CAN LANGUAGE MODELS LEARN TO REJECT THEIR OWN BAD REASONING STEPS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Verifier-guided decoding can prevent harmful reasoning steps from contaminating subsequent generation, but typically relies on an external learned verifier. Can a language model instead learn to reject its *own* bad reasoning steps? This requires distinguishing candidate steps sampled from the same problem and reasoning prefix. We define a prefix's *recoverability* as the probability that the frozen generator can complete it correctly. Diagnostic measurements show that most adjacent changes in recoverability are difficult to resolve with practical Monte Carlo budgets, even when cumulative degradation is detectable. Meanwhile, candidates sampled from a shared prefix exhibit a sparse low-recoverability tail, suggesting an opportunity for selective rejection. We introduce SELF-STEP REJECTION (SSR), which trains a lightweight LoRA acceptance gate on the generator's own backbone while keeping the base model frozen. SSR uses confidence-qualified *first-passage supervision*: steps before the first resolved crossing of a root-relative recoverability barrier are labeled as accepted, the crossing step is labeled as rejected, and unresolved steps and the subsequent suffix are excluded from supervision. Training combines pointwise classification with same-prefix pairwise learning, followed by group-relative policy refinement using final-answer correctness. At inference, the gate accepts candidates or resamples from the unchanged prefix, subject to a per-position attempt cap and an episode-level rejection budget, without an external learned verifier. Across three reasoning models and five mathematical reasoning benchmarks, SSR improves macro-average accuracy over single-pass decoding by 5.4–10.1 percentage points using 1.21–1.40× as many generated tokens, achieving the highest macro-average accuracy among the evaluated step-level methods for each generator, while full-solution scaling methods require 4.47–8.27× the single-pass token cost for comparable performance.

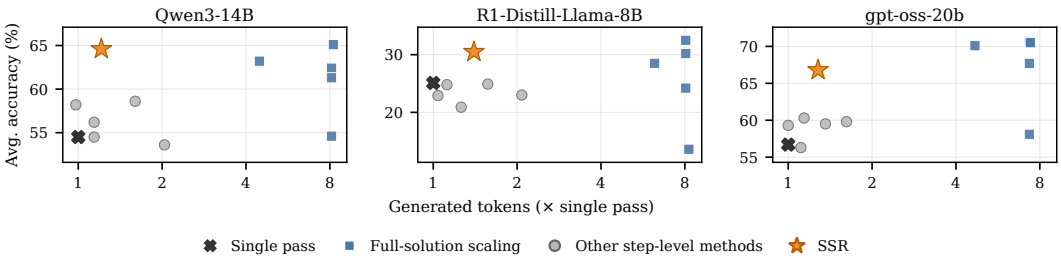

Figure 1: Macro-average accuracy across five benchmarks versus generated-token cost relative to single-pass decoding. Full-solution scaling includes Majority@8, ESC, ORM reranking, PRM reranking, and USC; other step-level methods include fixed-early resampling, low-confidence resampling, budget forcing, random rejection, and LATTS. SSR achieves the highest accuracy among step-level methods for all three generators and remains competitive with full-solution scaling at substantially lower token cost. See Table 1 for detailed results.

1 INTRODUCTION

Reasoning language models benefit from additional inference-time computation, but longer reasoning traces also create more opportunities to commit harmful intermediate steps (Wei et al., 2022; Guo

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.

et al., 2025; Muennighoff et al., 2025). Once a step is appended, subsequent generation is conditioned on it. A model may recover through reflection, but it may instead elaborate a plausible yet incorrect premise, reducing the chance of eventually reaching the correct answer.

A natural response is to verify each step before committing it. Process reward models score intermediate reasoning (Lightman et al., 2024; Wang et al., 2024), and verifier-guided decoders can reject low-scoring steps and sample alternatives (Uscidda et al., 2025). These approaches establish the utility of step-level intervention when a capable verifier is available. However, an external verifier may introduce additional model capacity or supervision from stronger teachers, leaving open whether the generator itself can learn to make these decisions. We therefore ask: *Given a reasoning prefix and one newly sampled step, can a language model decide whether to commit that step or reject it and sample again?*

We refer to this task as *same-prefix candidate rejection*, which requires distinguishing candidates sampled from the same problem and committed prefix. Global verification metrics may reward recognition of problem difficulty, trajectory depth, or accumulated errors without measuring this ability. We therefore train and evaluate candidate discrimination within shared prefixes.

A natural supervision signal is the change in *recoverability*: the probability that the frozen generator, continued from a prefix, eventually produces a correct answer. Yet adjacent changes in recoverability are difficult to estimate reliably because they involve differences between noisy Monte Carlo success rates. Across three reasoning models and five mathematical reasoning benchmarks, only 2.3–24.0% of adjacent recoverability differences are resolved with 16 rollouts per endpoint. Cumulative degradation can nevertheless become detectable even when individual transitions remain unresolved. This motivates supervision based on cumulative degradation relative to the problem's initial recoverability.

Cumulative degradation alone does not establish that resampling is useful. We therefore also examine candidates sampled from shared prefixes. These candidates exhibit variation beyond Monte Carlo noise, with a small low-recoverability tail. This suggests a selective intervention: accept most candidates immediately and spend additional computation on replacing the few that are likely to harm the continuation.

Our method, SELF-STEP REJECTION (SSR), freezes the generator and trains a lightweight LoRA (Hu et al., 2021) acceptance gate on the same backbone. It uses confidence-qualified *first-passage supervision* relative to the problem's initial recoverability: confidently acceptable steps before the first resolved crossing are labeled as accepted, the crossing step is labeled as rejected, and unresolved steps and the subsequent suffix are excluded from supervision. Training combines pointwise classification with same-prefix pairwise learning, directly encouraging the gate to distinguish accepted and rejected candidates that share the same state. We then refine the gate with a within-problem group-relative objective using final-answer correctness.

At inference, the adapter is enabled only for acceptance scoring. The decoder commits accepted candidates and resamples rejected ones from the unchanged prefix, subject to a per-position attempt cap and an episode-level rejection budget. Additional sampling is therefore selective and requires no external learned verifier.

Across three reasoning models and five mathematical reasoning benchmarks, SSR improves macro-average accuracy over single-pass decoding by 5.4–10.1 percentage points using 1.21–1.40$\times$ as many generated tokens. It achieves the highest macro-average accuracy among the evaluated step-level methods for each generator and remains competitive with full-solution scaling methods, which use 4.47–8.27$\times$ the single-pass token cost. Figure 1 summarizes this accuracy–cost trade-off.

Our contributions are:

- We formulate *same-prefix candidate rejection* and characterize its learning challenges: most adjacent recoverability changes are difficult to resolve at practical rollout budgets, while cumulative degradation is detectable and same-prefix candidates exhibit a sparse low-recoverability tail.
- We introduce SSR, combining confidence-qualified first-passage supervision, same-prefix pairwise learning, and group-relative policy refinement to enable budgeted step rejection using the generator's own backbone.
- Through extensive experiments, we demonstrate SSR's favorable accuracy–token trade-off against both step-level and full-solution scaling baselines.

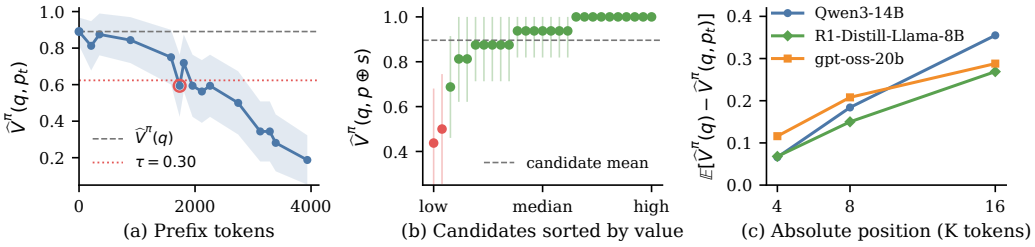

Figure 2: Why same-prefix candidate rejection is a distinct problem. (a) Recoverability along one trajectory evaluated at paragraph boundaries; the open circle marks the first crossing of a root-relative $\tau = 0.30$ barrier. (b) Candidate steps sampled from one shared prefix exhibit a small low-value tail. (c) Mean root-relative degradation increases with depth across the recoverability audit populations.

2 Same-Prefix Candidate Rejection for Reasoning Models

2.1 Problem Definition

Let q be a problem, π a frozen reasoning policy, and $p_t = (s_1, \ldots, s_t)$ a committed reasoning prefix. At decision t, the policy samples $s_t \sim \pi(\cdot \mid q, p_{t-1})$, and a gate either commits the candidate or rejects it and resamples from the same conditional distribution. The gate therefore receives $x_t = (q, p_{t-1}, s_t)$.

For any prefix p, define its recoverability under the frozen policy as

$$V^{\pi}(q, p) = \Pr_{y \sim \pi(\cdot|q,p)}[R(q, y) = 1], \tag{1}$$

where R is a binary final-answer checker. We write $V^{\pi}(q) = V^{\pi}(q, \varnothing)$ for the recoverability of the bare problem and use $\widehat{V}^{\pi}$ for Monte Carlo estimates. For a candidate step s, $p \oplus s$ denotes the extended prefix, so $V^{\pi}(q, p \oplus s)$ is the probability of eventual success after committing s and continuing under π.

This same-prefix comparison distinguishes candidate rejection from global verification. A verifier that separates easy problems from hard ones, or scores only properties accumulated across a trajectory, need not distinguish candidates available at the current state. Useful rejection instead requires

$$\mathbb{E}[V^{\pi}(q, p \oplus s) \mid s \text{ accepted}] > \mathbb{E}_{s \sim \pi(\cdot|q,p)}[V^{\pi}(q, p \oplus s)]. \tag{2}$$

We therefore train and evaluate the gate within a shared prefix.

2.2 Why Adjacent Recoverability Changes Are Difficult to Estimate

We estimate recoverability from independent continuations. A natural step-level target is the adjacent change $\Delta_t = V^{\pi}(q, p_t) - V^{\pi}(q, p_{t-1})$, but it is the difference between two noisy Bernoulli means.

Figure 2(a) shows one Qwen3-14B trajectory evaluated at paragraph boundaries. Estimated recoverability falls from 0.891 to 0.188, with temporary increases and no statistically resolved adjacent transition (Figure 3(a)). The mean adjacent change is approximately -0.05; a standard power calculation requires roughly 1,600 rollouts per endpoint to resolve an effect of this size with 80% power (Figure 3(b)). Appendix D.1 provides the power calculation.

Although individual changes are weak, their cumulative effect is measurable. Define the root-relative degradation

$$D_t = V^{\pi}(q) - V^{\pi}(q, p_t). \tag{3}$$

The trajectory in Figure 2(a) reaches $D_t = 0.703$, and its first crossing of a fixed root-relative barrier is clearly observable. Figure 2(c) shows the same increasing degradation with depth across the broader audit populations. The root serves as a difficulty-normalized reference rather than an assumed maximum. These observations motivate supervising the first measurable cumulative crossing rather than regressing noisy adjacent value changes.

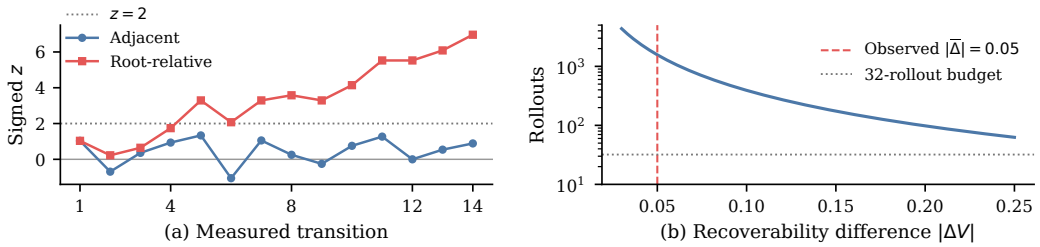

Figure 3: Cumulative degradation is statistically visible when local degradation is not. (a) Signed likelihood-ratio statistics for adjacent and root-relative changes along the measured trajectory. Positive values indicate a loss of recoverability; the dotted line at $z = 2$ provides an approximate significance reference. (b) Approximate rollouts per endpoint required to distinguish two Bernoulli success probabilities with 80% power.

2.3 Candidate Values Exhibit a Rare Bad Tail

Trajectory-level degradation alone does not imply that resampling from a fixed prefix is useful. We therefore sample candidate openings from a shared prefix and estimate their recoverability using independent continuations. Figure 2(b) reports 27 usable candidates, each evaluated with 16 continuations. Their variation exceeds the estimated Monte Carlo noise, indicating within-prefix differences. Most candidates retain high recoverability, with a median value of 0.938 and only two candidates at or below 0.5. This sparse low-value tail motivates selectively rejecting candidates while accepting most candidates without additional sampling. Together with measurable cumulative degradation, this supports training the acceptance gate with root-relative first-passage labels.

3 Self-Step Rejection

Self-Step Rejection (SSR) trains a same-model acceptance gate through first-passage supervision, same-prefix pairwise learning, and group-relative policy refinement, then deploys it for budgeted accept-or-resample decoding. Figure 4 illustrates the sparse inference-time intervention and the same-model generate–judge loop.

3.1 First-Passage Supervision

For each training problem, we sample one or more trajectories from the frozen generator π one step at a time, with each step containing at most L_{step} newly generated tokens. Let $\widehat{V}_0 = \widehat{V}^{\pi}(q)$ be the root estimate and $\widehat{V}_t = \widehat{V}^{\pi}(q, p_t)$ the estimate after committing step t. Ignoring estimation uncertainty, the first root-relative crossing for a degradation threshold τ is

$$t^{\star} = \min\left\{t : \widehat{V}_t \le \widehat{V}_0(1-\tau)\right\}. \tag{4}$$

If no step crosses the barrier, $t^{\star} = \infty$.

In practice, labeling stops at the first confidence-resolved crossing, which may occur later than the point-estimate crossing $t^{\star}$. Confidence intervals account for Monte Carlo uncertainty in the difference $\widehat{V}_t - (1-\tau)\widehat{V}_0$. Preceding steps confidently above the barrier are labeled ACCEPT, and the crossing step is labeled REJECT. The subsequent suffix remains unlabeled because, after the first detected barrier crossing, low recoverability may be inherited rather than caused by each subsequent step. Additional rollouts are allocated adaptively near the boundary; steps that remain unresolved are excluded from supervision. Appendix A gives the complete labeling procedure and edge cases.

3.2 Policy-Matched Acceptance Scoring

For a candidate step s_t, we form the gate input $x_t = (q, p_{t-1}, s_t)$. For scoring, we serialize x_t, append a fixed reserved marker, and run the same backbone with a LoRA adapter (Hu et al., 2021) Δ_ϕ enabled. Following token-based verifier readouts (Wang et al., 2024), let $\ell^{+}_{\phi}(x_t)$ and $\ell^{-}_{\phi}(x_t)$

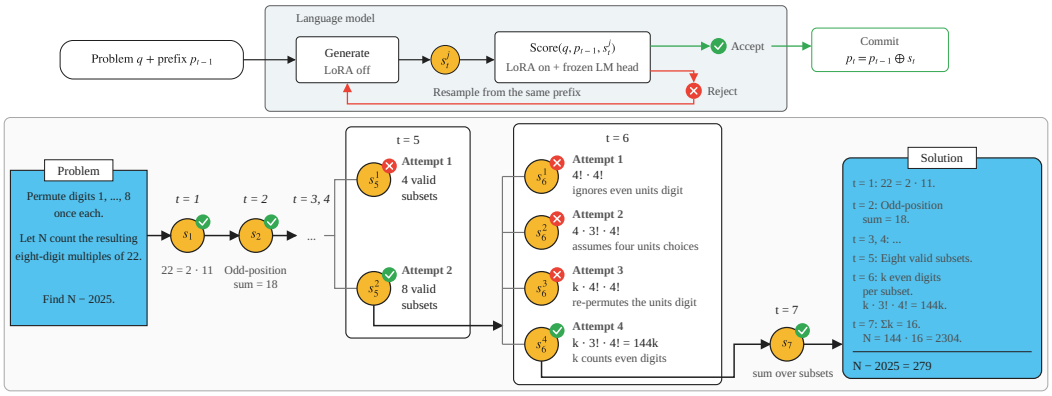

Figure 4: Top: SSR uses the frozen model to generate candidates and enables LoRA to score them. Acceptance extends the prefix; rejection triggers resampling from the unchanged prefix. Bottom: an actual decoding example with at most L_{step} new tokens per step, shown as reasoning summaries. Attempts at each position proceed from top to bottom. Resampling is bounded by a per-position attempt cap M and an episode-level rejection budget B.

denote the frozen language-modeling head's logits at the marker position for the existing vocabulary tokens + and −, respectively. We define the acceptance logit and normalized acceptance score as

$$z_\phi(x_t) = \ell_\phi^+(x_t) - \ell_\phi^-(x_t), \qquad \text{score}_\phi(x_t) = \frac{\exp \ell_\phi^+(x_t)}{\exp \ell_\phi^+(x_t) + \exp \ell_\phi^-(x_t)} = \sigma(z_\phi(x_t)) \,. \tag{5}$$

The score represents acceptance confidence rather than an estimate of recoverability. The base parameters θ, marker embedding, and language-modeling head remain frozen; only the LoRA adapter is trained. The adapter is disabled during candidate generation and enabled only for acceptance scoring.

3.3 Same-Prefix Supervised Objective

Let $\mathcal{D}$ contain the confidence-qualified examples, with $y_t = 1$ for ACCEPT and $y_t = 0$ for REJECT. The pointwise objective is weighted binary cross entropy:

$$\mathcal{L}_{\text{BCE}} = - \sum_{(x_t, y_t) \in \mathcal{D}} \omega_{y_t} \left[y_t \log \text{score}_\phi(x_t) + (1 - y_t) \log(1 - \text{score}_\phi(x_t)) \right] , \tag{6}$$

where ω_{y_t} is a class-dependent optimization weight. In implementation, this objective is evaluated with binary cross entropy on the logit difference $z_\phi(x_t)$ for numerical stability.

Pointwise accuracy can be high while a gate learns only problem difficulty or trajectory depth. We therefore enrich states near the crossing with multiple same-prefix candidates $s_t^1, \ldots, s_t^M$ sampled from the same prefix. Let $\mathcal{P}_t$ contain the confidence-qualified pairs (i, j) from this group, where candidate i is accepted and candidate j is rejected. Following RankNet (Burges et al., 2005), we add the pairwise objective

$$\mathcal{L}_{\text{pair}} = \sum_{(i,j) \in \mathcal{P}_t} \log \left(1 + \exp[-(z_i - z_j)]\right) , \qquad z_i = z_\phi(x_t^i). \tag{7}$$

The complete supervised objective is

$$\mathcal{L}_{\text{gate}} = \mathcal{L}_{\text{BCE}} + \lambda_{\text{pair}} \mathcal{L}_{\text{pair}}. \tag{8}$$

This objective directly rewards within-prefix ordering, reducing reliance on problem difficulty or trajectory depth alone.

3.4 Group-Relative Policy Refinement

We refine the supervised gate to improve final-answer accuracy under budgeted sequential decoding using a REINFORCE-style objective (Williams, 1992) with a group-relative baseline (Shao et al.,

2024). We optimize the gate as a stochastic binary policy $\rho_\phi(u_t \mid x_t)$ over $u_t \in \{\text{ACCEPT}, \text{REJECT}\}$:

$$\rho_\phi(\text{ACCEPT} \mid x_t) = \sigma\left(\frac{z_\phi(x_t) - \text{logit}(\eta)}{\text{temperature}}\right), \qquad \rho_\phi(\text{REJECT} \mid x_t) = 1 - \rho_\phi(\text{ACCEPT} \mid x_t), \quad (9)$$

where temperature controls action sampling and η is fixed during policy refinement and deployment. Training samples actions from ρ_ϕ; inference uses the deterministic threshold rule $\text{score}_\phi(x_t) \geq \eta$.

For each problem, we sample a group of G complete decoding episodes. Let $R_e \in \{0, 1\}$ denote final-answer correctness for episode e and define the within-problem advantage

$$A_e = R_e - \frac{1}{G}\sum_{e'=1}^{G} R_{e'}. \quad (10)$$

The group baseline compares each episode's outcome with other episodes of the same problem. With $\mathcal{T}_e$ denoting the gate decisions in episode e, we optimize

$$\mathcal{L}_{\text{RL}} = -\frac{1}{\sum_e |\mathcal{T}_e|}\sum_{e=1}^{G} A_e \sum_{t \in \mathcal{T}_e} \log \rho_\phi(u_{e,t} \mid x_{e,t}). \quad (11)$$

The environment enforces a per-position attempt cap and an episode-level rejection budget, bounding generation cost while the terminal reward optimizes correctness.

3.5 Sequential Accept-or-Resample Decoding

At inference time, the base model samples one candidate at a time. The acceptance gate commits the candidate when $\text{score}_\phi(x_t) \geq \eta$. Each position receives at most M candidate attempts. Rejected candidates are discarded and resampled from the unchanged parent prefix until either a candidate is accepted or the attempt cap is reached; in the latter case, the decoder commits the highest-scoring attempt. Across the episode, at most B candidates may be rejected. Once this budget is exhausted, subsequent candidates are committed directly. SSR generates additional candidates only after a rejection, allocating extra computation selectively within the prescribed budgets. Step-level best-of-K, in comparison, incurs the cost of sampling K candidates at every decision.

4 Experiments

4.1 Setup

We study Qwen3-14B (Yang et al., 2025), DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025), and gpt-oss-20b (Agarwal et al., 2025) as frozen reasoning generators. Our primary evaluation suite contains contest-mathematics problems from OlymMATH-Easy and OlymMATH-Hard (Sun et al., 2026), together with AIME 2025, HMMT February 2025, and HMMT February 2026 from MathArena (Dekoninck et al., 2026). We train the acceptance gate on a filtered subset of Omni-MATH (Gao et al., 2025), decontaminated against all evaluation benchmarks, and reserve a separate subset for development.

We compare SSR with the following baselines. Single pass is the no-scaling reference, while Majority@8 (Wang et al., 2022) and ESC ($w = 2$) (Li et al., 2024) represent fixed-budget and adaptive self-consistency. ORM reranking (Liu et al., 2026), PRM reranking (Zhang et al., 2025), and USC (Chen et al., 2023) select among eight complete solutions. We design fixed-early resampling, low-confidence resampling, and random rejection as controls for intervention timing and candidate selection. Budget forcing (Muennighoff et al., 2025) tests the effect of extending generation. LATTS (Uscidda et al., 2025) provides an external-verifier step-rejection baseline. Fixed early and random rejection use the same generation-budget constraints as SSR.

4.2 Implementation Details

Unless otherwise specified, all generators use a 32K context window and a per-step limit of $L_{\text{step}} = 2048$ new tokens, reduced when context space is insufficient. Each generator has a separate rank-16

Table 1: End-to-end accuracy (%) and generation cost, averaged over three evaluation runs. "Avg." is the macro-average and "Tok.×" is relative to single-pass generation. Gray rows are step-level methods; underlining and bold mark the best full-solution-scaling and step-level accuracy, respectively.

Model	Method	A25	Olym-E	Olym-H	HMMT25	HMMT26	Avg.	Tok.×
Qwen3-14B	Single pass	70.0	83.0	21.0	50.0	48.5	54.5	1.00
	Majority@8	76.7	94.0	33.0	56.7	51.5	62.4	8.11
	ESC ($w = 2$)	77.8	94.7	34.3	56.7	52.5	63.2	4.47
	ORM reranking	76.7	91.0	28.0	53.3	57.6	61.3	8.11
	PRM reranking	70.0	85.0	23.0	46.7	48.5	54.6	8.11
	USC	80.0	93.0	35.0	60.0	57.6	65.1	8.24
	Fixed early	73.3	90.0	28.0	53.3	48.5	58.6	1.60
	Low-conf. resample	73.3	87.0	22.0	**63.3**	45.5	58.2	0.98
	Budget forcing	66.7	81.0	22.0	50.0	48.5	53.6	2.04
	Random reject	66.7	86.7	23.3	53.3	42.4	54.5	1.14
	LATTS	70.0	80.0	33.3	43.3	**54.5**	56.2	1.14
	SSR	**77.8**	**95.7**	**35.0**	62.2	52.5	**64.6**	1.21
R1-Distill-Llama-8B	Single pass	33.3	36.0	5.0	30.0	21.2	25.1	1.00
	Majority@8	40.0	48.0	6.0	26.7	30.3	30.2	8.06
	ESC ($w = 2$)	35.6	44.0	7.0	25.6	30.3	28.5	6.23
	ORM reranking	40.0	46.0	10.0	30.0	36.4	32.5	8.06
	PRM reranking	30.0	37.0	7.0	16.7	30.3	24.2	8.06
	USC	10.0	31.0	5.0	10.0	12.1	13.6	8.27
	Fixed early	**40.0**	35.0	5.0	23.3	21.2	24.9	1.57
	Low-conf. resample	26.7	36.7	3.3	26.7	21.2	22.9	1.04
	Budget forcing	33.3	33.3	3.3	26.7	18.2	23.0	2.08
	Random reject	20.0	40.0	6.7	16.7	21.2	20.9	1.26
	LATTS	30.0	33.3	**16.7**	20.0	24.2	24.8	1.12
	SSR	37.8	**42.7**	10.3	**32.2**	**29.3**	**30.5**	1.40
gpt-oss-20b	Single pass	70.0	79.0	27.0	50.0	57.6	56.7	1.00
	Majority@8	86.7	93.0	37.0	66.7	69.7	70.6	7.34
	ESC ($w = 2$)	86.7	91.3	34.0	67.8	70.7	70.1	4.68
	ORM reranking	73.3	87.0	42.0	66.7	69.7	67.7	7.34
	PRM reranking	70.0	76.0	30.0	56.7	57.6	58.1	7.34
	USC	86.7	89.0	37.0	70.0	69.7	70.5	7.40
	Fixed early	76.7	80.0	34.0	56.7	51.5	59.8	1.62
	Low-conf. resample	76.7	81.0	28.0	53.3	57.6	59.3	1.00
	Budget forcing	73.3	80.0	33.0	56.7	54.5	59.5	1.36
	Random reject	**80.0**	80.0	33.3	56.7	51.5	60.3	1.14
	LATTS	66.7	70.0	33.3	**60.0**	51.5	56.3	1.11
	SSR	78.9	**81.7**	**45.7**	58.9	**68.7**	**66.8**	1.28

Table 2: Recoverability diagnostics across five benchmarks. $\mathbb{E}[|\Delta_t|]$: mean absolute adjacent change; resolved@16: fraction resolved with 16 rollouts per endpoint; Cand. SD: noise-corrected within-prefix candidate standard deviation; Tail: fraction significantly below the best candidate. Brackets indicate 95% problem-bootstrap confidence intervals.

Model	$\mathbb{E}[\|\Delta_t\|]$	Adj. resolved@16 (%)	Cand. SD	Tail (%)
Qwen3-14B	0.215 [0.175, 0.253]	24.0 [19.2, 28.2]	0.113 [0.027, 0.176]	6.0 [2.0, 12.0]
R1-Distill-Llama-8B	0.051 [0.014, 0.097]	2.3 [0.0, 9.2]	0.120 [0.058, 0.168]	11.0 [3.0, 21.0]
gpt-oss-20b	0.140 [0.071, 0.227]	18.7 [2.9, 47.2]	0.184 [0.067, 0.265]	11.5 [3.1, 21.9]

LoRA gate on the query, output, and MLP projections, with key and value projections unchanged. The adapter is enabled only for scoring, using the frozen language-modeling head's +/− logits at a fixed marker. Training combines first-passage supervision with group-relative policy refinement. Inference allows $M = 8$ candidate attempts per position and $B = 12$ rejections per episode. A threshold η, calibrated before refinement to a target rejection rate of 17.8%, remains fixed during refinement and inference. Appendix C provides decoding, optimization, calibration, and runtime details.

4.3 Main Results

Table 1 shows that SSR improves macro-average accuracy over single-pass decoding by 5.4–10.1 percentage points using 1.21–1.40× as many generated tokens, with gains across all 15 model–benchmark pairs. The learned gate distinguishes same-prefix candidates to reject low-value continuations before committing them. SSR's advantage over random rejection and LATTS at comparable token costs suggests that candidate discrimination matters beyond resampling alone, consistent with

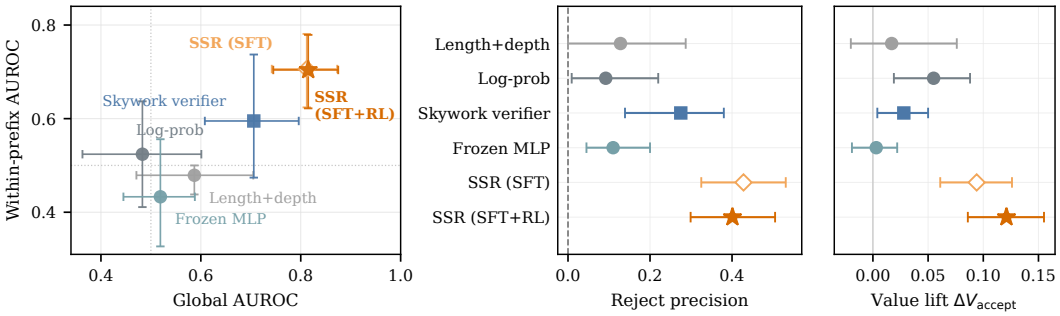

Figure 5: Left: verifier discrimination measured by global AUROC over pooled candidates and within-prefix AUROC over candidates sampled from the same prefix; error bars are 95% confidence intervals. Center: reject precision at a fixed rejection rate of 17.8%. Right: accepted-candidate value lift. Qwen3-14B is the generator throughout. Skywork verifier denotes Skywork-Reward-V2-Qwen3-8B (Liu et al., 2026).

Table 3: Training-component ablations on Qwen3-14B, averaged over three evaluation runs. Indented rows remove or replace one component of the full SSR recipe. "Avg." is the five-benchmark macro-average and "Tok.×" is relative to single-pass generation.

Method	A25	Olym-E	Olym-H	HMMT25	HMMT26	Avg.	Tok.×
Single pass	70.0	83.0	21.0	50.0	48.5	54.5	1.00
SSR (full)	**77.8**	**95.7**	**35.0**	**62.2**	**52.5**	**64.6**	1.21
Label all below-barrier steps	72.2	86.0	24.3	53.3	49.5	57.1	1.62
No same-prefix pairwise loss	73.3	88.0	26.3	54.4	50.5	58.5	1.14
No policy refinement	74.4	89.7	28.7	56.7	50.5	60.0	1.12

the diagnostics and ablations below. By resampling only after rejection and preserving the accepted prefix, SSR concentrates extra generation on local alternatives. This selective allocation helps explain its strong accuracy–cost trade-off: SSR achieves the highest step-level macro-average accuracy for all three generators and remains competitive with full-solution scaling. The strongest full-solution baseline for each generator achieves 0.5–3.8 percentage points higher accuracy but uses 7.34–8.24× the single-pass token cost. Figure 1 summarizes this trade-off.

4.4 Analysis

Problem characterization. Table 2 shows that only 2.3–24.0% of adjacent differences are resolved with 16 rollouts per endpoint, despite nonzero within-prefix variation across all three models. This suggests that noisy local estimates can obscure useful candidate differences, motivating supervision based on cumulative degradation. Meanwhile, only 6.0–11.5% of candidates occupy the low-value tail, supporting selective rejection while preserving most of the trajectory.

Same-prefix gate analysis. Figure 5 shows that strong global discrimination does not necessarily translate into effective within-prefix selection, where problem difficulty and prefix depth cannot distinguish candidates. SSR's stronger local discrimination is consistent with its pairwise training objective. Policy refinement raises accepted-candidate value lift from 0.094 to 0.121 with modest changes in discrimination metrics, suggesting improvements in acceptance decisions beyond offline ranking.

Training-component ablations. Table 3 shows that labeling all below-barrier steps reduces accuracy by 7.5 points and increases token cost, consistent with unnecessary rejection of steps that inherit earlier degradation. Removing pairwise learning costs 6.1 points, supporting direct supervision of local candidate preferences. Omitting policy refinement costs 4.6 points, suggesting that candidate discrimination benefits from further optimization for final-answer correctness under sequential decoding.

Sensitivity analysis. On Qwen3-14B (Appendix D, Tables 6 and 7), accuracy plateaus at target rejection rates of 17.8–30.0% while token cost rises, indicating diminishing returns from additional rejection. Step sizes of 1,024–4,096 tokens achieve similar accuracy, whereas shorter and longer steps perform worse. Short steps increase gate decisions, while long steps delay intervention, suggesting a balance between decision frequency and intervention timing. The default $L_{\text{step}} = 2048$ uses the fewest tokens within this accuracy plateau.

5 Related Work

Inference-time scaling and search. Sampling multiple complete solutions and aggregating their answers improves reasoning robustness (Wang et al., 2022; Cobbe et al., 2021). Tree-structured methods instead branch at intermediate states and use search or learned values to allocate computation (Yao et al., 2023; Luo et al., 2024). Adaptive methods control different aspects of this computation: ESC adjusts the number of sampled solutions through early stopping (Li et al., 2024), while s1 uses budget forcing to control the length of a single reasoning trace (Muennighoff et al., 2025). The effectiveness of scaling strategies also depends on problem difficulty, motivating compute allocation tailored to individual problems (Snell et al., 2024). We study allocation within a single decoding trajectory: the gate decides which candidates warrant replacement before commitment, concentrating extra generation on selected positions while preserving the accepted prefix.

Process supervision and verifier-guided decoding. Process reward models assign scores to intermediate reasoning steps (Lightman et al., 2024; Wang et al., 2024). Rollout-based supervision estimates successful completion from a prefix, reducing manual annotation while depending on the rollout policy and sample count. OmegaPRM lowers labeling cost by searching for the first error under an approximate monotonicity assumption (Luo et al., 2024), while Process Advantage Verifiers assess action-induced progress from a shared state (Setlur et al., 2025). For inference-time selection, Tree-PLV learns from same-prefix rollout preferences to guide tree search (He et al., 2024), whereas LATTS uses an external PRM or LLM critic to reject steps and trigger resampling (Uscidda et al., 2025). Our measurements motivate first-passage supervision when adjacent value changes are difficult to resolve but cumulative degradation is detectable. Building on verifier-guided decoding, we focus on learning a policy-matched acceptance gate from first-passage supervision and evaluating its ability to distinguish same-prefix candidates.

Self-evaluation and self-verification. Model representations can encode information useful for verification that ordinary generation does not express (Burns et al., 2022). Self-Refine uses the same model to generate feedback and iteratively revise its output without additional training (Madaan et al., 2023), although intrinsic self-correction without external feedback can fail to improve reasoning (Huang et al., 2024). Training-based approaches include SCoRe, which learns answer revision through multi-turn reinforcement learning (Kumar et al., 2025), and S^2R, which trains explicit self-verification and correction behaviors (Ma et al., 2025). These works share the goal of self-evaluation through generated feedback or revisions. Self-Step Rejection instead formulates same-prefix candidate rejection: a learned gate evaluates a candidate before commitment and, after rejection, resamples from the unchanged prefix. We introduce first-passage supervision for this acceptance gate, trained as a lightweight adapter while keeping the generator frozen.

6 Conclusion

We presented Self-Step Rejection, which trains a lightweight acceptance gate on the generator's own backbone to reject low-value candidate steps without an external learned verifier. The method combines root-relative first-passage supervision, same-prefix pairwise learning, and policy refinement while keeping the generator frozen. Across three reasoning models and five mathematics benchmarks, SSR improves macro-average accuracy over single-pass decoding by 5.4–10.1 percentage points using 1.21–1.40$\times$ as many generated tokens, achieving the highest macro-average accuracy among evaluated step-level methods for each generator. These results demonstrate the value of learning same-prefix candidate discrimination to guide selective resampling and improve the accuracy–token trade-off.

AI USE STATEMENT

We used generative AI tools to assist with writing and language editing during manuscript preparation. The authors reviewed and revised the AI-assisted text and take full responsibility for the final manuscript.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pp. 89–96, 2005.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023.

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

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. *arXiv preprint arXiv:2605.00674*, 2026.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Zhengyang Tang, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 100540–100569, 2025.

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.

Mingqian He, Yongliang Shen, Wenqi Zhang, Zeqi Tan, and Weiming Lu. Advancing process verification for large language models via tree-based preference learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2086–2099, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International conference on learning representations*, volume 2024, pp. 32808–32824, 2024.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 54523–54549, 2025.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *International Conference on Learning Representations*, volume 2024, pp. 14751–14768, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, et al. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. In *International Conference on Learning Representations*, volume 2026, pp. 133805–133838, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. S2r: Teaching llms to self-verify and self-correct via reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22632–22654, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 60808–60838, 2025.

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

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

Haoxiang Sun, Yingqian Min, Zhipeng Chen, Wayne Xin Zhao, and Ji-Rong Wen. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17438–17457, 2026.

Theo Uscidda, Matthew Trager, Michael Kleinman, Aditya Chattopadhyay, Wei Xia, and Stefano Soatto. Latts: Locally adaptive test-time scaling. *arXiv preprint arXiv:2509.20368*, 2025.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

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.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025.

A SELF-STEP REJECTION

A.1 FIRST-PASSAGE LABELING

First-passage labeling converts trajectory-level value estimates into step-level accept/reject supervision. For a trajectory $p_0, \ldots, p_T$, let $\widehat{V}_{\text{barrier}} = (1-\tau)\widehat{V}_0$ denote the root-relative barrier and define $\text{gap}_t = \widehat{V}_{\text{barrier}} - \widehat{V}_t$. The first confidence-resolved step with $\text{gap}_t > 0$ is labeled REJECT; all preceding confidence-resolved steps are labeled ACCEPT. The procedure then terminates and assigns no labels to the remaining suffix. This restriction associates the negative label with the first confidence-resolved crossing of the root-relative barrier, avoiding repeated attribution of inherited damage to later steps. We evaluate dense labeling of every below-barrier step as an ablation.

At each prefix, the rollout count is increased only while the confidence interval for gap_t intersects zero. An interval lying strictly below zero yields an ACCEPT label, whereas an interval lying strictly above zero yields the first REJECT label and terminates the scan. Prefixes that remain unresolved at $K_{\max}$ are marked gray and excluded from the supervised objective. This abstention rule avoids converting Monte Carlo ties into arbitrary labels.

If $\widehat{V}_0 = 0$, the relative barrier is also zero and contains no information about τ; Algorithm 1 therefore returns no labels. We likewise exclude candidate sets for which every evaluated continuation has zero value, because such sets contain no within-prefix signal for selecting one step over another. Labels are taken from the confidence-qualified decisions recorded during data collection rather than recomputed from the stored value estimates. Adaptive stopping changes the conditional distribution of the stored estimates, but not the recorded decision or committed candidate. The resulting first-passage data are positive-heavy, so we report an always-accept gate as the explicit no-intervention baseline.

The relative barrier scales the tolerated degradation with each problem's recoverability. A fixed absolute margin cannot trigger when the root estimate is below that margin. The root estimate is shared by all trajectories of a problem and can be computed once.

A.2 GROUP-RELATIVE POLICY REFINEMENT

Policy refinement samples G complete decoding episodes for each problem and uses final-answer correctness as the terminal reward. Gate actions are sampled according to Equation 9. The within-problem group mean provides a parameter-free baseline that reduces the influence of problem difficulty without requiring a learned value function. Rewards are terminal and returns are undiscounted, so all decisions in an episode share the group-centered advantage in Equation 10.

We center rewards by the group mean without standardizing by the group standard deviation. For binary rewards, the standard deviation is fully determined by the empirical success rate, $\sqrt{\widehat{\mu}(1-\widehat{\mu})}$. With $G = 8$, standardization would give a group with $\widehat{\mu} = 1/8$ approximately $1.5\times$ the group-level scaling of a balanced group with $\widehat{\mu} = 4/8$, emphasizing a near-degenerate group whose baseline is estimated from a single successful episode. Mean centering preserves the observed reward scale, and constant-outcome groups naturally contribute zero advantages. Algorithm 2 summarizes the update.

A.3 SEQUENTIAL ACCEPT-OR-RESAMPLE DECODING

At inference time, SELF-STEP REJECTION applies a sequential rejection policy at each committed prefix. The generator first samples a single candidate step, which is committed when its acceptance score is at least η. Otherwise, the candidate is discarded and a replacement is sampled from the unchanged prefix, with at most M total candidate attempts at each position. If every attempt is rejected, the highest-scoring candidate is committed. An episode-level budget permits at most B rejected candidates; after it is exhausted, subsequent candidates are committed directly. Algorithm 3 specifies this procedure. Additional candidates are sampled only after a rejection; a candidate accepted on the first attempt requires no additional candidate generation. Generation and acceptance scoring share the same backbone, with the acceptance-gate adapter activated only for scoring (Appendix C).

We calibrate η before policy refinement to match a prescribed rejection rate and hold it fixed during refinement and deployment. This on-policy calibration accounts for the additional samples induced by rejections and the resulting candidate distribution encountered by the gate. We consequently

Algorithm 1: Confidence-Qualified First-Passage Labeling

Input: Problem q; frozen policy π; sampled trajectory $p_0, \ldots, p_T$; threshold τ; confidence multiplier c; rollout schedule $\mathcal{K} = (K^{(1)}, \ldots, K^{(J)})$, where $K^{(J)} = K_{\max}$.
Output: Labeled prefixes $\mathcal{D}$.

Estimate $\widehat{V}_0 = \widehat{V}^{\pi}(q)$;
if $\widehat{V}_0 = 0$ **then**
  return $\varnothing$;
$\widehat{V}_{\text{barrier}} \leftarrow (1-\tau)\widehat{V}_0$;
$\mathcal{D} \leftarrow \varnothing$;
for $t \leftarrow 1$ **to** T **do**
  $j \leftarrow 1$;
  Draw $K^{(j)}$ continuations from p_t and estimate $\widehat{V}_t$;
  $\text{gap}_t \leftarrow \widehat{V}_{\text{barrier}} - \widehat{V}_t$;
  while *the confidence interval of* gap_t *contains* 0 *and* $j < J$ **do**
    $j \leftarrow j + 1$;
    Draw additional continuations to reach $K^{(j)}$ samples and update $\widehat{V}_t$ and gap_t;
  if $\text{gap}_t + c\widehat{\text{se}}(\text{gap}_t) < 0$ **then**
    Add (q, p_t, ACCEPT) to $\mathcal{D}$;
  else if $\text{gap}_t - c\widehat{\text{se}}(\text{gap}_t) > 0$ **then**
    Add (q, p_t, REJECT) to $\mathcal{D}$ and stop;
  else
    Mark p_t gray and exclude it from the supervised loss;
return $\mathcal{D}$;

Algorithm 2: Group-Relative Policy Refinement

Input: Training problems $\mathcal{Q}$; frozen generator π_θ; initialized gate policy ρ_ϕ; group size G; threshold η; sampling temperature temperature; attempt cap M; rejection budget B.

foreach *sampled problem* $q \in \mathcal{Q}$ **do**
  for $e \leftarrow 1$ **to** G **do**
    Decode one episode using π_θ and actions sampled from ρ_ϕ, subject to M and B;
    Extract and score the final answer to obtain $R_e \in \{0, 1\}$;
    Store all gate decisions $\mathcal{T}_e = \{(x_{e,t}, u_{e,t})\}_t$;
  $\overline{R} \leftarrow G^{-1} \sum_{e=1}^{G} R_e$;
  for $e \leftarrow 1$ **to** G **do**
    $A_e \leftarrow R_e - \overline{R}$;
    Accumulate $-A_e \sum_{t \in \mathcal{T}_e} \log \rho_\phi(u_{e,t} \mid x_{e,t})$;
Normalize by the number of stored gate decisions and update ϕ;

report the realized on-policy rejection rate together with its target. The two budgets complement one another: M limits concentration of sampling at one position, while B controls total rejection across the trajectory.

B DATASETS

Our evaluation suite contains 293 contest-mathematics problems from five test-only benchmarks. No evaluation problem is used to train the acceptance gate.

AIME 2025. We use all 30 problems from the 2025 American Invitational Mathematics Examination as a single test set, sourced from MathArena (Dekoninck et al., 2026). The problems cover algebra, combinatorics, geometry, and number theory and require a three-digit integer final answer.

Algorithm 3: SELF-STEP REJECTION Sequential Decoding

Input: Problem q; frozen generator π_θ; acceptance-gate parameters ϕ; threshold η; maximum candidate attempts M per position; episode rejection budget B; step-token limit L_{step}; context limit L_{ctx}.

Output: Final completion.

$p \leftarrow \varnothing$;
$n_{\text{reject}} \leftarrow 0$;
while *p is nonterminal and L_{ctx} is not exhausted* **do**
  $\mathcal{A} \leftarrow \varnothing$;
  committed $\leftarrow$ false;
  for $i \leftarrow 1$ **to** M **do**
    $L \leftarrow \min\{L_{\text{step}}, L_{\text{ctx}} - \text{tokens}(q, p)\}$;
    Sample a candidate step $s \sim \pi_\theta(\cdot \mid q, p)$ with at most L new tokens;
    if $n_{\text{reject}} \geq B$ **then**
      $p \leftarrow p \oplus s$;
      committed $\leftarrow$ true and break;
    score $\leftarrow \text{score}_\phi(q, p, s)$;
    Add (s, score) to $\mathcal{A}$;
    if score $\geq \eta$ **then**
      $p \leftarrow p \oplus s$;
      committed $\leftarrow$ true and break;
    $n_{\text{reject}} \leftarrow n_{\text{reject}} + 1$;
  if committed $=$ false **then**
    Select the highest-scoring candidate $s^\star$ from $\mathcal{A}$;
    $p \leftarrow p \oplus s^\star$;
return p;

OlymMATH-Easy/Hard. We use all 100 English-language problems from each of the easy and hard OlymMATH test sets (Sun et al., 2026). Both contain olympiad-style problems with exact symbolic answers spanning algebra, combinatorics, geometry, and number theory. Geometry and combinatorics are the largest subject groups in the easy set, whereas the hard set is balanced with 25 problems per subject.

HMMT February 2025/2026. We use all 30 problems from the February 2025 Harvard–MIT Mathematics Tournament and all 33 from the February 2026 tournament, sourced from MathArena (Dekoninck et al., 2026). In each year, the first 30 problems are divided evenly among algebra and number theory, combinatorics, and geometry. The 2025 answers include integers, fractions, radicals, and other exact expressions; the 2026 set additionally contains three longer proof-formatted problems with rule-verifiable final answers.

Training data. We construct the acceptance gate's training pool from the 4,428 Omni-MATH problems (Gao et al., 2025). After removing duplicates, we retain rule-verifiable problems with difficulty levels 4–8, decontaminate them against all 293 evaluation problems, and draw a stratified sample. The resulting pool contains 1,100 problems, including those from HMMT February and November, Putnam, national team selection tests, the IMO Shortlist, and USAMO. We use 1,004 problems for training and hold out 96 for policy-refinement development, including early stopping, checkpoint selection, and threshold calibration.

C IMPLEMENTATION DETAILS

C.1 STEP AND LABEL CONSTRUCTION

Unless otherwise specified, Qwen3-14B and DeepSeek-R1-Distill-Llama-8B use temperature 0.6 and top-p 0.95, while gpt-oss-20b uses temperature 1.0, top-p 1.0, and `reasoning_effort=medium`. All generators use a context limit of $L_{\text{ctx}} = 32{,}768$ tokens.

A reasoning step is one generation segment containing at most L_{step} newly generated tokens. Unless otherwise specified, we set $L_{\text{step}} = 2048$. For first-passage labeling (Algorithm 1), we use $\tau = 0.30$, confidence multiplier $c = 1$, and rollout schedule $\mathcal{K} = (8, 12, 16)$, with $K_{\max} = 16$. A step may terminate earlier when generation finishes, and its generation budget is dynamically reduced when fewer than L_{step} positions remain in the context window.

During label collection, each evaluated prefix permits at most $M = 8$ candidate attempts. Candidate continuation values are estimated within the remaining context,

$$L_{\text{available}} = L_{\text{ctx}} - \text{tokens}(q, p_t). \tag{12}$$

The resulting confidence-qualified decisions are converted into labels using the first-passage rule in Appendix A.1. A smaller generation cap is treated as an explicit budgeted condition rather than as the model's context limit. For reproducibility, each estimate retains its decoding configuration, seed, token counts, extracted answer, correctness, finish reason, and exact continuation or cutoff prefix.

C.2 Gate Architecture and Training

Input serialization. The acceptance-gate input concatenates the problem, committed prefix, candidate step, and a fixed marker token. The dynamically adjusted continuation budget keeps the problem, prefix, and candidate within L_{ctx}. The marker is used only for scoring and is excluded from the generation-token budget. We use the pretrained reserved token `<|fim_pad|>` as the marker and require that it not occur in any input segment. The marker is appended at the final input position.

Token-based readout. The frozen language-modeling head is evaluated at the marker position. Following the token-based readout of Math-Shepherd (Wang et al., 2024), the existing single-token vocabulary entries + and − verbalize ACCEPT and REJECT, respectively. If their logits are ℓ^+ and ℓ^-, training uses the logit difference $z = \ell^+ - \ell^-$ with binary cross entropy with logits, while inference uses the equivalent two-token softmax score $\text{score} = \exp(\ell^+)/[\exp(\ell^+) + \exp(\ell^-)]$. The marker embedding, vocabulary embeddings, and language-modeling head remain frozen; only the LoRA adapter is trained. Reusing the pretrained language-modeling head provides a stronger inductive bias under limited gate supervision, for which a newly initialized classification head readily overfits.

LoRA configuration. The only trainable parameters are rank-16 LoRA adapters with scaling factor 32, zero dropout, and no bias. They are attached in every transformer layer to `q_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. The key and value projections remain frozen. Generation and acceptance scoring maintain separate incremental key–value caches over the committed prefix.

Training, development, and test partitions are disjoint at the problem level. We optimize with AdamW (Loshchilov & Hutter, 2017) for three epochs at learning rate 3×10^{-4}, effective batch size eight, and gradient-norm clipping at 1.0. Class-dependent weighting addresses the imbalance in first-passage labels. For same-prefix accept/reject pairs, the pointwise losses are augmented with $\lambda_{\text{pair}} \,\text{softplus}[-(z_a - z_r)]$, with $\lambda_{\text{pair}} = 0.5$ and $\text{softplus}(x) = \log(1 + \exp x)$. This smooth ranking loss encourages a higher logit for the accepted candidate. Checkpoint selection uses within-prefix discrimination and accepted-value lift.

C.3 Policy-Refinement Configuration

Policy refinement uses $G = 8$ episodes per problem and $\text{temperature} = 1.0$. We set the target rejection rate to 17.8%, based on the rate measured under oracle-gated decoding. Before refinement, we calibrate η to the corresponding empirical quantile of the initial adapter's on-policy acceptance scores. The resulting threshold remains fixed throughout policy refinement and deployment; the realized rejection rate is reported separately in Table 4.

Each iteration draws 16 fresh training problems, sampling without replacement until the pool is exhausted and then reshuffling it. The resulting groups are combined into an effective batch of 256 gate decisions, and one AdamW update is applied at learning rate 5×10^{-5}, zero weight decay, and gradient-norm clipping at 1.0. Generation uses temperature 0.6 and top-p 0.95 for Qwen3-14B and DeepSeek-R1-Distill-Llama-8B, and temperature 1.0, top-p 1.0, and `reasoning_effort=medium` for gpt-oss-20b. All generators use 2048-token reasoning steps and a context limit of 32,768 tokens. Each position permits $M = 8$ total candidate attempts, and each episode permits $B = 12$ rejected

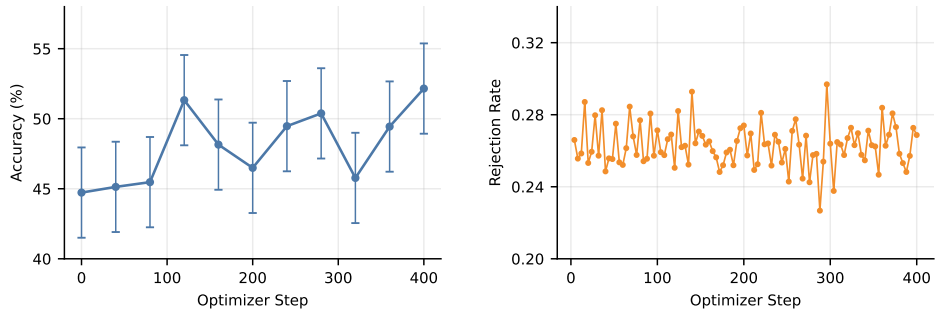

Figure 6: Held-out development accuracy (left) and realized on-policy rejection rate (right) over optimizer steps during policy refinement with Qwen3-14B.

Table 4: SSR deployment behavior on Qwen3-14B, averaged over three evaluation runs.

Metric	Mean
Candidate rejection rate	21.0%
Candidate attempts per committed step	1.3
Committed steps per problem	8.82
Gate evaluations per problem	10.7
Steps reaching the M-attempt cap	0.4%
Problems exhausting the B-rejection budget	3.0%

Table 5: Qwen3-14B wall-clock time per evaluation run, with generation on six H200 GPUs.

Method	Wall-clock time (h)
Single pass	1.09
SSR	1.26
ESC ($w = 2$)	4.85
Majority@8	8.81
USC	8.95

candidates. When the physical context is exhausted, we extract and score any available final answer; an episode with no extractable answer receives reward zero.

Training is bounded by a preset compute budget and development early stopping. Development evaluation samples four episodes for each of 96 held-out Omni-MATH problems. An evaluation more than two standard errors below the historical best is marked as degradation. Training stops after two consecutive degraded evaluations, and the checkpoint with the highest development accuracy is selected. Figure 6 summarizes the optimization trajectory: held-out accuracy follows a noisy upward trend, while the realized on-policy rejection rate remains broadly stable under the fixed η. Training uses a single H200 node, with a total wall-clock training time of approximately 200 hours per model.

C.4 Inference Runtime

Candidate generation runs with the LoRA adapter disabled through the adapter context, preserving the frozen generator exactly. Candidate evaluation enables the adapter and reads only the final marker position. Because the marker is guaranteed to be last, we request one output position instead of materializing logits of shape sequence-length by vocabulary-size. Generation and acceptance scoring each maintain an incremental key–value cache over committed steps. Scoring a new candidate processes only its new tokens and the appended marker rather than recomputing the full trajectory. Rejected candidates use temporary cache branches that are discarded after scoring. Table 4 shows that the learned gate rejects selectively: the per-position attempt cap is reached on only 0.4% of committed steps, and the episode budget is exhausted on 3.0% of problems. As shown in Table 5, SSR adds 0.17 hours over single-pass decoding while remaining substantially faster than full-solution scaling.

C.5 Evaluation Metrics

Gate metrics. Global AUROC is insufficient because it rewards problem-difficulty prediction. Our primary offline metrics are within-prefix pairwise AUROC, rejection precision at a fixed rejection rate, and the accepted-value lift

$$\Delta V_{\text{accept}} = \mathbb{E}[V^{\pi}(q, p \oplus s) \mid s \text{ accepted}] - \mathbb{E}_{s \sim \pi(\cdot|q,p)}[V^{\pi}(q, p \oplus s)]. \tag{13}$$

Table 6: Measured SSR sensitivity to the target rejection rate on the test suite with Qwen3-14B and $L_{\text{step}} = 2048$. Results are averaged over three evaluation runs.

Target rej. (%)	Tok.×	Accuracy (%)
7.5	1.07	60.3
10.0	1.10	61.0
12.5	1.14	62.1
15.0	1.17	63.2
17.8	1.21	64.6
22.0	1.27	64.6
30.0	1.37	64.7

Table 7: Measured SSR sensitivity to L_{step} on the test suite with Qwen3-14B at a fixed target rejection rate of 17.8%. Results are averaged over three evaluation runs.

L_{step}	Avg. steps	Tok.×	Accuracy (%)
256	63.3	1.27	54.7
512	31.7	1.27	54.4
1024	15.8	1.26	64.6
2048	8.82	1.21	64.6
4096	4.1	1.23	64.1
8192	2.3	1.21	62.2

We evaluate four baselines. Length+depth is a fitted two-feature probe using candidate token length and committed-prefix depth. Log-prob uses the frozen generator's mean per-token log-probability from a teacher-forced pass over the fully rendered prompt and requires no fitting. Frozen MLP is a one-hidden-layer probe trained on standardized, PCA-reduced final-token hidden states from a Transformer layer fixed before evaluation, with the prompt included. The Skywork verifier directly applies Skywork-Reward-V2-Qwen3-8B to the complete problem, prefix, and candidate sequence without task-specific training. All fitted probes are trained only on problem-grouped training folds.

End-to-end metrics. We report final-answer accuracy, generated-token cost relative to single-pass generation, candidate rejection rate, candidate attempts per committed step, M-cap fallback rate, B-budget exhaustion rate, and end-to-end wall-clock time. All online policies are evaluated in paired runs with deterministic problem/sample seeds. Threshold η, attempt cap M, episode rejection budget B, and checkpoint are selected on development data only.

D Additional Results

D.1 Power Analysis for Local Value Differences

For two Bernoulli proportions near success probability μ, the approximate number of rollouts per endpoint required to resolve an effect δ with 80% power is $n \approx 2(1.96 + 0.84)^2\mu(1-\mu)/\delta^2$. Using the magnitude of the mean adjacent change rounded to $\delta = 0.05$, the worst-case $\mu = 0.5$ calculation gives approximately 1,600 rollouts per endpoint.

D.2 Sensitivity Analysis

All sensitivity results in this subsection are measured with Qwen3-14B.

Target rejection rate. Table 6 varies the target rejection rate and calibrates η for each operating point. Accuracy improves as the gate is allowed to reject more candidates, then plateaus across target rates from 17.8% to 30.0% while token cost continues to increase. The default target of 17.8% lies near the start of this plateau.

Step size. Table 7 varies L_{step} with the target rejection rate fixed at 17.8%. Very short steps create many decision points and increase scoring and resampling cost. Very long steps delay intervention and yield lower accuracy without reducing generated-token cost below the 2048-token setting. Step sizes of 1,024–4,096 tokens achieve similar accuracy, with the default $L_{\text{step}} = 2048$ using the fewest generated tokens in this range.

D.3 Examples

We present two measured SSR trajectories that yield correct answers where the corresponding single-pass runs fail. On AIME 2025 Problem 10, Qwen3-14B rejects two candidates, one introducing an unstated structural constraint and the other leaving the counting argument unresolved, and returns the correct answer, 81 (Table 8). On HMMT February 2026 Problem 19, gpt-oss-20b rejects a faulty

Table 8: Abridged SSR trajectory for AIME 2025 Problem 10 with Qwen3-14B. Two candidates are rejected; omitted steps are accepted on their first attempt.

Step	Decision	Abridged candidate reasoning
$t = 1$	Accept	Identifies the row and 3×3-block constraints and begins reducing the problem to a combinatorial count.
$t = 2$, cand. 1	Reject	Incorrectly infers that entries must also be distinct within each column of a block and repeatedly recasts the block as a Latin-square-like object.
$t = 2$, cand. 2	Accept	Retains only the stated row and block constraints and represents each block by an ordered partition of $\{1, \ldots, 9\}$ into three row subsets.
$t = 3$–5	Accept	$\cdots$ Reduces the remaining task to counting the assignments for the second and third blocks after fixing the first block.
$t = 6$, cand. 1	Reject	Formulates the assignment as a constrained bipartite matching problem but does not obtain the required count.
$t = 6$, cand. 2	Accept	Derives the row-assignment count $\sum_{c=0}^{3} \binom{3}{c}^3 = 56$ and obtains $9! \cdot 56 \cdot (3!)^6$.
$t = 7$–10	Accept	$\cdots$ Verifies $9! \cdot 56 \cdot (3!)^6 = 2^{16}3^{10}5\,7^2$ and returns $2(16) + 3(10) + 5(1) + 7(2) = \boxed{81}$.

Table 9: Abridged SSR trajectory for HMMT February 2026 Problem 19 with gpt-oss-20b. One candidate is rejected; omitted steps are accepted on their first attempt.

Step	Decision	Abridged candidate reasoning
$t = 1$	Accept	Reads $\lvert A_i \rvert = i$ from the case $i = j$ and restates the objective as minimizing the eight maxima $\max A_d$ over the divisors of 250.
$t = 2$	Accept	Observes that the elements are arbitrary positive integers chosen by the construction, so only which labels are assigned to which sets matters.
$t = 3$, cand. 1	Reject	Assigns one element e_t to every A_k with $t \mid k$, computes that this yields $\lvert A_i \cap A_j \rvert \geq d(\gcd(i, j))$ rather than $\gcd(i, j)$, and on finding the count too small begins a prime-by-prime case analysis instead of revising the construction.
$t = 3$, cand. 2	Accept	Derives the structural lemma: for $d \mid n$, $\lvert A_n \cap A_d \rvert = d = \lvert A_d \rvert$ forces $A_d \subseteq A_n$, so the family is nested along divisibility. Assigns each element its minimal index m and concludes $e \in A_n$ iff $m \mid n$.
$t = 4$–7	Accept	$\cdots$ Writes $\lvert A_n \rvert = \sum_{m \mid n} f(m) = n$ for the count $f(m)$ of elements of minimal index m, giving $f = 1, 1, 4, 4, 20, 20, 100, 100$ on the divisors of 250, and reduces the objective to choosing each group's largest label $g(m)$, since $\max A_d = \max_{m \mid d} g(m)$.
$t = 8$–12	Accept	$\cdots$ Evaluates orderings of the eight groups. Labeling them in ascending order of size gives $g = 1, 2, 6, 10, 30, 50, 150, 250$ and the sum 499, while moving index 5 earlier raises the total to 502.
$t = 13$	Accept	Terminal step. Rules out the smaller candidate value 468, which would require $g(d) = d$ and violates the monotonicity $g(m) < g(n)$ for $m \mid n$, and returns $\boxed{499}$.

construction, replaces it with the nesting property $A_d \subseteq A_n$ used by the remainder of the solution, and returns the correct answer, 499 (Table 9).