NEGATIVE SELF-DISTILLATION: EVOLVING LLMS VIA LABEL-FREE AND GATED DENSE SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-Policy Self-Distillation (OPSD) has emerged as a popular paradigm for large language model (LLM) self-improvement, allowing models to act as their own teachers by leveraging privileged information such as ground-truth solutions. However, recent findings indicate that OPSD can severely degrade the performance of LLMs on complex reasoning tasks: By forcing the student to imitate an artificially confident reasoning trace conditioned on privileged information, OPSD inadvertently suppresses expressions of uncertainty and penalizes the exploratory, self-corrective behaviors required to solve challenging problems. To address this, we introduce Negative Self-Distillation (NSD), a new framework that optimizes LLMs by diverging from flawed reasoning rather than imitating privileged solutions. Instead of relying on ground-truth answers or external supervision, NSD uses the model itself to generate a question-specific negative condition (*e.g.*, acting as a "careless reasoner") and pushes the student's distribution away from this self-generated negative teacher. Naively applying unlearning objectives to achieve this divergence is problematic, as flawed reasoning tokens are confounded with basic linguistic tokens; indiscriminately penalizing both risks catastrophically degrading the model's foundational language capabilities. We resolve this by designing a dynamic gating mechanism that automatically identifies and isolates reasoning-critical tokens, ensuring gradient updates target only behavioral flaws while preserving the model's linguistic priors. Empirically, NSD consistently outperforms OPSD and other label-free, self-bootstrapping reinforcement learning (RL) baselines. Across seven mathematical reasoning benchmarks (AIME 24/25/26, HMMT, AMC, OlympiadBench, and MATH), NSD achieves average gains of 2.3%, 7.5%, and 6.0% for 1.7B, 4B, and 8B models, respectively. Further analyses show that NSD achieves higher training efficiency while preserving the self-correction behaviors crucial for complex reasoning. Code is available at https://anonymous.4open.science/r/NSD.

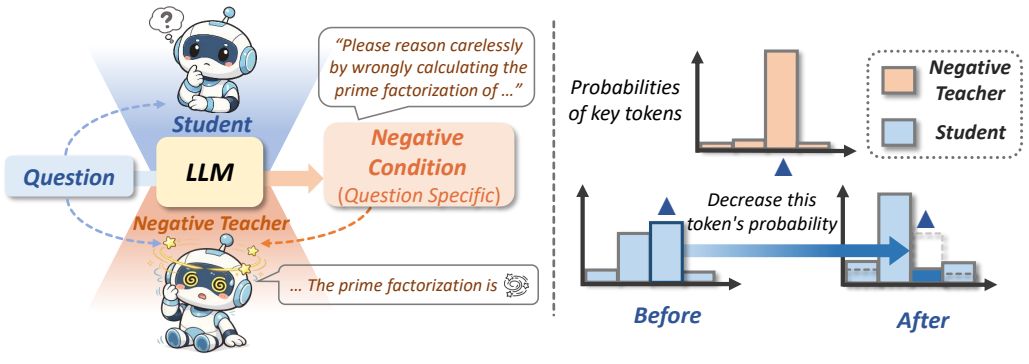

Figure 1: Overview of the NSD framework. (Left) We construct a negative teacher from the same base model via self-generated negative conditioning. (Right) The student model is optimized to diverge its distribution from that of the negative teacher.

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.

1 INTRODUCTION

Reinforcement Learning with Verifiable Rewards (RLVR) (Shao et al., 2024; Yu et al., 2025; Lambert et al., 2025) has emerged as an effective paradigm for enhancing the reasoning capabilities of large language models (LLMs). However, RLVR is often bottlenecked by computational inefficiency and training signal sparsity. These challenges arise because (1) sampling multiple rollouts per query is expensive, and rollouts within a group frequently receive identical rewards on exceptionally easy or difficult problems, leading to advantage collapse and vanishing gradients (Liao et al., 2026; Xu et al., 2026a; Zhang et al., 2025b); and (2) outcome-based rewards are applied uniformly across the entire generated sequence, which obscures fine-grained, token-level credit assignment. To mitigate these limitations, On-Policy Distillation (OPD) (Agarwal et al., 2024; Lu & Lab, 2025; Song & Zheng, 2026) utilizes a stronger, external teacher model to provide dense token-level supervision over the student model's self-sampled reasoning trajectories. While this approach successfully yields richer feedback, it introduces a practical constraint: obtaining a strictly superior external teacher that is both sufficiently capable of providing accurate dense supervision and compatible with the student's tokenizer is often impractical.

To circumvent the reliance on external teacher models, On-Policy Self-Distillation (OPSD) (Zhao et al., 2026a; Shenfeld et al., 2026; Hübotter et al., 2026) has been proposed as a scalable alternative. In OPSD, the model acts as its own teacher by utilizing privileged information (*e.g.*, ground-truth answers) to generate dense supervision signals for the student's self-sampled trajectories. However, because the OPSD teacher inherently knows the ground-truth solution, it tends to produce artificially confident and highly linear reasoning trajectories (Kim et al., 2026b; Harne et al., 2026). Consequently, forcing the student to minimize the divergence from this teacher distribution inadvertently suppresses high-entropy exploration, expressions of uncertainty, and self-corrective behaviors, which are essential for complex problem-solving.

Motivated by the observation that imitating a synthetically confident oracle can degrade natural reasoning processes, we explore an alternative training paradigm: optimizing the model to explicitly avoid flawed reasoning patterns. We introduce **Negative Self-Distillation (NSD)**, a fully self-bootstrapped framework that operates without external privileged data. Instead of utilizing a teacher conditioned on the correct answer, the model is prompted to generate a question-specific negative condition (*e.g.*, acting as a "careless reasoner") to instantiate a negative teacher. The student is then optimized to move its token distribution away from the negative teacher, encouraging it to avoid premature conclusions and other flawed reasoning patterns. Importantly, the negative signal is generated from the model itself and does not require ground-truth solutions or external annotations.

A central challenge, however, is that not every token assigned high likelihood by the negatively conditioned teacher corresponds to a reasoning error. A naive divergence or unlikelihood objective (Welleck et al., 2020) can also penalize ordinary linguistic tokens, degrading the model's pretrained linguistic priors. NSD therefore introduces a dynamic token-level gating mechanism that compares the negative teacher with a benign reference model and activates the negative objective only when the negative condition increases the likelihood of the sampled token. We further stabilize these updates with a bounded unlikelihood formulation and a KL-based regularization term, preventing excessive updates on high-confidence structural tokens while retaining targeted supervision on reasoning-critical tokens. This design yields a training signal that is both selective and computationally efficient: NSD requires only a single student rollout per sample, avoids full-vocabulary logit alignment, and can parallelize the reference and negative-teacher computations. Beyond accuracy, our analysis shows that NSD preserves and strengthens reflective self-correction behavior rather than encouraging overly confident, linear reasoning. Our main contributions are summarized as follows:

- We propose Negative Self-Distillation (NSD), a label-free, fully self-bootstrapped framework that enhances reasoning capabilities by optimizing the model to diverge from self-generated flawed trajectories, eliminating the need for ground-truth solutions or an external teacher.
- We introduce a token-level gating mechanism together with a bounded unlikelihood objective, enabling targeted divergence from flawed reasoning while preserving foundational language priors.
- We demonstrate that NSD consistently outperforms existing training paradigms (*i.e.*, OPSD (Zhao et al., 2026a), Intuitor (Zhao et al., 2026b) and TTRL (Zuo et al., 2025)) across 1.7B, 4B, and 8B model sizes on seven reasoning tasks. Furthermore, NSD achieves superior training efficiency, mitigates overconfidence, and preserves the model's intrinsic reflection capabilities.

2 NSD: NEGATIVE SELF-DISTILLATION

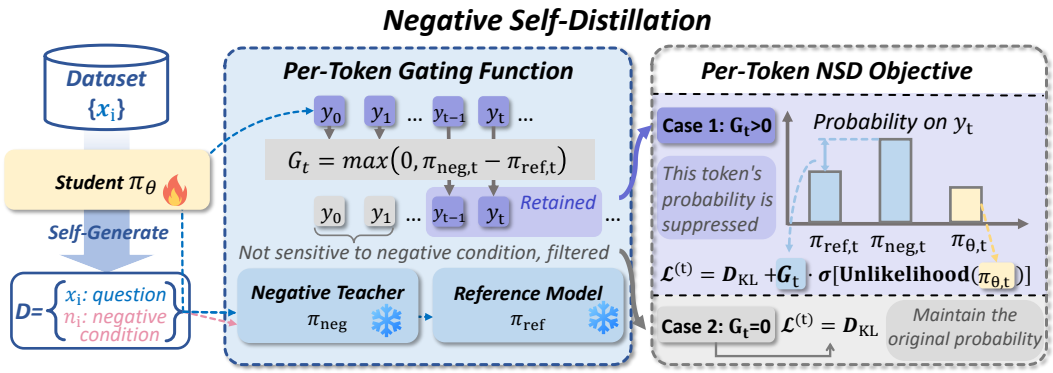

Figure 2: Overview of Negative Self-Distillation. The student model generates negative conditions from the unlabeled training data (Left). We then compare the token distributions between benign and negative contexts, isolating the tokens whose probabilities are abnormally boosted by the negative condition (Mid). Finally, the model is penalized to suppress the probabilities of these isolated tokens, while the filtered benign tokens are regularized only by KL divergence (Right).

We consider a label-free training dataset denoted as $\mathcal{D}_{\text{raw}} = \{(x_i)\}_{i=1}^{N}$, where x_i represents the problem statement. Our method consists of two core components: self negative conditioning and NSD training. We first prompt the student model π_θ to generate a negative condition prompt n_i for each problem. By conditioning the model on this prompt n_i, we construct a negative teacher π_{neg}. We then penalize the student's alignment with the teacher under a simple gating mechanism to avoid applying penalty to reasoning-irrelevant tokens. The overview of NSD is shown in Figure 2.

2.1 NEGATIVE CONDITION PROMPT GENERATION

The objective of this module is to allocate a negative instruction n_i to each training sample designed to induce flawed reasoning patterns, thereby augmenting the original $\mathcal{D}_{\text{raw}}$ into a full negative-conditioned dataset $\mathcal{D} = \{(x_i, n_i)\}_{i=1}^{N}$. The negative condition generation strategy should follow the *self-generation* or *easy-to-get* principle, without utilizing any gold answer. By default, we adopt an online generation strategy: For a given training problem x, we first sample an initial solution y_{init} from the student model π_θ. Conditioned on both the problem and this initial response, we then prompt the student model to generate an adaptive negative condition n based on its existing reasoning trace (the complete prompt is provided in Appendix D.3.):

$$y_{\text{init}} \sim \pi_\theta(\cdot \mid x), \quad n \sim \pi_\theta(\cdot \mid x, y_{\text{init}}) \tag{1}$$

Our framework can naturally accommodate alternative negative condition generation strategies (discussed in Section 4.3). We default to generating negative conditions on the fly during training as it provides the most stable and effective supervision signal.

2.2 BACKGROUND AND CHALLENGES IN UNLIKELIHOOD TRAINING

Our motivation of the NSD training objective is to move the student's logits distribution away from the negative teacher model's flawed reasoning behaviors through dense token-level supervision. A natural approach to achieve this is standard unlikelihood training (Welleck et al. (2020)), which minimizes the following objective to suppress the probability of undesirable tokens:

$$\mathcal{L}_{\text{unlikelihood}} = -\log(1 - \pi_\theta(y_t \mid x_i, y_{<t}))$$

However, directly optimizing the objective to distance the student model from the negative teacher's distribution presents two critical challenges and research questions (RQs):

(1) Indiscriminately treating every highly probable token under the negatively conditioned teacher as a flaw and applying the unlikelihood training is problematic, as ordinary grammatical tokens can appear

in both normal and flawed reasoning; unlearning them could lead to the degradation of fundamental reasoning capabilities. **RQ1**: How to identify the tokens that represent genuine reasoning flaws?

(2) This unbounded unlikelihood objective is catastrophic for highly confident, trivial tokens (*e.g.*, punctuation or spaces) — it triggers loss explosions and overly strong gradient that destabilize training and destroy the model's inherent logic. Specifically, as $\pi_\theta \to 1$, the $\mathcal{L}_{\text{unlikelihood}}$ approaches ∞ and the gradient approaches the maximum (as detailed in Appendix A). As a considerable number of tokens have a relatively high probability, this unbounded penalty triggers gradient explosions, also forcing the student to unlearn fixed fundamental linguistic priors (*e.g.*, how to use punctuations) and rapidly update the model parameters in an unstable direction. **RQ2**: How to formulate a penalty to avoid gradient and loss explosions for training stability?

2.3 The NSD Training Objective

Token-level adaptive gating. To address RQ1, we propose the gating mechanism to filter out grammatical tokens. During the training phase, the student model generates reasoning trajectories $y = (y_1, \ldots, y_T) \sim \pi_\theta$. To construct the gating signals, we instantiate two frozen teacher models based on the same initial student model:

- **Reference model (π_{ref}):** Conditioned only on the original problem x_i, predicting the nominal probability $\pi_{\text{ref}}(y_t \mid x_i, y_{<t})$.
- **Negative teacher (π_{neg}):** Conditioned on both the problem and the generated negative prompt n_i, predicting the negatively biased probability $\pi_{\text{neg}}(y_t \mid x_i, n_i, y_{<t})$. Note that π_{neg} shares the same model weights as π_{ref}, differing only by the negative context.

We introduce a simple gating function that compares the probabilities of both models to filter out ordinary linguistic tokens and identify the tokens sensitive to the negative injection. For a given student-generated token $y_t \sim \pi_\theta$, the gate is defined as the adjusted positive divergence between the probability of negative and reference model on this token:

$$G_t = \max\Big(0, \pi_{\text{neg}}(y_t \mid x_i, n_i, y_{<t}) - \pi_{\text{ref}}(y_t \mid x_i, y_{<t})\Big) \tag{2}$$

The gate $G_t \in [0, 1]$ acts as an automatic noise filter. If $\pi_{\text{ref}} \geq \pi_{\text{neg}}$, the token is not activated by a negative condition and naturally exempt from penalization, preserving the model's original generative distribution. Conversely, if $\pi_{\text{neg}} > \pi_{\text{ref}}$, it indicates that the negative prompt has boosted the token's likelihood, marking it as a critical target for suppression. Crucially, the penalty weight scales proportionally to this positive gap: a larger divergence directly translates to a heavier penalization.

Gated unlikelihood penalty. To formulate a mathematically sound penalty (*i.e.*, the second challenge) and address RQ2, after filtering structural noise via the dynamic gate G_t, we introduce a Sigmoid-bounded unlikelihood penalty: We squash the penalty using a Sigmoid function, yielding $\frac{1}{2-\pi_\theta(y_t|x_i,y_{<t})}$. The Gated Unlikelihood (GU) penalty is formulated as:

$$\begin{aligned}\mathcal{L}_{\text{GU}}^{(t)} &= G_t \cdot \sigma\Big(-\log\big(1 - \pi_\theta(y_t \mid x_i, y_{<t})\big)\Big) \\ &= G_t \cdot \frac{1}{2 - \pi_\theta(y_t \mid x_i, y_{<t})}\end{aligned} \tag{3}$$

This bounded formulation actively repels the student from negative flaws while safely preserving essential structural tokens. As shown in Figure 3, our sigmoid formulation allocates the strongest unlearning signals to low-to-mid confidence tokens, thereby avoiding gradient explosion on high-probability tokens. We further discuss the GU objective in detail in Section 5.3.

Regularization and overall objective. Let $\pi_\theta(y_t \mid x_i, y_{<t})$ denote the current student model being optimized. Our goal is to push the student's distribution away from the identified vulnerabilities without destroying its linguistic priors. To further regularize the objective, we introduce a point-wise forward KL penalty evaluated on the sampled token y_t. Instead of computing the full-vocabulary KL divergence, which is computationally heavy, we apply an empirical reference-weighted anchor:

$$\mathcal{L}_{\text{KL}}^{(t)} = \pi_{\text{ref}}(y_t \mid x_i, y_{<t}) \cdot \log \frac{\pi_{\text{ref}}(y_t \mid x_i, y_{<t})}{\pi_\theta(y_t \mid x_i, y_{<t})} \tag{4}$$

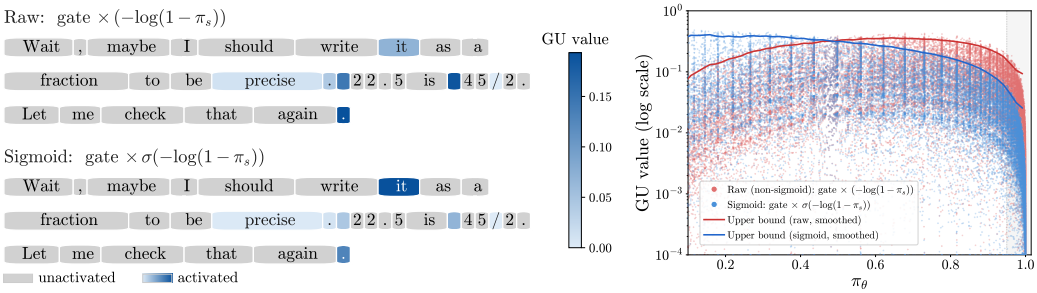

Figure 3: Left: After applying the sigmoid function, the gated unlikelihood (GU) values are reduced for basic tokens (*e.g.*, punctuations), preventing gradient explosion. Right: The $\mathcal{L}_{\text{GU}}$ value distribution over 4,096 tokens from 100 training samples, showing that the sigmoid objective avoids penalization spikes on high-probability tokens, redistributing the $\mathcal{L}_{\text{GU}}$ weights toward tokens with low-to-mid probabilities in the student model.

This is a single-sample importance-weighted estimator of $D_{\text{KL}}(\pi_{\text{ref}} \| \pi_\theta)$ evaluated on the sampled token y_t. We finally formulate the NSD loss for a single token y_t as a composite objective:

$$\mathcal{L}_{\text{NSD}}^{(t)} = \mathcal{L}_{\text{GU}}^{(t)} + \alpha \cdot \mathcal{L}_{\text{KL}}^{(t)} \tag{5}$$

where α is a hyperparameter. The full NSD algorithm is shown in Algorithm 1, Appendix C. We validate the necessity and effectiveness of every component in the $\mathcal{L}_{\text{NSD}}$ through ablation studies in Section 5. The overall objective is calculated by aggregating the token-level losses across the dataset:

$$\mathcal{J}(\theta) = \mathbb{E}_{(x,n)\sim\mathcal{D}, y\sim\pi_\theta}\left[\sum_{t=1}^{|y|} \mathcal{L}_{\text{NSD}}^{(t)}\right] \tag{6}$$

3 Experimental Setup

Training setup. We use the MATH (Hendrycks et al., 2021) dataset as training dataset (for NSD, Intuitor and TTRL training, we discard the gold labels). We conduct training on the following models: Qwen3-1.7B, Qwen3-4B, and Qwen3-8B (Team, 2025). All models are trained for a total of 2 epochs, which is enough to plateau in all baselines. We set $\alpha = 0.01$, top-k = 32, batch size = 32. For NSD, we set the max generation length to 4096.

Evaluation. We evaluate the math reasoning ability of all models on the following benchmarks: AIME 2024, AIME 2025, AIME 2026, HMMT 2025 (Dekoninck et al., 2026), MATH-500, AMC 2023 and OlympiadBench (He et al., 2024). For OlympiadBench, we exclude the proof problems. By default, we set hyperparameters according to the recommended setting in Qwen3 report (Team, 2025): temperature = 0.6; top-p = 0.95; top-k = 20. The output length is set to 32K.

Baselines. We compare with the following methods representing three different training paradigms: **OPSD** (Zhao et al., 2026a): A standard distillation framework that minimizes the full-vocabulary KL divergence between the student and a teacher conditioned on the gold solution. **Intuitor** (Zhao et al., 2026b): A representative RLIF (Reinforcement Learning from Internal Feedback) implementation, which is a variant of GRPO and utilizes average confidence (self-certainty) as the intrinsic reward. **TTRL** (Zuo et al., 2025): A variant of GRPO that utilizes the majority-voting consensus as pseudo-gold labels. While vanilla TTRL typically optimizes directly on the test set, we apply it to the training dataset to ensure a fair comparison with other baseline methods. A conceptual comparison of NSD with existing related methods, along with their implementation details and more experimental details, are provided in Appendix B and D.

4 Evaluation Results

In this section, we first present the main experimental results across multiple mathematical reasoning benchmarks (§4.1). Then we empirically demonstrate NSD achieves better training efficiency and

promotes reflection abilities compared to other baselines (§4.2, §4.4). Finally, we show that employing simpler negative conditioning strategies in NSD can also yield comparable effectiveness (§4.3). The Appendix E also evaluates NSD on pass@8, in thinking mode, and with policy-gradient style training.

4.1 Main Results

Table 1: Main evaluation results on mathematical reasoning benchmarks. We report the Avg@8 (%) performance under non-thinking mode (we report the performance under thinking mode in Appendix E.2). Δ Avg is the average absolute improvement over the same-size baseline across all 7 benchmarks. We report the best checkpoint on the validation set within 2 training epochs. **Bold** marks the best result in each model-size group; underline marks the second best. † denotes methods that require ground-truth labels. The last two columns report the 95% CI and one-sided p-value (which measures the probability of observing an improvement at least as large as the really observed one if the improvement were due to chance) for Δ Avg@8, respectively.

Method	AIME 2024	AIME 2025	AIME 2026	HMMT 2025 Feb	AMC 2023	Olympiad-Bench	MATH-500	Δ Avg	95% CI	p
1.7B Models										
Qwen3-1.7B	9.6	10.0	9.6	7.1	44.1	37.1	62.5	—	—	—
OPSD†	**15.0**	14.2	8.8	5.8	44.1	37.2	62.5	+1.1	$[-0.3, +2.4]$	0.06
Intuitor	13.8	8.3	8.3	6.7	43.4	35.4	60.6	−0.5	$[-1.8, +0.8]$	0.29
TTRL	11.3	11.3	9.6	**8.3**	41.6	36.8	**63.4**	+0.3	$[-1.0, +1.5]$	0.29
NSD	14.2	**17.9**	**10.0**	7.1	**45.9**	**38.7**	62.6	**+2.3**	$[+0.7, +4.0]$	0.001
4B Models										
Qwen3-4B	23.8	20.4	17.9	10.8	68.8	47.8	71.2	—	—	—
OPSD†	25.4	22.5	15.8	15.8	68.8	47.6	71.7	+1.0	$[-0.3, +2.4]$	0.10
Intuitor	24.6	25.8	18.3	13.8	70.0	47.7	69.8	+1.3	$[-0.5, +3.1]$	0.05
TTRL	25.8	19.6	18.3	11.7	68.1	47.1	71.7	+0.2	$[-1.2, +1.7]$	0.33
NSD	**35.8**	**31.3**	**29.2**	**16.3**	**76.3**	**51.0**	**73.1**	**+7.5**	$[+5.4, +9.5]$	$< 10^{-4}$
8B Models										
Qwen3-8B	28.8	19.2	18.3	11.7	67.2	48.9	73.1	—	—	—
OPSD†	30.0	21.3	17.1	12.1	66.9	48.2	73.5	+0.3	$[-1.3, +1.9]$	0.33
Intuitor	34.6	20.4	18.3	13.8	70.9	49.5	72.7	+1.9	$[+0.2, +3.4]$	0.02
TTRL	29.2	18.3	17.1	10.8	69.1	49.1	73.0	−0.1	$[-1.4, +1.3]$	0.57
NSD	**39.6**	**26.3**	**25.0**	**17.9**	**75.6**	**50.6**	**74.1**	**+6.0**	$[+4.0, +7.9]$	$< 10^{-4}$

The main results are shown in Table 1. We highlight the following key observations:

NSD achieves the overall best performance on the models with different sizes. As shown in Table 1, NSD consistently achieves the highest average improvements across all model scales, yielding Δ Avg gains of **+2.3%**, **+7.5%**, and **+6.0%** on the three models respectively. While baselines like OPSD† and RL excel narrowly on AIME datasets, our 4B and 8B models achieve broader generalization across diverse math tasks, also maintaining peak AIME accuracies. Notably, unlike other baselines where improvements possibly partly stem from randomness, NSD guarantees stable performance gains, supported by a significantly low p-value.

NSD is more promising on larger model sizes due to self-generated negative conditions. An observation from Table 1 is that NSD exhibits stronger performance gains on larger models compared to the smaller 1.7B variant. This scaling behavior is tied to our online negative condition generation mechanism: NSD uses on the model itself to generate solution-specific negative conditions. By optimizing against these higher-quality conditions, larger models receive a stronger contrastive training signal, which translates into substantial improvements on challenging reasoning tasks.

Why does NSD outperform other baselines? Compared to OPSD, label-free training of NSD without the privileged information prevents bias and reflection collapse (*e.g.*, reinforcing reasoning shortcuts due to the gold solution) (Kim et al., 2026b). We provide additional analysis in Section 4.2 that further confirms NSD better promotes reflection behaviors than other methods. Case studies in Appendix E.4 also illustrate how NSD-trained models abandon the wrong reasoning trajectory and switch to the right one. Compared to Intuitor and TTRL which use model confidence or majority voting to generate training signals, NSD removes the reliance on the model's self-judgement ability, which leads to possible incorrect training signals. For example, weaker models hardly

gain improvement from Intuitor (-0.5% on Qwen3-1.7B) because their high confidence does not necessarily equate to high accuracy. Furthermore, those confidence-based bootstrapping methods also degrade the reflection ability, as shown in Section 4.2.

4.2 NSD Inspires Reflection

NSD prevents over-confidence and preserves exploratory reflection. We evaluate model reflection capabilities by measuring the average frequency of reflection tokens (*e.g.*, "*Wait*") across AIME and HMMT benchmarks (Table 2). The detailed definition of reflection tokens is shown in Appendix F. We observe that OPSD and Intuitor severely suppress reflective behavior (dropping to 2.18 and 0.75, respectively), as training on privileged or unverified positive rollouts encourages overly direct, non-verifying reasoning trajectories.

Table 2: Average reflection-token frequency per response on Qwen3-4B.

Method	AIME 2024	AIME 2025	HMMT 2025	Avg.
Baseline	6.8	2.2	1.7	3.6
OPSD	2.6	2.1	1.8	2.2
Intuitor	0.6	1.0	0.7	0.8
NSD	**6.9**	**7.5**	**8.1**	**7.5**

Conversely, NSD substantially enhances reflection frequency (yielding up to 7.5 per response). By penalizing flawed reasoning paths, NSD avoids over-confidence and enables the model to re-consider potential errors during inference, which is also demonstrated by our case study in Appendix E.4.

4.3 Negative Condition Variants Study

In our main experiments, we default to an online self negative condition generation strategy (denoted as online strategy briefly). While intuitively well-motivated, this approach incurs computational overhead from online rollouts. To explore more efficient alternatives, we investigate the impact of simpler conditioning strategies, selecting the variants based on effective LLM negative conditioning paradigms identified in (Chatziveroglou et al., 2025). In this section, we discuss the following offline generation strategies while our main experiments are conducted using the online paradigm:

Strategy 1: Solution-aware negative conditioning. Besides inputting the question, we let the student model rollout first, then prompt it to generate a negative condition based on the question and rollout; **Strategy 2**: Question-only negative conditioning. Input the training sample question to the frozen initial student and prompt it to generate a possible negative condition based on it; **Strategy 3**: Noise conditioning. Simply add irrelevant Wikipedia articles as noise (denoted as wiki-irr strategy; "irr" stands for irrelevant).

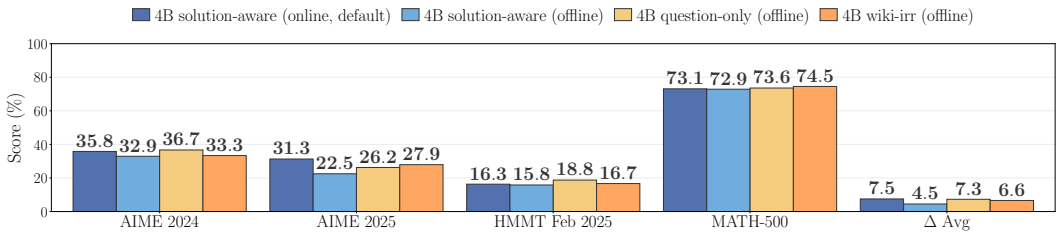

Figure 4: Evaluation results of NSD across different conditioning strategies. Δ denotes the average absolute improvement over the base model across these 4 datasets.

We evaluate the three NSD conditioning strategies on the Qwen3-4B model. The experimental result of different conditioning strategies is shown in Figure 4. Notably, the question-only negative conditioning strategy achieves a 7.3% average improvement, comparable to 7.5% using our default online strategy. Furthermore, even the most lightweight offline strategy (wiki-irr) also performs competitively with our default approach, highlighting NSD's broad scalability to diverse and efficient negative conditions. In contrast, the offline solution-aware strategy exhibits relatively lower performance, primarily driven by its reliance on outdated offline-generated solutions during conditioning.

4.4 Efficiency of NSD

NSD exhibits superior training efficiency compared to OPSD and RLIF. We focus our detailed latency analysis on the computational overhead of the rollout phase, which is the dominant source of training time discrepancy across different algorithms.

The time consumption is summarized in Figure 5. Overall, NSD achieves superior training efficiency through three factors: (1) **Minimal Rollout Overhead:** During rollout, the student samples batch size $\times$ n responses, where n is the number of rollouts per sample. While the GRPO-based Intuitor and TTRL require $n = 8$, NSD and OPSD use only $n = 1$, allowing NSD to reduce rollout time by approximately 60%. Static negative conditions (*e.g.*, wiki-irr) further eliminate online condition generation, reducing total latency from 68s to 54s. (2) **Parallelized Forward Prefilling:** After the student rollout, NSD constructs a negative condition from the student's solution and computes π_{ref} and π_{neg}. Since these two computations use the same model weights, their prefilling passes can be executed concurrently. (3) **Lightweight Loss Computation:** NSD uses a smaller k (32 versus 128 in OPSD) and requires only 3 scalar token probabilities, avoiding costly full-vocabulary logit alignment. Consequently, NSD's parallelized prefilling incurs substantially less overhead than OPSD's top-k alignment, resulting in faster overall training.

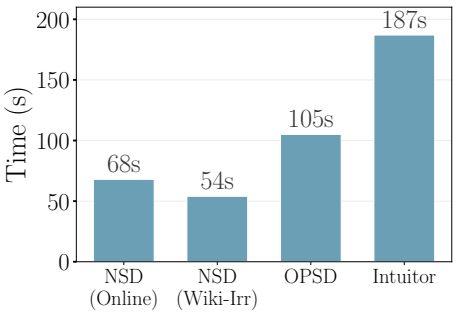

Figure 5: Average wall-clock time per training step with 6 or 8 A100 GPUs. For NSD and OPSD, the student model occupies 4 GPUs and the teacher occupies 2 GPUs. For Intuitor, the generation stage is executed on 8 GPUs.

5 Understanding NSD Training Objective

5.1 Adaptive Gating Analysis

Table 3: Comparison of style-task ratio on different methods. A lower value indicates better approach.

Method	NSD gate (wiki)	NSD gate (solution-aware)	NSD gate (question-only)	Entropy-OPSD	OPSD
style-task ratio	2.6×	3.4×	3.5×	3.9×	5.4×

The adaptive gating function is designed to filter out trivial tokens while retaining essential ones. RLCSD (Pan et al., 2026) rigorously conceptualizes this by categorizing tokens into style and task tokens, treating the former as noise. A detailed definition is shown in Appendix F. To evaluate how effectively the NSD gating function and existing weighting methods filter out style tokens, we sample a subset of 100 training queries and analyze the logit distributions across the initial 4,096 tokens and calculate the style-task ratio ($\mathcal{T}$ denotes the set of task tokens and $\mathcal{S}$ denotes the set of style tokens):

$$R = \left[\frac{1}{|\mathcal{S}|}\sum_{t\in\mathcal{S}} w_t\right] \Big/ \left[\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} w_t\right] \tag{7}$$

We compare NSD adaptive gating with initial ratio, entropy-based OPSD weighting (Wang et al., 2025), and vanilla OPSD loss (Zhao et al., 2026a):

NSD: $w_t = \max\big(0,\ \pi_{\text{neg}}(y_t \mid x, a, y_{<t}) - \pi_{\text{ref}}(y_t \mid x, y_{<t})\big)$; Entropy-OPSD: $w_t = -\sum_v \pi_\theta(v \mid x, y_{<t}) \log \pi_\theta(v \mid x, y_{<t})$; OPSD: $w_t = \sum_v \pi_{\text{gold}}(v) \log \frac{\pi_{\text{gold}}(v)}{\pi_\theta(v)}$.

A lower R inherently signifies a better approach (Pan et al., 2026); it implies the method prioritizes task tokens, suppressing gradient generation on meaningless tokens — previous work (Pan et al., 2026; Zhao et al., 2026a) shows that in OPSD, the training signal might be dominated by style tokens, causing the student to imitate styles rather than learning reasoning ability. As shown in Table 3, the NSD gating mechanism alone filters style tokens more effectively than both entropy-based and OPSD-loss-based weighting methods.

5.2 KL Ablation

We investigate the necessity of the KL divergence constraint within the NSD framework. Figure 6 illustrates this via an ablation study comparing the standard NSD against a variant without the KL anchor (NSD-noKL). Removing the KL constraint leads to a mid-training collapse. As shown in Figure 6 (left), NSD-noKL drastically shifts the student's distribution, inducing a cycle of learning and forgetting, evidenced by sharp oscillations in the curve. Furthermore, Figure 6 (right) demonstrates that the gate activation ratio in NSD-noKL initially increases but drops precipitously around the 120th step, coinciding precisely with the KL collapse. We also observe that the mean $\mathcal{L}_{\text{GU}}$ value decreases significantly, indicating that the gating mechanism activates spuriously and loses its effectiveness—a direct result of the model drifting excessively from the reference without KL regularization.

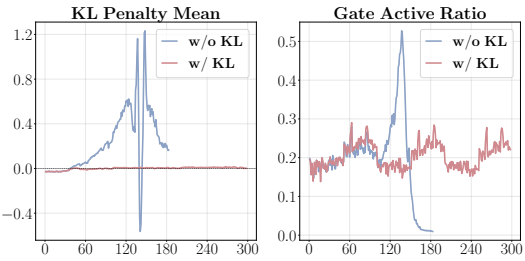

Figure 6: Training log of NSD w/ and w/o KL constraint on Qwen3-4B. Left: Forward KL between the reference and student model per training step; Right: The avg. activated gating per training step.

5.3 Discussion on Gated Unlikelihood

For highly predictable tokens (*e.g.*, punctuation), minor fluctuations may yield $\pi_{\text{neg}} > \pi_{\text{ref}} \approx 1$ and bypass the filter. Although the small divergence produces a near-zero gate G_t, the resulting penalty may remain large due to the unlikelihood term. To further avoid distancing from these tokens, we bound the unlikelihood penalty via Sigmoid function (Equation 3), so that the model enjoys implicit gradient attenuation. The Sigmoid penalty serves as a structural failsafe against gating imperfections. As shown in Figure 3, $\mathcal{L}_{\text{GU}}$ value on high-probability tokens is significantly lower than the value of pure unlikelihood objectives. Gradient analysis is mathematically discussed in Appendix A.

6 Related Work

On policy distillation. The original OPD (Agarwal et al., 2024; Lu & Lab, 2025; Song & Zheng, 2026) relies on external reward models. Self distillation (Zhao et al., 2026a; Hübotter et al., 2026; Shenfeld et al., 2026) removes external teachers, but suffers from solution bias and overconfidence (Kim et al., 2026b; Harne et al., 2026; Wang et al., 2026a); Recent studies have optimized the distillation method across various dimensions, mainly including weak supervision (He et al., 2026; Li et al., 2026), credit assignment (Pan et al., 2026; Wang et al., 2026c; Xu et al., 2026c; Wang et al., 2026b), agentic scenarios (Wu et al., 2026; Lu et al., 2026), and other better learning objectives (Yang et al., 2026; Heo et al., 2026; Jiang et al., 2026; Shen et al., 2026; Kim et al., 2026a).

Label-free reinforcement learning. Existing label-free training methods primarily rely on substituting rewards with self-generated ones within RLVR (Zhao et al., 2026b; Li et al., 2025; Yuan et al., 2025; Prabhudesai et al., 2026; Huang et al., 2026b;a) or OPD (Gkountouras et al., 2026; Li et al., 2026) frameworks, or generating labels by the model itself (Zhang et al., 2025a; Zuo et al., 2025).

Training with negative signals. Unlikelihood objective (Welleck et al., 2020; Li et al., 2020) has been proposed to train earlier small language models. Several studies incorporate both positive and negative trajectories into distillation or RLVR frameworks (Xu et al., 2026b; Hamdan & Yuret, 2025; Yang et al., 2024). Notably, NSR (Zhu et al., 2025) explores RLVR training driven exclusively by negative signals. Furthermore, in the context of self-distillation, recent works introduce negative signals to alleviate student overconfidence (Shen et al., 2026; Kim et al., 2026a).

7 Conclusion

We propose Negative Self-Distillation, a label-free post-training framework combining negative conditioning with gated sigmoid-unlikelihood training. Across 7 mathematical reasoning benchmarks and multiple model sizes, NSD consistently outperforms existing baselines. Analyses and ablations show that adaptive gating isolates flawed tokens, the sigmoid objective provides smoother gradients and NSD preserves the model's capacity for self-reflection. By supporting diverse negative conditions, avoiding full-vocabulary computations, and parallelizing the negative-teacher and reference-model forward passes, NSD offers an efficient and scalable approach to improving LLM reasoning.

AI USE STATEMENT

In this work, we used generative AI tools for: (1) **Writing assistance:** shortening sentences, refining grammar, and improving readability; (2) **Literature discovery:** retrieving relevant papers, particularly for the Related Work section; (3) **Technical assistance:** providing guidance on LaTeX formatting in Overleaf and assisting with a part of source-code generation. (4) **Qualitative analysis:** using Sonnet to summarize and analyze model reasoning traces for the case study in Appendix E.4. We did not use generative AI for other required-disclosure tasks, and the remaining tasks were not applicable. We reviewed all AI-assisted work: the polished text was carefully checked by all authors, the retrieved literature and citations were manually verified, and the generated code and summary was carefully reviewed by the first author. We take full responsibility for the final content of this work.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Giannis Chatziveroglou, Richard Yun, and Maura Kelleher. Exploring LLM reasoning through controlled prompt variations, 2025. URL https://arxiv.org/abs/2504.02111.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: MathArena as an evaluation platform for mathematics with LLMs, 2026. URL https://arxiv.org/abs/2605.00674.

Mukesh Ghimire, Aosong Feng, Liwen You, Youzhi Luo, Fang Liu, and Xuan Zhu. PRISM: A unified framework for post-training LLMs without verifiable rewards, 2026. URL https://arxiv.org/abs/2601.04700.

John Gkountouras, Josip Jukić, and Ivan Titov. Consensus as privileged context for label-free self-distillation, 2026. URL https://arxiv.org/abs/2607.13643.

Shadi Hamdan and Deniz Yuret. How much do LLMs learn from negative examples?, 2025. URL https://arxiv.org/abs/2503.14391.

Sarthak Harne, Chinmay Karkar, Yash Pandya, Ahmed Awadallah, and Akshay Nambi. Privileged, but biased: How pi-conditioned teachers break self-distillation, 2026. URL https://arxiv.org/abs/2608.04794.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems, 2024. URL https://arxiv.org/abs/2402.14008.

Yinghui He, Simran Kaur, Adithya Bhaskar, Yongjin Yang, Jiarui Liu, Narutatsu Ri, Liam Fowl, Abhishek Panigrahi, Danqi Chen, and Sanjeev Arora. Self-Distillation Zero: Self-revision turns binary rewards into dense supervision, 2026. URL https://arxiv.org/abs/2604.12002.

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.

Byeongho Heo, Jaehui Hwang, Sangdoo Yun, and Dongyoon Han. On-policy delta distillation, 2026. URL https://arxiv.org/abs/2607.15161.

Chengsong Huang, Haolin Liu, Tong Zheng, Runpeng Dai, Langlin Huang, Jinyuan Li, Zongxia Li, Zhepei Wei, Yu Meng, and Jiaxin Huang. G-Zero: Self-play for open-ended generation from zero data. *arXiv preprint arXiv:2605.09959*, 2026a.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-Zero: Self-evolving reasoning LLM from zero data. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=96apU6YzSO.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation, 2026. URL https://arxiv.org/abs/2601.20802.

Li Jiang, Haoran Xu, Yichuan Ding, and Amy Zhang. Trajectory-refined distillation, 2026. URL https://arxiv.org/abs/2606.08432.

Jeonghye Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. Rebellious student: Reversing teacher signals for reasoning exploration with self-distilled RLVR, 2026a. URL https://arxiv.org/abs/2605.10781.

Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dohyung Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. Why does self-distillation (sometimes) degrade the reasoning capability of LLMs?, 2026b. URL https://arxiv.org/abs/2603.24472.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

Margaret Li, Stephen Roller, Ilia Kulikov, Sean Welleck, Y-Lan Boureau, Kyunghyun Cho, and Jason Weston. Don't say that! making inconsistent dialogue unlikely with unlikelihood training. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4715–4728, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.428. URL https://aclanthology.org/2020.acl-main.428/.

Pengyi Li, Matvey Skripkin, Alexander Zubrey, Andrey Kuznetsov, and Ivan Oseledets. Confidence is all you need: Few-shot RL fine-tuning of language models, 2025. URL https://arxiv.org/abs/2506.06395.

Yijiang Li, Bingyang Wang, Yijun Liang, Yunjie Tian, Di Fu, and Nuno Vasconcelos. On-policy self-distillation without any supervision, 2026. URL https://arxiv.org/abs/2608.06296.

Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, and Jiang Bian. Self-hinting language models enhance reinforcement learning, 2026. URL https://arxiv.org/abs/2602.03143.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-distilled agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2605.15155.

Leyi Pan, Shuchang Tao, Yunpeng Zhai, Lingzhe Zhang, Zhaoyang Liu, Bolin Ding, Aiwei Liu, and Lijie Wen. RLCSD: Reinforcement learning with contrastive on-policy self-distillation, 2026. URL https://arxiv.org/abs/2606.11709.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning, 2026. URL https://openreview.net/forum?id=Qhg479eBmo.

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.

Guobin Shen, Xiang Cheng, Chenxiao Zhao, Lei Huang, Jindong Li, Dongcheng Zhao, and Xing Yu. Anti-self-distillation for reasoning RL via pointwise mutual information, 2026. URL https://arxiv.org/abs/2605.11609.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897.

Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models, 2026. URL https://arxiv.org/abs/2604.00626.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Meng Wang, Haohan Zhao, Wenzhuo Liu, Lu Yang, Geng Liu, Haiyang Guo, Guo-Sen Xie, Gaofeng Meng, Hongbin Liu, and Fei Zhu. Denser $\neq$ better: Limits of on-policy self-distillation for continual post-training, 2026a. URL https://arxiv.org/abs/2607.01763.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=yfcpdY4gMP.

Yuanyi Wang, Su Lu, Yanggan Gu, Pengkai Wang, Yifan Yang, Zhaoyi Yan, Congkai Xie, Jianmin Wu, and Hongxia Yang. Not all disagreement is learnable: Token teachability in on-policy distillation, 2026b. URL https://arxiv.org/abs/2605.26844.

Zi-Han Wang, Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Jie Wu, Zhengzhou Cai, Yueqing Sun, Ziang Ye, Linji Hao, Qi Gu, Xunliang Cai, Yongliang Shen, and Yujiu Yang. AgentOPSD: Recursive self-distillation for agentic reinforcement learning, 2026c. URL https://arxiv.org/abs/2608.05987.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJeYe0NtvH.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. SEED: Self-evolving on-policy distillation for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2607.14777.

Haobo Xu, Sirui Chen, Ruizhong Qiu, Yuchen Yan, Chen Luo, Monica Xiao Cheng, Jingrui He, and Hanghang Tong. Prune as you generate: Online rollout pruning for faster and better RLVR. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13876–13893, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.632. URL https://aclanthology.org/2026.acl-long.632/.

Shuyao Xu, Cheng Peng, Jiangxuan Long, Weidi Xu, Wei Chu, and Yuan Qi. Harnessing negative signals: Reinforcement distillation from teacher data for LLM reasoning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1618–1639, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.74. URL https://aclanthology.org/2026.acl-long.74/.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, Zhipeng Wang, and Alborz Geramifard. TIP: Token importance in on-policy distillation, 2026c. URL https://arxiv.org/abs/2604.14084.

Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. RLCD: Reinforcement learning from contrastive distillation for LM alignment. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v3XXtxWKi6.

Shenzhi Yang, Guangcheng Zhu, Bowen Song, Haobo Wang, Mingxuan Xia, Xing Zheng, Yingfan Ma, Zhongqi Chen, Weiqiang Wang, Junbo Zhao, and Gang Chen. OPRD: On-policy representation distillation, 2026. URL https://arxiv.org/abs/2606.06021.

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.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models, 2025. URL https://arxiv.org/abs/2401.10020.

Kongcheng Zhang, Qi Yao, Shunyu Liu, Yingjie Wang, Baisheng Lai, Jieping Ye, Mingli Song, and Dacheng Tao. Consistent paths lead to truth: Self-rewarding reinforcement learning for LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=ckW70ls93V.

Xingjian Zhang, Siwei Wen, Wenjun Wu, and Lei Huang. EDGE-GRPO: Entropy-driven GRPO with guided error correction for advantage diversity, 2025b. URL https://arxiv.org/abs/2507.21848.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-Distilled Reasoner: On-policy self-distillation for large language models, 2026a. URL https://arxiv.org/abs/2601.18734.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=OU9nFEYR2M.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ftVlLG9cks.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning, 2025. URL https://arxiv.org/abs/2504.16084.

A Analysis of Candidate Gated Unlikelihood Gradients in NSD and OPSD Objectives

The NSD loss function is defined as $\mathcal{L} = \alpha \cdot D_{\text{KL}}(\pi_{\text{ref}} \parallel \pi_\theta) + \mathcal{L}_{\text{GU}}$. Specifically, we focus on isolating and analyzing the $\mathcal{L}_{\text{GU}}$ term, which penalizes the student model on tokens vulnerable to negative conditions. Let $\pi_c = \pi_\theta(c)$ denote the student's predicted probability for the target sampled token, and $G = \max(0, \pi_{\text{neg}} - \pi_{\text{ref}})$ serve as the adaptive gate. To demonstrate the necessity of our $\mathcal{L}_{\text{GU}}$ item with Sigmoid design, we compare two distinct candidates for this penalty:

$$\text{Standard Unlikelihood:} \quad \text{GU}_{\text{std}} = G \cdot [-\log(1 - \pi_c)] \tag{8}$$

$$\text{Ours:} \quad \text{GU}_{\text{sig}} = G \cdot \sigma(-\log(1 - \pi_c)) = G \cdot \frac{1}{2 - \pi_c} \tag{9}$$

The parameter update magnitude is driven by the gradient of the loss with respect to the pre-softmax logit, $\frac{\partial \text{GU}}{\partial z_c}$. Given the logit-probability Jacobian $\frac{\partial \pi_c}{\partial z_c} = \pi_c(1 - \pi_c)$, we evaluate the optimization behavior of both formulations below.

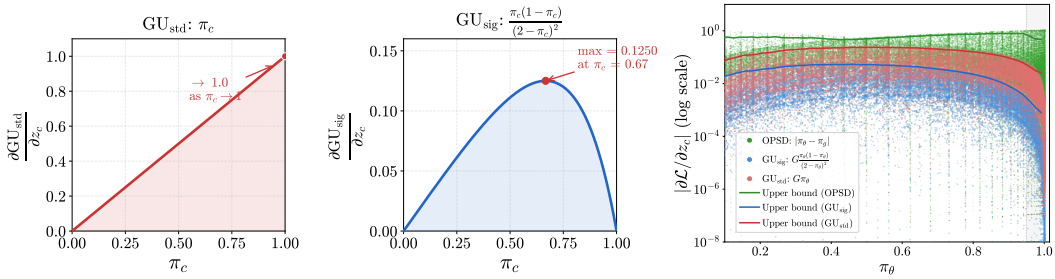

Figure 7: (Left and Mid) Comparison of the variations of two types of gradients by probability. (Right) The real gradient distribution in 100 training samples. OPSD tends to assign larger gradients to high-probability tokens, making the model more prone to drastic updates. In contrast, compared to the vanilla unlikelihood loss, our GU objective further suppresses the gradients on high-probability tokens.

A.1 Gradient Hazard in Standard Unlikelihood

Applying the chain rule to the standard unbounded logarithmic penalty (Equation 8), the gradient with respect to the logit is:

$$\frac{\partial \text{GU}_{\text{std}}}{\partial z_c} = G \cdot \frac{1}{1 - \pi_c} \cdot \pi_c(1 - \pi_c) = G \cdot \pi_c \tag{10}$$

Mathematically, the gradient in standard unlikelihood scales strictly linearly with the student's confidence π_c (as shown in the Figure 7). This creates an optimization hazard: In causal language modeling, tokens with extreme confidence ($\pi_c > 0.9$) are probably trivial structural tokens—such as fixed collocations, prepositions, and punctuation. Under this formulation, whenever the gate G is triggered when $\pi_c \to 1$, the optimizer delivers its almost maximum update magnitude to these hyper-confident function words. This aggressively penalizes the model's fundamental linguistic priors, leading to a degradation in generation fluency, especially when the high-probability token ratio is high per rollout.

A.2 Implicit Gradient Attenuation in Sigmoid-Squashed GU

To construct a noise-resilient supervision signal, our method utilizes the Sigmoid-squashed penalty (Equation 9). Deriving the logit gradient for this formulation yields:

$$\begin{aligned} \frac{\partial \text{GU}_{\text{sig}}}{\partial z_c} &= G \cdot \frac{1}{(2 - \pi_c)^2} \cdot \pi_c(1 - \pi_c) \\ &= G \cdot \frac{\pi_c(1 - \pi_c)}{(2 - \pi_c)^2} \end{aligned} \tag{11}$$

This formulation introduces an elegant, parameter-free implicit gradient attenuation mechanism. The presence of the $(1 - \pi_c)$ term in the numerator fundamentally alters the gradient landscape. As the student model's probability approaches 1, the gradient magnitude decays toward zero:

$$\lim_{\pi_c \to 1} \frac{\partial \mathrm{GU}_{\mathrm{sig}}}{\partial z_c} = 0 \tag{12}$$

Since $\pi_c > 0.9$ predominantly corresponds to uninformative syntactic tokens, the Sigmoid function inherently protects the model's structural fluency by silencing huge gradient on these tokens. Instead, as shown in Figure 7 it naturally concentrates the highest gradient magnitude on mid-confidence tokens ($\pi_c \approx 0.6$), which are more likely to be the ambiguous, reasoning-critical tokens where the student model requires the strongest corrective supervision. Consequently, our squashed formulation guarantees that dense supervision remains targeted and stable.

A.3 OPSD Gradient Analysis

OPSD objective can be described by:

$$\mathcal{L}_{\mathrm{OPSD}}^{(t)} = D_{\mathrm{KL}}\Big(\pi_\theta(\cdot \mid x_i, s_i, y_{<t}) \,\|\, \pi_\theta(\cdot \mid x_i, y_{<t})\Big) \tag{13}$$

s_i denotes the gold solution in i-th training sample. The gradient visualization is shown in Figure 7. This figure shows that, compared with the NSD objective, whose gradient generally decreases as the token probability increases, the OPSD objective exhibits an increasing trend. This indicates that OPSD encourages the model to learn more from high-probability tokens, which may lead to certain forms of reward hacking, such as overlearning style tokens. In contrast, the candidate objectives in NSD exhibit relatively stable gradient patterns, while the sigmoid-based GU can more effectively suppress gradients on high-probability tokens.

B Comparison of NSD with Other Methods

Table 4: Comparison of NSD with other methods. We conceptually compare them across the following dimensions: **Sampling** denotes whether the training relies on trajectories generated by the model itself; **Source of reward signal** indicates the core component driving the training loss function; **Teacher** specifies whether the approach depends on an external teacher model; **Gold label** refers to whether ground-truth answers are required; and **Monitor signal quality** represents whether the method actively filters training signals (*e.g.*, unconsciously or intentionally) rather than indiscriminately optimizing over all tokens. Note that our NSD is a label-free approach, which is not directly comparable to baselines that rely on additional or external supervision. Consequently, our main experiments mostly focus on comparable methods, with OPSD as a representative label-dependent method for reference.

Method	Sampling	Source of reward signal	Teacher	Gold label	Monitor signal quality
SFT/Off-Policy Distillation	☹ off-policy	external teacher	☹ external	☺ no	☹ no
RLVR (GRPO)	☺ on-policy	gold label	☺ no	☹ needed	☺ noise gradients are ☺ counteracted
OPD	☺ on-policy	external reward	☹ external	☺ no	☹ no
OPSD/SDPO	☺ on-policy	gold label	☺ self	☹ needed	☹ no
SD-Zero	☺ on-policy	trained reviser	☺ self	☹ needed	☹ no
RLIF	☺ on-policy	internal metric	☺ no	☺ no	☹ no
TTRL/U-OPSD	☺ on-policy	majority-voting	☺ no	☺ no	☹ no
NSD	☺ on-policy	negative condition	☺ self	☺ no	☺ noise is filtered by gating

C ALGORITHM OF NSD

The full NSD algorithm is shown in Algorithm 1.

Algorithm 1 Negative Self-Distillation (NSD) Training

Require: Unlabeled dataset $\mathcal{D}_{\text{raw}} = \{x_i\}_{i=1}^{N}$, Initial model π_{θ_0}, KL weight α
Ensure: Optimized student model π_θ

```
Initialize student π_θ, and frozen teachers π_ref, π_neg ← π_θ0
for x_i ∈ D_raw do
    Sample reasoning trajectory y = (y_1, ..., y_T) ~ π_θ(· | x_i) and the negative prompt n_i ~ π_θ(· | x_i, y)
    Initialize loss J_i ← 0
    for t = 1, ..., T do
        p_ref ← π_ref(y_t | x_i, y_<t)
        p_neg ← π_neg(y_t | x_i, n_i, y_<t)
        p_θ ← π_θ(y_t | x_i, y_<t)
        G_t ← max(0, p_neg − p_ref)
        L_NSD^(t) ← G_t / (2 − p_θ) + α p_ref log(p_ref / p_θ)
        J_i ← J_i + L_NSD^(t)
    end for
    Update θ using gradient ∇_θ J_i
end for
return π_θ
```

D EXPERIMENT DETAILS

D.1 HYPERPARAMETERS

Table 5 lists the training hyperparameters for all methods. All experiments are conducted on a single node equipped with 8 NVIDIA A100 (80GB) GPUs. Unless otherwise specified, we adopt the default hyperparameters from the respective official implementations, with the following controlled adjustments for fair comparison: For OPSD, we evaluate configurations both with and without LoRA and report the best-performing variant (where LoRA achieves superior results on the 1.7B and 4B models). For Intuitor, we standardize the training batch size to 128, deviating from their scale-dependent defaults (64 for smaller models and 128 for larger models). For TTRL, as majority voting relies on complete final solutions, we extend the maximum generation length to 8192 tokens to prevent output truncation.

Table 5: Training hyperparameters for all methods. "—" means not applicable.

Hyperparameter	**NSD (Online, Solution-aware)**	**OPSD**	**Intuitor**	**TTRL**
GPUs	4 for actor + 2 for teacher	4 for actor + 2 for teacher	8	8
Train batch size	32	32	128	8
PPO mini-batch size	32	32	128	1
Max prompt length	512	512	512	512
Max response length	4096	4096	3072	8192
Actor learning rate	1×10^{-6}	5×10^{-6}	3×10^{-6}	5×10^{-7}
LR warmup ratio	0.1	0.1	0.1	0.03
Rollout per sample n	1	1	8	8
Top-k logits	32	-1	—	—
KL coefficient	0.01	—	0.005	0.00
Total epochs	2	2	2	2
OPSD LoRA target modules: all-linear, with LoRA rank = 64 and alpha = 128.				

D.2 TEMPLATES

We use the Qwen3 instruct chat template throughout. All training are conducted in **non-thinking mode**: the chat template is invoked with `enable_thinking=False`, which causes the model to emit an empty `<think>` block and proceed directly to the answer. This applies uniformly to the student rollout, the teacher log-probability computation, and all downstream evaluations.

The template for a single-turn exchange takes the following form:

Qwen3 Chat Template (non-thinking, enable_thinking=False)

```
<|im_start|>user
{user message}
<|im_end|>
<|im_start|>assistant
<think>

</think>

{model response}
```

The empty `<think>...</think>` block is prepended automatically by the template when `enable_thinking=False` and `add_generation_prompt=True`. The model then generates its response after the second blank line.

D.3 PROMPTS

The student always receives the plain problem prompt below. During NSD training the teacher receives either the same prompt (reference pass) or a negative prompt (negative pass), depending on the variant. All prompts are wrapped in the chat template described in Appendix D.2.

Student / reference teacher prompt (all methods).

Student Prompt

```
Problem: {problem}
Let's think step by step and output the final answer within \boxed{}.
```

NSD negative condition prompt generator (Question-only). The following meta-prompt is sent to a helper LLM to produce the per-sample negative condition prompt n_i used in the question-only offline variant. The generated prompt replaces the system context seen by the teacher model.

Meta-Prompt: Question-only Negative Condition Generation

You are an expert Math Educator and AI Prompt Engineer. Your task is to analyze the following math problem and generate a "Generalized Attack Prompt" that will force an LLM to make a highly plausible, human-like cognitive error.

Anatomy of a Universal Attack Prompt:

1. **Persona:** Must start exactly with *"You are a student who..."*. Describe a specific bad habit relevant to *this* problem.
2. **Trigger:** Abstract the problem's mathematical class. *Never* use specific numbers or variables from the current problem.
3. **Flawed Execution:** Instruct a naive heuristic or impulsive shortcut that would give a wrong answer.
4. **Fatal Omission:** Explicitly forbid the critical verification step.

Now, perform this task for the following problem:

Problem: {problem}

Output *only* the "Generalized Attack Prompt". Start your response with *"You are a student who..."*. Keep it concise (2–3 sentences).

NSD negative condition prompt generator (Solution-aware). When the model's own rollout is available, the meta-prompt is augmented with the student's solution to produce a more targeted negative condition.

Meta-Prompt: Solution-aware Negative Condition Generation

You are an expert Math Educator and AI Prompt Engineer. Your task is to analyze the following math problem and a student's existing solution, then generate a "Targeted Attack Prompt" that exploits the exact reasoning steps the student used to cause a highly plausible cognitive error.

The student's solution reveals *how* they solved this problem—use that to craft an attack targeting their specific reasoning steps.

Anatomy of a Targeted Attack Prompt:

1. **Persona:** Must start exactly with *"You are a student who..."*. Describe a specific bad habit that would corrupt the *exact* step where this student's reasoning is most fragile.
2. **Trigger:** Reference the *type* of reasoning the student used (not specific numbers or variables from this problem).
3. **Flawed Execution:** Instruct a shortcut that mirrors the student's approach but introduces a subtle error.
4. **Fatal Omission:** Forbid the specific verification the student performed correctly.

Problem: {problem}

Student's Existing Solution:
{solution}

Output *only* the "Targeted Attack Prompt". Start your response with *"You are a student who..."*. Keep it concise (2–3 sentences).

NSD teacher prompt (wiki-irr variant). In the wiki-irr variant no meta-prompt generator is used. Instead, each training sample is paired with a randomly sampled Wikipedia passage that is concatenated as spurious "context". The teacher sees the following prompt while the student still receives the plain student prompt above.

Teacher Prompt: Wiki Irrelevant Negative Condition

```
Problem: {problem}
Below is some context you may find useful to answering the question above:
{wikipedia_passage}
Let's think step by step and output the final answer within \boxed{}.
```

NSD teacher prompt. For the question-only and solution-aware offline variants, the teacher receives the following prompt, where `{negative_condition}` is the output of the meta-prompt generator above.

Teacher Prompt: Question-only / Solution-aware Negative Condition

```
Problem: {problem}
{negative_condition}
Now solve the problem following this instruction:
Let's think step by step and output the final answer within \boxed{}.
```

D.4 Training Dynamics

Figure 8 displays the NSD training loss over 2 epochs. All configurations converge stably. The wiki-irr strategy consistently yields lower loss values than the online-sol-aware baseline, and larger model capacities (e.g., 8B) naturally achieve lower overall training loss, validating the scalability and stability of our framework.

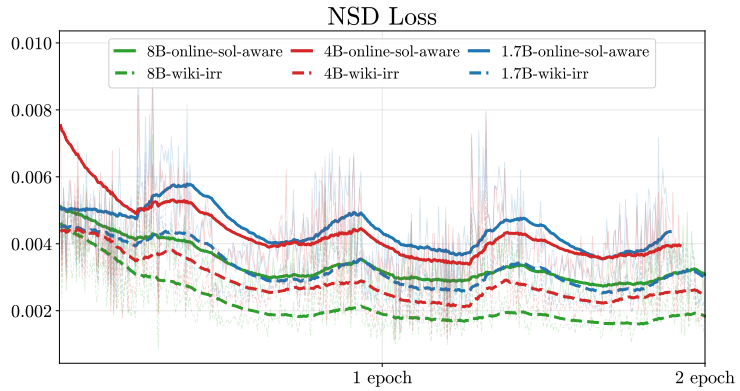

Figure 8: Training loss curves of NSD over two epochs across different model scales. Dashed lines represent the wiki-irr strategy, while solid lines denote the online-sol-aware strategy.

E Additional Experimental Results

We conduct several supplementary evaluations in this section. First, we evaluate our models under the Pass@8 metric in Appendix E.1. Second, we report the performance under thinking mode in Appendix E.2. NSD continues to outperform all baselines under these settings. Third, in Appendix E.3, we investigate an alternative objective formulation that treats the negative of the loss as an advantage signal for policy-gradient optimization, demonstrating that the NSD framework is scalable to policy-gradient-style training paradigms.

E.1 Pass@8 Performance

We report the performance of pass@8 in Table 6.

Table 6: Main evaluation results reported as pass@8 (%): at least one of 8 sampled solutions is correct. Same evaluation setting as Table 1. Δ Avg is the average absolute improvement over the same-size baseline across all 7 benchmarks. **Bold** marks the best result in each model-size group; underline marks the second best. † denotes methods that require ground-truth labels.

Method	AIME 2024	AIME 2025	AIME 2026	HMMT 2025 Feb	AMC 2023	Olympiad-Bench	MATH-500	Δ Avg
1.7B Models								
Qwen3-1.7B	16.7	23.3	13.3	16.7	70.0	57.8	77.6	—
OPSD†	**40.0**	23.3	13.3	16.7	72.5	59.6	77.6	+3.9
Intuitor	30.0	23.3	16.7	13.3	75.0	57.0	76.2	+2.3
TTRL	30.0	26.7	23.3	13.3	77.5	58.7	77.0	+4.4
NSD	33.3	**36.7**	**23.3**	**16.7**	**77.5**	**60.9**	**77.6**	**+7.2**
4B Models								
Qwen3-4B	50.0	40.0	40.0	20.0	95.0	67.0	81.6	—
OPSD†	40.0	46.7	36.7	30.0	92.5	66.4	81.6	+0.0
Intuitor	50.0	53.3	36.7	26.7	95.0	66.1	80.0	+2.0
TTRL	63.3	40.0	46.7	23.3	90.0	64.3	81.6	+2.2
NSD	**60.0**	**63.3**	**53.3**	**30.0**	**95.0**	**68.3**	**82.0**	**+8.3**
8B Models								
Qwen3-8B	56.7	30.0	43.3	23.3	92.5	66.8	82.0	—
OPSD†	46.7	43.3	40.0	23.3	87.5	67.6	81.8	−0.6
Intuitor	60.0	40.0	36.7	26.7	95.0	69.0	81.8	+2.1
TTRL	56.7	33.3	36.7	20.0	92.5	66.8	82.0	−1.0
NSD	**70.0**	**46.7**	**60.0**	**40.0**	**95.0**	**69.0**	**81.8**	**+9.7**

E.2 Performance on Thinking Mode

We evaluate the performance of all methods under the thinking mode, reporting the results of the best-performing checkpoints evaluated under the thinking mode. As shown in Table 7, NSD also outperforms all other baselines overall.

Notably, both OPSD and NSD achieve more improvements on challenging datasets such as AIME and Olympiad Bench, aligning with the observations from the non-thinking setting. On datasets with limited headroom for improvement (*e.g.*, AMC), all methods perform comparably to the base model. Furthermore, we observe that the Intuitor-trained model tends to over-think, causing many responses to exceed the maximum generation length limit (even after extending it to 38k tokens), which leads to a severe degradation in accuracy. This phenomenon is also discussed in the previous works (Zhao et al., 2026b; Ghimire et al., 2026). Interestingly, the wiki-irr strategy enhances the model's reasoning robustness in long-context scenarios, granting it a distinct advantage in thinking mode.

Table 7: Thinking mode evaluation results reported as avg@8 (%). Truncated samples (no `\boxed{}` answer due to token budget exhaustion) are counted as incorrect. Δ Avg is the average absolute improvement over the same-size baseline across all 6 benchmarks. **Bold** marks the best result in each model-size group; underline marks the second best.

Method	AIME 2024	AIME 2025	AIME 2026	HMMT 2025 Feb	AMC 2023	Olympiad-Bench	Δ Avg
1.7B Models							
Qwen3-1.7B	50.0	32.1	39.6	24.2	81.2	49.9	—
OPSD†	44.2	35.0	39.6	25.8	83.8	50.0	+0.2
Intuitor	40.8	29.6	29.6	22.5	74.4	47.0	−5.5
TTRL	41.7	30.0	35.0	27.1	80.3	49.1	−2.3
NSD	51.7	38.3	**42.9**	27.5	**85.3**	50.4	+3.2
NSD-wiki	**55.0**	**41.2**	40.8	**28.8**	85.0	**52.0**	**+4.3**
4B Models							
Qwen3-4B	75.8	69.1	67.5	46.0	97.2	45.9	—
OPSD†	76.2	69.8	67.2	46.2	96.6	46.7	+0.2
Intuitor	52.9	45.8	51.2	39.6	90.9	43.9	−12.9
TTRL	72.5	64.3	65.1	46.0	96.6	44.4	−2.1
NSD	**77.3**	**73.3**	**67.7**	**48.4**	97.8	**57.9**	**+3.5**
NSD-wiki	76.2	71.2	65.8	47.5	**99.4**	57.3	+2.7
8B Models							
Qwen3-8B	75.0	66.2	**67.5**	47.5	**97.2**	58.3	—
OPSD†	70.8	57.5	62.5	43.8	92.2	57.7	−4.5
Intuitor	70.0	61.7	61.2	44.2	94.1	57.2	−3.9
TTRL	75.4	67.1	65.4	46.7	94.4	58.2	−0.7
NSD	75.0	69.2	67.1	48.3	96.9	58.7	+0.6
NSD-wiki	**79.2**	**69.6**	67.1	**51.2**	96.6	**59.3**	**+1.9**

E.3 Alternative Objective: Policy Gradient Optimization

While the NSD loss $\mathcal{L}_{\text{NSD}}^{(t)}$ can be directly backpropagated as a supervised objective, we find it also fits a sampled-token advantage policy-gradient framework, following the spirit of Zhao et al. (2026a) and Lu & Lab (2025). For each token y_t in a student rollout $y \sim \pi_\theta(\cdot \mid x_i)$, we define a token-level advantage as the NSD loss:

$$A_t = -\mathcal{L}_{\text{NSD}}^{(t)} \tag{14}$$

Intuitively, a token with high NSD loss receives a strongly negative advantage, signaling the policy to reduce its probability. Conversely, tokens with low NSD loss receive near-zero or positive advantage, leaving their probabilities unchanged. We treat A_t as a constant with respect to θ and optimize the

student via the standard policy gradient surrogate objective:

$$\mathcal{J}_{\text{PG}}(\theta) = \mathbb{E}_{(x,a)\sim\mathcal{D},y\sim\pi_\theta}\left[\sum_t A_t \log \pi_\theta(y_t \mid x_i, y_{<t})\right] \tag{15}$$

We evaluate the NSD based on the alternative objective under the same setting as our main experiment. The result is shown in Table 8. Compared to models optimized with the $\mathcal{J}$ objective (Eq. 6), NSD trained under $\mathcal{J}_{\text{PG}}$ (Eq. 15) achieves superior performance on the 1.7B model (+5.0% on average). However, on the 4B and 8B models, the $\mathcal{J}$-objective NSD yields better overall results. Notably, the wiki-irr strategy consistently performs best under the policy gradient setting, while the online solution-aware strategy emerges as the second best. This discrepancy arises because the gradients are truncated by the advantage function, decreasing the capture of richer gradient signals. In contrast, wiki-irr utilizes noise to introduce more generalized interference (causing an overall degradation of the model's reasoning capabilities in long contexts), which ultimately makes it a more effective strategy in this regime.

Table 8: Evaluation results of NSD with different negative conditioning strategies on mathematical reasoning benchmarks. The models are trained based on the NSD policy gradient objective in Eq. 15.

Method / Variant	**AIME 2024**	**AIME 2025**	**HMMT Feb 2025**	**MATH-500**	**Δ Avg**
1.7B Models					
NSD (Solution-aware, Offline)	15.8	14.2	5.4	63.2	+2.4
NSD (Solution-aware, Online)	15.8	12.5	7.5	63.3	+2.5
NSD (Wiki-irr, Offline)	**20.0**	<u>15.0</u>	**9.6**	**64.4**	**+5.0**
4B Models					
NSD (Question-only, Offline)	30.4	24.6	16.2	72.7	+4.4
NSD (Solution-aware, Offline)	**33.8**	22.5	14.6	72.1	+4.2
NSD (Solution-aware, Online)	31.3	25.0	16.3	**74.0**	<u>+5.1</u>
NSD (Wiki-irr, Offline)	<u>33.8</u>	**25.4**	**17.1**	<u>73.5</u>	**+5.9**
8B Models					
NSD (Solution-aware, Offline)	30.8	23.3	12.5	73.6	+1.9
NSD (Solution-aware, Online)	**35.0**	21.7	13.8	73.6	+2.8
NSD (Wiki-irr, Offline)	<u>34.2</u>	**28.8**	**19.2**	<u>73.9</u>	**+5.8**

E.4 Case Study

A case study is shown in Table 9. These results indicate that NSD-trained models more readily explore novel and correct solutions that are entirely absent from the outputs of both the base and OPSD-trained models. Furthermore, by prompting an external LLM (Sonnet) to analyze the reasoning traces of each response, we observe that the NSD-trained model engages in several reflection steps, successfully circumventing erroneous trajectories that commonly trap the base model.

F Definition of Task, Style, and Reflection Tokens

Considering the similar task setting with RLCSD (Pan et al., 2026), we follow their definition of both task and style tokens:

(1) empty or whitespace-only → **style**;

(2) matches any of the math regexes (a digit `\d`; an arithmetic operator in $+-=*/<>$ $\times\div\leq\geq\neq$; a LaTeX command `\[A-Za-z]+`; a double backslash; or one of `$ ^ _`) → **task**;

(3) the normalized form is in the math wordlist {`mod, prime, factor, gcd, lcm, log, ln, sin, cos, tan, exp, integral, sqrt, boxed, frac, sum, prod, pi, alpha, beta, gamma, theta, delta, lambda, mu, sigma, infty, leq, geq, neq, cdot, times, div`} → **task**;

(4) pure punctuation or a literal newline token (`\n`, `\\n`) → **style**;

(5) the normalized form is in the discourse wordlist (connectives `therefore, so, thus, hence, then, because, since`; hedges `wait, maybe, perhaps, seems, okay, ok, well, now, first, next, finally, actually, alternatively, however`; scaffolding `step, answer, let, lets`; closed-class function words `is, are, us, we, the, a, an, of, to, for, in, on, by, at, as, and, or, but, if, yes, no, this, that, these, those, it, its, be, been, being, have, has, had, do, does, did, will, would, should, could, can, may`) → **style**;

(6) otherwise → **neutral**.

The following tokens are considered as reflection tokens:

```
wait, actually, hmm, let me reconsider, let me rethink, i made an
error, i made a mistake, that's wrong, that is wrong, incorrect,
reconsider, rethink, re-examine, let me check, let me verify,
double check, double-check, going back, revisit, on second thought
```

Table 9: Case study on AIME 2025 II #12 comparing Qwen3-4B baseline, OPSD, and NSD. The table shows a summary from Sonnet. Numbers denote correct samples out of 8 independent draws. ✓ correct; ✗ incorrect.

Problem	**Baseline**	**OPSD**	**NSD (Ours)**
AIME 2025 II #12 — Geometry. Let $A_1A_2\cdots A_{11}$ be a non-convex simple 11-gon satisfying: (1) $[A_iA_1A_{i+1}]=1$ for $2\le i\le 10$; (2) $\cos(\angle A_iA_1A_{i+1})=\frac{12}{13}$ for $2\le i\le 10$; (3) perimeter $=20$. Express $A_1A_2+A_1A_{11}=\frac{m\sqrt{n}-p}{q}$ (n squarefree, no prime divides all of m,p,q); find $m+n+p+q$.			
Pass@8	0/8 ✗	0/8 ✗	**4/8** ✓
Key reasoning	From $\cos\theta=\frac{12}{13}$ derives $\sin\theta=\frac{5}{13}$, hence $\lvert A_1A_i\rvert\cdot\lvert A_1A_{i+1}\rvert=\frac{26}{5}$. The product constraint gives an alternating sequence $a_2=x$, $a_3=\frac{26}{5x}$, $a_4=x$, … Attempts to use the perimeter but *conflates the sum of radii from A_1 with the polygon perimeter*, obtaining $5x+\frac{26}{x}=20$ which has no clean closed form. After extensive numerical trials, **guesses the symmetric solution** $x=\frac{13}{\sqrt{5}}$ (i.e. $a_2=a_{10}$), giving $A_1A_2+A_1A_{11}=\frac{13}{\sqrt{5}}+2\sqrt{5}=\frac{23\sqrt{5}}{5}$, so $m=23,n=5,p=0,q=5$. **Final: 33** ✗	Same product relation and alternating sequence. Applies Law of Cosines: since $x_ix_{i+1}=\frac{26}{5}$, each inner-polygon side satisfies $d^2=x_i^2+x_{i+1}^2-\frac{48}{5}$, so all 9 inner sides are equal. Correctly writes the perimeter equation $a+9d+\frac{26}{5a}=20$ and sets $S=a+\frac{26}{5a}$. *Instead of solving for S, minimises S via AM–GM*: $\min(a+\frac{26}{5a})=2\sqrt{\frac{26}{5}}=\frac{2\sqrt{130}}{5}$, and incorrectly **treats this minimum as the answer**, concluding $A_1A_2+A_1A_{11}=\frac{2\sqrt{130}}{5}$, so $m=2,n=130,p=0,q=5$. **Final: 137** ✗	Same product relation and alternating sequence. Applies Law of Cosines; all 9 inner sides equal d. Writes the perimeter equation $a+9d+\frac{26}{5a}=20$. Then **attempts a symmetric-guess approach**: tests $x=\sqrt{26/5}$, $x=2$, $x=13/5$, $x=13/\sqrt{5}$ in turn, each time verifying numerically that the two expressions for d^2 do *not* agree. Sets $S=a+\frac{26}{5a}$, so the perimeter equation gives $d=\frac{20-S}{9}$. Rewrites d^2 via Law of Cosines: $d^2=a^2+\frac{676}{25a^2}-\frac{48}{5}=\left(a+\frac{26}{5a}\right)^2-\frac{52}{5}-\frac{48}{5}=S^2-20$. Substituting $d=\frac{20-S}{9}$ yields $\left(\frac{20-S}{9}\right)^2=S^2-20$, which expands to $4S^2+2S-101=0$. Quadratic formula: $S=\frac{-2\pm\sqrt{1620}}{8}=\frac{9\sqrt{5}-1}{4}$ (positive root), so $m=9,n=5,p=1,q=4$. **Final: 19** ✓
Reflection	*No self-correction.* After the perimeter approach yields no clean form, the model **commits to a guess** ($x=\frac{13}{\sqrt{5}}$) without checking whether it satisfies the original constraints, and submits the result directly.	*No self-correction.* The model sets up the correct equation structure but **replaces the constraint with its relaxation**: once the AM–GM bound is computed it is treated as the solution, with no attempt to verify that the minimum is actually attained.	After the guessing strategy fails on multiple candidates ($x=\sqrt{26/5}$, 2, $\frac{13}{5}$, $\frac{13}{\sqrt{5}}$), the model **explicitly abandons the approach** ("*Hmm. Maybe my approach is not working. Alternative idea:*") and **reframes the problem** around the aggregate variable $S=a+\frac{26}{5a}$, turning an intractable system into a single quadratic $4S^2+2S-101=0$.

G LIMITATIONS

NSD relies on the student model's inherent capacity to generate negative conditions. Consequently, this approach may be less effective for extremely small or weak models that struggle to produce meaningful negative contrasts for optimization. However, given the rapid capability scaling of modern foundational models, this capacity bottleneck is expected to diminish naturally in future architectures or in stronger models.

Under the online strategy, NSD requires negative-condition generation and two forward passes through the two same frozen models. Nevertheless, we explored alternative conditioning strategies, including efficient generation-free methods like the wiki-irr strategy, which can mitigate the rollout costs while maintaining competitive performance. Moreover, executing the two forward passes in parallel at each training step effectively minimizes overall wall-clock latency.