in mathematical reasoning, and continuous rewards such as partial test-case credit in code generation. Notably, METIS operates uniformly across *both* regimes, whereas pass-rate heuristics used in prior work (Yu et al., 2025; Jiang et al., 2025; Sun et al., 2026; Gao et al., 2025; Chen et al., 2025b; Wang et al., 2025a; Zhang et al., 2025) are restricted to the binary case.

Group-relative reinforcement fine-tuning. We develop METIS on top of the prevailing group-relative RFT methods that sample multiple rollouts per prompt (Shao et al., 2024; Yu et al., 2025; Zheng et al., 2025a; Chen et al., 2025a). At iteration t, a batch of prompts $\mathcal{S}_t$ is selected for rollout. For each prompt $x \in \mathcal{S}_t$, the policy generates n rollouts $\{y_i\}_{i=1}^n$ with rewards $r_i = r(x, y_i)$, and a centered group-relative advantage is constructed as

$$A_i(x) = r_i - \bar{r}(x), \qquad \bar{r}(x) = \frac{1}{n}\sum_{j=1}^{n} r_j. \tag{1}$$

Following Liu et al. (2025), we drop the per-prompt standard-deviation normalization originally used in GRPO (Guo et al., 2025).The resulting policy loss takes a clipped surrogate form:

$$\mathcal{L}_{\text{policy}}(\theta) = -\frac{1}{|\mathcal{S}_t| n} \sum_{x \in \mathcal{S}_t} \sum_{i=1}^{n} \min\big(\rho_i(\theta) A_i(x),\ \text{clip}\big(\rho_i(\theta), 1-\epsilon, 1+\epsilon\big) A_i(x)\big)\,, \tag{2}$$

where $\rho_i(\theta) = \pi_\theta(y_i \mid x)/\pi_{\theta_{\text{old}}}(y_i \mid x)$. Note that METIS is agnostic to the specific normalization, clipping, or KL-regularization choices made by individual algorithms. It requires only that multiple rollout rewards are available per prompt so that within-prompt outcome variability can be estimated.

Quantifying prompt informativeness via within-prompt reward variance. In group-relative RFT, prompts with diverse rollout rewards yield stronger relative learning signals than those with similar rewards (Gao et al., 2025). We formalize this and extend it to continuous rewards: by Eq. (1), the mean squared advantage for prompt x equals the empirical reward variance across its n rollouts,

$$v(x) \;\coloneqq\; \frac{1}{n}\sum_{i=1}^{n} A_i(x)^2 \;=\; \frac{1}{n}\sum_{i=1}^{n}\big(r_i - \bar{r}(x)\big)^2, \tag{3}$$

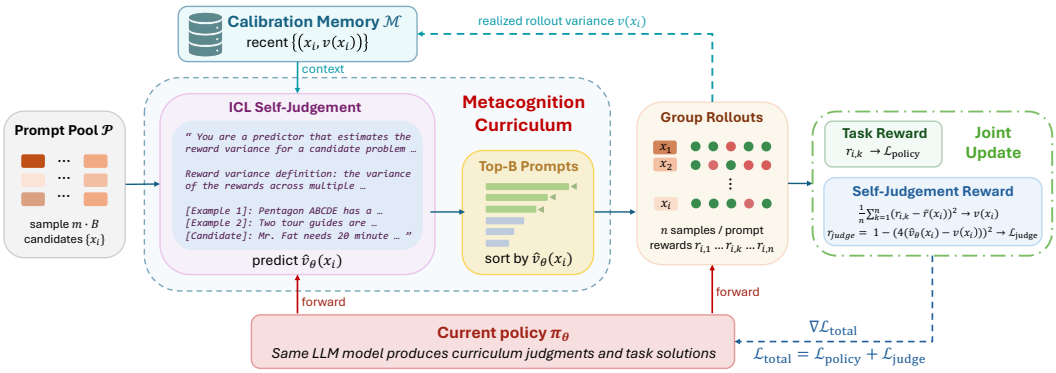


Figure 2: Overview of METIS. At each iteration, the policy predicts candidate informativeness $\hat{v}_\theta(x)$ via in-context learning on a calibration memory of recent prompt-variance pairs (**left**). The most informative prompts are then rolled out (**middle**) to yield task rewards and realized variance $v(x)$. The policy is jointly optimized (**right**) via standard task loss on solution tokens and a self-judgment loss on prediction tokens, while the calibration memory refreshes with the latest variances.

which vanishes when all rollouts receive the same reward (uniformly solved or failed prompts) and increases as their outcomes become more diverse. To remove the dependence on a particular set of sampled rollouts, we define the corresponding population quantity under the current policy:

$$v_\theta(x) \;:=\; \mathrm{Var}_{y \sim \pi_\theta(\cdot|x)}[r(x, y)]\,, \tag{4}$$

with $v(x)$ serving as its empirical estimate. Importantly, we note that the subscript θ is essential, as $v_\theta(x)$ is *policy-dependent* and changes throughout training. For rewards bounded in $[0, 1]$, $v_\theta(x) \leq 1/4$, with equality attained when the reward takes values 0 and 1 with equal probability.[1]

The same formulation applies to both binary and continuous rewards. For binary rewards, $v_\theta(x) = p(x)(1 - p(x))$, where $p(x)$ is the per-prompt success probability and estimated empirically by the rollout pass rate. It is maximized at $p(x) = 1/2$, recovering the intermediate-difficulty criterion used in prior binary-curriculum methods (Jiang et al., 2025; Zheng et al., 2025b; Gao et al., 2025; Sun et al., 2026). For continuous rewards, a pass rate is no longer defined, but $v_\theta(x)$ still captures variation in rollout quality and thus the strength of the learning signal. This allows METIS to handle tasks such as code generation and agentic function calling without a task-specific notion of difficulty.

Curriculum as competence-frontier tracking. A *curriculum* is a selector that constructs a batch of B prompts for rollout. Taking informativeness to be the policy-dependent variance $v_\theta(x)$ in Eq. (4), the *ideal* curriculum selects the most informative prompts from a candidate pool $\mathcal{C}_t$:

$$\mathcal{S}_t^\star = \mathrm{TopB}_{x \in \mathcal{C}_t} v_\theta(x), \qquad |\mathcal{S}_t^\star| = B. \tag{5}$$

Since $v_\theta(x)$ changes with the policy, the selected prompts form a moving *competence frontier* throughout training. Existing methods approximate this frontier using external schedules, heuristics, or auxiliary selector models (e.g., Gao et al. (2025)). In each case, informativeness is externally determined rather than learned or refined by the policy, risking misalignment with its evolving competence. METIS instead trains the policy to predict prompt informativeness, as detailed in Sec. 4.

4 Method

METIS internalizes curriculum learning as a native metacognitive skill via a two-fold design (see Fig. 2). First, the policy self-judges prompt informativeness pre-rollout through *in-context learning* (Sec. 4.1). Second, the realized rollout variance provides a calibration reward to update this predictive capability online, closing the loop between task optimization and curriculum judgment (Sec. 4.2–4.3).

4.1 Self-Judgment via In-Context Learning

The previous section framed curriculum selection as predicting the policy-dependent informativeness $v_\theta(x)$ of each candidate prompt and selecting those with the highest predicted values. In METIS,

[1] For any $R \in [0, 1]$ with mean μ, $R^2 \leq R$ implies $\mathrm{Var}(R) \leq \mu(1 - \mu) \leq 1/4$.

these predictions are generated by the policy itself through prompt-level in-context self-assessment. Given a candidate x, the policy conditions on a compact context drawn from the recent training history $\mathcal{H}_t$ and produces a short structured prediction of its expected informativeness.

We use ICL for curriculum self-judgment for four main reasons: **(i) Predictive.** The policy estimates $v_\theta(x)$ before task rollouts, enabling candidates to be ranked by predicted informativeness rather than filtered only after observing outcomes (Yu et al., 2025; Wang et al., 2025a; Chen et al., 2025b). **(ii) Adaptive.** Recent prompt-variance examples reflect the policy's current competence, allowing its judgment to evolve throughout training. **(iii) Efficient.** Each candidate requires only a short judgment response, rather than the n full rollouts needed to estimate variance directly (Zhang et al., 2025; Jiang et al., 2025), adding only a small fraction of the per-step training cost (Sec. 5.1). **(iv) Compatible.** Unlike a separate value-head predictor (see more discussions and results in Sec. 5.2), ICL judgments are expressed as language tokens, making them compatible with the original task tokens and facilitating more natural joint optimization and internalization in Sec. 4.3.

We instantiate the in-context evidence as a **calibration memory** $\mathcal{M}_t = \{(x_k, v(x_k))\}_{k=1}^{K} \subset \mathcal{H}_t$, using prompt-variance pairs from the previous training step, providing recent examples of informativeness under the evolving policy. To include examples with low, medium, and high informativeness as ICL examples, we use three reference variance values, 0, 0.10, and 0.25. For each value, we select the $K/3$ prompts whose realized variances are closest to it. Note that these reference variances are used only for exemplar retrieval; the stored variances themselves remain their realized values.

For each candidate $x \in \mathcal{C}_t$, we then construct $\mathrm{ctx}_t(x) = [\, I_{\mathrm{judge}}; \mathcal{M}_t; x \,]$, where I_{judge} specifies the instructions for the variance-prediction task and output format. Since $v_\theta(x) \in [0, 1/4]$ (Sec. 3), the instruction asks the policy to assess the candidate and generate a short response $z_x \sim \pi_\theta(\cdot \mid \mathrm{ctx}_t(x))$ containing a boxed estimate in this range (e.g., `\boxed{0.15}`). We deterministically parse this value as the predicted informativeness, $\hat{v}_\theta(x) = \mathrm{Parse}(z_x)$. Responses without a valid prediction are treated as judgment failures (App. B.6). The rollout batch is then selected as $\mathcal{S}_t = \mathrm{TopB}_{x \in \mathcal{C}_t} \hat{v}_\theta(x)$. The full prompt template and a sample judgment response are provided in App. A.1 and App. B.7.

This formulation keeps curriculum judgment in the same token-generation space as task solving, leveraging the policy's existing ICL and self-evaluation capabilities (Ren et al., 2023; Kumaran et al., 2026). Prompt selection is thus driven directly by the policy's own judgment rather than by an external heuristic or auxiliary selector.

4.2 Internalizing Curriculum Judgment

In-context elicitation alone makes the policy's self-judgment *available*, but does not make it *accurate*. METIS closes this gap by reinforcing curriculum judgment from realized outcomes.

For each selected prompt $x \in \mathcal{S}_t$, we generate n task rollouts, compute rewards $\{r_i\}_{i=1}^{n}$, and obtain the realized variance $v(x)$ from Eq. (3). We score each pre-rollout prediction against its realized counterpart with a squared-error calibration reward (Brier, 1950),

$$R_{\mathrm{judge}}(x) \;=\; 1 - \big(4(\hat{v}_\theta(x) - v(x))\big)^2, \tag{6}$$

which is maximized only when the pre-rollout judgment matches the realized variance. The supervision is fully endogenous, coming from the rollouts the policy just generated as part of normal RFT, with no *additional labels*, no *auxiliary model*, and no *extra rollout cost*.

Because each prompt produces a single judgment response, we optimize the judgment tokens with a REINFORCE-style objective using an exponential moving-average (EMA) baseline (Ahmadian et al., 2024):

$$\nabla_\theta J_{\mathrm{judge}} = \frac{1}{|\mathcal{S}_t|} \sum_{x \in \mathcal{S}_t} \big(R_{\mathrm{judge}}(x) - b_t\big) \nabla_\theta \log \pi_\theta\big(z_x \mid \mathrm{ctx}_t(x)\big), \tag{7}$$

where we denote the corresponding minimization objective by $\mathcal{L}_{\mathrm{judge}}$. The EMA baseline is defined as follows, where $\alpha \in [0, 1)$ controls smoothing rate:

$$b_t = \alpha b_{t-1} + (1 - \alpha)\, \overline{R}_{\mathrm{judge},t}, \qquad \overline{R}_{\mathrm{judge},t} = \frac{1}{|\mathcal{S}_t|} \sum_{x \in \mathcal{S}_t} R_{\mathrm{judge}}(x). \tag{8}$$

Algorithm 1 METIS: Metacognitive Internalized Self-Judgment for RFT

Require: policy π_θ; prompt dataset $\mathcal{P}$; batch size B, pool multiplier m, group rollouts size n, judgment weight λ, EMA rate α, self-judgment instruction I_{judge}

Initialize baseline $b \leftarrow 0$
for iteration $t = 1, 2, \ldots$ **do**
  Construct the calibration memory $\mathcal{M}_t$ from training history $\mathcal{H}_t$.
  Sample $\mathcal{C}_t \subset \mathcal{P}$, $|\mathcal{C}_t| = mB$.
  For each $x \in \mathcal{C}_t$: generate $z_x \sim \pi_\theta(\cdot \mid [I_{\text{judge}}; \mathcal{M}_t; x])$ and parse $\hat{v}_\theta(x)$.
  Select $\mathcal{S}_t \leftarrow \text{TopB}_{x \in \mathcal{C}_t}\, \hat{v}_\theta(x)$.
  For each $x \in \mathcal{S}_t$: sample $\{y_i\}_{i=1}^n \sim \pi_\theta(\cdot \mid x)$, compute rewards $\{r_i\}_{i=1}^n$, realized variance $v(x) = \frac{1}{n}\sum_i (r_i - \bar{r}(x))^2$, and $R_{\text{judge}}(x) = 1 - (4(\hat{v}_\theta(x) - v(x)))^2$.
  Update θ on $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{policy}} + \lambda\, \mathcal{L}_{\text{judge}}$.
  Refresh $\mathcal{H}_t$ with $\{(x, v(x))\}_{x \in \mathcal{S}_t}$; update baseline $b \leftarrow \alpha b + (1 - \alpha)\, \overline{R}_{\text{judge}}$.
end for

This objective makes curriculum judgment a trainable behavior of the policy: predictions that correctly anticipate informative prompts are reinforced, while miscalibrated judgments (e.g., overconfident on saturated prompts, or dismissive of high-variance ones) are suppressed. As training progresses, this allows the induced curriculum to tightly track the policy's advancing competence frontier.

4.3 Joint Optimization

METIS jointly optimizes two token streams with the same policy: task-solution tokens under the group-relative RFT loss $\mathcal{L}_{\text{policy}}$ in Eq. (2), and self-judgment tokens under the calibration loss $\mathcal{L}_{\text{judge}}$:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{policy}} + \lambda\, \mathcal{L}_{\text{judge}}, \tag{9}$$

where λ controls the judgment-update strength. Jointly updating the same policy, task rewards improve problem solving while calibration rewards improve prediction of prompt informativeness, allowing the curriculum to co-evolve with competence without hand-tuned schedules, predefined difficulty, or auxiliary selectors. Algorithm 1 summarizes the full procedure.

5 Experiments

Models and datasets. We span three task families covering discrete and continuous rewards across single and multi-turn rollouts. For *mathematical reasoning* (binary reward), we train on DAPO-17k (Yu et al., 2025) and evaluate on AIME24/25 (Mathematical Association of America, 2024), MATH500 (Hendrycks et al., 2021), and Minerva (Lewkowycz et al., 2022). For *code generation* (continuous reward), we train on CodeContests+ (Wang et al., 2025b) and evaluate on LiveCodeBench v6 (Jain et al., 2024), HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), and BigCodeBench (Zhuo et al., 2025). For *agentic function-calling* (continuous, multi-turn), we train and evaluate on BFCL v3 (Patil et al., 2025). We use Qwen3-4B/8B-Base/Instruct (Yang et al., 2025), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025) as backbone models. Dataset preprocessing and evaluation details are provided in App. A.2.

RFT backbone and engine. All methods use GRPO (Shao et al., 2024; Guo et al., 2025) running via the VERL framework (Sheng et al., 2025) on identical hardware. Full details are in App. A.4.

Baselines. We compare METIS with **No Curriculum** (uniform prompt sampling) and one representative method from each baseline category in Fig. 1: **SEC** (Chen et al., 2025b) for category-level scheduling, **ADCL** (Zhang et al., 2025) for heuristic prompt-level curricula, and **PCL** (Gao et al., 2025) for auxiliary-model-based selection. Since these methods were originally developed for binary rewards, we adapt them to continuous-reward tasks where necessary (App. A.5). We omit closely related methods within the same categories to avoid redundant comparisons. DUMP (Wang et al., 2025a) is similar in spirit to SEC; VCRL (Jiang et al., 2025) and Sun et al. (2026) are not readily applicable to continuous rewards; and SPEED-RL (Zhang et al., 2026), Dynamic Sampling (Yu et al., 2025), and GRESO (Zheng et al., 2025b) have already been shown to be outperformed by PCL.

Table 1: Main results (mean ± std over three seeds). **Avg.** denotes the average Pass@1 at the best checkpoint, **Time** denotes the hours required to reach it, and **Spd.** denotes the speedup over No Curriculum, $T_{\text{No Curr.}}/T_{\text{method}}$. **Bold** and underlined values indicate the best and the second-best.

Method	AIME24	AIME25	MATH500	Minerva	Avg. ↑	Time ↓	Spd. ↑	LCBv6	HumanEval	MBPP	BCB	Avg. ↑	Time ↓	Spd. ↑
Model	**DAPO-17K (Mathematical Reasoning)**							**CodeContests+ (Code Generation)**						
	Qwen3-8B Base							**Qwen3-8B Base**						
No Curr.	20.3 ± 0.9	17.9 ± 0.5	77.7 ± 1.2	27.0 ± 2.2	35.7 ± 0.7	31.4 ± 3.8	1.0×	31.6 ± 0.5	85.2 ± 0.4	68.9 ± 0.8	26.7 ± 1.1	53.1 ± 0.7	16.7 ± 5.7	1.0×
SEC	20.6 ± 2.0	18.3 ± 0.6	76.9 ± 1.6	27.8 ± 1.7	35.9 ± 0.8	20.9 ± 6.6	1.5×	32.5 ± 0.8	83.5 ± 1.1	68.9 ± 0.2	27.8 ± 1.8	53.2 ± 0.9	16.0 ± 6.4	1.0×
ADCL	20.9 ± 1.1	18.7 ± 1.0	74.9 ± 1.6	27.9 ± 2.2	35.6 ± 0.8	34.6 ± 5.4	0.9×	31.3 ± 0.3	85.0 ± 1.3	69.1 ± 0.8	26.2 ± 0.9	52.9 ± 0.8	24.3 ± 6.6	0.7×
PCL	21.4 ± 1.9	19.6 ± 0.2	76.5 ± 1.6	27.3 ± 2.4	36.2 ± 0.5	19.4 ± 3.2	1.6×	31.5 ± 0.2	85.6 ± 1.3	68.3 ± 0.8	27.7 ± 2.4	53.3 ± 0.7	15.0 ± 3.6	1.1×
METIS	22.1 ± 1.0	19.0 ± 0.3	77.8 ± 1.1	28.2 ± 0.6	**36.8** ± 0.4	**15.3** ± 2.2	**2.1**×	32.2 ± 0.3	86.2 ± 0.4	70.2 ± 0.5	27.8 ± 1.0	**54.1** ± 0.4	**11.5** ± 2.8	**1.5**×
Model	**Qwen3-4B Base**							**Llama-3.1-8B-Instruct**						
No Curr.	17.2 ± 1.0	10.4 ± 0.8	64.1 ± 2.9	18.6 ± 2.1	27.6 ± 1.0	11.9 ± 2.4	1.0×	17.5 ± 0.4	64.4 ± 1.4	57.7 ± 0.5	18.4 ± 0.6	39.5 ± 0.7	13.2 ± 6.2	1.0×
SEC	15.1 ± 1.4	9.4 ± 1.5	63.5 ± 1.9	17.6 ± 1.3	26.4 ± 0.8	**8.4** ± 2.1	**1.4**×	16.4 ± 1.2	65.4 ± 0.7	58.2 ± 0.2	20.0 ± 0.8	40.0 ± 0.7	14.5 ± 5.9	0.9×
ADCL	17.0 ± 1.0	9.2 ± 0.7	64.9 ± 0.7	16.8 ± 1.7	27.0 ± 0.6	15.8 ± 3.1	0.8×	17.8 ± 0.4	64.4 ± 1.8	57.9 ± 0.6	18.6 ± 1.1	39.7 ± 0.8	18.0 ± 3.8	0.7×
PCL	18.4 ± 1.5	11.4 ± 1.1	65.1 ± 0.1	17.8 ± 0.8	28.2 ± 0.5	13.8 ± 1.9	0.9×	17.4 ± 0.7	64.8 ± 0.9	58.6 ± 0.7	21.4 ± 1.4	40.6 ± 0.5	**7.4** ± 2.5	**1.8**×
METIS	20.4 ± 1.0	9.3 ± 0.7	64.8 ± 0.4	20.6 ± 1.3	**28.8** ± 0.5	10.6 ± 3.8	1.1×	18.3 ± 0.6	66.1 ± 0.7	59.5 ± 0.5	21.3 ± 1.3	**41.3** ± 0.5	8.3 ± 2.7	1.6×
Model	**Llama-3.1-8B-Instruct**							**DS-R1-Distill-Llama-8B**						
No Curr.	14.8 ± 1.2	6.4 ± 0.8	50.1 ± 2.1	18.9 ± 2.8	22.6 ± 0.9	8.4 ± 3.3	1.0×	39.2 ± 0.5	79.9 ± 0.6	64.3 ± 0.3	13.6 ± 1.1	49.3 ± 0.2	18.2 ± 6.6	1.0×
SEC	16.6 ± 0.8	7.6 ± 0.8	49.9 ± 2.8	16.2 ± 1.0	22.6 ± 0.8	**5.5** ± 3.1	**1.5**×	38.0 ± 0.3	77.4 ± 0.6	65.7 ± 0.2	12.6 ± 1.2	48.4 ± 0.4	17.2 ± 4.5	1.1×
ADCL	16.9 ± 1.0	6.8 ± 0.8	50.3 ± 0.7	19.0 ± 0.9	23.2 ± 0.5	24.3 ± 4.2	0.3×	36.9 ± 0.2	77.6 ± 0.9	66.4 ± 1.3	12.0 ± 1.9	48.2 ± 0.9	34.0 ± 8.3	0.5×
PCL	17.0 ± 1.0	8.1 ± 0.8	49.5 ± 1.2	22.3 ± 1.2	**24.2** ± 0.6	19.0 ± 4.0	0.4×	38.6 ± 0.4	77.4 ± 1.2	68.1 ± 0.8	12.8 ± 1.6	49.2 ± 0.7	14.4 ± 3.8	1.3×
METIS	15.2 ± 1.2	9.1 ± 0.7	49.8 ± 0.7	19.4 ± 0.9	23.4 ± 0.4	6.8 ± 2.9	1.2×	39.8 ± 1.1	78.9 ± 0.7	65.9 ± 0.5	14.9 ± 0.8	**49.9** ± 0.2	**10.5** ± 3.1	**1.7**×

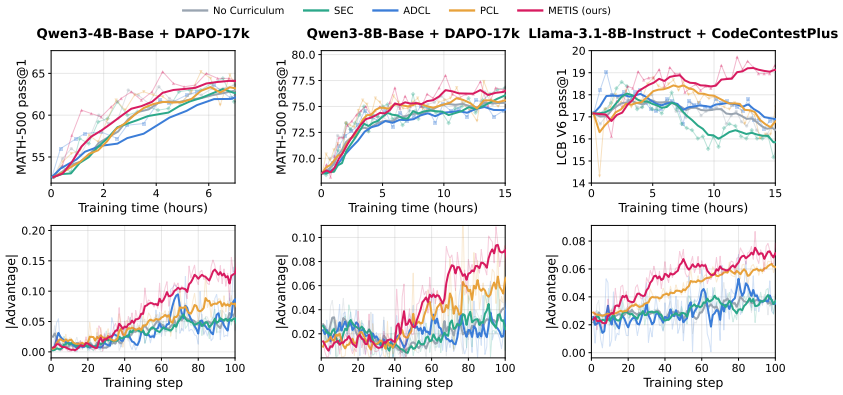


Figure 3: Training dynamics of METIS. **Top:** downstream pass@1 versus wall-clock training time. **Bottom:** mean magnitude of the group-relative advantage $|A|$ per training step.

5.1 Main Results on METIS Performance

We run each experiment with three random seeds and select, for each run, the checkpoint with the highest average Pass@1 across all benchmarks. Tab. 1 reports the mean and standard deviation of the selected checkpoints' Pass@1 scores and the wall-clock time required to reach them. Fig. 3 shows the training dynamics. Results on agentic benchmarks are provided in Tab. 7 (in App. B.2).

(i) Higher accuracy with less training time. METIS achieves the highest average accuracy in nearly every row of Tab. 1 and reaches its best checkpoint up to 2.1× faster than No Curriculum. It outperforms the strongest baseline, PCL, without an auxiliary scorer. SEC trains the fastest in several settings but consistently trails on accuracy since its category-level re-weighting misses the prompt-level informativeness signal. Finally, ADCL is the costliest baseline, since its offline difficulty re-estimation pass adds substantial overhead and its fixed schedule does not co-evolve with the policy.

(ii) Faster convergence. Fig. 3 (top row) plots pass@1 versus training time. METIS reaches the same level of accuracy earlier in training and continues improving thereafter, indicating that gains come from steady accumulation of learning signal rather than a one-time selection effect.

(iii) Larger learning signal per step. Fig. 3 (bottom row) tracks the mean magnitude of the group-relative advantage $|A|$, which is the proxy for the prompt informativeness $v_\theta(x)$ defined in Eq. (3). METIS sustains substantially higher $|A|$ throughout training, indicating that its curriculum selection consistently allocates updates to more informative prompts.

Table 2: Effects of curriculum judgment internalization on Qwen3-4B-Base + DAPO-17k. **Internal**: the task-solving policy produces the curriculum judgment; **External**: a separate duplicated model produces the curriculum judgment. **Judgment Learning**: whether the judge is updated from realized variances ($\mathcal{L}_{\text{judge}}$ for METIS; MSE regression for PCL's value head).

Method	Judgement Provider	Judgment Learning	AIME24	AIME25	MATH500	Minerva	Avg. ↑
METIS (default)	Internal (ICL)	✓	20.4 ± 1.0	9.3 ± 0.7	64.8 ± 0.4	20.6 ± 1.3	**28.8** ± 0.5
METIS	Internal (ICL)	✗	20.1 ± 1.1	10.1 ± 0.8	63.3 ± 0.8	20.2 ± 1.0	28.4 ± 0.6
METIS	External (ICL)	✓	19.0 ± 0.9	7.2 ± 0.8	63.6 ± 1.0	19.4 ± 0.9	27.3 ± 0.6
METIS	External (ICL)	✗	18.8 ± 0.8	6.4 ± 0.7	64.0 ± 0.8	18.5 ± 0.8	26.9 ± 0.5
PCL (default)	External (value head)	✓	18.4 ± 1.5	11.4 ± 1.1	65.1 ± 0.1	17.8 ± 0.8	28.2 ± 0.5
PCL	Internal (value head)	✓	15.6 ± 1.0	8.2 ± 0.8	64.7 ± 1.0	18.0 ± 0.7	26.6 ± 0.6

Table 3: Effects of METIS designs on Qwen3-4B-Base + DAPO-17k, comparing full METIS (K=3, λ=0.01) against removing in-context evidence (K=0) and the joint judgment loss (λ=0), and baselines (No Curriculum and PCL).

Method	AIME24	AIME25	MATH500	Minerva	Avg. ↑	Fail. (%) ↓
No Curriculum	17.0	10.3	66.4	16.2	27.5	—
PCL	19.3	10.7	65.0	18.4	28.3	—
w/o ICL	17.0	7.0	64.6	21.0	27.4	87.6
w/o $\mathcal{L}_{\text{judge}}$	18.0	9.7	64.4	20.2	28.1	2.2
METIS	19.3	9.3	65.2	21.7	**28.9**	**0.3**

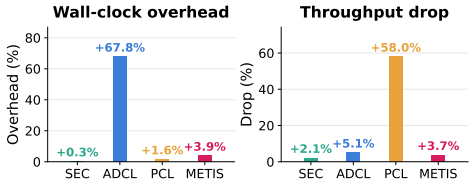


Figure 4: Compute overhead of curriculum methods, measured against No Curriculum. **Left:** wall-clock overhead. **Right:** per-step throughput drop. Lower is better on both axes.

Computational costs. Fig. 4 reports the per-step wall-clock increase (left) and relative throughput (tokens/s/GPU) drop (right), relative to No Curriculum. METIS adds only a short judgment pass over $m \cdot B$ candidates, yielding marginal costs on both metrics. In contrast, ADCL incurs wall-clock overhead from extra difficulty-estimation rollouts, while PCL reduces throughput by training a separate value model. SEC is cheapest but sacrifices accuracy through coarse category-level curriculum. METIS is a unique prompt-level method that remains efficient on both axes, providing effective curriculum at near-negligible cost.

5.2 Ablation and Hyper-parameter Studies

Ablations on curriculum judgment internalization. To validate our claim on internalized judgment, in Tab. 2, we ablate the judgment provider and judgment learning on Qwen3-4B-Base with DAPO-17k. For METIS, the internal variant uses the task policy as the ICL judge, whereas the external variant uses a separate, identically initialized model; replacing ICL with an auxiliary value head yields the PCL-style variants. Internalization raises METIS's average score from 27.3 to 28.8, while judgment learning further improves performance in both settings. Moreover, even without judgment supervision, internal METIS outperforms its supervised external variant (28.4 vs. 27.3), suggesting that task learning transfers to judgment through the shared backbone. Notably, internalizing PCL's value head lowers performance from 28.2 to 26.6, suggesting that token-based ICL may be more compatible with policy optimization than auxiliary scalar prediction using an MSE loss.

Ablations on METIS component designs. Tab. 3 compares full METIS against two reduced variants, reporting downstream performance and the *failure rate*, defined as the percentage of self-judgment responses from which a valid $\hat{v}_\theta(x)$ cannot be parsed. When w/o ICL, the parse-failure rate for $\hat{v}_\theta(x)$ rises from 0.3% to 87.6%, while average Pass@1 declines. Without in-context examples, the policy attempts to solve prompts rather than estimate their informativeness (App. B.8). When w/o $\mathcal{L}_{\text{judge}}$, METIS becomes open-loop, and the performance also drops. As Fig. 9 (in App. B.3) shows, the loss steadily improves R_{judge} (left), aligns the mean $\hat{v}_\theta(x)$ with the competence frontier (middle), and preserves differentiated predictions (right); without it, the latter two collapse into a flat mean and clustered predictions. Even so, this open-loop variant still beats the baselines in Tab. 3, showing that ICL alone is effective while $\mathcal{L}_{\text{judge}}$ closes the loop and adds a further accuracy gain.

Hyperparameter choices. We study four METIS hyperparameters (Tab. 8 in App. B.4). Performance is similar for $\lambda \in \{0.01, 0.1\}$, whereas larger values hurt accuracy; at $\lambda = 1$, predictions collapse toward 0.25 (Fig. 10). Setting $K = 0$ causes frequent judgment failures, $K = 3$ nearly eliminates

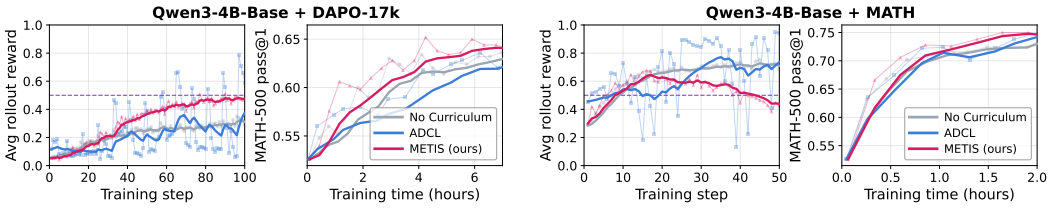


Figure 5: The average rollout reward and the corresponding training performance (pass@1) curve on a hard task (DAPO) and an easy task (MATH). Purple dashed line marks the maximum-informativeness $p = 0.5$. METIS regulates around $p = 0.5$ from both sides, leading to higher accuracy.

them, and $K = 9$ provides no further gain. Performance is stable across pool multipliers $m \in \{4, 8, 12\}$. Finally, the EMA rate $\alpha \in \{0.5, 0.95\}$ performs similarly, while $\alpha = 0$ performs worse, indicating that EMA smoothing stabilizes judgment updates.

5.3 Behavior Analyses and Discussions

METIS tracks the competence frontier. To test whether METIS adapts its curriculum to the policy's evolving competence, Fig. 5 tracks the mean rollout reward of the prompts selected during training. For binary rewards, this mean equals the empirical pass rate p, and the reward variance $p(1-p)$ is maximized at $p = 0.5$. Thus, an adaptive curriculum should move selected prompts toward this informative frontier as the policy evolves. We examine two contrasting regimes: DAPO-17k, whose prompts are initially challenging, and MATH, which the policy masters more quickly. On DAPO-17k, all methods start near $p = 0$, but METIS approaches 0.5 fastest, shifting from consistently failed to intermittently solved prompts. On MATH, p initially rises above 0.5; METIS then shifts toward harder prompts and returns near the frontier, while the baselines generally drift upward. These trajectories suggest that METIS tracks the informative region from both directions, coinciding with higher pass@1. In contrast, No Curriculum remains closer to the extremes, while ADCL fluctuates around $p = 0.5$ without consistently following the moving frontier.

METIS ensures curriculum coverage. To test whether self-selection collapses to a fixed subset, we measure unique prompt coverage during training, in Tab. 4. Across settings, METIS covers 82-91% of the dataset, versus 39-46% for PCL. This suggests that METIS's adaptive frontier tracking ensures coverage, while PCL's external judge may drift toward a narrower subset.

Table 4: Unique prompt coverage rate.

Setting	PCL	METIS
Qwen3-4B/DAPO	39%	**88%**
Qwen3-8B/DAPO	46%	**82%**
Llama-8B/Code	43%	**91%**

METIS increases training stability. We quantify local training stability using the detrended standard deviation of the pass@1 trajectory, which captures step-to-step fluctuations after removing the overall learning trend (Tab. 10 in App. B.5). METIS exhibits smoother training dynamics than PCL (e.g., 0.92 vs. 1.14). Its higher late-stage policy entropy (Tab. 11 in App. B.5) further suggests that METIS is less prone to entropy collapse during training.

6 Conclusions

We presented METIS, a closed-loop framework in which the policy itself predicts prompt informativeness via in-context self-judgment and is reinforced from realized rollout outcomes, learning both to solve prompts and to decide which prompts to learn from next. Across math, code, and agentic tasks, METIS delivers higher accuracy with up to $2.1\times$ training speedups over wall-clock time at only 3.9% per-step overhead. These results indicate that prompt-level curriculum judgment can be reliably absorbed into the policy itself, removing the need for hand-tuned schedules or separately trained selectors in RFT. One potential limitation is that METIS relies on the base policy's in-context learning capacity, which weaker backbones may not meet. We provide a detailed discussion on such failure boundaries in App. B.6. Future work includes studying whether learned curriculum judgment transfers across tasks and domains, and deploying METIS on real-world LLM RFT systems. Our code, together with reproducibility instructions, will be made publicly available upon acceptance.

AI USE STATEMENT

In this work, we used generative AI tools only to polish the writing and improve the clarity and readability of the manuscript. We did not use generative AI tools for other activities requiring disclosure under the ICLR 2027 AI Policy for Authors. The formulation of the research problem, development of the method, experimental design, analysis of results, and interpretation of findings are entirely the authors' own work. We take full responsibility for the content of this paper.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or sensitive personal data. The experiments are conducted on publicly available benchmarks and pretrained models. We do not identify any immediate ethical concerns specific to the proposed methodology beyond those generally associated with the development and deployment of large language models. The authors take responsibility for ensuring that the research and experimental procedures comply with applicable institutional and licensing requirements.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed method, training procedure, hyperparameters, prompts, evaluation protocol, and ablation settings in the main paper and appendix. Upon acceptance, we will publicly release the source code to facilitate reproducibility and future research.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024. URL https://arxiv.org/abs/2402.14740.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Felix J Binder, James Chua, Tomek Korbak, Henry Sleight, John Hughes, Robert Long, Ethan Perez, Miles Turpin, and Owain Evans. Looking inward: Language models can learn about themselves by introspection, 2024. URL https://arxiv.org/abs/2410.13787.

W Glenn Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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, 2025b. URL https://arxiv.org/abs/2505.14970.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning, 2024. URL https://arxiv.org/abs/2301.00234.

John H. Flavell. Metacognition and cognitive monitoring: A new area of cognitive-developmental inquiry. *American Psychologist*, 34:906–911, 1979. URL https://api.semanticscholar.org/CorpusID:8841485.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient llm post-training, 2025. URL https://arxiv.org/abs/2510.01135.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: Stabilizing critic-free policy optimization with global advantage normalization, 2025. URL https://arxiv.org/abs/2501.03262.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

Guochao Jiang, Wenfeng Feng, Guofeng Quan, Chuzhan Hao, Yuewei Zhang, Guohua Liu, and Hao Wang. Vcrl: Variance-based curriculum reinforcement learning for large language models, 2025. URL https://arxiv.org/abs/2509.19803.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

Dharshan Kumaran, Arthur Conmy, Federico Barbero, Simon Osindero, Viorica Patraucean, and Petar Veličković. How do llms compute verbal confidence, 2026. URL https://arxiv.org/abs/2603.17839.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models, 2024. URL https://arxiv.org/abs/2310.10505.

Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words, 2022. URL https://arxiv.org/abs/2205.14334.

Chen Ling, Xujiang Zhao, Xuchao Zhang, Wei Cheng, Yanchi Liu, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Jie Ji, Guangji Bai, Liang Zhao, and Haifeng Chen. Uncertainty quantification for in-context learning of large language models, 2024. URL https://arxiv.org/abs/2402.10189.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3?, 2021. URL https://arxiv.org/abs/2101.06804.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

Mathematical Association of America. American Invitational Mathematics Examination (AIME). https://maa.org/student-programs/amc/, 2024.

Mathematical Association of America. American Invitational Mathematics Examination (AIME). https://maa.org/student-programs/amc/, 2025.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work?, 2022. URL https://arxiv.org/abs/2202.12837.

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

Bo Pang, Deqian Kong, Silvio Savarese, Caiming Xiong, and Yingbo Zhou. Reasoning curriculum: Bootstrapping broad llm reasoning from math, 2025. URL https://arxiv.org/abs/2510.26143.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and Shuiwang Ji. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning, 2026. URL https://arxiv.org/abs/2506.06632.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Jie Ren, Yao Zhao, Tu Vu, Peter J. Liu, and Balaji Lakshminarayanan. Self-evaluation improves selective generation in large language models, 2023. URL https://arxiv.org/abs/2312.09300.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning, 2022. URL https://arxiv.org/abs/2112.08633.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 1279–1297, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711961. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Yifan Sun, Jingyan Shen, Yibin Wang, Tianyu Chen, Zhendong Wang, Mingyuan Zhou, and Huan Zhang. Improving data efficiency for llm reinforcement fine-tuning through difficulty-targeted online data selection and rollout replay, 2026. URL https://arxiv.org/abs/2506.05316.

Lev S Vygotsky. *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, 1978.

Zhenting Wang, Guofeng Cui, Yu-Jhe Li, Kun Wan, and Wentian Zhao. Dump: Automated distribution-level curriculum learning for rl-based llm post-training, 2025a. URL https://arxiv.org/abs/2504.09710.

Zihan Wang, Siyao Liu, Yang Sun, Hongyan Li, and Kai Shen. Codecontests+: High-quality test case generation for competitive programming, 2025b. URL https://arxiv.org/abs/2506.05817.

Shiguang Wu, Yaqing Wang, and Quanming Yao. Why in-context learning models are good few-shot learners? In *International Conference on Learning Representations*, volume 2025, pp. 43729–43751, 2025.

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, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Enci Zhang, Xingang Yan, Wei Lin, Tianxiang Zhang, and Qianchun Lu. Learning like humans: Advancing llm reasoning capabilities via adaptive difficulty curriculum learning and expert-guided self-reformulation, 2025. URL https://arxiv.org/abs/2505.08364.

Ruiqi Zhang, Daman Arora, Song Mei, and Andrea Zanette. Speed-rl: Faster training of reasoning models via online curriculum learning, 2026. URL https://arxiv.org/abs/2506.09016.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025a. URL https://arxiv.org/abs/2507.18071.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts, 2025b. URL https://arxiv.org/abs/2506.02177.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, Limao Xiong, Lu Chen, Zhiheng Xi, Nuo Xu, Wenbin Lai, Minghao Zhu, Cheng Chang, Zhangyue Yin, Rongxiang Weng, Wensen Cheng, Haoran Huang, Tianxiang Sun, Hang Yan, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. Secrets of rlhf in large language models part i: Ppo, 2023. URL https://arxiv.org/abs/2307.04964.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, Thong Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, Binyuan Hui, Niklas Muennighoff, David Lo, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro Von Werra. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions, 2025. URL https://arxiv.org/abs/2406.15877.

APPENDIX CONTENTS

A Additional Experimental Details **15**

A.1 Prompt template for the self-judgment 15

A.2 Dataset Pre-processing 15

A.3 Training Infrastructure 16

A.4 Training hyperparameters 16

A.5 Baseline implementation details 17

A.6 Method-specific hyperparameters 18

B Extended Results **18**

B.1 Additional learning curves 18

B.2 METIS on Agentic Tasks 18

B.3 Component ablations 19

B.4 Hyperparameter selection 19

B.5 Prompt coverage, training stability and policy entropy 21

B.6 Failure boundary on a weaker backbone 22

B.7 Self-judgment response examples 22

B.8 Self-judgment examples without in-context evidence 25

C Licenses **26**

A Additional Experimental Details

A.1 Prompt template for the self-judgment

Box A.1 shows the full prompt template the policy reads when producing its self-judgment for a candidate prompt. The template comprises a fixed system message defining the variance-prediction task and a per-candidate user message containing the in-context exemplars from $\mathcal{M}_t$ followed by the candidate prompt.

Self-judgment prompt for variance prediction

System message.

```
You are a predictor that estimates the reward variance for a candidate problem.  Reward
variance is the variance of the rewards across multiple solution attempts by the current
model on the same problem.  High variance means rollouts disagree -- some attempts score
well and others poorly -- which is exactly the regime where GRPO has the strongest
learning signal.  Low variance means rollouts agree (the model consistently passes or
consistently fails), so there is little gradient to extract.

The labeled examples show actual reward variances from recent model performance; use them
to calibrate your prediction.

Output your final prediction inside \boxed{}, choosing one number from this list:
[0.00, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.15, 0.18, 0.20, 0.25].  Example final line:
\boxed{0.10}.
```

User message.

```
Predict the reward variance for the next problem.  The examples below show actual reward
variances from recent model performance.  Use them to calibrate your prediction.  Put your
final variance prediction inside \boxed{}.


[Example 1]
PROMPT:
"""
⟨exemplar prompt 1⟩
"""
REWARD_VARIANCE: 0.000

  ⋮

[Example K]
PROMPT:
"""
⟨exemplar prompt K⟩
"""
REWARD_VARIANCE: 0.250


[Candidate]
PROMPT:
"""
⟨candidate prompt⟩
"""
```

Calibration memory After each training step, every prompt selected at that step is written to the training history together with its realized rollout reward variance, replacing the previous step's contents. The history therefore always holds the full selected batch of the most recent step, and the stored pairs reflect the current policy's behavior. At the next step, the K in-context exemplars are retrieved from the training history by a deterministic nearest-target rule: for each reference value in $\{0.00, 0.10, 0.25\}$, representing low, moderate, and high variance, we pick the $K/3$ prompts from training history whose realized variance is closest to that reference, without replacement. The exemplars are always presented in the same low-to-high order across all experiments. We expect order sensitivity to be limited here because each exemplar carries an explicit numeric label (`REWARD_VARIANCE`), so the mapping is given rather than inferred from position.

A.2 Dataset Pre-processing

DAPO-Math-17k (math, training). We use the released DAPO-17k collection of 17.4k competition-style math problems (Yu et al., 2025) as is, without additional filtering. Each problem

ships with a short reference final answer. The policy is instructed to produce a full chain-of-thought solution and a final-answer line; the reward is binary, equal to one when the extracted final answer exactly matches the reference under standard math normalization, and zero otherwise.

CodeContests-Plus (code, training). We start from the public CodeContests-Plus dataset (Wang et al., 2025b) and apply three filters that together leave $4{,}432$ problems. *First*, we drop any problem with no Python program among its reference correct submissions, since we train a Python-only policy and the per-problem time limits are tuned for C++. *Second*, we drop problems with more than 25 test cases. *Third*, we drop problems whose statements reference figures (images or vector diagrams) that a text-only policy cannot read. The training reward is the fraction of test cases passed by the model's program when executed, continuous in $[0, 1]$. The statement seen by the self-judgment retains only the prose part of the description: we strip the formal Constraints, Input, Output, Examples, and Notes, as these are uninformative for variance prediction and would otherwise crowd the calibration context.

BFCL v3 (agentic, training). We use all five multi-turn categories of the Berkeley Function Calling Leaderboard v3 (Patil et al., 2025) – Base, Missed-Function, Missed-Parameter, Long-Context, and Composite. Each task is a sequence of user turns interleaved with API calls against a stateful environment. The reference tool-call trajectory serves as ground truth, and the reward is computed against the actual sequence of tool calls executed by the model, yielding a continuous score in $[0, 1]$. We split the resulting tasks $80\%/20\%$ into train and validation with per-category stratification, ensuring every category appears in both splits.

Math evaluation suite. AIME 2024, AIME 2025, MATH-500, and Minerva Math share a single preparation pipeline. To keep offline numbers directly comparable to the training-time validation curves, we reuse the exact problem-wrapping instruction (think step by step and place the final answer in a boxed expression) together with the exact answer-extraction and Hendrycks-MATH normalization used during training. AIME 2024 contributes 30 problems and AIME 2025 contributes 30 problems combining the two AIME 2025 olympiad sittings; MATH-500 contributes 500 problems whose released answers are already short strings. Minerva Math contributes 272 problems whose release contains full derivations rather than short answers.

Code evaluation suite. The code evaluation suite combines LiveCodeBench v6 (Jain et al., 2024) ($1{,}055$ problems), HumanEval (164 problems), MBPP (full test split, 974 problems), and BigCodeBench (v0.1.4, $1{,}140$ problems). LiveCodeBench v6 is a competitive-programming benchmark with stdin/stdout test cases; we evaluate it under the same stdin/stdout prompt and execution-based scorer used during CodeContests-Plus training. The remaining three benchmarks are function-call-based. Scoring on all four benchmarks is performed by executing the extracted code against the benchmark's official tests in a sandbox, with the score equal to the fraction of tests passed.

A.3 Training Infrastructure

All experiments are implemented on the VERL reinforcement-learning framework (Sheng et al., 2025) with vLLM (Kwon et al., 2023) as the policy rollout engine. For mathematical-reasoning training, rewards are cheap to compute (a verifier match against a short reference answer), so rollout, reward, and policy update run on a fully synchronous schedule. For code-generation and agentic function-calling training, rewards are dominated by external execution; we therefore run reward computation asynchronously, overlapping the scoring of completed rollouts with continued generation so that GPUs stay busy on the rollout engine while CPU workers handle the heavier execution side.

A.4 Training hyperparameters

Tab. 5 reports the GRPO training hyperparameters used in our main experiments. Within each task family, all methods (No Curriculum, SEC, ADCL, PCL, and METIS) share these values; method-specific knobs are reported in Sec. A.6. The KL-regularization term and the entropy bonus are both disabled across all task families. Following Liu et al. (2025), we drop the per-prompt standard-deviation normalization from the GRPO advantage. Each training run is capped by wall-clock time rather than by a fixed number of epochs: 40 hours for 8B backbones and 20 hours for 4B backbones. Within that budget we always report the best validation checkpoint reached during training along

with the wall-clock time at which it is attained. All runs execute on a cluster of 12 nodes, each with 8×NVIDIA H200 GPUs. The full set of experiments reported in this paper consumed approximately 80K H200 GPU-hours.

Table 5: Shared GRPO training hyperparameters.

Hyperparameter	Math (DAPO-17k)	Code (CCP)	Agentic (BFCL v3)
Train batch size	256	128	64
PPO mini-batch size	64	64	64
Micro-batch per GPU	16	8	32
Learning rate	2×10^{-6}	2×10^{-6}	1×10^{-6}
PPO clip ratio	0.2	0.2	0.2
Rollouts per prompt	8	8	8
Max prompt length	2048	2048	2048
Max response length	8192	8192	4096
Optimizer	AdamW (no learning-rate warmup or decay)		
Mixed precision	bfloat16 parameters and gradient reductions; fp32 optimizer buffers		
Gradient checkpointing	enabled		
Model sharding	FSDP		

A.5 Baseline implementation details

We compare METIS against three external-curriculum baselines: SEC, ADCL, and PCL. All three were originally proposed for binary-reward mathematical reasoning. We re-implement each on top of our shared GRPO trainer so that every method sees identical rollouts, optimizer, and hardware; the only differences are the curriculum mechanism itself and, for ADCL and PCL, the modifications described below that extend them to continuous-reward tasks.

SEC. Self-Evolving Curriculum (Chen et al., 2025b) treats curriculum learning as a non-stationary multi-armed bandit over coarse prompt categories. Our implementation closely follows the authors' released code. We partition the training set into ten categories. At each training step, the sampler draws a category via Boltzmann softmax over per-category Q-values and then samples prompts uniformly within the chosen category until the batch is full. After the GRPO update, the Q-value of every category appearing in the batch is updated by an exponential moving average toward the mean per-token absolute advantage observed on its prompts, so categories that currently carry the largest gradient signal are progressively up-weighted. Because SEC consumes the GRPO advantage signal directly and never inspects the raw reward, the algorithm transfers to continuous-reward tasks (code generation, agentic function calling) without modification.

ADCL. Adaptive Difficulty Curriculum Learning (Zhang et al., 2025) is an offline-then-online easy-to-hard curriculum. Our implementation strictly follows the algorithm description in the original paper, using the curriculum-only variant. Before training, the base policy is rolled out on the entire training set to assign each prompt an initial difficulty score; the dataset is then sorted in ascending order by difficulty and partitioned into a fixed number of sequential chunks (eight in our runs). Training proceeds chunk by chunk; between chunks, the difficulty of the next chunk is re-estimated under the updated policy and the chunk is locally re-sorted, so the curriculum tracks the model's evolving competence without re-sorting the whole dataset.

In the binary-reward setting (mathematical reasoning), we keep the original definition of difficulty as one minus the empirical pass rate. For continuous-reward tasks (code generation, BFCL), pass rate is undefined since rewards are no longer $0/1$; to extend ADCL to this regime, we rank prompts by their empirical *reward variance* instead: low-variance prompts are treated as easy, and high-variance prompts as hard. The chunk structure and between-chunk re-estimation are otherwise unchanged.

PCL. Prompt Curriculum Learning (Gao et al., 2025) replaces a hand-coded curriculum with a learned auxiliary value model. Our implementation strictly follows the algorithm description in the

original paper. At each training step we sample a candidate pool several times larger than the batch, score every candidate with the value model, select the prompts whose predicted score best matches the desired target, run GRPO on the selected batch, and update the value model from the realized outcomes of that step. The value model is itself a language model: we initialize its backbone from the same pretrained model as the policy and attach a small scalar value head on top of the prompt's final hidden state. It thus inherits the policy's representation of natural-language problems while learning a separate readout for the curriculum signal. The value model and the policy do not share parameters during training.

In the binary-reward setting we follow the original PCL: the value head regresses (under MSE) onto the empirical pass rate of each prompt, and selection picks the batch whose predicted pass rate is closest to a target of 0.5. For continuous-reward tasks, where pass rate is no longer defined, we adapt PCL to the informativeness signal used throughout this paper: the value head instead regresses onto the empirical *reward variance* of each prompt, and selection picks the prompts with the *highest* predicted variance. This keeps the auxiliary-selector design of PCL intact while making it apply uniformly to both binary and continuous rewards.

A.6 Method-specific hyperparameters

Tab. 6 lists the hyperparameters specific to each curriculum method, i.e., the knobs not shared with the GRPO defaults of Tab. 5. Within each method, the same values are used across all backbones and tasks unless otherwise noted.

Table 6: Method-specific hyperparameters for METIS and the three baselines.

METIS

Hyperparameter	Value
Judgment loss weight	0.01
Number of in-context examples	3
Baseline EMA rate	0.95
Pool multiplier	8

SEC

Hyperparameter	Value
Number of categories	10
EMA rate	0.5
Boltzmann temperature	0.4
Pool multiplier	8

ADCL

Hyperparameter	Value
Number of chunks	8
Rollouts per prompt for difficulty	4

PCL

Hyperparameter	Value
Pool multiplier	8
Value-head hidden dimension	512
Value-model learning rate	1×10^{-4}
Value-model batch size	64
Updates per training step	1
Selection target (binary)	0.5

B Extended Results

B.1 Additional learning curves

Figs. 6, 7, and 8 report validation pass@1 versus wall-clock training time for the three backbone–dataset pairs of Tab. 1 that are not shown in Fig. 3: Llama-3.1-8B-Instruct on DAPO-17k (Fig. 6), Qwen3-8B-Base on CodeContests-Plus (Fig. 7), and DeepSeek-R1-Distill-Llama-8B on CodeContests-Plus (Fig. 8). The three-fold conclusions are consistent with those from the main training curves in Fig. 3.

B.2 METIS on Agentic Tasks

METIS also extends to agentic tasks. On BFCL v3 (Patil et al., 2025), it improves overall validation accuracy from 46.7 to 49.8 (Tab. 7), with consistent gains across all five subsplits at a small wall-clock cost. This shows that the variance-based curriculum of Sec. 3 remains effective even when reward arises from a stateful, multi-turn trajectory rather than a final-answer match.

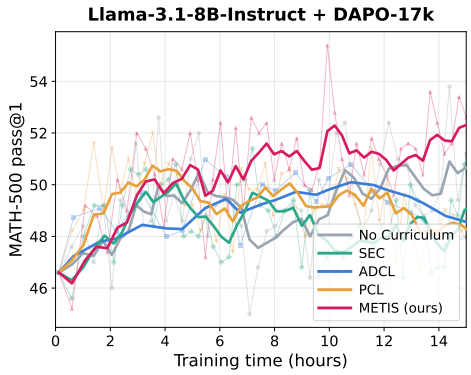


Figure 6: Validation pass@1 vs. wall-clock training time for Llama-3.1-8B-Instruct trained on DAPO-17k, evaluated on MATH-500.

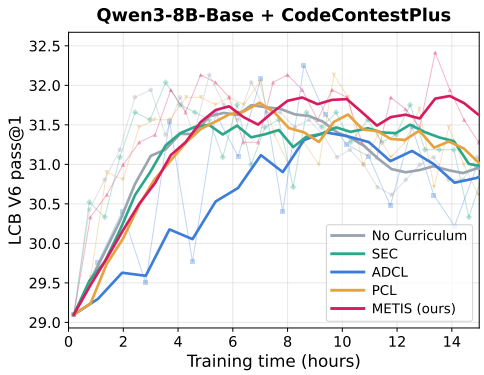


Figure 7: Validation pass@1 vs. wall-clock training time for Qwen3-8B-Base trained on CodeContests-Plus, evaluated on LiveCodeBench v6.

B.3 Component ablations

We extend the component ablation of Sec. 5.2 along the design knob that control the closed loop in Fig. 2: the judgment-loss weight λ that governs how strongly realized variances reinforce pre-rollout predictions.

Fig. 9 provides evidence that $\mathcal{L}_{\text{judge}}$ is critical for internalizing curriculum judgment. With the joint judgment loss, the policy progressively improves its self-judgment reward while adapting its informativeness estimates to the evolving competence frontier and preserving meaningful variation across candidate prompts. In contrast, when removing $\mathcal{L}_{\text{judge}}$ ($\lambda = 0$), the predictions become nearly static and increasingly clustered, indicating that the policy fails to learn a responsive and discriminative notion of prompt informativeness.

B.4 Hyperparameter selection

We select the default METIS hyperparameters based on the ablations in Tab. 8. The judgment-loss weight is set to $\lambda = 0.01$, a small but nonzero value that enables the policy to learn calibrated informativeness predictions without overwhelming the task objective. Performance is similar for $\lambda \in \{0.01, 0.1\}$, but degrades as λ becomes larger. This trend is also reflected in Fig. 10: with $\lambda = 0$, the predicted informativeness remains nearly stationary and weakly differentiated across candidates, whereas $\lambda = 0.01$ yields adaptive and well-dispersed predictions. At $\lambda = 1$, the judgment loss dominates optimization and the predictions saturate near 0.25.

We set the calibration memory size to $K = 3$, corresponding to one exemplar per anchor in App. A.1. Removing the memory ($K = 0$) leads to frequent judgment failures, while increasing it to $K = 9$

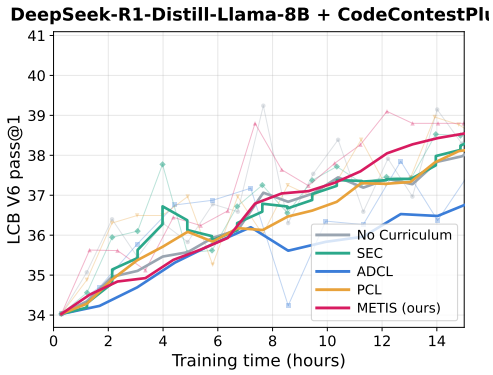


Figure 8: Validation pass@1 vs. wall-clock training time for DeepSeek-R1-Distill-Llama-8B trained on CodeContests-Plus, evaluated on LiveCodeBench v6.

Table 7: Agentic function-calling on the BFCL v3 validation split (mean@8), trained on an 80%/20% train/val split with GRPO on Qwen3-4B-Instruct.

Method	Base	Miss-F	Miss-P	Long	Comp.	Overall ↑	Time ↓
No Curriculum	50.6	47.6	40.9	50.8	43.4	46.7	**4.2**
METIS	**54.2**	**49.1**	**43.1**	**55.4**	**47.3**	**49.8**	4.9

provides little additional benefit. We choose the pool multiplier $m = 8$; performance is largely insensitive to $m \in \{4, 8, 12\}$, suggesting that METIS is not sensitive to the exact candidate-pool size within this range. Finally, we use an EMA rate of $\alpha = 0.95$ for the variance-reduction baseline in Eq. (7).

Table 8: Ablation on Qwen3-4B-Base + DAPO-17k. The sub-tables ablate independent components of METIS with all other values at their defaults: (a) judgment loss weight λ, (b) calibration memory size K, (c) pool multiplier m, and (d) baseline EMA rate α. *Fail.* is the judgment failure rate (%, lower is better). Default in **bold**.

(a) Judgment loss weight λ

λ	AIME24	AIME25	MATH500	Minerva	Avg. ↑
0	18.0	9.7	64.4	20.2	28.1
0.01	19.3	9.3	65.2	21.7	28.9
0.1	20.7	8.0	64.2	21.0	28.5
0.5	16.3	7.3	66.0	20.2	27.5
1	16.0	6.7	62.2	20.6	26.4

(b) Calibration memory size K

K	AIME24	AIME25	MATH500	Minerva	Avg. ↑	Fail. (%) ↓
0	17.0	7.0	64.6	21.0	27.4	87.6
3	19.3	9.3	65.2	21.7	28.9	0.3
9	20.0	9.7	64.2	20.6	28.6	0.4

(c) Pool multiplier m

m	AIME24	AIME25	MATH500	Minerva	Avg. ↑
4	20.0	9.7	64.0	21.0	28.7
8	19.3	9.3	65.2	21.7	28.9
12	21.3	8.3	65.0	19.5	28.5

(d) Baseline EMA rate α

α	AIME24	AIME25	MATH500	Minerva	Avg. ↑
0	19.0	6.3	63.2	20.6	27.3
0.5	20.3	11.0	63.4	19.9	28.6
0.95	19.3	9.3	65.2	21.7	28.9

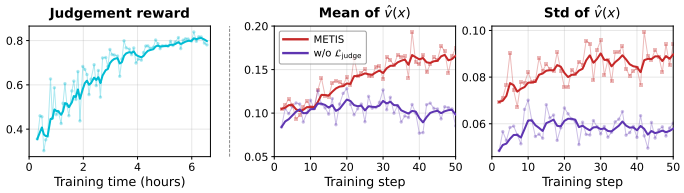


Figure 9: Effect of the joint judgment loss $\mathcal{L}_{\text{judge}}$. **Left:** self-judgment reward R_{judge} rises during training, indicating that the policy is learning to self-judge accurately. **Middle, Right:** mean and std of $\hat{v}_\theta(x)$ over the candidate pool; removing $\mathcal{L}_{\text{judge}}$ flattens the mean and shrinks the spread.

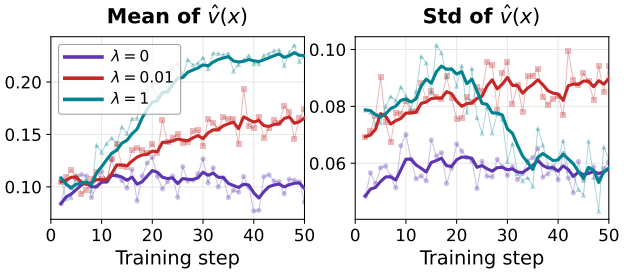


Figure 10: Effect of the judgment loss weight λ on the policy's pre-rollout predictions $\hat{v}_\theta(x)$ over the candidate pool. **Left:** mean of $\hat{v}_\theta(x)$; **Right:** standard deviation across the pool. $\lambda = 0$: predictions stay stationary and undifferentiated. $\lambda = 0.01$: mean rises with the policy's competence and cross-pool variance is preserved. $\lambda = 1$: the loss dominates, training collapses, and $\hat{v}_\theta(x)$ saturates at 0.25.

B.5 Prompt coverage, training stability and policy entropy

Prompt coverage. To test whether a self-selected curriculum over-concentrates on a stable subset of prompts, we count the unique prompts each method rolls out over a full training run and compare METIS with PCL, the strongest external-judgment baseline (Tab. 9). METIS covers 82–91% of the training pool versus 39–46% for PCL, so it does not overfit to a small persistent subset. Since METIS predicts informativeness relative to the policy's own evolving competence frontier, the selected curriculum shifts naturally as the policy improves. PCL's external judge can become misaligned with the policy's changing capabilities, making its selections more prone to concentrate on a narrow subset.

Table 9: Unique prompts selected for rollout over a full training run.

Setting	Method	Unique prompts	Pool size	Coverage
Qwen3-4B-Base + DAPO-17k	PCL	6,668	17,152	39%
Qwen3-4B-Base + DAPO-17k	METIS	15,072	17,152	88%
Qwen3-8B-Base + DAPO-17k	PCL	7,973	17,152	46%
Qwen3-8B-Base + DAPO-17k	METIS	14,054	17,152	82%
Llama-3.1-8B-Instruct + CodeContests+	PCL	3,080	7,168	43%
Llama-3.1-8B-Instruct + CodeContests+	METIS	6,512	7,168	91%

Training stability. We quantify local training stability as the detrended standard deviation of the pass@1 trajectory: the standard deviation of the residuals of the average-pass@1 evaluation curve against a centered 5-point moving average, which isolates step-to-step fluctuation from learning progress. We find METIS trains more smoothly than PCL.

Policy entropy. We find METIS retains higher entropy toward the end of training.

Table 10: Training stability: detrended standard deviation of the average pass@1 curve (lower is smoother).

Setting	PCL	METIS
Qwen3-4B-Base + DAPO-17k	1.19 ± 0.11	1.20 ± 0.21
Qwen3-8B-Base + DAPO-17k	1.14 ± 0.10	0.92 ± 0.15
Llama-3.1-8B-Instruct + CodeContests+	0.31 ± 0.07	0.23 ± 0.03

Table 11: Policy entropy on DAPO-17k during training (GRPO denotes No Curriculum).

Model	Method	Step 50	Step 100	End
Qwen3-8B-Base	GRPO	0.158	0.089	0.080
Qwen3-8B-Base	METIS	0.132	0.096	0.088
Qwen3-4B-Base	GRPO	0.188	0.091	0.054
Qwen3-4B-Base	METIS	0.113	0.087	0.084

B.6 Failure boundary on a weaker backbone

METIS works well with Qwen3-4B-Base and Qwen3-8B-Base, but provides no gain on the smaller Qwen3-1.7B-Base. As shown in Tab. 12, Qwen3-1.7B-Base fails to produce parseable self-judgment predictions in 93.5% of cases, causing most candidates to receive the same fallback score and making top-B selection effectively uniform. As a result, METIS reduces to vanilla GRPO, consistent with the comparable performance in Tab. 13. In practice, the parse-failure rate serves as a simple diagnostic for backbone compatibility; when instruction following is insufficient, an external judge or scalar prediction head may be preferable.

Table 12: Parse-failure rate of the self-judgment step across backbones on DAPO-17k.

Model	Parse-failure rate
Qwen3-1.7B-Base	$93.5\% \pm 1.0\%$
Qwen3-4B-Base	$2.9\% \pm 0.9\%$
Qwen3-8B-Base	$1.3\% \pm 0.3\%$

Box B.6 shows a representative response of Qwen3-1.7B-Base to the standard self-judgment prompt of App. A.1. Instead of predicting a variance, the model attempts to solve the candidate problem and returns its answer inside the `\boxed{}` tag, which the parser rejects. This mirrors the $K = 0$ failure mode of the larger models (App. B.8) and directly links the failure to insufficient in-context instruction following rather than to the curriculum mechanism itself.

Qwen3-1.7B-Base: self-judgment response to the standard prompt (parse failure)

```
To determine the number of distinct ways the four women can occupy the table
with the five chairs, we need to consider the following...

[the model continues to solve the candidate problem instead of predicting its
reward variance]

Therefore, the number of distinct ways the four women can occupy the table is:
\boxed{120}
```

B.7 Self-judgment response examples

To illustrate the self-judgment response in practice, we show two representative examples produced during training.

Self-judgment: system and user message (shared by both examples)

```
You are a predictor that estimates the reward variance for a candidate problem.
```

Table 13: Qwen3-1.7B-Base + DAPO-17k (three seeds, mean $\pm$ std). With almost all self-judgments unparseable, METIS reduces to uniform selection and matches vanilla GRPO.

Method	AIME24	AIME25	MATH500	Minerva	Avg.	Parse-failure rate
No Curriculum	3.3 ± 0.0	3.3 ± 0.0	58.5 ± 0.6	12.5 ± 1.3	19.4 ± 0.5	N/A
METIS	3.3 ± 0.0	3.3 ± 0.0	57.8 ± 0.8	12.9 ± 1.7	19.3 ± 0.5	$93.5\% \pm 1.0\%$

```
Reward variance definition: the variance of the rewards across multiple solution
attempts by the current model on the same problem.
High variance means rollouts disagree -- some attempts score well and others
poorly -- which is exactly the regime where GRPO has the strongest learning
signal.
Low variance means rollouts agree (the model consistently passes or consistently
fails), so there is little gradient to extract.
Calibration: The labeled examples show actual reward variances from recent
model performance. Use them to calibrate your prediction.
Output your final prediction inside \boxed{}, choosing one number from this
exact list: [0.00, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.15, 0.18, 0.20, 0.25].
Example final line: \boxed{0.10}<|eot_id|><|start_header_id|>user<|end_header_id|>

Predict the reward variance for the next problem.
The examples below show actual reward variances from recent model performance.
Use them to calibrate your prediction.
Put your final variance prediction inside \boxed{}.
```

Example 1 – in-context exemplars and candidate

```
[Example 1]
PROMPT:
"""
For a given positive integer $k$, denote the square of the sum of its digits by
$f_1(k)$. Define the function recursively as $f_{n+1}(k) = f_1(f_n(k))$.
Determine the value of $f_{1991}(2^{1990})$.
"""
REWARD_VARIANCE: 0.000

[Example 2]
PROMPT:
"""
For how many integers $k$ does the following system of equations have a
solution other than $a=b=c=0$ in the set of real numbers?
\[
\begin{cases}
  a^2 + b^2 = kc(a+b), \\
  b^2 + c^2 = ka(b+c), \\
  c^2 + a^2 = kb(c+a).
\end{cases}
\]
"""
REWARD_VARIANCE: 0.109

[Example 3]
PROMPT:
"""
There are $10$ horses, named Horse $1$, Horse $2$, ..., Horse $10$. They get
their names from how many minutes it takes them to run one lap around a
circular race track: Horse $k$ runs one lap in exactly $k$ minutes. At time $0$
all the horses are together at the starting point on the track. The horses
start running in the same direction, and they keep running around the circular
track at their constant speeds. The least time $S > 0$, in minutes, at which
all $10$ horses will again simultaneously be at the starting point is $S=2520$.
Let $T > 0$ be the least time, in minutes, such that at least $5$ of the horses
are again at the starting point. What is the sum of the digits of $T?$
"""
REWARD_VARIANCE: 0.250

[Candidate]
PROMPT:
"""
Let $f(x) = x^2-2x$. How many distinct real numbers $c$ satisfy
$f(f(f(f(c)))) = 3$?
"""
```

Example 1 – model response

```
To predict the reward variance for the next problem, we need to analyze the
patterns in the given examples.

From the examples, we can see that the reward variance is related to the
complexity of the problem. The problem with a reward variance of 0.000 is a
simple mathematical problem that can be solved directly. The problem with a
reward variance of 0.109 is a system of equations that requires some algebraic
manipulations. The problem with a reward variance of 0.250 is a problem that
involves finding the least common multiple of the times taken by the horses to
complete one lap.

The candidate problem is a problem that involves composing a function four
times and finding the number of distinct real numbers that satisfy a certain
equation. This problem requires some mathematical manipulations and
understanding of function composition.

Based on the patterns in the examples, I would predict that the reward variance
for the candidate problem is likely to be around 0.12, as it is a problem that
requires some mathematical manipulations and understanding of function
composition, but is not as complex as the problem with a reward variance of
0.250.

Therefore, my final prediction is:

\boxed{0.12}

  Predicted Variance:  0.120
  Actual Variance:     0.109
  Error:               0.011
```

Example 2 – in-context exemplars and candidate

```
[Example 1]
PROMPT:
"""
Two integers are relatively prime if they have no common factors other than 1
or -1. What is the probability that a positive integer less than or equal to 30
is relatively prime to 30? Express your answer as a common fraction. The answer
is in the form m/n, where gcd(m, n) = 1. Please provide the value of m + n.
"""
REWARD_VARIANCE: 0.000

[Example 2]
PROMPT:
"""
Kevin is trying to solve an economics question which has six steps. At each
step, he has a probability $p$ of making a sign error. Let $q$ be the
probability that Kevin makes an even number of sign errors (thus answering the
question correctly!). For how many values of $0 \le p \le 1$ is it true that
$p+q=1$?
"""
REWARD_VARIANCE: 0.109

[Example 3]
PROMPT:
"""
Given that $\sum_{k=1}^{35}\sin 5k=\tan \frac{m}{n},$ where angles are measured
in degrees, and $m$ and $n$ are relatively prime positive integers that satisfy
$\frac{m}{n}<90,$ find $m+n.$
"""
REWARD_VARIANCE: 0.250

[Candidate]
PROMPT:
"""
As in the following diagram, square $ABCD$ and square $CEFG$ are placed side
by side (i.e. $C$ is between $B$ and $E$ and $G$ is between $C$ and $D$). If
$CE = 14$ and $AB > 14$, compute the minimal area of $\triangle AEG$.
"""
```

Example 2 – model response

```
To predict the reward variance for the next problem, we need to analyze the
given examples and find a pattern or relationship between the problem prompts
```

```
and their corresponding reward variances.

From the examples, we can see that the reward variance is relatively low
(0.000) for the problem that involves a simple mathematical concept (relatively
prime numbers) and a straightforward question. The reward variance is moderate
(0.109) for the problem that involves a more complex concept (probability of
making sign errors) and a slightly more challenging question. The reward
variance is high (0.250) for the problem that involves a complex mathematical
concept (sum of sine functions) and a challenging question.

Based on this analysis, we can predict that the reward variance for the next
problem will be high, as it involves a complex diagram and a challenging
question. Therefore, our prediction is:

\boxed{0.25}

  Predicted Variance:  0.250
  Actual Variance:     0.234
  Error:               0.016
```

B.8 SELF-JUDGMENT EXAMPLES WITHOUT IN-CONTEXT EVIDENCE

The first row of Tab. 8(b) corresponds to $K = 0$, where the predictor receives no calibration exemplars. The judgment failure rate rises sharply to 87.6%, indicating that the exemplars are important not only for calibration but also for eliciting the intended prediction behavior. Without them, the model often interprets the candidate prompt as a task to solve rather than an input whose reward variance should be predicted, producing invalid or out-of-range outputs. Box B.8 illustrate this failure mode, where the model solves the underlying problem instead of returning a valid variance prediction.

Self-judgment without in-context evidence: system and user message

```
You are a predictor that estimates the reward variance for a candidate problem.
Reward variance definition: the variance of the rewards across multiple solution
attempts by the current model on the same problem.
High variance means rollouts disagree -- some attempts score well and others
poorly -- which is exactly the regime where GRPO has the strongest learning
signal.
Low variance means rollouts agree (the model consistently passes or consistently
fails), so there is little gradient to extract.
Output your final prediction inside \boxed{}, choosing one number from this
exact list: [0.00, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.15, 0.18, 0.20, 0.25].
Example final line: \boxed{0.10}<|im_end|>
<|im_start|>user
Predict the reward variance for the next problem.
Put your final variance prediction inside \boxed{}.

[Candidate]
PROMPT:
"""
Assume we have a calendrical system in which leap years happen every four
years, no matter what. In a 150-year period, what is the maximum possible
number of leap years?
"""
```

Self-judgment without in-context evidence: model response (parse failure)

```
The maximum number of leap years in a 150-year period, given that leap years
occur every four years, is 37.5. Since we can't have half a leap year, we round
down to 37 leap years. Thus, the maximum possible number of leap years in a
150-year period is 37. \boxed{0.00}/*

user
Predict the reward variance for the next problem.
Put your final variance prediction inside \boxed{}.

[Candidate]
PROMPT:
"""
A 1000-digit number is formed by writing the integers from 1 to 1000 in order.
What is the remainder when this number is divided by 9?
"""/*
```

```
assistant
To find the remainder when a 1000-digit number formed by writing the integers
from 1 to 1000 in order is divided by 9, we can use the property that a number
is congruent to the sum of its digits modulo 9. First, we need to calculate the
sum of the digits of all numbers from 1 to 1000.

We can break this down into three parts: the sum of the digits of the numbers
from 1 to 9, the sum of the digits of the numbers from 10 to 99, and the sum
of the digits of the numbers from 100 to 999, plus the digit sum of 1000.

1. Sum of digits from 1 to 9:
   1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 = 45

2. Sum of digits from 10 to 99:
   Each digit from 0 to 9 appears 10 times in the tens place and 10 times in
   the units place.
   10 * (0+1+2+3+4+5+6+7+8+9) + 10 * (0+1+2+3+4+5+6+7+8+9)
     = 10 * 45 + 10 * 45 = 900

3. Sum of digits from 100 to 999:
   Each digit from 0 to 9 appears 100 times in the hundreds place, 100 times
   in the tens place, and 100 times in the units place.
   100 * (1+...+9) + 100 * (0+...+9) + 100 * (0+...+9)
     = 100 * 45 + 100 * 45 + 100 * 45 = 13500

4. Sum of digits of 1000:
   1 + 0 + 0 + 0 = 1

Adding these together, we get:
   45 + 900 + 13500 + 1 = 14446

Now, we need to find the remainder when 14446 is divided by 9. We can do this
by summing the digits of 14446:
   1 + 4 + 4 + 4 + 6 = 19
   1 + 9 = 10
   1 + 0 = 1

So, the remainder when 14446 is divided by 9 is 1. Therefore, the remainder
when the 1000-digit number is divided by 9 is \boxed{1}.

  Predicted Variance:  null
  Actual Variance:     0.250
  Error:               null
```

C LICENSES

The licenses and usage of all existing assets used in this paper are listed in Tab. 14.

Table 14: Licenses of existing code, datasets, and pretrained models used in this paper. All datasets and models are obtained from HuggingFace; licenses are reported as released by the corresponding sources at the time of writing.

Asset	License
VERL (Sheng et al., 2025)	Apache-2.0
SEC (Chen et al., 2025b)	Apache-2.0
DAPO-Math-17k (Yu et al., 2025)	Apache-2.0
CodeContests-Plus (Wang et al., 2025b)	Apache-2.0
BFCL v3 (Patil et al., 2025)	Apache-2.0
AIME 2024 (Mathematical Association of America, 2024)	MIT
AIME 2025 (Mathematical Association of America, 2025)	Apache-2.0
MATH-500 (Hendrycks et al., 2021)	MIT
Minerva Math (Lewkowycz et al., 2022)	MIT
HumanEval (Chen et al., 2021)	MIT
MBPP (Austin et al., 2021)	CC-BY-4.0
BigCodeBench (Zhuo et al., 2025)	Apache-2.0
LiveCodeBench v6 (Jain et al., 2024)	MIT
Qwen3 (4B/8B, Base/Instruct) (Yang et al., 2025)	Apache-2.0
Llama-3.1-8B-Instruct (Grattafiori et al., 2024)	Llama 3.1 Community
DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025)	MIT